MS-2000 XY Piezo Inverted Microscope Stage Instruction Manual

Applied Scientific Instrumentation, Inc.

29391 W. Enid Rd.
Eugene, OR 97402-9533 USA

Phone: (800) 706-2284
(541) 461-8181
Fax: (541) 461-4018
Web: www.ASIimaging.com
E-mail: Support@ASIimaging.com

for Controller Firmware Version 8.0, and later

Table of Contents

INTRODUCTION

This manual pertains to ASI's MS-2000 microscope stage control system. The MS-2000 is a compact, highly functional, computer-controlled stage system that can be configured in several different ways, depending upon the needs of the user. The basic system consists of the MS-2000 control unit and an XY stage. ASI's precision Z-axis focus control drive can be incorporated to create an integrated XYZ system. For ultimate accuracy and repeatability of positioning, the MS-2000 can be configured with precision linear encoders on any axis. Autofocus and laser feedback options are also available for automated processes and ergonomic ease of use.

This manual will describe the installation, operation, and programming for basic system components, plus sections for applicable options. Please contact ASI regarding addition options if you wish to upgrade your system.

Features and Capabilities of an MS-2000 System

- Closed-loop DC servo motor control of the X, Y, and Z axes for precise positioning and highly repeatable focusing
- Sub-micron repeatability on all axes
- Wide dynamic speed range with adjustable trapezoidal move profiles
- Compact ergonomic tabletop control unit is 3½ x 9 x 6½ inches (9 x 23 x 16½ cm)
- Back-lit LCD display shows X, Y, and Z coordinates
- Smooth adjustable dual-range joystick control
- Microprocessor control with RS232-C serial and USB communications
- Z-axis clutch for easy switching between manual and motor-driven focus control
- X and Y axis Hall-effect limit sensors
- Electronic torque limit on drives minimizes damage by runaway stage
- Configurable autofocus parameters
- "Zero" button for setting "Home" position
- Other functions including programmable positioning patterns and scans

Installing the MS-2000 XY Stage onto your Inverted Microscope

The procedure has three parts:
Removing the original stage.
Install the MS-2000 XY stage on the microscope
Connecting the cables on the MS-2000 control unit

Part 1 – Remove the Original Stage

Before removing or installing the stage, the objectives should be removed from the nosepiece and stored in a safe and secure place until after the stage has been installed and aligned.

Remove the retaining bolts and carefully remove the original microscope stage assembly.

Please refer to your microscope's manual for any further information about removing the original stage.

Part 2 – Install the MS-2000 XY Stage on the Microscope

Mount the stage with new screws supplied with the ASI stage.
Slide the stage onto the microscope with the connector to the right, and the Y-axis leadscrew to the right. Align the screw holes on the front and rear of the stage with the holes located on the microscope. Secure the stage to the microscope with the supplied screws and Allen wrench.

Part 3 – Connecting the Cables on the MS-2000 Control Unit

The XY stage's 5 ft. cable connects to the controller's 25-pin female D-connector. If included, the Z-axis drive's 6 ft. cable connects to the controller's 15-pin male D-connector. An RS232 cable can be connected to the controller's 9-pin female Serial IN D-connector, and to a serial communications port on a host computer. A USB cable can be connected to the USB B-connector on the back of the controller, and to a USB A-port on a host computer. The cable from the power supply module is plugged into the 3-pin male circular connector on the back of the controller, and the module's power cord connected to a proper power source.

Installing ASI Slide Inserts Into The ASI Stage

The easiest way to install the ASI slide insert into the stage is to slide the insert in from the right, towards the spring clips, as shown in figures 1 through 3 below.

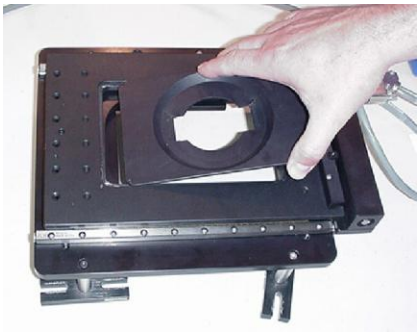


Figure 1. Slide the insert from the side of the stage that does not have the spring clips.

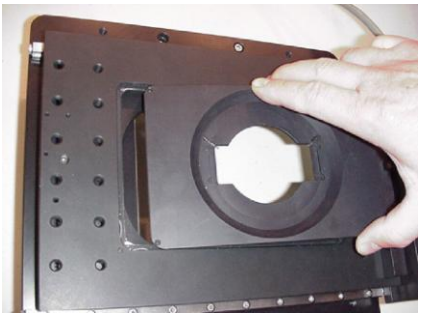


Figure 2. Slide the insert so that it is pressed against both the back and front guides. In this manner the front spring clip is engaged first and the side spring clip engaged last.

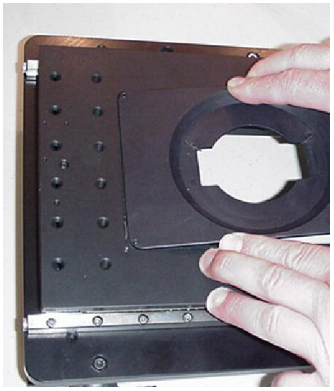
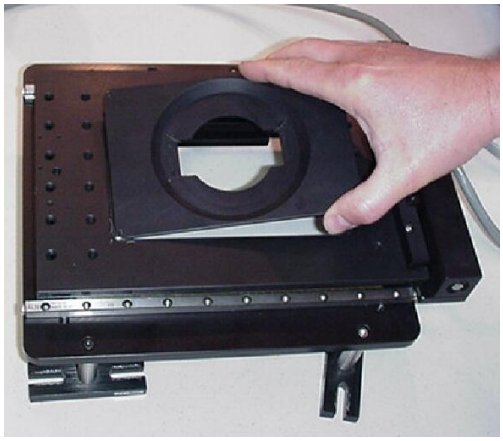
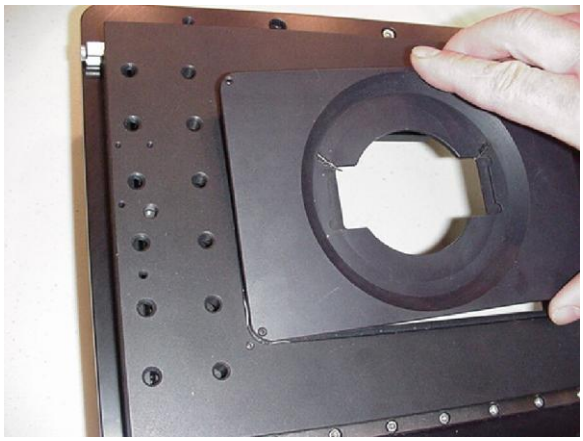


Figure 3. When the insert engages the spring clips, press down with one hand while continuing to slide the insert into place with the other hand.





Figures 4 & 5: If you try to use the corner of the insert to compress both of the spring clips at once, it will be difficult to install the inserts.

Setting Limits on Inverted MS-2000 Stages

The two limits on the top of the stage, shown in figures 1a and 1b below, are used to limit the range of motion in the Y-axis, and are located in the top plate of the stage. The two limits, shown in figures 2a and 2b below, are used to limit the range of motion in the X-axis, and are located in the middle plate of the stage. The X-axis limit adjustment setscrews can be accessed by moving the Y-axis towards the front of the microscope. You will need a .050-inch Allen wrench that was supplied with the stage to make these adjustments.

Do not over-tighten the limit switch's setscrews. Tighten only until the limit assembly does not move. Over-tightening the upper assemblies causes them to bend outward and even break. Also, ensure that the upper limit assemblies are held firmly against their rail to prevent them from touching or scratching the leadscrew cartridge.

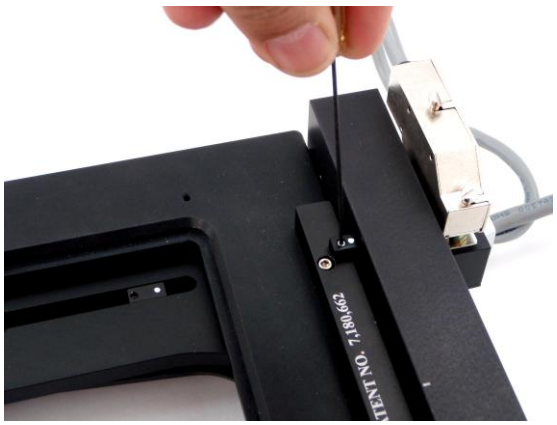


Figure 1a: Loosen the setscrew so that you can move the Y-axis lower limit located at the rear right of stage. **Do not over-tighten.**

Figure 1b: Loosen the setscrew so that you can move the Y-axis upper limit located at the front right of stage. **Do not over-tighten.**

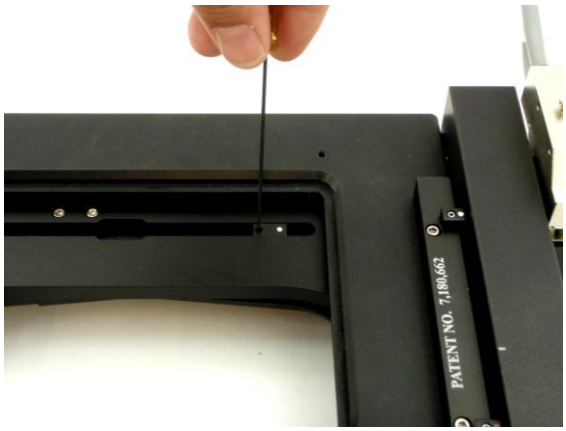


Figure 2a:. Loosen the setscrew so that you can move the X-axis lower limit located at the right rear of the middle plate of stage. **Do not over-tighten.**

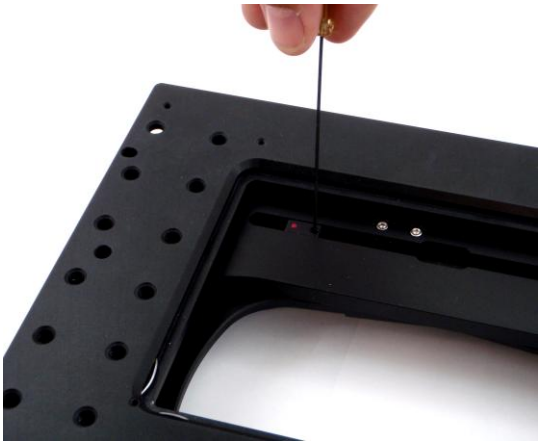


Figure 2b: Loosen the setscrew so that you can move the X-axis upper limit located at the left rear of the middle plate of stage. **Do not over-tighten.**

The limits are set at the factory to allow the longest range of travel in the X and Y-axes. However, the limits have been designed to allow you, the user, to control the range of travel of the stage. This can be done to prevent the stage from running into the microscope, or parts of the microscope such as the objective, shown in the example below. Please note that if you set the limits as shown in the example below you will be limiting the stage's travel to scan a 25 x 75 mm slide. This is just an example, and the limits can be set to cover any subset of the full range of travel. Please also note that, if the limits are set such that the stage encounters a mechanical obstruction before the limits are reached, the stage does have over-current protection. However, this is provided only as an emergency safety measure, and over time the stage can be damaged if the limits are not properly set.

Please refer to the above figures 1a through 2b and the figures below, as an example of how to set the limits to prevent the slide insert from hitting the objective.

Setting the Y-axis Lower Limit

Please refer to figure 1a above for loosening the setscrew that secures the Y-axis lower limit.

Setting the Y-axis lower limit:

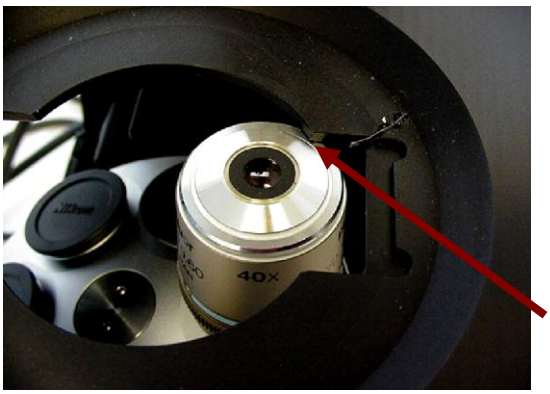


Figure 3a: Move the stage in Y-axis so that it almost hits the objective as shown.

Figure 3b: After loosening the setscrew as shown in figure 1a, move the limit until the "**L**" appears in the Y-axis display as shown in figure 3c. **Do not over-tighten.**

Figure 3c: Lower limit "**L**" displayed on Y-axis

Setting the Y-axis Upper Limit

Please refer to figure 1b above for loosening the setscrew that secures the Y-axis upper limit.

Setting the Y-axis upper limit:

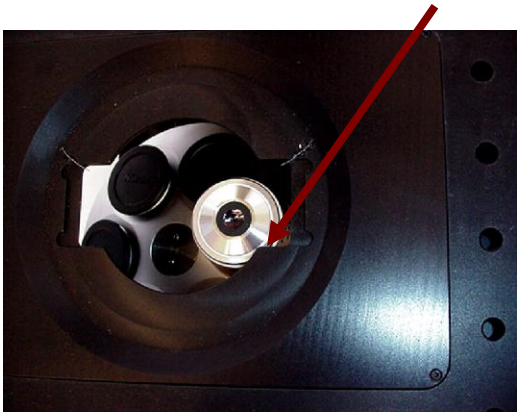


Figure 4a: Move the stage in Y-axis so that it almost hits the objective as shown.

Figure 4b: After loosening the setscrew as shown in figure 1b, move the limit until the "**U**" appears in the Y-axis display as shown in figure 4c. **Do not over-tighten.**

Figure 4c: Upper limit "**U**" displayed on Y-axis

X-axis Range of Travel Limits

The X-axis limits are located on the middle stage plate, towards the rear of the microscope, as shown below in figure 5.

Figure 5

Setting the X-axis Lower Limit

Please refer to figure 2a above for loosening the setscrew that secures the Y-axis lower limit.

Setting the X-axis lower limit:

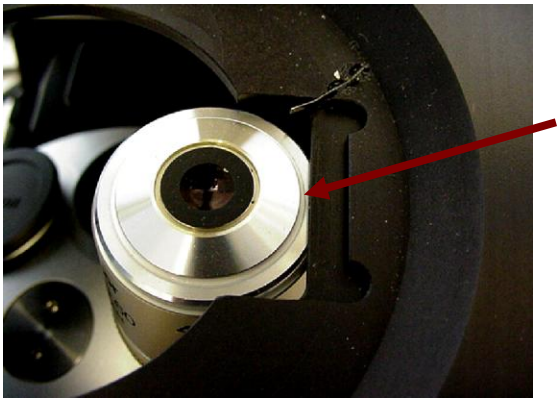


Figure 6a: Move the stage in X-axis so that it almost hits the objective as shown.

Figure 6b: After loosening the setscrew as shown in figure 2a, move the limit until the "**L**" appears in the X-axis display as shown in figure 6c. **Do not over-tighten.**

Figure 6c: Lower limit "**L**" displayed on X-Axis.

Setting the X-axis Upper Limit

Please refer to figure 2b for loosening the setscrew that secures the Y-axis upper limit.

Setting the X-axis upper limit:

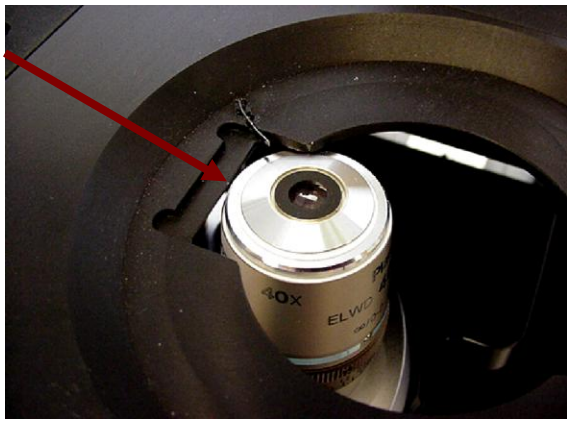


Figure 7a: Move the stage in X-axis so that it almost hits the objective as shown.

Figure 7b: After loosening setscrew as shown in figure 2b, move the limit until the "**U**" appears in the X-axis display as shown in figure 7c. **Do not over-tighten.**

Figure 7c: Upper limit "**U**" displayed on X-axis

.

Piezo-Z Top-Plate Option for ASI XY stage

The PZ-2000 series stage top plates are designed to provide a high resolution, and highly repeatable Z position of the microscope stage. The top plate of the stage accepts standard K-size slide inserts that are available for any sample, i.e., slides, petrie dishes, multi-well plates, etc. The slide insert is moved in the Z-axis via a piezo element with a range of 150 µm with nanometer accuracy (300 µm & 500 µm range are also available). By moving the sample in the Z-plane, any objective can be used, eliminating twisting wires or needed spacers as required when a piezo element is put onto a single objective.

Installation and Handling

The piezo top plate is built into the PZ-2000 series stage, so no special installation steps are required for the top plate option. The stage is mounted on the microscope according to the instructions in the MS2000 Instruction manual. The cable from the piezo-Z stage should be plugged into the connector on the MS2000 controller labeled **PZ STAGE.** An appropriate stage insert should be inserted into the opening in the stage top. The insert should slip neatly into the top, being held in place by flat springs on two sides of the insert.

Warning – The piezo stage top is a delicate precision device.
Avoid:

- Dropping the stage or dropping heavy objects on the stage top.
- Appling large forces or torques to the stage driven top plate.

ASIs Drive Electronics for Piezo Top-Plate

ADEPT's Features:

- Closed loop operation with *strain gauge* feedback.
- Auto *strain gauge* calibration on startup.
- Accepts inputs from MS2000 (digital) or external (analog) sources.
- Nanometer precision with low noise and drift.

Figure 1 ADEPT Card

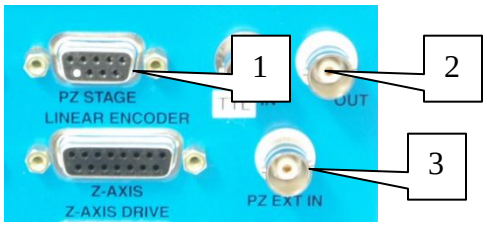
Introduction

ADEPT is an add-on module that extends the MS2000 controller's capability to control *piezo top-plate*. The user can position the *piezo top-plate* through MS2000s serial commands or manual inputs like the knob, or through an external signal source. It can also be used also with other MS2000 modules like **AUTOFOCUS, CRIFF, and SEQUENCER** etc

ADEPT can operate either in open loop or closed loop mode. In closed loop mode, it uses high sensitivity semiconductor *strain gauge* as sensors to position the *piezo top-plate* with nanometer precision and transient time less than 25ms.

MS2000 performs calibration on ADEPT card on startup, so the system maintains calibration.

Connectors



1. Piezo Top-plate Connector
2. Sensor Out
3. External Input

Figure 2 MS2000 Connections

External Input and Output

Apart from using the ADEPT card to position the piezo top-plate through MS2000, this card can also get positioning information through 0-10V analog signal.

When the controller is set to external input mode (using the `PZ Z` command or holding the @ button down for 6 sec), ADEPT positions the *piezo top-plate* based on the voltage applied to EXT IN or external input. The analog signal should be between 0 to 10 volts, where 10 represents lower limit, and 0 volts represents higher limit. For every 1 volt change, the piezo top-plate changes $\frac{1}{10}$ the piezo range. Example, if it's a 150um *piezo top-plate* 1 volt represents 15um, 0.1 volt represents 1.5um. We recommend that frequency of the signal be kept less than 10Hz for long moves, so as to give the *piezo top-plate* time to settle down and come to a complete stop.

In external input mode, the letters "XT" are display next to the axis position on LCD, and the position displayed is extracted from the strain gauge feedback. This position is accurate to 1um.

Sensor Out is a 0 to 10 volt analog signal that corresponds to position of the *piezo top-plate*. It is a buffered output of the *strain gauge* sensors.

Command Set

Apart from the regular axis commands like **`MOVE, MOVREL, ZERO, INFO`** (refer to MS2000 Programming section) here are some ADEPT specific commands. Most users do not need to use these commands unless configuration are changed or problems are encountered.

Command: PZ

Format `PZ X=[1 to 255] Y=[25 to 5101] Z=[0 to 3]`

Function

- X argument sets the *zero adjust potentiometer* of the ADEPT card. Only values between 1 to 255 are accepted. The setting is stored in a non volatile memory on the ADEPT board. This is one of the settings that are automatically picked during both long and short auto calibration. Please refer the *calibration* section for its usage.

- Y Argument sets the gain of the feedback stage. Only values between 25 to 5100 are accepted. The setting is stored in a non volatile memory on the ADEPT board. This is one of the settings that is automatically picked during long auto calibration. Please refer the *calibration* section for its usage.

- Z Argument sets the board in various modes, Open Loop, Closed loop, MS2000 input and External input.

Z Argument sets the Mode of Operation of the ADEPT board.

PZ Z =	Mode of Operation
0	MS2000 input, Closed loop (default)
1	External input, Closed Loop
2	MS2000 input, Open loop
3	External input, open loop

In Open Loop mode, a set voltage is applied to the piezo and the feedback from strain gauge is ignored. Useful during system calibration.

In Closed Loop mode, the voltage applied to the piezo is adjusted according to the feedback coming from the strain gages. This is the default mode of operation.

MS2000 input, in this mode the MS2000 controller generates the positioning input for the *piezo top-plate*. This is the default mode of operation.
In External input mode, the *piezo top-plate* is positioned according to 0 to 10V analog signal provided by the user. Every one volt change moves the piezo $\frac{1}{10}$ the range. We recommend that frequency of the signal be kept less than 10Hz for long moves, to give the *piezo top-plate* sufficient time to settle and come to a complete stop.

The user can toggle between external input mode and MS2000 input mode by holding the "@" button down for 3+ sec. Use a clock if you are having difficulty.

The modes will revert back to default state, i.e. MS2000 input with Closed Loop (`PZ Z=0`) when system is powered off. Use the **`ss z`** command to save your preference.

Command: PZC

Format `PZC X=[0 or 1] Y=[0,1,2,3] Z=[1 to 100] F=[1 to 100] ,or PZC`

Function `PZC` when entered alone runs the *auto calibration* routine that sets various internal parameters for optimal operation of the *piezo top-plate*. `:A` is returned on completion, `:N-5` is returned if the routine failed.

- X argument sets the auto calibration type to perform. 0 is for short calibration (default) i.e. only strain gauges offset is adjusted. While 1 is long calibration routine, with adjusts both strain gauge offset and the feedback gain. You will need a length gauge to run the full calibration routine. `ss z` command is not applicable, settings will revert back to default when controller restarts.
- Y argument sets the axis index to which the length gauge is assigned. Default is 0 i.e. X index in a 4 axis build. `ss z` command not applicable,

settings will revert back to default on controller restart.

- Z argument sets the delay between routine runs, default is 35 i.e. 35ms. Units are in milliseconds. `ss z` command not applicable, settings will revert back to default on controller restart.
- F argument sets the position where controller moves the *piezo top-plate* before adjusting the strain gauge offset. Accepts values between 1 to 100, units are %, default is 50 i.e. middle of the piezo range. `ss z` command **is** applicable, settings will be saved between controller restarts.

Please use `HALT` command to stop a running calibration routine; else the routine will leave incorrect settings on the ADEPT card.

Command: PZINFO

Format `PZINFO`

Rely
```
Voltages @ Pos1>
HV   : 147 V
Sout : 4 V
PZout: 65 V
I2C Check> DAC[OK] SWITCH[OK] DigPot[OK]
DigPot> SGoffset: 110 Gain: 96
Closed Loop
MS2000 IN
SG Offset [OK]
```

Function When issued MS2000 replies with all relevant ADEPT card settings to aid in trouble shooting. Please refer to the Troubleshooting section for explanation of the message.

Button: @

Format `@ long press i.e. holding down @ button for 3+sec`

Function Toggle between external input mode and MS2000 input mode.

Troubleshooting

- On startup if the axis with *piezo top-plate* comes up as disabled with either a COD error of 140 or 141, it indicates one of the startup self test for the ADEPT card has failed. Use the PZINFO command to further investigate.
- If you're noticing any odd behavior the PZINFO command is a good way to debug the error. When this serial command is issued , MS2000 controller performs a series of test and returns the result that looks like this:

```
Voltages @ Pos1>
HV   : 147 V
Sout : 4 V
PZout: 65 V
I2C Check> DAC[OK] SWITCH[OK] DigPot[OK]
DigPot> SGoffset: 110 Gain: 96
Closed Loop
MS2000 IN
SG Offset [OK]
```

  - Voltage @ Pos1 indicates the Voltage on High Voltage Rail as HV, Voltage on Sensor Out as Sout and Voltage being applied to the *piezo top-plate* as PZout at the current position. High Voltage rail should always be in high 140s and more, if not it indicates a problem on the board.
  - I2C Check pings the various digital ICs on the ADEPT board and returns OK or BAD. OK indicates that the IC is powered and is communicating with MS2000. BAD indicates a problem.
  - DigPot returns the current calibration settings, strain gauge offset and feedback gain.
  - Next it indicates whether the card is in open loop or closed Loop mode.
  - Then it displays the current source of piezo top-plate position. MS2000 IN or EXT IN.
  - SG Offset is a quick routine to checks the validity of the *strain gauge* offset setting, and returns OK or BAD. This routine will only check the strain gauge offset, and will not try to correct it. Issue a `PZC` command to run calibration routine; it will automatically pick the correct value.

Calibration

Note: All ADEPT boards are calibrated in the factory and undergo thorough testing. Users do not have to perform any calibration. This section is for your understanding of the *piezo top-plate's* working.

For optimal closed loop operation, when a *strain gauged piezo top-plate* and ADEPT board are paired together calibration is performed. Every piezo system is calibrated in factory, and a shorter self calibration is performed by MS2000 on startup. So ideally users do not have to worry about calibrations, offsets etc. Here is a short description of what they involve.

Strain Gauge Offset: ADEPT uses two semiconductor strain gauges installed in the *piezo top-plate* in half bridge configuration for feedback. One strain gauge flexes as the *top-plate* moves while the other is installed in a non flexing position. As the top-plate moves up and down, the flexing strain gauge's resistance relative to the unflexing strain gauge increases and decreases, the sensor conditioning circuit interprets this as a voltage. The *strain gauge* offset parameter is the setting for a digital potentiometer that adjusts the resistance of the unflexing strain gauge so that at a position of our choosing (set by `PZC F` command) it has the same resistance as the flexing strain gauge. This setting can be viewed and adjusted with `PZ X` serial command. It is automatically picked by the MS2000 controller during auto calibration routine, and also during the shorter self calibration routine on startup.

When a `PZINFO` command is issued, among one of the check it runs is for the strain gauge offset and it returns a `SG Offset [OK]` or `[BAD]`

Feedback Gain: When the piezo top-plate moves, the strain gauge interprets this as the change in resistance. The feedback gain is the parameter that helps the sensor conditioning circuit interprets this change in resistance as a voltage, and then into distance moved.

Performance

Electrical Performance

- Capable of Applying -24V to 150V to the Piezo actuators
- Maximum continuous output current of 13mA
- 11msec Transient Response time (10%-90%) for moves below 30% travel range with 600grams load.

Characteristics

Name	PZ-2150	PZ-2300	PZ-2500
Piezo Travel Range (+- 5%)	150um	300um	500um
Piezo smallest move/resolution *	2.2nm	4.5nm	7.6nm
Maximum Load for full range travel in Kilogram	2Kg	1Kg	1Kg
Transient Response time **	11 – 15 millisec		
External Analog input(BNC)	0 to 10 Volts		
Maximum Input Frequency	20 Hz		

*This is with MS-2000 controller's 16Bit DAC. User may improve this thru a high resolution signal generator, and the system operating in external input mode.

** Time took to travel 10%-90%, for moves below 30% travel range and with 600 grams load.

Transient Response

Figure 3 the above graph shows the step/transient response of ASIs piezo top-plate for a 30um move. The rise time is 11msec.

The above graph is the transient response of a 150um ASI piezo top-plate. Red waveform is the user commanded input or desired output. Green waveform is the actual piezo top-plate position interpreted from sensor out. Here it shows the piezo top-plate travel 30um with a rise time (10% to 90%) of 11 msec. The piezo top-plate was loaded with 600grams payload.

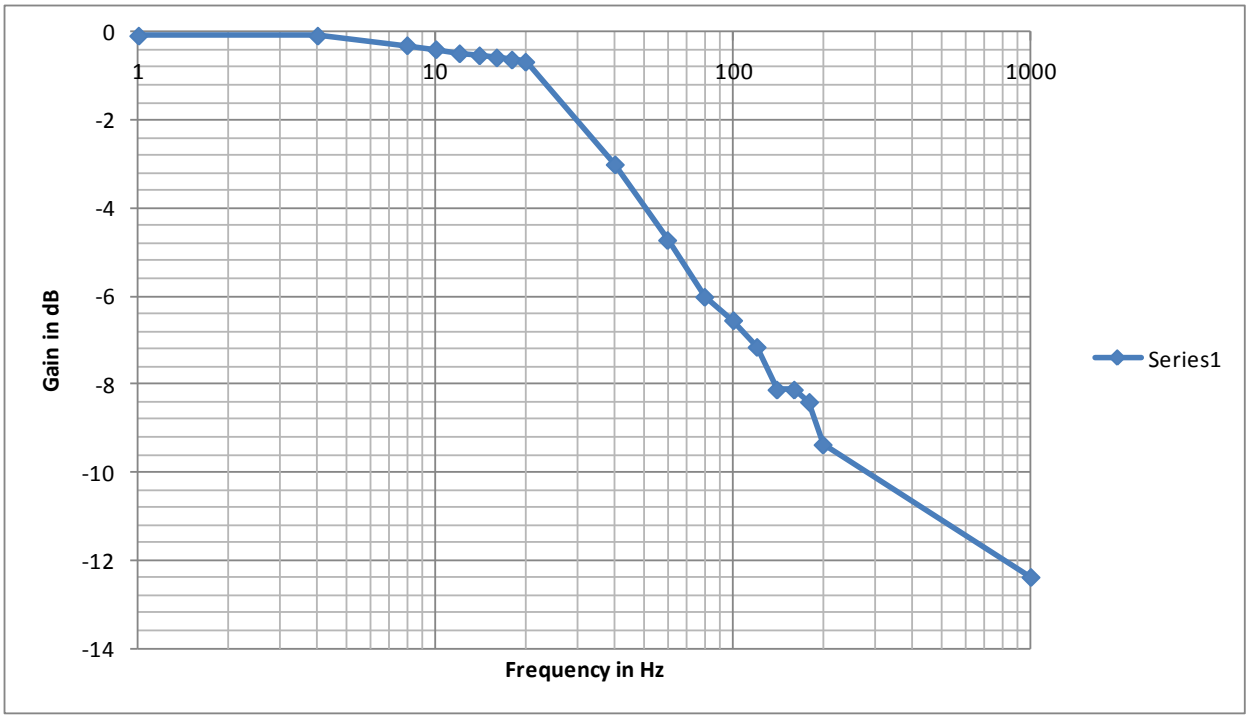
We observed that the transient tends to increase for longer moves.

Travel Range	150um	300um	500um
30%	11-12ms	9 - 10.5ms	10 - 14.5ms
50%	10-12ms	9 - 14.5ms	26 - 30ms
70%	13.5-17ms	14.5 - 17ms	33 - 45ms

All values listed above are with 600grams load.

Frequency Response

In this test a sine wave of 2 Volt peak to peak and 5Vrms of varying frequency is applied thru external input and the response of the top-plate was interpreted on sensor out is observed and its amplitude is recorded. This is used to determine what the maximum possible input frequency is. Top-plate is loaded with 600grams payload.



Observation: As the moves come more frequently, the piezo top-plate may not have sufficient time to finish its move and settle in and the move is left incomplete. The above graph shows that over 10Hz or more that 10 commanded moves a second, there are a lot of incomplete moves and at 40Hz the top-plate only travels half the commanded distance.

Top-plate Load capacity

During testing we found the 150um can carry 2Kg load and the 300um and 500um can carry 1Kg load without any noticeable drop in performance. The Top-plates will continue to perform at high loads with some loss in performance like travel range, accuracy and repeatability. The maximum load we tested was 4Kg, at which point the top-plates lost 20% of their travels.

Caution: Mass above 5Kgs may damage the top-plate. Always try to have the mass evenly distributed on the top-plate.

Change log

Last Updated	Comments
8/16/2010	Rough Draft
4/25/2011	Updated the Performance section
4/26/2011	Performance section edited again
5/26/2011	Characteristic performance added
6/17/2011	Load and transient response performance data added.

OPERATION

Front Panel Controls

XY Joystick

The XY Joystick is spring loaded to return to a zero movement center when not in use. The speed at which the stage moves is linear function of the degree to which the joystick is pushed away from the center. The direction of deflection can be controlled by the settings of the DIP switches on the back panel of the box (see *Back Panel Controls*). Depressing the button on top of the joystick will toggle the speed range of the joystick. In the high-speed range, the stage will travel up to the maximum speed of the motors; in the low-speed range, the speed for maximum deflection is reduced to 5 – 10% of maximum speed. The speed settings for the joystick may be programmed and saved in firmware. See the JSSPD command.

Command Encoder Knob

The Command Encoder Knob is usually used to control the Z-axis stage. The relative speed of the knob can be set with the JSSPD command and saved in firmware. The command knob can be attached to any axis by using the JS command.

Zero Button

The Zero Button allows the user to set all three axes coordinates to zero. Upon pressing the button, the LCD will display the change. Pressing the button also cancels any and all serial-controlled movement commands. The zero button also acts as a HALT button to stop undesired motion. Pressing the zero button briefly will halt motion and zero the coordinates; pressing and holding down the zero button for more than 1 second will halt motion, but not alter the coordinate settings.

Home Button

The Home Button sends the stage back to the zero coordinates.

@ Button

The @ Button is programmed for special functions. On most controllers this button is used with the Multi-Point Save/Move feature (see *Special Functions* below).

Rocker Switch - Clutch Enable

The Clutch Enable Switch allows the user to disconnect the Z-Axis motor from the microscope's fine focus knob by setting the switch to "Disengaged" (no dot on switch). When switching from "Disengaged" to "Engaged" (white dot on switch), the current position of the Z-Axis is locked-in, canceling any previously given Move commands. When the drive is "Disengaged," the feedback encoder still provides position information for the Z-axis LCD display. In some system configurations, the rocker switch is not used, or used for other special purposes.

LCD Screen

The Liquid Crystal Display (LCD) screen shows the current position coordinates of the axes with status information displayed to the right. A dim back illumination allows users to clearly view the screen even in a darkened room without causing light pollution.
The LCD display has four display modes selected by DIP switches 1 & 2 on the back panel. The display modes have the following characteristics:

Mode 1 - SW 1 & 2 DOWN Normal Display with Controller Firmware Version Line Shown

```
X: 23.12345 mm : fB
Y: -3.12345 mm : UM
Z:  1.12345 mm : E
     MS2000 v8.8d
```

This display shows the stage position in millimeters with five digits of precision with the status indicators on the right side.

Mode 2 - SW 1 DOWN & SW 2 UP Normal Display with Status Line Shown

```
X: 23.12345 mm : s
Y: -3.12345 mm : U
Z:  1.12345 mm : E
HRR 001:003 00:23:05
```

The status line at the bottom of the display indicates the command set (H high or L low), the XY encoder mode (R rotary or L linear), and the Z encoder mode (R rotary or L linear). The next two numbers show the next position to move-to for ring buffer, and the number of positions stored, respectively, separated by a colon (:). Controllers with the Auto-Focus option display the focus value on this line as well. On the right side is a time clock. Some error codes are displayed in place of the clock for a few seconds after they occur.

Mode 3 - SW 1 & 2 UP Dual Display with Status Line Shown

```
X23.12342>23.12345s
Y-3.12340>-3.12345U
Z 1.12345> 1.12345E
HRR 002:003 COD: 22
```

In this mode, two sets of number are shown for each axis. The number on the left is the current position reported by the axis encoders. The number on the right is the target position that the controller is trying to achieve.

Mode 4 - SW 1 UP & 2 DOWN Dual Display with Firmware Version Line Shown

There are several status indicators that may appear on the right side of the axis line display (in display columns 19 or 20). The meaning of these indicators is listed in the table below.

Table 1: Status Indicators in Order of Priority per Column

Status Letter	Meaning	Column	Priority	Comments
D	Disabled / Disengaged	19	highest	Axis is disabled if run-away error condition is detected
L	Lower Limit Engaged	19	mid	
U	Upper Limit Engaged	19	mid	
s	Slow Joystick Control	19	lower	
f	Fast Joystick Control	19	lower	
E	Clutch Engaged	19	lowest	Microscope Z-drive only
P	Axis in PAUSE state	20	highest	Axis is BUSY while paused.
B	Axis BUSY	20	high	
M	Motor Active	20	mid	Servos are turned on.
W	Command Wheel	20	lowest	Used on controllers where the command wheel may be switched between axes with the '@' button.

The status indicators can help you understand how the controller is set up and working.

The f, s, & W indicators tell you which axes are being controlled by the manual control devices as well as the speed range for the devices. Momentarily depressing the joystick button will switch the speed indicator from 's' to 'f' or vise versa.

Should the stage be moved into either a hardware or software limit switch, the 'U' or 'L' indicators will appear. Further movement into the limit is prohibited.

When a commanded move is issued to an axis via a computer command, the 'B' indicator will appear until the axis reaches target to the accuracy specified by the PC error variable. Should the stage drift further from the current target by more than the E drift error variable, the motors will re-engage and the 'M' will appear as the right status indicator. The 'M' will disappear when the stage is again within the PC error variable of the target. When using manual input devices (joystick or knob), the 'M' will appear as the motors attempt to keep the XY stage and Z drive at the location specified by the input devices.

If excessive servo errors are encountered, the axis will be disabled and the 'D' will appear. This is a safety feature to limit motion under run-away conditions or in the event of a stage crash.

Back Panel

USB Port

To use the USB port, you need to install the necessary drivers onto your computer. The drivers are furnished on a CD shipped with the controller and are available for download from http://www.asiimaging.com/wk_usb_support.html.

To obtain and install the drivers and associated files from the CD, open ...*\CP2101 USB-Serial\MS2000X\Host Driver Installation Files\WK_USB.zip*.

Create a temporary folder on your computer and extract all 20 files from WK_USB.zip into that folder. Run PreInstaller.exe. This application will step you through two separate installations: one for the USB low-level driver, and the other for the virtual serial port driver.

With these drivers installed, you can test for correct operation by using your computer's Control Panel and its subfunctions System/Hardware/Device Manager/Ports (COM & LPT). Before you connect and power up your ASI controller, expand the Ports (COM & LPT) listing and note any COM*x* devices present. When you power up the connected controller, you will see a new COM*x+1* appear. For example, if you see COM1 before powering up the controller, then after powering it up you will see COM1 and COM2.

(If you do not see "Ports (COM & LPT)" when the drivers are installed and the controller connected and powered up, then the computer may not fully support USB and RS-232. Certain inexpensive laptops have been observed with this defect. The workaround solution for this problem is to use a Serial Port PCI Card. Alternately, your computer may work with a Keyspan USA19HS High Speed USB / Serial Adapter. This device, a cable with two connectors, plugs into your USB port, and the other end is a serial port connector. With either the Keyspan or the Serial Card, you connect the device to the controller via a serial null modem cable. A serial null modem cable is furnished with each ASI controller and widely available at computer stores. Note that if the words "NULL MODEM" are not stamped on the connectors of a serial cable, it is probably not a null modem cable.)

The USB drivers on your computer will create a virtual serial port whenever the computer is connected to a powered-up controller. This virtual serial port operates like an RS-232 port as described below.

RS-232 Ports

The two 9-pin RS-232 ports allow serial commands to talk to and through the MS-2000. The IN port attaches to the PC computer via a null modem RS-232 Serial Cable to allow serial commands to control and get information from the MS-2000. The null modem cable switches the RX and TX lines and terminates possible PC handshaking lines allowing for asynchronous communication without handshaking. The OUT port is controlled by a second UART on the microcontroller. As a default it is configured as a "pass though" so serial traffic sent to the controller from the PC is echoed directly on the OUT port. Special functions are supported that use this port for dedicated purposes (e.g. triggered encoder reporting).

Fuse

The MS-2000 uses a 1A, 250V, fast blow, 5x20mm standard fuse.

Power Input

The MS-2000 uses a 24V 1.25A universal input, switching DC power supply. The power supply is connected and disconnected from the circuits via the ON/OFF power switch.

X-Y Stage Connector

This DB-25 connector is used to connect the MS-2000 to the X/Y stage via a four-foot cable.

Z-Axis Connector

This DB-15 connector is used to connect the MS-2000 to the Z-Axis drive assembly via a four-foot cable.

Linear Encoder Connectors

X ,Y & Z linear encoder connectors are located on the back panel. Heidenhain encoders utilize labeled DB-15F connectors. If the encoders are cross connected, the affected axes will behave erratically.

BNC Connectors

Two BNC connectors are provided, labeled IN and OUT. The connectors are wired to the internal board connector SV1. The IN connector is usually wired to IN0, the buffered TTL input channel. On piezo Z-axis systems, the OUT connector is connected to the analog DAC output that is used for control of the piezo system. On non-piezo systems, the OUT connector is usually wired to OUT0, the buffered TTL output channel.

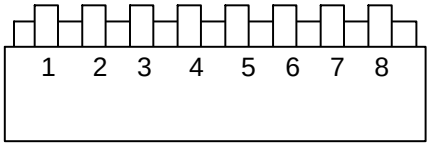
Reset Button

The reset button causes a hardware-level reboot of the microprocessor, which re-initializes the MS-2000 system.

Dip-Switches

The Dip-Switches allow the user to modify the configuration of the MS-2000's input and output devices. Switches 1-2 select the LCD screen options. Switches 4 and 5 set up the serial baud rate for the RS-232 and USB interfaces. Switches 3 & 6 select between linear and rotary encoders for the XY and Z axes, respectively. Switches 7-8 adjust the deflection of the joystick. The controller must be reset for most new DIP switch settings to take effect.

Dip Switch Settings



SWITCH	DOWN	UP	COMMENT
DIP SW 1	LCD Show Actual Position Only (normal)	Displays Position Actual > Target	
DIP SW 2	LCD 4th line: Firmware Version	LCD 4th line: config / status / clock	4th line format depends on specific firmware build
DIP SW 3	XY Linear Encoder	XY Rotary Encoder	Reset controller after changing switch
DIP SW 4	Baud Rate Selector – see chart below		Reset controller after changing switch
DIP SW 5			
DIP SW 6	Z Linear Encoder	Z Rotary Encoder	Reset controller after changing switch
DIP SW 7	Joystick Y deflection Reversed	Joystick Y deflection Normal	Reset controller after changing switch
DIP SW 8	Joystick X deflection Reversed	Joystick X deflection Normal	

Switch 4	Switch 5	Baud Rate
UP	UP	9600
UP	DOWN	19200
DOWN	UP	28800
DOWN	DOWN	115200

Special Functions and Features

Several special features have been incorporated into the stage control firmware beginning with version 6.0a. Several of these functions are standard on every controller, others are only supported with special hardware modifications or options; each are discussed in turn.

Configuration Flags

Beginning with firmware version 8.0, a set of configuration flags are read upon startup which determines the axis profiles for standard build firmware. These flags determine whether linear or rotary encoders are to be used, and the type of motor / lead screw combination used for the various axes. The configuration flags may be changed using the "CCA X" commands or by switching the encoder DIP switches. When a configuration flag is changed for an axis, new default parameter settings are used for that axis. On most controllers the "CCA X?" command will show the existing configuration and show the other configurations available in the firmware.

Build Configuration

Users often request special features for their systems. Often there are special firmware modules that are included to provide custom functionality. The BUILD X [BU X] command lists the firmware basic build flavor and all of the special firmware modules that are included in the controller. The following list describes some of these modules that may be present in your controller.

LL COMMANDS *	Low Level Command set is included
RING BUFFER *	Internal 50 position Ring Buffer is supported
SEARCH INDEX *	Supports ability to search for the index on linear encoders
TRACKING	Firmware module to support PhotoTrack system
AUTOFOCUS	Video autofocus scanning firmware module
SCAN MODULE	Supports 1-d and 2-d programmable scan patterns.
ARRAY MODULE	Firmware module for x,y moves in array pattern.
SPEED TRUTH	Query on SPEED command returns internal calculated speed used.
CRIFF	Firmware for the CRIFF focus system.
DAC OUT	WRDAC command sends specified voltage to SV1-Pin 5.
PREPULSE	Module to add predictive TTL output trigger pulse
PEDALS	Support for foot pedals to control Z-axis and Zoom systems
MULTIAXIS MOVES	Supports circular and spiral moves directly from the controller
CLOCKED POSITIONS	Supports motorized objective turrets, filter turrets & filter wheels
TTL_REPORT_INT	TTL IN0 used for interrupt-driven encoder position reporting
ENC_INT	Interrupt line is used for encoder pulse counting with the **SCAN MODULE**

IN0_INT * TTL IN0 used for a variety of interrupt driven functions selected using the TTL command.

* These modules are often included in standard builds.
If you see something you want but don't have, contact ASI.

Power Down Coordinate Save

Beginning with firmware version 8.1, powering down the MS-2000 controller will automatically cause the current positions to be saved to non-volatile memory so they can be restored upon startup. The shutdown procedure watches for power failure and immediately turns off the motor drivers before saving the position coordinates. Any power interruption will shut down operation. The user can always reset the stage coordinate origin using the ZERO button, however the actual position of the preset firmware limits remain unchanged with this operation. To reset the controller with default firmware limits and with zero stage coordinates, press the RESET button on the back of the controller. With the RESET operation, the current stage position will be lost.

Upon loss of power, the controller will send the character 'O' out the serial port. After successfully saving positions, the controller will send the character 'K'.

Save Settings to Non-Volatile Memory

The MS-2000 controller allows the user to customize various parameter settings and then save the settings to non-volatile memory to be used on subsequent power-ups. The controller is shipped with general purpose default setting suitable for most users. The user can always return to the default settings unsuitable parameters are saved. See the SAVESET command in the Programming section of the manual.

Post-Move Control Options

The behavior of the stage and controller at the completion of a move can be controlled with several programmable parameters. The best method can depend upon the particular application, the thermal and vibration environment, whether linear encoders are used, speed required, etc. The various options are set using the MAINTAIN [MA] command codes for each axis. The Finish Error and Drift Error tolerances are set with the PCROS [PC] and ERROR [E] commands respectively. The WAIT [WT] command can be used to enter a PAUSE state or control the motor drivers following a move. The table below shows how the various command options can be used.

MAINTAIN code [MA]	Description	Axis STATUS changes at end of move		Consequence of setting a WAIT time [WT]
		NOT BUSY	BUSY	
0	**Default** – HYSTERESIS Motor and servo turns off when position error is less than the *Finish Error*. Motor turns on again when error is more than the *Drift Error*. Drift-out and re-correct can occur 18 times per 0.5 sec. before fatal position error is set	ε< *Finish Error* [PC] LCD shows '**B**' until ε< *Finish Error*	ε> *Drift Error* [E] LCD shows '**M**'	PAUSE state is entered at end of move trajectory. BUSY state is not cleared until the WAIT expires.

	and motion is halted.			
1	UNLIMITED_TRIES: Like HYSTERESOUS above, but drift and re-correction can occur indefinitely without error.	ε< *Finish Error* [PC]	ε> *Drift Error* [E]	Same as above.
2	SERVOS_ON: Servo remains active and correcting errors until HALT command is received. Leaving the servos on indefinitely will cause the controller to run warm.	ε< *Finish Error* [PC]	Never	Same as above
3	SERVOS_WAIT: Servo remains active and correcting errors for the time set by the WAIT command following the move completion.	ε< *Finish Error* [PC]	Never	Motor driver and servo remain active for the WAIT time following completion of the move – then turn off.

ε is the final position error

Leaving the servos on can improve the positioning, especially on some linear encoded stages, and especially when a second axis remains in motion after one axis has finished moving. Turning the drivers off and using error threshold hysteresis means that most of the time there is no power applied to the motors so they cannot move. This is also the most power efficient mode.

Multi-Point Save/Move

Often there is a need to mark several positions to later revisit. The MS-2000 controller has a ring-buffer with up to 50 locations that the user can load with position information and then visit sequentially. The current stage position can be saved to the buffer by depressing the button on top of the joystick and holding it for longer than one second. (A short tap of the button toggles the joystick speed) You can move to the next position of interest and again save the position in the buffer by holding down the joystick button. Continue this procedure to save all positions of interest.

Save locations can be revisited by pressing the @ button briefly. Each press of the @ button advances to the next position. When you reach the last position, the next press of the @ button will take you back to the first position.

Holding the HOME button down for longer than one second will clear all the stored positions in the ring buffer.

The ring buffer may be preloaded with values via the serial command LOAD. Serial commands can also be used to advance to the next position as well as to control which axes will be affected by the move commands. See LOAD, RBMODE, and TTL commands in the Programming section of this manual. Contact ASI for details.

Constant Velocity Moves – (with firmware Version 8.1+)

The MS-2000 controller now uses a full closed-loop trajectory-driven algorithm for all commanded moves. This means that the move velocity is controlled as part of the digital feedback loop. Many users need to have high precision slow speed control. The MS-2000 now

provides the smoothest control possible in a motorized stage. To achieve this unsurpassed control, we have had to impose some small restrictions in terms of the acceptable velocity values. The controller has a minimum controlled speed of one encoder count per 64 servo cycles. The table below shows the slowest controlled speed for various stage configurations:

	6.35 mm Pitch Lead-screw Stage	**1.59 mm Pitch Lead-screw Stage**	**10nm resolution Linear Encoder equipped Stage**
XY Stage: 0.5 ms servo loop	0.69 μm/sec	0.17 μm/sec	0.31 μm/sec
XYZ Stage: 0.75 ms servo loop	0.46 μm/sec	0.12 μm/sec	0.21 μm/sec

The controlled stage speed must be an integer-multiple of the minimum speed. For example, an XYZ stage with 6.35 mm pitch lead-screw could be programmed to move 0.46 μm/s, 0.92 μm/s, 1.38 μm/s, etc., but not values between the integer-multiple of the slowest speed. Be aware that at the very slowest speeds, the condition and cleanliness of the stage, and the calibration of the analog stage driver circuitry can have a dramatic effect on the smoothness of operation. Please request tech note *TN120 Slow Speed Considerations* for further information.

TTL Controlled I/O Functions

Buffered TTL input (IN0) and output (OUT0) are available on internal connector SV1 pins 1 & 2 respectively. These lines may be connected to the IN and OUT BNC connectors on the MS-2000 back panel. The TTL command allows the user to select which functions are active for the IN0 and OUT0 lines. Various functions supported by the TTL command include:

- Triggered moves or Z-stack acquisitions
- Triggered asynchronous serial stage position reporting
- Output pulses upon move completion
- Output gated during constant speed motion.

The TTL input functions require the **`IN0_INT`** firmware module. The output functions are available in all builds.

Automated 1-D or 2-D Scanning

Systems the with **`SCAN_MODULE`** firmware addition have some special commands that make raster scanning very easy and well controlled. With the SCAN, SCANR, and SCANV commands, you can define a raster area and the number of raster lines. The stage will scan each line at constant speed, followed by rapid retrace. Hardware line sync signals available on SV1 pin 7 for the X or Y axis, as selected by internal jumper JP1 (1&2 X-axis; 2&3 Y-axis). With the **`ENC_INT`** firmware module encoder transitions can be counted to provide a "pixel" clock for an external recording device.

Synchronous Encoder Reporting

The **TTL_REPORT_INT** firmware module allow for external TTL synchronized position reporting. The position reports are sent to the auxiliary serial port on the MS-2000-WK in a binary format so that rapid, low jitter position reporting is possible for real-time positioning tasks. Contact ASI for details.

Tracking Features

ASI's PhotoTrack system uses the **TRACKING** firmware module along with hardware available from ASI that allows the stage to latch on and track spots of fluorescent illumination from labeled organisms. A quadrant PMT is used to provide sensitive and rapid position feed back information from the illuminated target. Contact ASI for details.

Safety, Diagnostic and Alignment Features

Motor-driver current-limits prevent the motors from fully powering into hard stops. In addition, the MS-2000 controller is constantly monitoring the positions of the motors under its control. Situations that may result in run-away conditions (such as reversed polarity motors or encoders), or situations where the motor is not able to follow the desired move trajectory (e.g., when mechanical interference limits the motion), cause the controller to detect an error condition. The motors are immediately turned off and the offending axis is disabled. If this happens, a 'D' character is displayed as the status indicator on the LCD display. The user should correct the problem and then reset the controller to regain control of the disabled axis.

Controllers using firmware Version 6.0 and newer utilize a motor driver circuit where all analog circuit alignment is done either automatically in firmware, or via serial commands. The user need not open the case to adjust the drive-circuit feedback and zeroing levels.

The firmware also keeps track of any internal error conditions that may arise during operation and saves the last 255 error codes in a buffer that may be read out for diagnostic purposes (see the serial DUMP command). The controller also has a built-in "move buffer" that holds move dynamic information for up to 200 servo cycles. The user may utilize this buffer to attempt advanced tuning of the controller for special applications or extremely fast or slow moves. Please see the section *MS-2000 Optimal Alignment Procedures* for a full discussion of these issues.

Clocked Devices

The MS-2000 controller supports clocked rotational devices, such as motorized objective nosepiece turrets, filter cube turrets, and filter wheels. These devices move to discrete clocked positions. Manual control is usually via the **@** button to advance to the next position. The serial MOVE command is used to move the devices to a specific integer-value clocked position. The current position is reported using the WHERE command.

Default Settings, Saved Parameters, Configuration Flags, Limits and Positions

The controller keeps track of several sets of flags, parameters and other saved configuration variables. It is important to understand how changing some of these parameters affect the other sets. The parameter sets are discussed below, starting with the most permanent settings and continuing to the least permanent settings.

Motor Driver Alignment Parameters

The motor driver alignment parameters are set with the AA and AZ commands. These commands set nonvolatile digital potentiometers located in the driver circuitry. These settings are independent of the other parameters settings and are immediately saved on the digital pots themselves.

Configuration Flags

The configuration flags are set with the "CCA X=n" command, and by the position of DIP Switches #3 and #6 (the DIP switches change between linear and rotary mode for the XY stage and Z-axis drive respectively). The configuration flags allow the loaded firmware to work with a variety of specific hardware configurations. Specific lead screw pitch, linear encoder resolution, encoder type, and piezo Z-axis range are some examples of the configurations settings that can be changed. (See the CCA command for details). The configuration flags are usually only changed when the controller is first set up for a particular set of hardware, when new firmware is loaded using the ASI Updater, or when changing between linear and rotary encoder for the stage. When a configuration flag is changed, it is immediately saved in nonvolatile memory; the controller must be restarted for the new configuration to be implemented. *Whenever any configuration flag is changed, the controller restores any Saved Parameter settings back to the Factory Defaults settings.*

Saved Parameters

There are many operating parameters that can be changed in the controller. These include such things as error tolerances (E and PC commands), speed and acceleration times (S and AC commands), servo parameters settings (KP, KI, and KD commands), and many others. All of these operating parameters have Factory Default settings that have been determined to be appropriate for most typical situations. A user may find that a change to some parameter value will improve the performance of the system for their application. When parameter values are changed using a serial command, the new parameter immediately becomes active in the controller. Third party software vendors can change parameter settings "on the fly" using their software and the changes will remain active as long as the controller remains powered on and not RESET. Parameter changes can be made persistent using the "SS Z" command, which saves all parameter settings to nonvolatile flash memory. Users wishing to make a one-time permanent change to a parameter setting can use a terminal program to communicate with the controller, make the parameter change, and then make the change persistent with the "SS Z" command. The new parameters will be used on subsequent power down/up or controller RESET. The user can restore the Factory Default parameter settings any time using the serial command "SS X".

Saved Stage Positions, Limit and Home Positions

The controller watches the power line voltage so that it can detect when the controller is being turned off. There is sufficient stored charge in the controller's power supply to allow the controller to save the stage position and a few other variables as power is being shut down. The variables that are saved are the Stage Axis positions, the programmable Upper and Lower limit locations, and the Home location all expressed in the current coordinated reference system. When power is restored, the controller loads the saved information into its working memory and clears the data from the Saved Position nonvolatile memory locations to ready those storage locations for when power is again shut off. *If the controller is RESET (without turning off the power) current locations are NOT saved, and the controller will come up with axis positions at zero and default Limit and Home positions; the Saved Position information will be lost.* Successful shutdown is indicated by "**OK**" broadcast on the serial port upon power-off.

Internal I/O connector details

Special user requirements often require custom external wiring. The MS-2000-WK controller has an internal board connector with several I/O lines that are often wired to the external BNC connectors for user connections. There may be occasions where the functions required are not wired to external connectors. The table below shows the connector wiring and the firmware modules that are required to take advantage of the I/O functions. The BU X command will list which modules are present in the loaded firmware. On most controllers the IN BNC is connected to TTL IN0.

EXT I/O – SV1

PIN	DESCRIPTION	FUNCTION	FIRMWARE Modules
1	TTL IN0	INPUT –TTL input w/ processor interrupt *(Usually wired to the IN BNC)*	**IN0_INT** used for external TTL triggered tasks – see TTL command for specific functions.
			ENC_INT use to count encoder pulses (selected with JP2) in conjunction with **SCAN** firmware.
			TTL_REPORT_INT used for triggered position reporting.
2	TTL OUT0	TTL OUTPUT – *(Usually wire to the OUT BNC on systems without* **PIEZO***)*	All builds – see TTL command for specific functions.
3	GND	Ground for all I/O	
4	TTL IN0 - OUT	OUTPUT – IN0 buffered and inverted	Can be used as buffered encoder pulses OUTPUT with **ENC_INT.**
5	PZ-DAC OUT	ANALOG OUTPUT from 16 bit DAC *(Wire to OUT BNC on* **PIEZO** *systems)*	On systems with a **PIEZO** axis this is the control voltage.
			DAC_OUT with WRDAC command, provides external analog output.
6	TTL IN1	INPUT – Auxiliary TTL input	
7	SCAN SYNC	OUTPUT for **SCAN MODULE** SYNC pulse	**SCAN MODULE** - selects sync source from JP1 to clock the sync flip-flop.

Internal Jumper JP1 selects the encoder flag signal that is used for the SYNC flip-flop. JP1 1-2 selects the X-axis; JP1 2-3 selects the Y-axis.

Internal Jumper JP2 selects the encoder signals that are counted during scanning. JP2 1-2 selects the X-axis; JP2 2-3 selects the Y-axis.

Please contact ASI if you need assistance configuring the controller for special functions.

Electrical Characteristics

External Modular Power Supply

AC Input:	100-240 VAC, 50/60 Hz,	0.8 A Standard Supply 1.5 A High-Current Supply Option
DC Output:	+24 VDC,	1.25 A Standard Supply 2.5 A High-Current Supply Option
Fuse:	1 Amp 2 Amp	Standard Supply High-Current Supply

Indoor use only

WARNINGS

1. Ensure power switch is in the OFF position before plugging in the power cord.
2. Do not unplug or plug-in devices / cables when power is on.
3. Do not remove the cover; no user serviceable parts are inside.
4. For indoor use only.
5. Keep clear of moving equipment. ASI Stages have current limits on the motors to prevent excessive traveler force from doing permanent bodily harm, but pinches can still be painful.
6. Protection provided by the equipment may be impaired if the equipment is used in a manner not specified by ASI.
7. In the event of device failure, contact ASI: (541) 461-8181
(800) 706-2284
International: 011-541-461-8181

Back Panel Connector Pin-outs

X-Y Stage DB-25F Connector

PIN	SIGNAL	INFORMATION
1	X Mot -	X Motor -
2	X GND	X Encoder Ground
3	X Enc Ch A	X Encoder Channel A
4	Y Mot -	Y Motor -
5	Y GND	Y Encoder Ground
6	Y Enc Ch A	Y Encoder Channel A
7	N.C.	Not Connected
8	N.C.	Not Connected
9	N.C.	Not Connected
10	X Lim U	X Upper Limit
11	+5V	+5V (X-limits)
12	Y Lim U	Y Upper Limit
13	+5V	+5V (Y-limits)
14	X Mot +	X Motor +
15	+5V	+5V (X-encoder)
16	X Enc Ch B	X Encoder Channel B
17	Y Mot +	Y Motor +
18	+5V	+5V (Y-encoder)
19	Y Enc Ch B	Y Encoder Channel B
20	N.C.	Not Connected
21	N.C.	Not Connected
22	X Lim L	X Lower Limit
23	GND	Ground (X-limits)
24	Y Lim L	Y Lower Limit
25	GND	Ground (Y-limits)

Z-Axis Drive & Optional F-Axis DB-15M Connector

PIN	SIGNAL	INFORMATION
1	F Enc Ch B	F Encoder Channel B
2	Z Lim L	Z Lower Limit
3	F Lim L	F Lower Limit
4	F Mot -	F Motor -
5	Z Enc Ch B	Z Encoder Channel B
6	GND	Ground
7	CLTCH	Clutch (+24V)
8	Z Mot +	Z Motor +
9	F Enc Ch A	F Encoder Channel A
10	Z Lim U	Z Upper Limit
11	F Lim U	F Upper Limit
12	F Mot +	F Motor +
13	Z Enc Ch A	Z Encoder Channel A
14	+5V	+5V
15	Z Mot -	Z Motor -

RS-232 Serial In DB-9F Connector

PIN	SIGNAL	INFORMATION
2	R In	Receive
3	T Out	Transmit
5	GND	Signal Ground
1,4,6-9	N.C.	Not Connected

RS-232 Serial Out DB-9M Connector

PIN	SIGNAL	INFORMATION
2	T Out	Transmit
3	R In	Receive
5	GND	Signal Ground
1,4,6-9	N.C.	Not Connected

Circular Power Connector

PIN	SIGNAL	INFORMATION
1	+24V	+24V Power From Modular Supply.
2	GND C	Case Ground
3	GND S	Supply Ground

USB Connector

PIN	SIGNAL	INFORMATION
1	VBUS	USB VBUS
2	D+	Data +
3	D-	Data -
4	GND	Ground

XY Axis Linear Encoder (optional) DB-9M Connector

PIN	SIGNAL	INFORMATION
1	X Enc Ch A	X Encoder Channel A
2	X Enc Ch B	X Encoder Channel B
3	GND	Signal Ground
4	N.C.	Not Connected
5	+5V	+5V Power
6	N.C.	Not Connected
7	Y Enc Ch A	Y Encoder Channel A
8	Y Enc Ch B	Y Encoder Channel B

9	N.C.	Not Connected

Z Axis Linear Encoder (optional)
DB-15F Connector

PIN	SIGNAL	INFORMATION
1	Z Enc Ch A	Z Encoder Channel A
2	GND	Signal Ground
3	Z Enc Ch B	Z Encoder Channel B
4	+5V	+5V Power
5-15	N.C.	Not Connected

IN BNC (optional)

PIN	SIGNAL	INFORMATION
Center	TTL IN	V_{IH} > 3.2V V_{IL} < 1.3V
Outer	GND	Signal Ground

OUT BNC (optional - TTL)

PIN	SIGNAL	INFORMATION
Center	TTL OUT	V_{OH} > 4.4V V_{OL} < 0.1V I_O Max: ±50mA
Outer	GND	Signal Ground

OUT BNC (optional - Analog)

PIN	SIGNAL	INFORMATION
Center	Analog OUT	0-10 VDC I_O Max: ±3mA
Outer	GND	Signal Ground

PROGRAMMING

The following section describes the RS-232 serial command set that the MFC-2000 and MS-2000 controllers use when communicating with a host computer. **Please note that the commands shown here include the XY stage command set. The XY related commands DO NOT apply to the MFC-2000 unit.** If you don't need to know everything, just use the quick reference below to get started. Details of each command, including examples, follow.

Quick Reference – Main Operating Commands

<del> or <bs> - Abort current command and flush input buffer

Command	Shortcut	Description
CDATE	CD	Returns Date/Time current firmware was compiled
HALT	****	Halts all serial commands being executed
HERE	H	Writes a position to an axis position buffer
HOME	!	Tells stage to go to physical limit switches
INFO	I	Returns a screen full of information about the axis
MOTCTRL	MC	Enables/Disables motor control for axis
MOVE	M	Writes a position to an axis target buffer
MOVREL	R	Writes a relative position to target buffer
RDSBYTE	RB	Returns a Status Information byte for an axis
RDSTAT	RS	Same as RDSBYTE, in decimal ASCII format.
RESET	~	Resets the MFC-2000 and MS-2000 controller
SPEED	S	Sets the maximum velocity/speed of axis
SPIN	@	Causes axis to spin motor at given DAC rate
STATUS	**/**	Returns B-Busy, N-Not Busy
UNITS	UN	Toggles LCD units – mm or in – when DIP switch 2 is down
WHERE	W	Returns current position
ZERO	Z	Sets all axes to zero/set position to origin

Quick Reference – Customization Commands

These commands support setup parameters. In most cases, these commands would be used only once after the unit is powered up.

Command	Shortcut	Description
ACCEL	AC	Changes/Displays ramp time in milliseconds
BACKLASH	B	Changes axis backlash correction motion constant
BENABLE	BE	Enables/Disables buttons
ERROR	E	Changes/ Displays max position error allowable before the controller will start re-correcting position.
JOYSTICK	J	Enables/Disables/Assigns manual control input for an axis
JSSPD	JS	Sets/Displays % Max speed for joystick ranges
MAINTAIN	MA	Makes axis hold its position indefinitely.
PCROS	PC	Changes/Displays position error at which controller considers a move to be complete
SAVESET	SS Z	Saves current set-up parameters to FLASH memory.
SETLOW	SL	Sets/Displays lower firmware limit switch for an axis
SETUP	SH	Sets/Displays upper firmware limit switch for an axis

RS-232 Communication

The MFC-2000 and MS-2000 utilize an RS-232 serial link to connect with any computer with an RS-232 serial port in order to utilize all of the controller's abilities. The current setup for the serial link is: 9600 baud, no parity, eight data bits, one stop bit, and no flow control (9600: 8 : N : 1 : None). This serial control feature can be accessed through terminal programs such as Telix, ProComm, and HyperTerminal. The MFC-2000 and MS-2000 command set mimics Ludl's command set, so that software written with drivers for Ludl stages should be able to run an MFC-2000 focus controller and MS-2000 stage without modification[1]. The MFC-2000 and MS-2000 also have an abbreviated version of the commands that helps cut down on typing time and serial bus traffic.

Format

The MFC-2000 and MS-2000 control instruction set is implemented using the following format:

`COMMAND X=?????? Y=?????? Z=?????? <Carriage Return>`

The COMMAND is a string of ASCII characters such as **MOVE** or **HOME**, which must be followed by a space. All commands are case-insensitive.

Next are the axis parameters. (Bracketed "[ ]" parameters are optional.) The axis name is given, followed immediately by an equal sign and the axis parameter value. Each axis must be separated from the one before by one blank space. One or more axes may be specified on a single command line. An axis symbol typed without an "=" assignment is assumed to mean "=0", or the command format may not require a parameter value (e.g., **INFO X**). Commands will accept integer or decimal numbers. Internal truncation or rounding will occur if fractional decimals are of no meaning to the command.

Standard Axis Names: X and Y are stage controls, Z is focus control, F is a special axis and is used for zoom, rotary, or Piezo-Focus control when there is a standard Z/Focus control drive in use, potentially giving the user two separate focus controls. On four-axis systems, the fourth axis may be named F, T, or M depending on the application. Axis names are shown on the LCD.

Valid Examples:

(Typed commands are in **THIS TYPEFACE**; computer replies are in ***THIS TYPEFACE***.)

`MOVE X=1234`
`MOVE X=1234 Z=1234.5`
`MOVE X=1234 Y=1234 Z=1234`
`MOVE X Y Z` (This is evaluated as **`MOVE X=0 Y=0 Z=0`**)

All commands are completed with a Carriage Return (ASCII hex code: 0D). The MFC-2000 and MS-2000 controllers receive ASCII characters one at a time and place them into their memory buffer. With the exception of single hex code commands like the tilde (**~**), the controller will not process a command in the memory buffer until the Carriage Return (<CR>) has been received.[2]

---

[1] Unlike the Ludl command set, the MS-2000 and MFC-2000 controllers do not repeat the last command when a <CR> is received without a command. The MS-2000 and MFC-2000 do not use Modul or Point Id's. Valid axis labels are dependent on the controller. An axis parameter without an assignment (=) is assumed to be an assignment of zero, unlike the Ludl command set which returns the current setting.

[2] ASI's Control Character Bracketed Command Set, e.g., <Ctrl G><Ctrl H>, cause the memory command buffer to be emptied. This allows the daisy-chaining of other peripherals, such as ASI's SC-2 shutter controller, on the RS-232 line without causing unrecognized command errors to be reported back by the MS-2000 and MFC-2000.

Reply

Upon receiving a Carriage Return <CR>, the MFC-2000 and MS-2000 will process the command stored in its command buffer, clear the command buffer, and return a reply.

When a command is recognized, the MFC-2000 and MS-2000 send back a colon '***:***' (hex code: 3A) to show that it is processing the command. When processing of the command is complete, an answer is returned with any requested information, typically beginning with the letter ***A***. In some cases, the answer part of the reply is delayed until the completion of the command. The reply is terminated by a carriage return and a linefeed character (<CR><LF>). In the examples below, the <CR> and <CR> <LF> are implied.

Examples:

```
MOVE X
:A
WHERE X
:A 0
MOVE X=4 Y=3 Z=1.5
:A
WHERE X Y Z
:A 4 3 1.5
```

Error Codes

When a command is received that the MFC-2000 and MS-2000 cannot interpret, for one reason or another, an error message is returned in the following format:

:N<error code>

The error codes are as follows:

-1 Unknown Command
-2 Unrecognized Axis Parameter (valid axes are dependent on the controller)
-3 Missing parameters (command received requires an axis parameter such as x=1234)
-4 Parameter Out of Range
-5 Operation failed
-6 Undefined Error (command is incorrect, but for none of the above reasons)
-7..20 Reserved for filterwheel.
-21 Serial Command halted by the HALT command
-30-39 Reserved

WARNING: When using the MS-2000's RS232 OUT port to daisy-chain to another device, be aware that the MS-2000 will monitor all serial communications, responding with a ***:N-1*** error for foreign commands when a <CR> is received. This can be avoided by using "Control Command" or "Control Command Bracketed" command sets. Any Control Commands (see ASCII chars 0 - 26) will cause the MS-2000 to delete all characters received in its input buffer, thus avoiding error responses.

Query of Parameters

Most commands used to set parameter values can be queried for the current values using the question-mark syntax:

```
CMND X? Y? Z? F?
```

The controller will respond with **CMND**'s current settings, e.g.

:A X=0 Y=1 Z=10 F=2

This feature is most useful when using a terminal program to change controller parameters to verify that you have made the changes that you think you did, or to check present settings.

MFC-2000 and MS-2000 Command Set

Command: ACCEL

Shortcut: AC

Format: ACCEL [X=*time*] [Y= *time*] [Z= *time*]

Function: This command sets the amount of time in milliseconds that it takes an axis motor speed to go from the start velocity to the maximum velocity and then back down again at the end of the move. At a minimum, this acceleration / deceleration time must be greater than t_step (the amount of time it takes for the controller to go through one loop of its main execution code. Use the **INFO** command to determine the t_step).

Example:

```
AC X=50 Y=50 Z=50
:A
AC X? Y? Z?
:X=50 Y=50 Z=50 A
```

The command in this example will make the controller take 50 milliseconds to accelerate the motors on each axis during a move command. When the controller gets within 50 milliseconds of finishing the move, it will begin to decelerate the motors back down to the start velocity where the pulses take over to bring the axes within the pulse crossover position error.

Command: AALIGN

Shortcut: AA

Format: AALIGN X [Y] [Z]

AALIGN X=n [Y=n] [Z=n]

AALIGN X? Y? Z?

Function: Performs self-calibration of axis motor drive circuit. With just the axis name as the argument, automatic alignment is initiated. If a value n is specified, the value is written directly into the axis potentiometer. **WARNING** – The stage will move during the AALIGN command.

Example:

AA X? Y? Z?	Queries the current AA parameters.
:A X=83 Y=78 Z=59	
AA X=85	Sets the X axis potentiometer to 85.
:A	

Command: AFCONT (Requires Autofocus Hardware - See Autofocus Manual)

Command: AFLIM (Requires Autofocus Hardware - See Autofocus Manual)

Command: AFOCUS (Requires Autofocus Hardware - See Autofocus Manual)

Command: AFSET (Requires Autofocus Hardware - See Autofocus Manual)

Command: AFMOVE (Requires Autofocus Hardware - See Autofocus Manual)

Command: AHOME (**ARRAY** firmware module required, Version 8.7+)

Shortcut: AH

Format: AH [X=*x0*] [Y=*y0*]

Function: Used with the ARRAY command to set the coordinate location of the first array position, (1,1). Without arguments, the command set the current location to the (1,1) location. Otherwise, *x0* and *y0* are the coordinates expressed in millimeters.

Command: AIJ (**ARRAY** firmware module required, Version 8.7+)

Shortcut: IJ

Format: AIJ [X=*i*] [Y=*j*]

Function: Used with the ARRAY command to move to array location (*i*,*j*), where *i* and *j* are the indices of the desired array location. The AHOME location is position (1,1).

Command: ARRAY (**ARRAY** firmware module required, Version 8.7+)

Shortcut: AR

Format: AR [X=*Nx*] [Y=*Ny*] [Z=*Δx*] [F=*Δy*]

Function: The ARRAY command sets up a grid of points that can be traversed automatically with simple TTL control or with the RBMODE or AIJ commands. The size of the array is *Nx* by *Ny* points, with points spaced apart distance *Δx* and *Δy*. The location of the first point in the array is set with the AHOME command.

Without arguments, the **AR** command starts self-scanning of the array. When the stage arrives on target, it will delay for a period of time set by the command **RT Z=*time_delay*** before continuing on to the next position.

Command: AZERO

Shortcut: AZ

Format: AZERO [X] [Y] [Z]

Function: Automatically adjusts the zero balance of the motor drive card.

Command: BACKLASH

Shortcut: B

Format: BACKLASH [X= *distance*] [Y= *distance*] [Z= *distance*]

Function: This command sets (or displays) the amount of distance in millimeters to travel to absorb the backlash in the axis' gearing. This backlash value works with an anti-backlash routine built into the controller. The routine ensures that the controller always approaches the final target from the same direction. A value of zero (0) disables the anti-backlash algorithm for that axis.

Example:
```
B X=.05 Y=.05 Z=0
:A
```

The command in this example will make the controller move the X and Y axes to a location 50 microns away from the final target before moving to the final target, while the anti-backlash algorithm for the Z axis is disabled.

Command: BCUSTOM (Version 9.1+)

Shortcut: BCA

Format: BCA [X=@ *Normal Press*] [Y = @ *Long Press*] [Z= @ *Extra Long Press*] [F= **Home** *Long Press*] [T= **Home** *Extra Long Press*]

Function: Several MS-2000 modules have functions associated with @ and **Home** button presses on the controller. When two or more of these modules are installed in a system, they contest for the button functions. The BCUSTOM command lets the user reconfigure the button function in the field as per their convenience.

Example: The Autofocus system has the following button functionality

On *normal* @ button press Do *Autofocus*

On *long* ***Home*** button press Do *Auto Calibration*

And for RING BUFFER

On *normal* @ button press Do *move to next RING BUFFER position*

On *long* @ button press Do *load current position into RING BUFFER*

When these modules are shipped together the action that results from a *normal* @ button press is contested. Using the BCA command this can be resolved.

`BCA X? Y? Z? F? T?` when issued , will return

```
X=1 Y=1 Z=1 F=1
7: AT_RING_BUFFER
9: AT_AFOCUS
```

'1' indicates that the current configuration is set at the factory. And the number & and 9 are symbols for Ring Buffer and Auto Focus respectively. So if the factory picked RING BUFFER to have button functionality, then all the places where there is a contest RING BUFFER function has priority over AUTOFOCUS. So

On *normal* @ button press Do *move to next RING BUFFER position*

On *long* @ button press Do *load current position into RING BUFFER*

But on *long* ***Home*** button press Do *Auto Calibration*, as it wasn't contested by RING BUFFER

So if you wanted *normal* @ button press to do autofocus instead of RING BUFFER function, you would set it by issuing the

`BCA X=9` serial command

Now its

On *normal* @ button press Do *Autofocus*

On *long* @ button press Do *load current position into RING BUFFER*

On *long* ***Home*** button press Do *Auto Calibration*

If you set `BCA Y=9`, *long* @ button press will NOT do *load current position into RING BUFFER.* As this function was given to autofocus , but autofocus doesn't have a action for *long* @ button press, so this will cause the controller not to act on *long* @ button presses.

The settings of BCUSTOM are automatically saved in non-volatile memory when changed, they will be available even on controller restart.

List of Modules and their symbols

Symbol	Module
2	Smart Move
3	Toggle Switches
4	PhotoTrack, Focal Pt and Laser Track
5	Filter Wheel and other Clocked position actuators
6	Array Module
7	Ring Buffer
8	Scan Module
9	Auto Focus
10	CRISP and CRIFF
11	XYZ Knob
12	XYZF Knob
13	ADEPT
14	RAMM load
15	Zoom

Command: BENABLE

Shortcut: BE

Format: BENABLE [X=*Toggle*] [Z=*Enable_Byte*]

Function: Enables or disables button functions. *Toggle=0* disables all buttons and pulses. *Toggle=1* enables all buttons and pulses (default settings). Specific buttons can be enabled/disabled by explicitly setting the *Enable_Byte*. The bits are set to one (1) when enabled or zero (0) when disabled, and are defined as follows:

Bit 0:	"Zero" Button
Bit 1:	"Home" Button
Bit 2:	"@" Button
Bit 3:	Joystick Button
Bit 4:	Reserved
Bit 5:	Zero button zeros Z axis only (Version 6.1z and later)
Bit 6:	Reserved
Bit 7:	Reserved

Command: BUILD

Shortcut: BU

Format: BUILD [X]

Function: This command returns the firmware "Build" version. BU X shows various configuration options and build-modules that are present in the firmware.

Example: **BU X**

STD_XYZ	shows that the firmware build was for a Standard XYZ system
Motor Axes: X Y Z	shows axis names for motor axes
Axis Types: x x z	shows axis type for each of axis (see table below) (9.2c+)
CMDS: XYZFTR	shows argument names pseudo-axis commands
BootLdr V:0	shows version of boot-loader program
Hdwr REV.E	shows main-board hardware revision
LL COMMANDS	list of optional firmware modules present
RING BUFFER	…
SEARCH INDEX	…
IN0_INT	…
DAC OUT	…

```
#define XY_STAGE           'x'   // conventional XY Stage Axis
#define Z_FOCUS_MOTOR      'z'   // motorized Z focus Axis
#define PIEZO_FOCUS  'p'   // Piezo Z focus Axis
#define OBJECTIVE_TURRET 'o'     // Objective changer - clocked position device
#define FILTER_CHANGER   'f'     // Filter changer - discrete position device
#define THETA_STAGE  't'   // Rotational stage, continuous motion, clocked
#define LINEAR_STAGE       'l'   // Generic linear motorized stage
#define LINEAR_PIEZO       'a'   // Generic linear piezo stage
```

```
#define ZOOM        'm'   // Zoom magnification motor axis
```

Command: CDATE

Shortcut: CD

Format: CDATE

Function: This command returns the date and time the current firmware was compiled.

Example: **CD**

Dec 19 2008:16:19:59

This example shows that the firmware running was compiled on December 19th year 2008 at 4:19:59 PM.

Command: CNTS

Shortcut: C

Format: CNTS [X=*nx*] [Y=*ny*] [Z=*nz*] [F=*nz*]

Function: Changes axis' encoder counts per mm. For example, doubling this number would cause a given number of mm to be converted internally to twice as many encoder counts as before. A command to move the stage 2 mm would instead cause it to move 4 mm. MOST USERS DO NOT NEED THIS FUNCTION!

In version 7.4d and later, this parameter can be saved to non-volatile memory.

In version 7.4d and later, piezo movement is controlled by this parameter. For piezo only, the formula for calculating this parameter is as follows:

Cnts = $(6.5536 * 10^7)$ / *d*

where *d* is the total range of movement in microns. For example, if the range of movement is -100um to +100um, then *d* = 200, and Cnts = 327680.

***Note**: In version 7.4d and later, for a piezo device, always set CNTS first, then limits (SL and SU) afterward.*

Example **C X=13490.4**

:A

Changes the encoder constant on the X-axis to 13490.4 counts/mm. The default values for this parameter are restored upon reset and should not require user modification.

Command: CUSTOMA

Shortcut: CCA, CA

Format: **CCA X=*n*** version 8.0 + & LX-4000 only.

Function: Configuration flags are set according to the table below for builds with STNDRD_XY and/or STNDRD_Z axes profiles. Configuration flags are changed one at a time for each execution of the CCA command. The changes will not take effect until the controller is restarted. Issue the RESET command to activate the new configuration.

CCA X=	Description	Display	Comment
5	XY Leadscrew Coarse Pitch (6.35 mm - Standard)	**B**	Firmware default
6	XY Leadscrew Fine Pitch (1.59 mm)	**A**	
7	XY Leadscrew Super Coarse (12.7 mm)	**C**	
8	XY Leadscrew Ultra Fine (0.635 mm)	**U**	0.635 Leadscrew post 9.0e , 0.317mm pre 9.0e
15	XY GTS Motor/Fine Pitch (1.59 mm)	**a**	
16	XY GTS Motor/Coarse Pitch (6.35 mm)	**b**	
17	XY GTS Motor/Super Coarse (12.7 mm)	**c**	
18	XY Leadscrew Ultra Coarse (25.4 mm)	**D**	
28	XY SISKIYOU Motor/Leadscrew	**S**	
42	XY Maxon Direct-Drive (1.59 mm)	**x**	
43	XY Maxon Direct-Drive (3.18 mm)	**e**	
44	XY Maxon Direct-Drive (6.35 mm)	**X**	
21	XY Linear Encoder 10 nm resolution	**1**	Firmware default
22	XY Linear Encoder 20 nm resolution	**2**	
51	XY Linear Encoder 5nm resolution	**K**	Ver 9.0e+
52	XY Linear Encoder 2.5nm resolution	**L**	Ver 9.0e+
30	XY Limit Polarity – Normally Open	**o**	Firmware default
31	XY Limit Polarity – Normally Closed	**c**	
9	Z Scope Drive 100 µm/rev. (50 nm enc. resolution)	**N**	Firmware default
10	Z Scope Drive 200 µm/rev. (50 nm enc. resolution)	**Z**	
19	Z Scope Drive 100 µm/rev. (25 nm enc. resolution)	**H**	
11	Z Leadscrew Coarse Pitch	**B**	
12	Z Leadscrew Fine Pitch	**A**	
13	Z Leadscrew Super Coarse Pitch	**C**	

CCA X=	Description	Display	Comment
14	Z Leadscrew Ultra Fine Pitch	U	
29	Z SISKIYOU Motor/Leadscrew	S	
26	ZF Linear Encoder 10 nm resolution	1	Leadscrew devices only. LE resolution is 50nm on scope drives.
27	ZF Linear Encoder 20 nm resolution	2	
53	XY Linear Encoder 5nm resolution	K	Ver 9.0h+
54	XY Linear Encoder 2.5nm resolution	L	Ver 9.0h+
32	ZF Limit Polarity – Normally Open	o	Firmware default
33	ZF Limit Polarity – Normally Closed	c	
34	Piezo Range 50 µm	f	
23	Piezo Range 100 µm	1	
35	Piezo Range 150 µm	S	Firmware default
24	Piezo Range 200 µm	2	
36	Piezo Range 300 µm	3	
25	Piezo Range 350 µm	t	
37	Piezo Range 500 µm	5	
1	XY Linear Encoders Used	L	Use DIP SW 3 (See Note 1)
2	XY Rotary Encoders Used	R	Use DIP SW 3 (See Note 1)
3	Z Linear Encoders Used	L	Use DIP SW 6 (See Note 1)
4	Z Rotary Encoders Used	R	Use DIP SW 6 (See Note 1)
20	Reserved for LX-4000 LE Flag		
26	Reserved for Tracer Enable		
70	The joystick and knob are always enabled, and the device assignments cannot be changed. The JOYSTICK command has no effect.	J	Version 8.8i and all later 8.8x. Version 9.0f and later.
71	The joystick and knob can be disabled, and the device assignments can be changed. The JOYSTICK command works normally.	j	Firmware default. Version 8.8i and all later 8.8x. Version 9.0f and later.

Note 1: Applies to LX-4000 systems only. On MS-2000 and MS-4000 systems, use DIP Switch #3 for XY linear encoders and DIP Switch #6 for Z-axis linear encoders instead of this CCA setting.

Example: **CCA X=6** Sets to XY stage for 1.59mm pitch lead screws.

A:

Query: **CCA X?** Returns string representing current state of flags

A: XY:RA Z:RN Shows XY stage is rotary encoded, lead screw pitch A (1.59mm), and Z-drive is rotary encoded, 100µm/turn scope motor drive.

A: XY:RAj Z:RN Shows XY stage is rotary encoded, lead screw pitch A (1.59mm), JOYSTICK command works normally for all axes, and Z-drive is rotary encoded, 100µm/turn scope motor drive. Version 8.8i and all later 8.8x; version 9.0f and later.

XY:F or **Z:F** indicate that the XY or Z settings are Fixed by the firmware build and cannot be changed using the CCA command.

A listing of the valid CCA X configuration flags is displayed for firmware builds where sufficient space is available.

Example:

```
A: XY:RB Z:RN PF:2
A: XY:RBJ Z:RN PF:2  Version 8.8i and all later 8.8x; version 9.0f and later.

5  XY B PITCH 4/in
6  XY A PITCH 16/in
7  XY C PITCH 2/in
8  XY 0 PITCH 80/in
18 XY D PITCH 1/in
21 XY 1 XYLE 10nm
22 XY 2 XYLE 20nm

9  Z N SCOPE 100u/T
10 Z Z SCOPE 200u/T
11 Z B PITCH 4/in
12 Z A PITCH 16/in
13 Z C PITCH 2/in
14 Z U PITCH 80/in
19 Z H SCOPE 100u/T 25nm

23 P 1 100um RANGE
24 P 2 200um RANGE
25 P 3 350um RANGE
```

Format: **CCA Y=*n*** version 7.3a and later

Function: Sets number of move repetitions. Default value is zero. That is, a MOVE command causes the system to initiate one move to the given position. If $n > 0$, then the move will be initiated more than once as a means to achieve fine adjustment and a more stable landing. This parameter is saved in non-volatile memory by the **SS Z** command.

Example: **CCA Y=3** All moves will be initiated four times.

```
A:
```

Format: **CCA Z = *n*** version 7.4d and later.

Function: Sets system configuration flags according to following table.

CCA Z=	Description	Display	Comment
1	X axis movement direction is positive (default). [+]	**+**	
2	X axis movement direction is negative. [-]	**-**	
3	Y axis movement is positive (default)[+] (Note: In the MS-4000, the default direction value for the Y axis is -1)	**+**	
4	Y axis movement is negative. [-]	**-**	
5	Z axis movement is positive (default)[+]	**+**	
6	Z axis movement is negative. [-]	**-**	
7	F axis movement is positive. [+]	**+**	
8	F axis movement is negative. [-]	**-**	
9	Disengage clutch [D]	**D**	
10	Engage clutch[E]	**E**	
11	Enable LCD display[O]	**O**	
12	Disable LCD display[F]	**F**	
13	CLOCKED DEVICES take shortest path	**S**	
14	CLOCKED DEVICES do not take shortest path	**L**	
20	The joystick and knob are always enabled, and the device assignments cannot be changed. The JOYSTICK command has no effect.	**J**	Version 8.8i and all later 8.8x. Version 9.0f and later.
21	The joystick and knob can be disabled, and the device assignments can be changed. The JOYSTICK command works normally.	**j**	Firmware default. Version 8.8i and all later 8.8x. Version 9.0f and later.
15	Disable ADEPT piezo self test on startup	**N**	Version 9.2d and later
16	Enable ADEPT piezo self test on startup	**C**	Firmware default, Version 9.2d and later

Note: A few products have different axis names. When in doubt, call ASI.

Format: CCA Z?

Function: Transmits string of sign characters for all active axes, then clutch and display status characters.

Note 1: When the direction of an axis is negative, upper limit settings must be negative values, and lower limit settings must be positive values.

Format: **CCA F?** version info TBD

Function: Returns number of, names of, and types of supported axes. The axis type identifiers are defined below in Table 2.

Reply format:

2*n*+1 bytes, where *n* is the number of supported axes.

Byte 1: number of axes, defined as follows:

‘1’ (0x31): 1 supported axis

‘2’ (0x32): 2 supported axes

‘3’ (0x33): 3 supported axes

‘4’ (0x34): 4 supported axes

‘5’ (0x35): 5 supported axes

. . .

Byte 2: name of axis[0]

Byte 3: type identifier of axis[0]

. . .

Byte 2*n*: name of axis[*n*-1]

Byte 2n+1: type identifier of axis[*n*-1]

Examples:

A one-axis system:

CCA F?

A: 1Z1

A two-axis XY system:

CCA F?

A: 2X0Y0

A three-axis XYZ system.

CCA F?

A: 3X0Y0Z1

A four axis XYZF system with piezo F

CCA F?

A: 4X0Y0Z1F2

A three-axis XYZ system with piezo Z

CCA F?

A: 3X0Y0Z2

A four-axis XYZP system with piezo P

CCA F?

A: 4X0Y0Z1P2

Axis Type Identifiers	Description
0x00-0x2F	unused
'0' (0x30)	Motor-driven stage axis
'1' (0x31)	Motor-driven focusing axis
'2' (0x32)	Piezo-driven focusing axis
'3' (0x33)	Motor-driven focusing axis with autofocus
'4' (0x34)	Piezo-driven focusing axis with autofocus
'5' (0x35)	Filterwheel
'6' (0x36)	Filter turret (AZ100)
'7' (0x37)	Motorized nosepiece
'8' (0x38)	Micrometer actuator (TIRF)
0x39…0xFF	unused

Table 2 CCA F? Axis Type Identifiers

Command: CUSTOMB

Shortcut: CCB

Format: **CCB Z=*n***

Function: Planar correction functions.
CCB Z=1
CCB Z=2
CCB Z=3
Store current xyz location values x1,y1,z1…x3,y3,z3 respectively.

CCB Z=4 Calculate coefficients for planar correction function, and enable planar correction.
CCB Z=5 Disable planar correction.
CCB Z=6 Displays *actual corrected* current Z position as long integer.
Note: WHERE Z displays the *intended* position of Z based on the most recently sent MOVE, MOVEREL, or HERE command.
CCB Z=7 Re-initialize to zero all Planar variables including x1,y1,z1…x3,y3,z3 and planar correction function coefficients. Disable planar correction.

Example: For CCB Z=1 through CCB Z=6
CCB Z=*n*

```
:A
```

CCB Z=6

```
:A Z=12345
```

Command: DACK

Shortcut: D

Format: DACK [X=*nx*] [Y=*ny*] [Z=*nz*]

Function: Sets motor speed control ratio, in mm/sec, of movement per DAC count. A DAC count is a value change of one (1) in the 8-bit integer written to the motor speed control register. MOST USERS DO NOT NEED THIS FUNCTION!

Example:

```
D X=.055
:A
```

Incrementing/decrementing the motor speed control register by one DAC count increases/decreases X-axis stage speed by 0.055 mm/sec.

Command: DUMP

Shortcut: DU

Format: DUMP [X] [Y]

Function: Dump internal buffers to terminal screen. DU, without arguments, dumps the Trajectory Buffer. DU X clears Trajectory Bbuffer. DU Y dumps Error Buffer. See the *Error Codes for MS-2000 Diagnostics* section below.

The MS-2000 controller has several built-in diagnostic capabilities that are useful for troubleshooting difficulties and for tuning the servo motion parameters. It is often useful to see how well the servo motion is tracking the theoretical trajectory. The controller has a built-in buffer that can hold 200 move steps. For best results, restrict testing to a single axis at a time; otherwise information from multiple axes will be interleaved in the dump buffer. Any motion from any axis will write information into the dump buffer until it is full.

Examples: **`DU X`** [Clears the dump buffers]

Then make a short move, e.g.: **`M X=12345`** [Moves about 1.23 mm]

After the move is complete, you can dump the buffer to the screen:

`DU` [Dumps Trajectory Buffer]

`DU Y` [Dumps Error Buffer]

Command: ENSYNC (Version 8.5+)

Shortcut: ES

Format: ENSYNC [X= *position*] [Y= *position*]

Functions: This command lets the user set a position, in millimeters - absolute, which will toggle a TTL output when the stage crosses that position. When ENSYNC is issued, the TTL output is reset low. Whenever the stage crosses the ENSYNC position, the output will toggle low to high and if crossed again, from high to low. ENSYNC will only work with one axis at a time, either X or Y and depends on how JP1 is jumped. (JP1-1&2 = X axis, JP1 – 2&3 = Y axis) The TTL output is available on pin SV1-7. Contact ASI for additional details on these modifications. *Warning—units of the position info is millimeters rather than tenths of microns.*

Command: EPOLARITY

Shortcut: EP

Format: EP X=*value* Y=*value* Z=*value F=value*

EP X? Y? Z? F?

Function: Supported by version 8.0 and later. Values are -1 and 1. Adapts the firmware to the counting direction of the motor encoders. This setting is normally set by ASI and not changed.

Command: ERROR

Shortcut: E

Format: ERROR [X= *position*] [Y= *position*] [Z= *position*]

Function: This command sets the Drift Error setting. This setting controls the crossover position error (in millimeters) between the target and position at which the MFC-2000 and MS-2000 controller considers an axis to be too far out of position. When this limit is reached, the controller will re-attempt to move the axis back within the Finish Error (**PC**) limit. The current value for this setting can be viewed using the **INFO** command or by placing a **?** after the axis name. Entries of zero value, e.g., **ERROR X=0**<CR>, are ignored.

Examples: **E X=.0004**

:A

Input values equal to or less than zero are acknowledged by "***:A*** ", but ignored.

The command in this example would cause the controller to consider a difference between the target and the current position greater than 400nm to be too large. If this large of an error were detected, the controller would re-engage the move algorithm to place the position error back inside of the Finish Error (**PC**) limit.

Command: HALT

Shortcut \ (the backslash character)

Format: HALT

Function: This command will stop all active motors.

Reply: If there are no errors, a positive reply of "***:A***" will be returned. If the "**HALT**" command is given while a commanded move is *in motion*, the controller will reply with the ***:N-21*** error.

Example: **HALT**
:A

Command: HERE

Shortcut: H

Format: HERE axis=*position* [axis=*position*] [axis=*position*]

Function: Assign the specified number to the axis's current position buffer. The unit of measurement is in tenths of microns. This defines the current position to be a specific distance from the origin (0), i.e., the origin may change.

Reply: If there are no errors, the positive reply "***:A***" will be sent back from the controller.

Example: **`H X=1234 Y=4321 Z`**

:A

The X position will change to 123.4 microns from the origin, Y will change to 432.1 microns, and the Z will be zeroed. The LCD display immediately shows the change.

Command: HOME

Shortcut: ! (the exclamation point character)

Format: HOME *axis* [*axis*] [*axis*]

Function: Move specified axis motors toward their *HOME* position. The default location for the *HOME* position (1000 mm) is far past the positive limit of the stage travel. If a hardware or firmware limit switch is encountered, the motor will stop.

Reply: If there are no errors, an "***:A***" is returned.

Example: **`! X Y Z`**
:A

The X and Y-axis motors will start moving towards the *HOME* position. A **`HALT`** command can stop the motors.

Note: The stage will be positioned at the limit switches or at the previously defined *HOME* position at the completion of this command. See **`SETHOME`**.

Command: INFO

Shortcut: I

Format: I [X] [Y] [Z] [F]

Function: This command returns the current values of various variables and constants that control the way the specified axis performs, as well as its current status.

Example: **I X**

```
Axis Name ChX:          X          Limits Status: f
Input Device :       JS_X [J]      Axis Profile :STD_CP_ROT
Max Lim      :    110.000 [SU]     Min Lim      :  -110.000 [SL]
Ramp Time    :        100 [AC] ms  Ramp Length  :     25806 enc
Run Speed    :    5.74553 [S]mm/s  vmax_enc*16  :     12520
Servo Lp Time:          3       ms Enc Polarity :         1 [EP]
dv_enc       :        368          LL Axis ID   :        24
Drift Error  :   0.000400 [E]  mm  enc_drift_err:        18
Finish Error :   0.000024 [PC] mm  enc_finsh_err:         1
Backlash     :   0.040000 [B]  mm  enc_backlash :      1815
Overshoot    :   0.000000 [OS] mm  enc_overshoot:         0
Kp           :        200 [KP]     Ki           :        20 [KI]
Kv           :         15 [KV]     Kd           :         0 [KD]
Axis Enable  :          1 [MC]     Motor Enable :         0
CMD_stat     :    NO_MOVE          Move_stat    :      IDLE
Current pos  :     0.0000       mm enc position :         0
Target pos   :     0.0000       mm enc target   :         0
enc pos error:          0          EEsum        :         0
Lst Stle Time:          0       ms Av Settle Tim:         0 ms
Home position:    1000.00       mm Motor Signal :         0
mm/sec/DAC_ct:    0.06700 [D]     Enc Cnts/mm   :  45397.60 [C]
Wait Time    :          0 [WT]     Maintain code:         0 [MA]
```

The **INFO** dump shows **command shortcuts** inside the square brackets, which you can use to change parameters, where applicable.

Command: JOYSTICK (updated w/ V8.8b)

Shortcut: J

Format: JOYSTICK [X±] [Y±] [Z±] or JOYSTICK [X=*dev*] [Y=*dev*] [Z=*dev*]

Function: This command enables (+) or disables (–) the input from the default manual control device for the axis (joystick or knob). If you specify an input device number *dev*, the axis specified will be connected to that input device. The table below shows the valid device assignments:

1.	NONE	
2.	DEFAULT	
3.	Joystick – X deflection	(X-axis default)
4.	Joystick – Y deflection	(Y-axis default)
5.	Standard Control Knob	(Z-axis default)
6.	X-Wheel	(special hardware required)
7.	Y-Wheel	"
8.	ADC CH1 – For ADC_FOLLOW or ADC_LOCK operation.	
9.	Foot switch	
10.	JX and X-wheel combo	(special hardware required)
11.	JY and Y-wheel combo	"
12.	CRIFF knob	(used for CRIFF system)
22	Z-Wheel	(TG-1000 only)
23	F-Wheel	(TG-1000 only)

Reply: If there are no errors, the positive reply "***:A***" will be returned from the controller.

Example: **`J X+ Y+ Z-`**

`:A`

The above command enables the default X and Y joystick control and disables the Z control knob.

Example: **`J X? Y?`**

`A: X=2 Y=3`

Here the query shows that the X & Y axes use the X & Y joystick driver.

Versions 8.8e and later: To set a default value that can be saved in nonvolatile memory, add 100 to the argument. Exceptions are 0 (NONE) and 1 (DEFAULT).

Example: **`J X=105`**

`:A`

This makes X-Wheel the default X axis manual control device. This is a setting that can be saved with the SAVESET command.

Example session:

```
 1. J X?
 2. :A X=2
 3. J X=1
 4. :A
 5. J X?
 6. :A X=2
 7. J X=105
 8. :A
 9. J X?
10. :A X=2
11. J X=1
12. :A
13. J X?
14. :A X=5
15. SS Z
16. :A
17. RESET
18. :RESE?
19. :J X?
20. :A X=5
```

In this session, the default manual input device is changed to X-Wheel in line 7. Line 11 sets the manual input device to whatever the default value is, which is now X-Wheel (5). Line 15 saves the settings. After the reset (line 17), the manual input device is set on startup its new saved default value, X-Wheel.

Command: JSSPD

Shortcut: JS

Format: JSSPD [X=*high*] [Y=*low*] [Z=*knob_speed*] [F=*xy_knobs_spd*] [T=*xy_knobs_mul*]

Function: This command sets the relative motor speed for maximum deflection of the joystick to the values specified. Values between 0.1 and 100 (%) are acceptable. Pressing the Joystick button toggles between the *high* and *low* settings. *Knob_speed* is a signed value that sets the relative speed and direction of the encoder knob.

xy_knobs_spd and *xy_knobs_mul* are used to set the relative speed and range multiplier of the XY_KNOBS if installed on the system.

In version 9.0h *only*, the joystick or knob speed cannot be set to zero.

Reply: If there are no errors, the positive reply "**:A**" will be sent back from the controller. The query reply tells you which attributes you have set instead of the standard X Y response.

Query Example:

```
JS X? Y?
:A JS_FAST=100 JS_SLOW=5
```

Command: KADC (For **CRIFF** and **AF-DUAL** Systems)

Shortcut: KA

Format: KA Z=*n*

Function: Adjusts a gain parameter in the servo loop where *n* is a signed integer. Use to change the polarity and gain of the feedback. Default *n=1*, (use *n=5* for PZ-2000 CRIFF).

Reply: "**:A**" is returned upon receipt of the command.

Query: **KA Z?** returns the current value.

```
:A Z=1          (for example)
```

Command: KD

Shortcut: KD

Format: KD [X=*kd*] [Y=*kd*] [Z=*kd*]

Function: Sets the servo derivative error term constant, the integer value *kd*. Usually set to zero (0). Especially useful when inertia is a factor to improve settling time and stability. MOST USERS DO NOT NEED TO USE THIS FUNCTION!

Command: KI

Shortcut: KI

Format: KI [X=*ki*] [Y=*ki*] [Z=*ki*]

Function: Sets the servo integral error term constant, the integer value *ki*. Larger values of ki reduce the time for small errors to be corrected at the finish of a move, but decreases stability if set too large. MOST USERS DO NOT NEED TO USE THIS FUNCTION!

Command: KP

Shortcut: KP

Format: KP [X=*kp*] [Y=*kp*] [Z=*kp*]

Function: Sets the servo proportional error term constant, the integer value *kp*. Larger values of kp increase the stiffness of the response to loss of position, but decreases stability if set too large. MOST USERS DO NOT NEED TO USE THIS FUNCTION!

Command: LCD

Format: LCD "*string*"

Function: Displays the quoted *string* on the bottom line of the LCD in place of the version information (DIP SW #2 DOWN).

Command: LED **(LED DIMMER** Firmware Required**)**

Format: LED [X= *0 to 99*]

LED X?

Function: Sets the brightness of ASIs LED illuminator by generating PWM thru TTL out. TTL out mode should be set to '9' (i.e. `TTL y=9`). Enable out from the LED illuminator should be connected to TTL out on controller. This setting can be saved in non-volatile memory using the **SAVESET** command.The PWM frequency prior to firmware ver 9.2c was 20KHz. The PWM frequency in and after firmware version 9.2c is 1.3Khz.

NOTE: If you are encountering flickering, try adjusting your shutter speed to integer multiples of the PWM frequency.

Command: LLADDR

Shortcut: LL

Format: LLADDR X=*xaddr* Y=*yaddr* Z=*zaddr*

LLADDR X? Y? Z?

Function: Sets the address of the axis used by the low-level command set. The default values are **X=24**, **Y=25**, and **Z=26**. Some systems require **X=1**, **Y=2**, and **Z=3**. This setting can be saved in non-volatile memory using the **SAVESET** command.

Command: LOAD (**RING BUFFER** Firmware Module Required**)**

Shortcut: LD

Format: LOAD [X=*xposition*] [Y=*yposition*] [Z=*zposition*]

Function: The **LOAD** function places a set of position coordinates in the next available internal ring-buffer memory location. The position values are expressed as floating point numbers representing tenths of a micron, the same as the MOVE command. For example, **LOAD Z=0.1** denotes a Z-axis movement of 1/100 of a micron, or 10 nanometers. Up to 50 position coordinates may be loaded into the buffer. The coordinates for the next move may be queried by using the command **LD X? Y? Z?**. Setting the current buffer position and initiating moves to locations stored in the buffer can be done using the **RBMODE** and **TTL** commands (see below), or by using a front panel button. The **LOAD** operation increments the *number-of-positions* counter which can be displayed on the LCD screen (controlled by DIP switch settings). To clear the buffer, type **RM X=0**. (See the **RBMODE** command.)

The current stage position may be loaded into the ring-buffer by pressing the Joystick button for 3 seconds and releasing.

Command: LOCK (For **CRIFF** and **AF-DUAL** Systems)

Shortcut: LK

Format: LK [X] [Y] [Z]

Function: Without argument, advances to the next system state until the **Cal_OK** state is reached. Once a good calibration is obtained, a subsequent **LK** command initiates the **Lock** state in which the servo loop error signal is supplied from the focus system. For CRIFF the lock is made at current location reference. (See **RELOCK** command.)

LK X returns the current system state code.

LK Y returns current focus error signal.

LK Z Unconditionally advances to the next system state.

Reply: "**:A**" is returned upon receipt of the command.

Command: LOCKRG (For **CRIFF** and **AF-DUAL** Systems)

Shortcut: LR

Format: LR Z=*lock_range*

Function: The Z parameter of the LOCKRG command allows the user to control the maximum excursion of the stage before the system generates an error condition and unlocks. The value *lock_range* is in millimeters. The default value is 0.050mm.

Reply: "***:A***" is returned upon receipt of the command.

Query: **LR Z?** returns the status of the lock and the lock range

:A 0.05

Command: LOCKSET (For **AF-DUAL** Systems)

Shortcut: LS

Format: LS Z=*focus_trim*

Function: The command directly sets the *focus_trim* value normally adjusted with the control knob after locking.

Reply: "***:A***" is returned upon receipt of the command.

Query: LS Z? returns the current reference value.

:A Z=-48 (for example)

Command: MAINTAIN

Shortcut: MA

Format: MAINTAIN [X=*code*] [Y=*code*] [Z=*code*] [F=code]

Query : MAINTAIN X? Y? Z? F?

Function: The maintain command specifies the behavior of the controller after move completion. Move commands complete when the stage moves to within the *finish error* tolerance of the target position (PCROS command). The actions for various *code* values are:

code = 0 [default] Post-move, when the controller detects drift from target specified by the *drift error* value, it will return the stage axis to the target several times (18) within a timeout period (~0.5 sec.) before declaring a move error code 60 and giving up further attempts.

code = 1 Post-move, the controller will indefinitely continue to try to reach target when drifts greater than the *drift error* are detected.
With *codes* 0 and 1, the motor drivers are turned off when the stage reaches the *finish error* tolerance.

code = 2 The motor drivers remain on and the servo loop remains active. (Version 8.5+)

code = 3 Drivers remain on and servos active for the post-move time set by the WAIT command. The system BUSY is released when the *finish error* tolerance is first achieved. Setting the WAIT time sufficiently long can stabilize post-move drifts during data recording, but then allow for less power consumption of the driver amplifiers when waiting between moves.

Reply: If there are no errors, the positive reply "***:A***" will be sent back from the controller.

Command: MOTCTRL

Shortcut: MC

Format: MOTOCTRL [X±] [Y±] [Z±]

Function: This command enables (+) or disables (-) the controller's ability to control the motor of a certain axis. The motor control voltage is set to zero and the position feedback control is not monitored when the motor is in disable (-) mode. The electronics of the controller will attempt to keep the motor from moving while disabled, however, it should be noted that this is an open-loop brake control only, and any movement or drift is not corrected.

Reply: If there are no errors, the positive reply "***:A***" will be sent back from the controller.

Example: **MC X+ Y+ Z-**

:A

This example shows that the X and Y motor control is enabled, but disables the Z motor control.

Command: MOVE

Shortcut: M

Format: MOVE axis= *position* [axis= *position*] [axis= *position*]

Function: Move one or more axis motors to an absolute *position*. The unit of measurement is in tenths of microns. If no *position* is specified, 0 (the origin) is assumed.

For devices with CLOCKED POSITIONS (turrets and filter wheels), the *position* is an integer value between one and the number-of-positions.

Reply: A positive reply of "***:A***" is sent back when the command is received correctly. Reception of the reply does not mean the end of execution, and the command **STATUS** can be used to determine if the move has been completed.

Examples: **M X=1234 Y=4321 Z**

:A

The controller will move the X-axis to position 123.4 microns from the origin using the maximum set speed (see **SPEED**). Simultaneously, it will move the Y-axis to position 432.1 microns, and the Z-axis to the zero (0) position.

During this movement, the Joystick and Encoder inputs will be locked-out and cannot alter the target positions entered. The motors will stop when they have reached their target or when their limit switch is encountered. To stop the motors during a serial **MOVE** command, use the **HALT** (\ ) command.

Command: MOVREL

Shortcut: R

Format: MOVREL axis=*distance* [axis=*distance*] [axis=*distance*]

Function: Move one or more axis motor a distance relative from its current position. This command is very similar to the **MOVE** command. The unit of measurement is also in tenths of microns.

Reply: A positive reply of "***:A***" is sent back when the command is received correctly. Reception of the reply does not mean the end of execution, and the command **STATUS** can be used to determine if the move has been completed.

Examples: **R X=1234 Y=-321 Z**

:A

The controller will move the X-axis an additional 123.4 microns in the positive direction at the maximum set speed (see **SPEED**). Simultaneously, the Y-axis will move 32.1 microns in the negative direction, while the Z-axis will not move at all.

During this movement, the Joystick and Encoder input will be locked-out and cannot alter the target positions entered. The motors will stop when they have reached their target, or if their limit switch is encountered. To stop the motors during a serial **MOVREL** command, use the **HALT** (****) command.

Command: PCROS

Shortcut: PC

Format: PCROS [X=*distance*] [Y= *distance*] [Z= *distance*]

PCROS [X?] [Y?] [Z?]

Function: This command sets/displays the Finish Error setting, which controls when the motor algorithm routines will turn off. The setting controls the crossover position error (in millimeters) between the target and position at which the MFC-2000 and MS-2000 controller will stop attempting to move the stage closer to achieving the position=target. This is value also determines the maximum error allowable before a move is considered complete. This value is usually set to the value of the smallest move step size according to the encoder resolution. The current value for this setting can be viewed using the **INFO** command.

Example: **PC X=.00005 Y=.00002 Z=.00005**

:A

Values equal to or less than zero are acknowledged by "***:A*** ", but ignored.

The command in this example will make the controller consider a **MOVE** command complete when the difference between the target and the current position is 50 nm for X, 20 nm for Y, and 50 nm for Z. **Warning:** If the **PCDOS** value is extremely small, moves may take an excessively long time to complete.

Command: MULTIMV (**MULTIAXIS_MOVES** firmware required v8.7+)

Shortcut: MM

Format: MM [X=*radius*] [Y= *speed*] [Z= *width*] [F=*mode*] [T?]

Function: The MULTIMV command allows several common multi-axis move patterns to be executed. Presently the patterns supported include **circles** and **spirals**. If users have other special requirements, they should contact ASI for assistance.

The command, without any arguments, initiates the multi-axis pattern move.

The patterns are initiated from the current stage position. The movement is parameterized in terms of the *speed* (feed rate) in mm/sec and pattern parameters. For **circles**, the *radius* in millimeters is the only required parameter. For **spirals** the *width* per spiral turn in millimeters is required as well as the maximum *radius*.

The *mode* is a bit-mapped character that determines the characteristics of the motion. The mode bits are used according to the following table.

Bit	Set	Clear
0	Lead-in Move Used	No Lead-in Move
1	Controlled acceleration along path, set by ACCEL command, to programmed *speed*.	No controlled acceleration
2	Move pattern repeated indefinitely	Only single cycle of move pattern executed
3	Reserved	
4	Reserved	
5	Reserved	
6	Motion pattern selector bits 6 & 7: 00 Reserved 01 Circle 10 Reserved 11 Spiral	
7		

Circles:

Lead-in move assumes start location is center of circle and moves out to X → X + *r* before the circular motion is started.

Spirals:

Spirals start at current location. Presently, no lead-in move is programmed. The spiral equation is $r = width \times \theta / 2\pi$. Motion continues to the maximum *radius.* If *mode* BIT2 is set, the motion then continues spiraling inward, and continues inward and outward until halted.

Query: **`MM T?`** returns the Multi-move state code and the current theta position in radians.

Command: PEDAL (Requires Foot Pedal Hardware/Firmware)

Shortcut: PD

Format: PEDAL X=*distance* Y=*rate* Z=*multiplier*

PEDAL [X?] [Y?] [Z?]

Function: This command sets/displays the dual-pedal footswitch controls for controllers with this feature. The command is set up as follows: X = Pedal Step Increment size, in millimeters. Y = Rate when pedal is held down, as an integer proportional to a speed in millimeters per second, Z = an integer multiplier used when the pedal controls a zoom axis.

Warning: User must ensure that the Rate given in this command is not greater than the maximum speed of the axis being controlled by the pedals. Entering an invalid value may result in unexpected errors and failures.

Reply: If there are no errors, a positive reply of "**:A**" followed by the startup sequence.

Examples:

```
PD X=0.02 Y=8 Z=5
:A
PD X? Y?
:A X=0.02000 Y=8.00000
```

Command: RBMODE (**RING BUFFER** Firmware Module Required)

Supported by firmware version 6.0e and higher.

Shortcut: RM

Format: RBMODE [X=*control*] [Y=*axis_byte*] [Z=*buffer_pointer*]

Function: Provides control of move and save operations involving the controller's internal 50-position ring-buffer. (Also, see the **LOAD** command.)

The command, without any arguments, performs the same operation that a TTL IN0 input pulse would control as determined by the current *IN0_mode*. See TTL command.

A move to the Next Position may be initiated by:

1 a TTL pulse when the appropriate *IN0_mode* is selected (see **TTL** command, **IN0_INT** Firmware Module Required).

2 a short press and release of the **@** button (as long as other special functions are not utilizing the **@** button).

3 by the **RM** command without arguments.

Setting the argument variables has the following effects:

control: 0 - Clears the RING_BUFFER. (**RING_BUFFER** firmware required.)

(0 – Starts ARRAY scan, firmware version 9.2c and earlier)

1 - Starts ARRAY scan. (**ARRAY_MODULE** firmware required.)

axis_byte: 1-7: Binary value determines which axes are commanded to move, or which axes positions are reported using *IN0_mode* =5. Bit 0: X-Axis; Bit 1: Y-axis; Bit 2: Z-axis. Default is *axis_byte*=3, XY enabled, Z disabled.

buffer_pointer: sets the pointer to the buffer position for the next move.

Command: RDADC

Shortcut: RA

Format: RA [X] [Y] [Z] [F]

Function: Returns the present values on the MS2000's 4-channel ADC. The X and Y channels are used for the joystick. The Z and F channels may be used for special applications, e.g. Autofocus or ADC_LOCK and ADC_FOLLOW modes of controlling the stage. Special firmware is required for these applications.

Example: **RA X Y**

:A 128 128

Shows typical ADC values for a centered joystick.

Reply: **:A Z=135** (for example)

Command: RDSBYTE

Shortcut: RB

Format: RDSBYTE *axis* [*axis*] [*axis*]

Function: Requests the MS-2000 to respond with the Status Byte. The number is one byte, which can be broken down into 8 bits that represent the following internal flags:

Bit 0: 0 = No commanded move is in progress. 1 = A commanded move is in progress. This bit is synonymous with the STATUS command. If the bit is set, then STATUS returns '**B**', otherwise STATUS returns '**N**'.

Bit 1: 0 = The axis is disabled. It can be renabled by one of the following: High Level command **`MC <axis>+`**, cycling the clutch switch for the Z-axis, Low Level StartMotor command (hex 47), or a system reset. This feature is available in versions 6.2c and later; 1 = The axis is enabled.

Bit 2: 0 = Motor is inactive (off), 1 = Motor is active (on).

Bit 3: 0 = Joystick/Knob disabled, 1 = Joystick/Knob enabled

Bit 4: 0 = Motor not ramping, 1 = Motor ramping

Bit 5: 0 = Ramping down, 1= Ramping up

Bit 6: Upper limit switch: 0 = open, 1 = closed

Bit 7: Lower limit switch: 0 = open, 1 = closed

Reply: **:** <*byte as hexadecimal*>

Examples: **`RB X`**

:<0x8A>

`RB X Y`

:<0x8A><0x02>

The X-axis example value of 0x8A means the following:

B7: 1 - X Axis is at its lower limit
B6: 0 - X Axis upper limit switch open
B5: 0 - Ramping down, if ramping
B4: 0 - Motor not ramping
B3: 1 - Joystick/Knob is enabled
B2: 0 - Motor power is off.
B1: 1 - X Axis is enabled
B0: 0 - No commanded move is in progress

Note: Motor power can be on while a commanded move is not in progress and the stage appears not to be moving. This happens when the motor is either making a final adjustment to a commanded move or when it is applying a force to maintain the stage position.

Command: RDSTAT

Shortcut: RS

Format: RDSTAT *axis* [*axis*] [*axis*]

Function: Same as **RDSBYTE**, except the data is returned in ASCII decimal format.

Examples: **RS X**

:A 138

Command: RELOCK (For **CRIFF** and **AF-DUAL** Systems)

Shortcut: RL

Format: RL

Function: Turns on the CRIFF laser and initiates a **LOCK** state using previously saved reference values. Same as LOCK for AF-DUAL systems.

Reply: "**:A**" is returned upon receipt of the command.

Command: RESET

Shortcut: ~

Format: RESET

Function: This command causes the controller to do a software reset. A software reset reinitializes all variables back to their pre-assigned values.

Reply: If there are no errors, a positive reply of "***:A***", followed by the startup sequence.

Example: **~**

:A

Command: RT

Shortcut: RT

Format: RT [X=*report_time*] [Y=*pulse_length in ms*] [Z=*delay_time*] [F=*num_aves*]

Function: The X argument Sets the time interval between report events when using *IN0_mode* = 5, TTL triggered serial interface asynchronous reporting. The *report_time* value has an acceptable range from 20 to 32700 milliseconds. The default value is 200ms.

The Y argument sets the length of the TTL output pulse in ms when using any *OUT0_mode* that triggers a TTL pulse.

The Z argument sets the post-move delay time for sequenced arrays.

The F argument sets *num_aves*, the power-of-two exponent for the number of samples to be averaged. Used with the CRIFF system.

Reply: "**:A**" is returned upon receipt of the command.

Command: RUNAWAY

Shortcut: RU

Format: RU X=*n*

Function: This command sets the servo loop error limit before the motors will be disabled. The value *n*, is the distance in millimeters that the internal servo target and the actual position can differ before the motor is disabled. Default is 1 to 2 mm. If spurious disable conditions are encountered, increase this number. For more sensitive crash protection, decrease this number.

Reply: A positive reply of "**:A**" is sent back when the command is received correctly.

Example: **RU X=5** Sets runaway sensitivity to 5mm on all axes.

:A

Command: SAVESET

Shortcut: SS

Format: SAVESET Z - saves settings to flash memory

SAVESET Y - restores previously saved settings after a SAVESET X

SAVESET X - will reload factory defaults upon next power-up

Function: SAVESET allows the user to save current parameters settings to Flash memory.

Reply: Upon the start of execution of this command, the controller will reply with a "**:**".

When the execution is complete, an "**A**" will follow the colon.

Note 1: During the time interval between the "**:**" and the "**A**", no serial or manual moves should be given.

Note 2: In Versions 6.1u and later (see VERSION command), limit settings (see SETLOW and SETUP) are saved if and only if the SAVEPOS command is issued *after* the command SAVESET Z.

Example: **SS Z** Saves current settings to flash memory.

:A

Command: SAVEPOS

Shortcut: SP

Format: SP [X=*inhibit*]

Function: Starting with Version 8.1 the axis positions and soft limit locations can be automatically saved when power is turned off. If this action is not desired, setting *inhibit*=1 will prevent power down saves. (Default is *inhibit* = 0) If the command is given without argument, a save position shutdown will be initiated whereby the axes will be halted, positions saved to flash, and the controller placed in a non-responsive condition until power is cycled.

Reply: Upon the start of execution of this command, the controller will reply with a "**:**". When the execution is complete, an "**A**" will follow the colon.

When a power down condition is detected, an "O" is transmitted. After the positions are successfully saved, a "K" is sent.

Note 1: During the time interval between the "**:**" and the "**A**", no serial or manual moves should be given.

Note 2: See Note 2 in the SAVESET section.

Command: SCAN **(SCAN** firmware required**)**

Shortcut: SN

Format: SCAN [X=*scan_axis*] [Y=*scan_axis*] [Z=*scan_axis*] [F=*pattern*]

Function: Sets which axes are to be used for 2-D raster scan. The fast-scanned raster axis (horizontal) is defined by *scan_axis* = 1; the slow-scanned axis (vertical) is defined by *scan_axis* = 2. Single axis scans (1-D) requires setting the unused axes *scan_axis* = 0, and the driven axis as *scan_axis* = 1.

The scan *pattern* may be set to *0 for RASTER* scans or *1 for SERPENTINE* scans.

Without arguments, the command SCAN initiates (or stops) a scan using parameters set with the **SCANR** and **SCANV** commands. (See below.)

Command: SCANR **(SCAN** firmware required**)**

Shortcut: NR

Format: SCANR [X=*start*] [Y=*stop*] [Z=*enc_divide*] [F= *#_pixels*]

Function: Sets up raster scan *start* and *stop* positions, with the position values expressed in millimeters. During scanning, the stage will move past both of these positions slightly, so that when scanning within the range specified, the scan proceeds with uniform speed (set by the SPEED command). On units equipped with hardware position Sync, the output pulse goes high as the stage crosses the *start* position. On systems with the **ENC_INT** firmware module, an output pulse will occur every *enc_divide* number of encoder counts. If the user specifies the *#_pixels*, the *stop* position will be calculated based upon the *enc_divide* and *start* position.

Command: SCANV **(SCAN** firmware required**)**

Shortcut: NV

Format: SCANV [X=*start*] [Y=*stop*] [Z=*number_of_lines*] [F=*overshoot*]

Function: Sets up the slow-scan (vertical) *start* and *stop* positions, with the position values expressed in millimeters. The stage will move to the start position before beginning the scan. The scan range will be divided into *number_of_lines* lines. Following a completed horizontal scan, the stage will move vertically to the next scan line. The processes will conclude when the stage has moved to the vertical *stop* position and completed the last horizontal scan. Single axis, 1-D scans will be repeated *number_of_lines* times. The *overshoot* parameter sets the amount of extra motion to account for the acceleration ramp at the start and stop of the trace. An *overshoot*=1.0 sets the pre and post move distances equal to the ramp up and down distances. Using a larger number will allow for more time to reach constant speed before the active sweep region.

Command: SECURE (special hardware and **U_SERVO_LK** firmware module needed)

Shortcut: SECURE

Format: SECURE [X= *p*]

Function: With stages equipped with Micro Servo lock mechanism, this command is used to lock or unlock samples on the stage. The value of *p* determines the position of the lever arm and can be any decimal number between 0.0 and 1.0. A value of 1.0 fully retracts the lever. The best value for a particular well plate model may vary and can be determined experimentally.

Example: **SECURE X=1.0** (fully opens lever)

SECURE X=0.25 (closes lever for typical well plate)

Reply: ***:A***

SECURE

:N-3 (Error at axis required)

SECURE Y=0

:N-2 (invalid axis)

SECURE X?

:N-2 (invalid operation)

Command: SETHOME (Version 8.0+)

Shortcut: HM

Format: HM X= *position* [Y= *position*] [Z= *position*]

Function: This command sets/displays a fixed hardware *HOME* location for an axis in units of millimeters. The *HOME* position is considered a fixed hardware location and is adjusted properly when the controller's coordinate system is altered with the **HERE** or **ZERO** function. The *HOME* position is automatically remembered and recalled through a power cycle and does not need to be saved using the **SAVESET** command.

Reply: If there are no errors, a reply of "***:A***" is returned.

Example: **HM X?**

:A X=1000.000

In the above example the default location for the *HOME* position for the X-axis is returned.

Command: SETLOW

Shortcut: SL

Format: SETLOW X= *position* [Y= *position*] [Z= *position*]

Function: This command sets/displays the lower firmware limit switch for an axis. The *Limit* positions are considered fixed hardware locations and are adjusted properly when the controller's coordinate system is altered with the **HERE** or **ZERO** function. The *Limit* positions are automatically remembered and recalled through a power cycle and do not need to be saved using the **SAVESET** command. ***Note:*** *If this value is equal to or greater than the value for* **SETUP**, *then the controller will operate incorrectly.*

Reply: If there are no errors, a positive reply of "***:A***" followed by the startup sequence.

For the Z axis only, input values equal to or greater than the current **SETUP** parameter value are acknowledged by "***:A***" but ignored.

Example: **SL X=-50 Y=-50 Z?**

:A Z=-110.000

In the above example, the lower limit for the X and Y axes have been set to 50 millimeters from the origin in the negative direction. Note that the **Z?** resulted in the controller returning the current position of the Z lower firmware limit switch.

Note 1: If this value is equal to or less than the value for **SETLOW**, *then the controller will operate incorrectly. See also Note 2.*

Note 2: When the direction of an axis is negative (see CCA Z=xxx), upper limit settings must be negative values, and lower limit settings must be positive values.

Command: SETUP

Shortcut: SU

Format: SETUP X= *position* [Y= *position*] [Z= *position*]

Function: Same as **SETLOW** command (see above) but for upper firmware limit switch.

Note 1: If this value is equal to or less than the value for **SETLOW**, *then the controller will operate incorrectly. See also Note 2.*

Note 2: When the direction of an axis is negative (see CCA Z=xxx), upper limit settings must be negative values, and lower limit settings must be positive values.

Command: SI

This command has two distinct functions depending on whether the system uses linear encoders (**SEARCH INDEX)** or rotary encoders (**SEEK LIMITS**).

(For linear encoders: **SEARCH INDEX** firmware required -- Linear Encoder Stages & Version 8.4+ Heidenhain XY Encoders only)

This functionality is available by request from ASI. It is not included with standard firmware.

Shortcut: SI

Format: SI [X=*center value*] [Y=*center value*] [Z=*center value*]

SI X? [Y?] [Z?]

Function: This Command searches for the physical centers of the stage and marks it with a user inputted value. Software limits are reset to default.

Reply: If there are no errors, a positive reply of "**:A**" is sent back.

Example: **SI X=0**

:A

In the example, the controller searches for the center of X-axis and sets it to zero.

SI Y=20000

:A

In the example, the controller searches for the center of Y-axis and sets it to 2mm.

SI Y=0

:N-5

N-5 indicates center of axes could not be found. This could be because previous center value is same as the new value, or hardware and software issues.

(For rotary encoders: **SEEK LIMITS** firmware required -- Rotary Encoder Stages. This is supported by Version 8.8e and above.)

Format: SI axis = direction [axis = direction] [axis = direction]

where direction ∈ {1, -1}

If direction is 1, then the stage seeks the upper limit. If direction is -1, then the stage seeks the lower limit.

Function: The stage moves to the hardware limit, backs away 3 mm, then approaches the limit slowly enough to maximize repeatability of the result. The recommended procedure is as follows, with SI and HERE commands using one or more axis arguments:

Send SI command.
Poll with STATUS command until 'N' is received.
Send HERE command with desired real world position.

Reply: If there are no errors, a positive reply of "**:A**" is sent back.

Example:
```
SI X=1 Y=-1
:A
```

Command: SPEED

Shortcut: S

Format: SPEED [X=*maximum_speed*] [Y=*maximum_speed*] [Z=*maximum_speed*]

SPEED X? [Y?] [Z?]

Function: Sets the maximum speed at which the stage will move. Speed is set in millimeters per second. Maximum speed is = 7.5 mm/s for standard 6.5 mm pitch leadscrews.

Reply: If there are no errors, a positive reply of "**:A**" is sent back.

Example:
```
S X=1.23 Y=3.21 Z=0.2
:A
```

In the example, the X-axis maximum speed is set to 1.23 mm/s, the Y-axis is set to 3.21 mm/s, and Z-axis is set to 0.2 mm/s.

Command: SPIN

Shortcut: @

Format: SPIN X=*rate* [Y= *rate*] [Z= *rate*]

Function: Tells controller to 'spin' the motor of specified axis at a rate expressed as its DAC value, a bit value from 0 to 128.

Reply: If there are no errors, a positive reply of "***:A***" is sent back.

Example:
```
@ X=100 Y=-100 Z
:A
```

This example shows a command that will instruct the X-axis turn at a motor rate of 100 DAC bits in one direction, the Y-axis at the same rate but in the other direction, and stop any rotation or motion of the Z-axis.

NOTE: To stop rotation, give a value of zero, or just the type the axis letter without an assignment as shown in the example above, or use the **HALT** (\ ) command.

NOTE: The **HALT** command will not return an ***:N-21*** when stopping a **SPIN** command.

Command: STATUS

Shortcut: /

Format: STATUS

Function: Inquires regarding the motor status of all axes. Queries the controller whether or not any of the motors are still busy moving following a serial command. Using the shortcut / is the preferred method for rapid polling of the controller for a busy state. The / is handled quickly in the command parser.

Reply: The positive reply can come in two forms:

N - there are no motors running from a serial command

B - there is a motor running from a serial command

Example:
```
MOVE  X=12345
:A
STATUS
B
/
N
```

In this example, the command **MOVE** started the X-axis moving towards the position 1.2345 millimeters from the origin. The first **STATUS** command returned a "***B***" showing that the motor is still busy moving towards the target. The second time, the **STATUS** command returned an "***N***" signifying that the **MOVE** command is finished and there is no longer any motor movement.

Command: STOPBITS

Shortcut: SB

Format: STOPBITS X=*n*

STOPBITS X?

Function: Sets the number of stop bits, *n*, to be used for RS232 serial communication. The default is one (1) stop bit; the other option is two (2) stop bits. Use the **SAVESET Z** command to retain the new stop bit setting after power off.

Command: TTL (version 8.5+)

Format: TTL [X=*IN0_mode*] [Y=*OUT0_mode*] [Z=*aux_IO_mode*] [F=*OUT0_polarity*]

Function: The MS2000 controller has a buffered TTL input (*IN0*) and output (*OUT0*) port as well as several unbuffered I/O ports. The signals *IN0* and *OUT0* are found on the board connector SV1 pin1 and 2 respectively. On many controllers these signals are connected to the IN and OUT BNC connectors on the back of the controller. The *IN0_mode* and *OUT0_mode* parameters set with this command determine the character of the I/O pins.

IN0_mode: **0** - turns off TTL IN0 controlled functions; TTL interrupt DISABLED.

1 – TTL IN0 initiates a Move-to-Next-Position of the stored positions in the Ring Buffer pointed to by the *buffer_pointer*. When the *buffer_pointer* reaches a value equal to the number of saved positions, it resets to the first position, allowing cyclic repetitions to the saved locations. See RBMODE and LOAD command.

2 - TTL IN0 repeats most recent relative move (See **MOVREL**) For example, begin a session by issuing the command **MOVREL X=0 Y=0 Z=0.5**, and each subsequent move to Next Position will cause the Z axis to move 0.05 micron. This function can be used for repetitive relative moves of any axis or combination of axes. You may directly set the *dZ* value with the **ZS** command's X parameter.

3 – TTL IN0 initiates an autofocus operation on systems with autofocus installed.

4 – enables TTL IN0 controlled Z-stacks. (See **ZS** command).

5 – enables TTL IN0-started position reporting via the serial interface. Information is asynchronously sent out the serial interface every *report_time* interval, where *report_time* is set with the RT command. Data returned in the serial stream are the elapsed time in milliseconds since the TTL trigger, followed by the position of each axis enable by the *axis_byte*. On TRACKING systems, the PMT sum signal is also reported. Reporting is toggled on and off by the TTL input pulse.

6 – TTL interrupt ENABLED; use with TTL triggered position reporting.

7 – TTL commanded ARRAY move to next position.

9 – Used with CRISP focus lock. TTL IN0 HIGH engages lock if system is in READY state. TTL IN0 LOW will cause system to UNLOCK is locked already.

OUT0_mode: **0** – TTL OUT0 unconditionally set LOW.

1 – TTL OUT0 unconditionally set HIGH.

2 – generates TTL pulse at end of a commanded move (**MOVE** or **MOVREL**). The pulse duration is set with command **RT Y=???.**

3 – output TTL OUT0 gated HIGH during axis index 0 (X) constant speed move.

4 – output TTL OUT0 gated HIGH during axis index 1 (Y) constant speed move.

5 – output TTL OUT0 gated HIGH during axis index 2 (Z) constant speed move.

8 – TTL OUT0 timed arrival pre-pulse output. See RT command. Requires **PREPULSE** firmware module

9 – TTL OUT0 PWM and MicroServo Output. See the LED or the SECURE command. Requires LED_DIMMER or USERVO firmware module

aux_IO_mode: Not Used Yet.

OUT0_polarity: **1** – default polarity, **-1** inverts polarity of TTL OUT0.

Command: UM (Units Multiplier)

Shortcut: UM

Format: UM [X=n] [Y=n] [Z=n]

Function: Specifies the multiplier for most serial commands such as MOVE and WHERE. Default values are 10000 (/mm), setting the default input scaling to 0.1μm/count. The sign of the Units Multiplier can be used to change the relative direction of motion for commanded moves. However, using the "CCA Z" command is the recommended procedure for changing the stage direction. The Units Multiplier can be saved with the "SS Z" command.

Reply: If there are no errors, a positive reply of "*:A*" is returned.

Command: UNITS

Shortcut: UN

Format: UNITS

Function: Toggles between millimeters and inches shown on the LCD display when DIP Switch 2 is down.

Reply: If there are no errors, a positive reply of "**:A**" is returned.

Command: UNLOCK (For **CRIFF** or **AF-DUAL** Systems)

Shortcut: UL

Format: UL

Function: This command unlocks the servo from the focus system and returns control to encoder feedback from the Z-axis drive. The CRIFF laser is turned off and the CRIFF system is placed in the **Laser_OFF** state. Current CRIFF lock reference values are saved for eventual use by the **RELOCK** command.

Reply: "**:A**" is returned upon receipt of the command.

Command: VB (Version 8.5+)

Shortcut: VB

Format: VB [X=*binary_code*] [Y=*TTL IN1 state* (read only)] [Z=*read_decimal_places*]

Function: Adds serial communication verbose modes for special functions. The *binary_code* is the sum of the bit values for the desired functions from the list below. The Y argument allows the TTL *IN1* input state to be directly queried via serial command. The number of decimal places for the WHERE command is set by *read_decimal_places.*

Bit 0	1	Send character '**N**' upon **completion of a commanded move**.
Bit 1	2	Send '**p**' for **joystick** quick-press and release, '**P**' for long-press.
Bit 2	4	Send 'H' for **TTL IN1** low-to-high transition; 'L' for high-to-low.
Bit 3	8	Changes the reply termination for <CR>+<LF> to just <CR>
Bit 4	16	Move and Move Rel will print the new Target Position.
Bit 5	32	Axes positions reported upon **completion of a commanded move**.

Example: VB X=7 turns on the first three of the above functions.

Command: VECTOR (Version 8.5+)

Shortcut: VE

Format: VE [X=*x_velocity*] [Y= *y_velocity*] [Z= *z_velocity*]

Function: The VECTOR command causes the stage to immediately ramp up to the velocity value specified by the command. The command arguments are expressed in units of mm/sec. The stage will continue indefinitely at the commanded velocity until the controller receives another command. A value of zero for the velocity component will halt motion on that axis. The controller will accelerate the stage to the commanded velocity at the rate specified by the ACCEL and SPEED commands until the commanded velocity is obtained.

Query: VE X? [Y?] [Z?]

Returns the current speed increment for the servo trajectory generator in units of mm/sec.

Reply: "**:A**" is returned upon receipt of the command.

Command: VERSION

Shortcut: V

Format: VERSION

Function: Requests controller to report which firmware version it is currently using.

Reply: If there are no errors, a positive reply of "**:A**" will be returned, followed by the version number.

Example:
```
V
:A Version: USB-8.6a
```

Command: WAIT

Shortcut: WT

Format: WAIT [X=*msecs*] [Y=*msecs*] [Z=*msecs*]

Function: Sets the length of time *msec*, in milliseconds, the controller will pause at the end of a move. The Busy status is not cleared during this Pause state. Additionally, a "**P**" is displayed on the LCD display when in the Pause state. During the Pause state, the servo loop remains actively attempting to position the axis on target.

Example:
```
WT X=20
:A
```

Sets the wait time for the X-axis to 20 ms.

Command: WHERE

Shortcut: W

Format: WHERE *axis* [*axis*] [*axis*]

Function: Returns the current position of the device for the axis specified.

Reply: If there are no errors, a positive reply of "**:A**" will be followed by the current position, in tenths of microns.

Example:
```
W X Y Z
:A 1234.5  432.1 0
```

In this example, X is 123.45 microns from the origin, Y is 43.21 microns from the origin, and Z is sitting on the origin.

Notes: No matter which order the X, Y, and Z's are specified in the **WHERE** command, the reply will always be in the order X, Y, Z.

The reporting precision of the **WHERE** command can be changed with the Setup Control Commands (below). Default includes a single fractional digit, which represents 10 nanometer precision. If fractional decimals cannot be handled by the user's software, use the appropriate Setup Control Command (below) so only integer data is returned (100 nanometer precision).

Command: WHO

Shortcut N

Format: WHO

Function: Inquires the controller to reply with its name. Allows computer software to automatically determine what stage instrument is attached at the end of the serial line.

Reply: If there are no errors, the MFC-2000 and MS-2000 will reply with a positive response of "***:A***", followed by its name.

Example: **N**
:A ASI-MS2000-XYBR-Zs-USB

Command: WRDAC (firmware 8.4f+)

Format: WRDAC X=n

Function: Lets the user set the voltage on header pin SV1-5 on WK2000 board. The voltage can be varied between 0 and 10 Volts, with an accuracy of 0.1V. Maximum Output drive current is 35mA. Input value in volts. Does not work with Piezo units.

Reply: If there are no errors, a positive response of "***:A***" will be returned.

Example **WRDAC X=1.1**

:A (Voltage on PIN SV1-5 is 1.1Volts)

WRDAC X=20 OR -1

:N-4 (Parameter out of range)

Command: ZERO

Shortcut: Z

Format: ZERO

Function: Writes a zero to the position buffer of all axes. Allows the user to set current position as the origin.

Reply: If there are no errors, a positive response of "**:A**" will be returned.

Example **Z**

:A

After the reply, the indicators on the LCD should all be zeros.

Command: Z2B (revised version 8.6d+)

Format: Z2B *current_axis_letter=new_axis_letter_ascii_code*

Function: Allows the user to change the axis name for a motor axis. The *current_axis_letter* must be one of the motor axes names listed with the "**BU X**" command. The *new_axis_letter_ascii_code* must be the decimal ASCII code for the desired axis name for letters between upper case 'A'(65) and 'Z'(90). For the change to take effect, the new setting must be saved to flash memory using "**SS Z**", followed by a hardware reset. The new axis name will remain in effect unless default settings are restored to the controller.

Reply: If there are no errors, a positive response of "**:A**" will be returned from the controller.

Example **Z2B Z=66** … change to "**B**" axis name.

:A

SS Z … required to save new name setting to flash.

:A

Command: ZF (requires ZFLOCK module)

Shortcut: ZF

Format: ZF [X=*0 or 1*] [Y=0 or 1]

Function: When enabled the controller slaves one of the axis to another, resulting in the slave axis mirroring all of the master axis's moves. Issuing ZF serial command enables or disabling the slaving. The designation of the master and slave can be swapped with the Y argument. Setting Y to 0, makes Z axis the master and F the slave. Setting Y to 1, makes F axis the slave and F the master.

Reply: If there are no errors, a positive reply of "**:A**" will be returned.

Example: **ZF X=1** Enables the slaving; slave axis will mirror all of the master axis's moves

:A

ZF X=0 Disables the slaving; slave axis will not mirror master axis's moves, and behave as an independent axis.

:A

ZF Y=0 Makes Z axis the master and F the slave.

:A

ZF Y=1 Makes F axis the slave and F the master.

:A

Command: ZS

Shortcut: ZS

Format: ZS [X=*dZ*] [Y=*n*] [Z=*mode*] [F= *stack_timeout*]

Function: Sets parameters for use with TTL triggered Z movement. User must set TTL X=4 for this trigger mode to be active. When a positive TTL edge is detected, the Z-axis is moved by an amount *dZ* (expressed in 10^{th} microns units). This move distance is repeated for *n* TTL triggered moves. If *mode*=1, the stage will step in the opposite direction for *n* moves, then turn around again, repeating a triangular waveform cycle. If *mode*=0 the stage will return to the original position after *n* moves and repeat a saw-tooth waveform cycle.

The stage will move to the starting position upon receiving the first TTL pulse after waiting more than *stack_timeout* milliseconds (default 500ms) from the previous pulse.

Reply: If there are no errors, a positive reply of "**:A**" will be returned.

Example: **ZS X=10 Y=20 Z=1** Setup to do twenty 1 micron slices with triangular pattern.

:A

SETUP CONTROL COMMANDS

Currently, the only way to toggle between the High-Level and the Low-Level command format is through the Setup Control Commands.

The following are special commands used to setup different properties of the MS-2000 and MFC-2000. The MS-2000 and the MFC-2000 recognizes these two-byte commands by their prefix byte 255. These commands mimic the Ludl Interface Control Commands and expand upon them.

Command	Description
255 65 **Alt[255] A**	Switch to High-Level Command Format (note: the post-byte "**A**" must be in upper-case)
255 66 **Alt[255] B**	Switch to Low-Level Command Format
255 82 **Alt[255] R**	Reset Controller
255 72 **Alt[255] H**	Return hundredth of a micron precision for High Level **WHERE** command.
255 84 **Alt[255] T**	Return tenth of a micron precision for High Level **WHERE** command.

Error Codes for MS-2000 Diagnostics

Error codes are dumped to the screen with the last error code shown first using the '**DU Y**' command. The table below lists the meanings of the error codes as of this publication.

Error Number*	Error Description
1-9	OVERTIME – **RECOVERABLE.** Error caused by competing tasks using the microprocessor.
20-22	AXIS DEAD – **FATAL.** No movement for 100 cycles; axis halted.
30-32	EMERGENCY STOP – **FATAL.** Getting further from the target; axis halted.
34	UPPER LIMIT – Upper Limit reached. (axis unspecific)
35	LOWER LIMIT – Lower Limit reached. (axis unspecific)
43	CRISP HALTED – Unexpected Halt Motion to locked CRISP axis.
45	ADC_LOCK_OOR – Out-of-range error on ADC input.
46	ADC_FOLLOW_ERR – Error attempting to follow an analog ADC input.
47	SERVO_LOCKED – Commanded motion attempted to servo-locked axis.
59	I2C NAK ERROR – followed by I2C chip address.
60-62	ADJUST-MOVE ERROR – Failed to clear '**M**' soon enough. **FATAL**
85	SCAN LOST PULSES – During a scan, missing pulses were detected.
86	SCAN INCOMPLETE – During a scan, terminated before completing the row.
87	TTL REPORT OVERRUN – TTL trigger too soon following previous triggered report.
90-92	ERROR_LARGE – **RECOVERABLE.** Error large. Motor set to FULL SPEED; hope to catch up.
100-102	INDEX NOT FOUND – Search for encoder index failed.
140	ADEPT HV LOW – Piezo drive card insufficient high voltage.
141	ADEPT I2C DEAD – Communication to ADEPT card failed.
142	PIEZO READ POS
143	PIEZO WRITE POS
144	PIEZO MOVE ERR
145	PIEZO READ POS1
146	PIEZO INIT
147	PIEZO POS ERROR
148	Autofocus 200um safety limit Encountered
149	I2C_BAD_BUSY ERROR
173	I2C_AXIS_ENABLE_ERR1
174	I2C_AXIS_ENABLE_ERR2
175	I2C_AXIS_MUTE1_ERR
203	I2C_NACK_ERROR
205	ERR_TTL_MISMATCH I2C bus error.
255	10 MINUTE CLOCK – Provides time reference for error dump list.
300	Autofocus Scan failed due to insufficient contrast
302	Clutch Disengaged, Engage clutch to do Autofocus

* Where multiple errors are listed, the last digit indicates the axis number that is in error. On three-axis units X=0, Y=1, and Z=2; on single-axis MFC units, Z=0.

FATAL errors cause the controller to halt motion on the axis that has the error. A commanded move will not be completed to the desired precision if a **FATAL** error occurs.

RECOVERABLE errors do not stop the controller from attempting to complete a commanded move. Large numbers of recoverable errors should be taken as a warning. Frequent servo errors (numbers 90-92) often

mean that the speed is near or exceeding the stage maximum. Frequent overtime errors (numbers 1-9) often mean that competing processes, such as over-frequent serial status requests, are using too much CPU time.

SETUP CONTROL COMMANDS

Currently, the only way to access the low level format is through the Setup Control Commands

The following are special commands used to setup different properties of the MS-2000 and MFC-2000. The MS-2000 and the MFC-2000 recognizes these two-byte commands by their prefix byte 255. These commands mimic the Ludl Interface Control Commands and expand upon them.

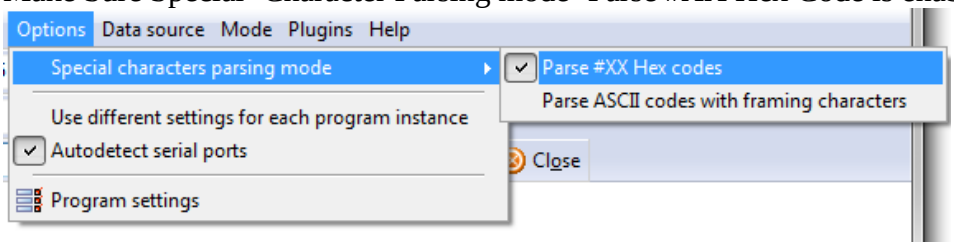
For HyperTerminal

Command	Description
255 65 Alt[255] 'A'	Switch to High Level Command Format (note: the post-byte "A" must be in caps)
255 66 Alt[255] 'B'	Switch to Low Level Command Format
255 82 Alt[255] 'R'	Reset Controller
255 72 Alt[255] 'H'	Return hundredth of a micron precision for High Level WHERE command.
255 84 Alt[255] 'T'	Return tenth of a micron precision for High Level WHERE command.

Note: Remote Switch Scanning and Transmission Delay is not supported

For Advanced Serial Port Monitor

Make Sure Special>Character Parsing mode>Parse #XX Hex Code is enabled



Command	Description
#FF#41	Switch to High Level Command Set
#FF#42	Switch to Low Level Command Set
#FF#52	Reset Controller

LOW LEVEL FORMAT

This serial RS-232 interface is used to hook up the MS-2000 and MFC-2000 to a PC with a protocol that imitates the Ludl Low Level command set. The purpose of the low level protocol is to provide a simple interface between a PC program and the MS-2000 and the MFC-2000, without ASCII conversion. The high level protocol is designed to allow direct human interface capability by displaying all numbers and commands in ASCII characters. The high level format is slow due to the extended transmission of ASCII characters as well as the time consumed converting back and forth from 3 byte memory stored numbers and multiple byte ASCII character numbers stored in strings. The low level format deals strictly with numbers that identify modules, commands, data_size, and data represented in 1 to 6 bytes in 2's compliment form.

NOTE: These commands apply to MS-2000 Controller firmware version 3.2 and forward.

The low level format is formed by the following 8 bit bytes:

BYTE1: Axis Identification
BYTE2: Command
BYTE3: Number of data bytes to be exchanged for this command
BYTES 4 thru 9: Data Bytes, mostly in 2's compliment form in the order of: Least Significant Byte, Middle Byte, Most Significant Byte
LAST BYTE: The ASCII colon character **(:)** flags the end of the serial command

All values specified through this section of the manual use the following format:

000000	*Decimal*
0x0000	*Hexadecimal*
Ctrl<A>	*ASCII character pressed with Ctrl held down*
Alt[0000]	*Decimal number typed with Alt held down*
'A'	*ASCII character typed in*

RS 232 Timeout: The normal Ludl 2-second timeout is not implemented. The MS-2000 clears its buffers whenever a colon (:) is received, thereby eliminating any error-prone characters received serially.

SERIAL DELAY: Due to the use of higher speed computers, there is no longer any need to delay serial communication replies; therefore, serial delays are not supported by the MS-2000.

WARNING: When using the RS-232 OUT port to daisy chain RS-232 devices, it must be taken into consideration that the MS-2000 monitors all serial traffic on the line. Although the Low level command set will not respond with an error to a foreign command like the high level command set, it is possible for the correct sequence of numbers to be entered which would match an actual command. This would result in the MS-2000 executing an unwanted command. ***It is recommended that the RS-232 OUT port is not used with the low level command set.***

Commands are generally broken into five groups. In three special cases the number-of-data-bytes group is omitted to speed up the communication process.

Data values are broken into 8-bit bytes for the data length times, and then each byte is sent out through serial channel to the interface, from LSB to MSB.

The ASCII colon (:) character is defined as the end-of-command code, and used to terminate the command loading sequence at which time the controller clears the serial buffer and attempts to process the command. If the command has errors and cannot be processed and executed, it is ignored.

Note: The MS-2000 does not support parity check.

Group 1 /Byte 1: Axis Identifier
This one byte character identifies which axis or control function the command is for.

X Axis:	Dec: 24 (Hex: 0x18)	Keyboard: Ctrl<X>
Y Axis:	Dec: 25 (Hex: 0x19)	Keyboard: Ctrl<Y>
Z Axis:	Dec: 26 (Hex: 0x1A)	Keyboard: Ctrl<Z>
F Axis:	Dec: 27 (Hex: 0x1B)	Keyboard: ESC

The following are reserved for future use:

Autofocus	*Dec:01(Hex: 0x01)*	*Keyboard: Ctrl<A>*
Controller	*Dec:03(Hex: 0x0)3*	*Keyboard: Ctrl<C>*
Scan	*Dec:03(Hex: 0x03)*	*Keyboard: Ctrl<S>*

Group 2 / Byte 2: Command Identifier
This is a single byte instruction code. These codes are listed in this manual. If the end-command ':' is received at this point, then the command is aborted and ignored.

Group 3 / Byte 3: Data Size
This is a single byte that gives the number of data bytes for this instruction. This value can also be found in command listing for different commands. Although the range of this variable is from 0 to 255, the MS-2000 only supports 0 to 6 up to firmware version 3.3.

Exceptions: There are 3 commands that do not use this data group: '?'-request status, 'G'-start motor / function, 'B'-stop motor / function.

Group 4 / Bytes 4-?: Data Bytes
This group holds the data for the command whether the command is sending or receiving information. The number of bytes for this group varies with each command and is stated in Group 3.
Numerical information is broken down into the 8 bit bytes. These are transmitted in the order of Least **S**ignificant **B**yte, **M**iddle **B**yte, then **M**ost **S**ignificant **B**yte. Positive numbers are divided

down using the base of 256. Numbers that may go negative are sent in 2's compliment. For more information, see the examples for individual commands.

Group 5: Last Byte
This is a one-byte end-of-command character '**:**'. The MS-2000 will not recognize a command until this character is received. When the ':' is received, the MS-2000 goes to a subroutine which then pulls Groups 1-4 out of the serial port buffer, and then searches the buffer until the ':' is found. Any information between Group 4 and the ':' is ignored.

COMMAND LISTING

The following are commands formatted by the MS-2000 shown in Decimal, and keyboard / ASCII form. The first command, Read Status, give examples that explain in depth the formatting which will be used for the rest of the examples.

Command: Read Status

Dec: 63 **Hex:** 0x3f **Keyboard:** ? **Data Size:** None

Description: The MS-2000 will respond to this command in the following manor. If the motor signal is not zero or there is a command being executed and the axis motor is enabled, the controller will return an upper case **B**. Otherwise, it will return a lower case **b**.

Example: Command: 24 63 58
Reply: 66

The above is an example of a stream of bytes that a PC would send serially to the controller and the controller's reply.
In the above example the 24 represents the X axis, the 63 represents the Read Status command and the 58 is the colon which signifies the end of the command. The reply 66 is the decimal code for the ASCII character B, which means the axis is currently busy.

Example: Ctrl<X>?:*b*

The above example shows a way to enter this command using a terminal screen where the Ctrl<X> means that the Ctrl key is held down while the key capital X is pressed. This enters the axis identifier for the X axis. The ? stands for the command Read Status and the : signifies the end of the command.
The b is the controller's response, which means the axis is not busy. Notice that with the low level command set there are no spaces, carriage returns or line feeds. Note that, for the sake of easy recognition of the computer response, all computer responses in this manual will be either labeled so, or be printed in *italics*.

Command: Read Motor Position

Dec: 97 **Hex:** 0x61 **Keyboard: a Data Size:** 3

Description: Requests the MS-2000 to respond with the current stage position in two's compliment form using 3 bytes. The response is in tenths of microns.

Example: Command: 24 97 03 58
Reply in Dec: 160 134 01

The above is an example of a stream of bytes that a PC would send serially to the controller and the controller's reply.
In the above example the 24 represents the X axis, the 97 represents the Read Motor Position command. The 3 means that the controller should return 3 bytes of data, and the 58 is the colon, which signifies the end of the command. In the reply are three bytes: lsb:160, the mb:134, and the msb:01.

Conversion: 160+(134*256)+(1*256*256)=100000 tenths of a micron or 10 millimeters from the origin.

The example below shows the same example above as it would appear on a computer serial port terminal program such as Hyperterminal (see command 63 for this manuals formatting information). As can be seen the numbers 160 134 01 correspond to non-legible ASCII characters. For this reason it is next to impossible to use a terminal program with the low level command set.

<X>a<C>: *áå_*

Note: As can be seen in the Read Motor Position Command, many low level commands are incompatible with terminal screens, so no terminal screen example will be given throughout the rest of the manual for those commands.

Command: Read Increment Value

Dec: 100 **Hex:** 0x64 **Keyboard:** d **Data Size:** 3

Description: Requests the MS-2000 to respond with current setting for the distance of increment moves. The number is a three byte two's compliment number representing a position offset in tenths of a micron.

Example: Command: 24 100 03 58
Reply in Dec: 160 134 01

The above is an example of a stream of bytes that a PC would send serially to the controller and the controller's reply.
In the above example the 24 represents the X axis, the 100 represents the Read Increment Value command. The 3 means that the controller should return 3 bytes of data, and the 58 is the colon, which signifies the end of the command. In the reply are three bytes: lsb:160, the mb:134, and the msb:01.

Conversion: 160+(134*256)+(1*256*256)=100000 tenths of a micron or 10 millimeters from the origin.

Command: Read Identification

Dec: 105 **Hex:** 0x69 **Keyboard: i Data Size:** 6

Description: Requests the MS-2000 to respond with the identification code for the Axis Id. The response for X, Y, and Z axis' is EMOT :. The fifth and sixth bytes are spaces (ASCII code 32).

Note: The MS-2000 does not support consecutive 105 commands to read the version information.

Example: Command: 24 105 58
Reply in Dec: 69 77 79 84 32 58
Reply Converted to ASCII: EMOT :

The above is an example of a stream of bytes that a PC would send serially to the controller and the controller's reply.
The example below shows the same example above as it would appear on a computer serial port terminal program such as Hyperterminal (see command 63 for this manuals formatting information).

<X>i:*EMOT :*
^--there is a space here

Command: Read Motor Position and Status

Dec: 108 **Hex:** 0x6C **Keyboard: l Data Size:** 4

Description: Requests the MS-2000 to respond with the current stage position in two's compliment form using 3 bytes followed by the status byte. The response is in tenths of microns. See command 126 (Read Status Byte) for more information on the status byte.

Example: Command: 24 108 03 58
Reply in Dec: 160 134 01 20

The above is an example of a stream of bytes that a PC would send serially to the controller and the controller's reply.
In the above example the 24 represents the X axis, the 97 represents the Read Motor Position command. The 3 means that the controller should return 3 bytes of data, and the 58 is the colon, which signifies the end of the command. In the reply are three bytes, the lsb:10, the mb:1, and the msb:2, plus the status byte. This can be translated as follows:

160+(134*256)+(1*256*256)=100000 tenths of a micron or 10 millimeters from the origin.

See command 126 (Read Status Byte) for more information on the status byte.

Command: Read Current Speed

Dec: 111 **Hex:** 0x6F **Keyboard:** o **Data Size:** 2

Description: Requests the MS-2000 to respond with current instantaneous value of the servo speed trajectory. The number returned is a signed two-byte number representing the velocity in μm/s.

Example: Command: 24 111 02 58
Reply in Dec: 78 02

The above is an example of a stream of bytes that a PC would send serially to the controller and the controller's reply.
In the above example the 24 represents the X axis, the 111 represents the Read Current Speed command. The 02 means that the controller should return 2 bytes of data, and the 58 is the colon, which signifies the end of the command. The reply is made up of an lsb and msb, which would convert as follows:

78+(2*256) =590 μm/s
or 0.59 mm/second

Command: Read Ramp Time

Dec: 113 **Hex:** 0x71 **Keyboard:** q **Data Size:** 1

Description: Requests the MS-2000 to respond with current setting for the time to ramp up and down. This is a one-byte number between 1 and 255. It represents the number of milliseconds the ramp from start speed to maximum speed at the beginning of a move and from maximum speed to start speed at the end of a move will take.

Example: Command: 24 113 01 58
Reply in Dec: 78

The above is an example of a stream of bytes that a PC would send serially to the controller and the controller's reply.
In the above example the 24 represents the X axis, the 113 represents the Read Ram Time command. The 1 means that the controller should return 1 byte of data, and the 58 is the colon, which signifies the end of the command. The reply 78 means that the controller will allow 78 milliseconds for ramping up and down.

Command: Read Start Speed
Dec: 114 **Hex:** 0x72 **Keyboard:** r **Data Size:** 2
Description: Dummy function. Do not use.

Command: Read Maximum Speed

Dec: 115 **Hex:** 0x73 **Keyboard:** s **Data Size:** 2

Description: Requests the MS-2000 to respond with current setting for the maximum speed the stage is allowed to move. The number returned is a straight two-byte number representing a speed μm/s.

Example: Command: 24 114 02 58
Reply in Dec: 78 02

The above is an example of a stream of bytes that a PC would send serially to the controller and the controller's reply.
In the above example the 24 represents the X axis, the 115 represents the Read Maximum Speed command. The 02 means that the controller should return 2 bytes of data, and the 58 is the colon, which signifies the end of the command. The reply is made up of an lsb and msb, which would convert as follows:

78+(2*256) =590 μm/s
or 0.59 mm/second

Command: Read Target Position

Dec: 116 **Hex:** 0x74 **Keyboard:** t **Data Size:** 3

Description: Requests the MS-2000 to respond with current target position. The number is a three byte, two's compliment, number representing a position offset in tenths of a micron.

Example: Command: 24 116 03 58
Reply in Dec: 160 134 01

The above is an example of a stream of bytes that a PC would send serially to the controller and the controller's reply.
In the above example the 24 represents the X axis, the 116 represents the Read Target Position command. The 3 means that the controller should return 3 bytes of data, and the 58 is the colon, which signifies the end of the command. In the reply are three bytes: lsb:160, the mb:134, and the msb:01.

Conversion: 160+(134*256)+(1*256*256)=100000 tenths of a micron or 10 millimeters from the origin.

Command: Read Status Byte

Dec: 126 **Hex:** 0x7E **Keyboard:** ~ **Data Size:** 1

Description: Requests the MS-2000 to respond with the Status Byte. The number is one byte, which can be broken down into 8 bits that represent the following internal flags:

Bit 0: 0 = No Motor Signal, 1 = Motor Signal (i.e., axis is moving)
Bit 1: Always 1, as servos cannot be turned off
Bit 2: 0 = Pulses Off, 1 = Pulses On
Bit 3: 0 = Joystick/Knob disabled, 1 = Joystick/Knob enabled
Bit 4: 0 = motor not ramping, 1 = motor ramping
Bit 5: 0 = ramping up, 1= ramping down
Bit 6: Upper limit switch: 0 = open, 1 = closed
Bit 7: Lower limit switch: 0 = open, 1 = closed

Example: Command: 24 126 58
Reply: 138

The above is an example of a stream of bytes that a PC would send serially to the controller and the controller's reply.
In the above example the 24 represents the X axis, the 126 represents the Read Status Byte command. The 58 is the colon, which signifies the end of the command. The reply can be broken into its individual bits as follows:

B7: 1 - Axis is at upper limit
B6: 0 - Lower limit switch open
B5: 0 - Ramping down if ramping
B4: 0 - Not ramping
B3: 1 - Joystick is enabled
B2: 0 - Pulses are not being used
B1: 1 - Servo Encoders are in use
B0: 0 - Motors are not turned on

Command: Start / Enable Motor

Dec: 71 **Hex:** 47 **Keyboard:** G **Data Size:** 0

Description: Enables the function. Mainly used to turn on / start / enable the motor for an axis specified. Does not give or receive data so the data field is omitted and the end character ':' follows directly.

Example: **Command:** 24 71 58
Response: There is no response

Command: Stop / Disable Motor

Dec: 66 **Hex:** 42 **Keyboard:** B **Data Size:** 0

Description: Disables the function. Mainly used to turn off / stop / disable the motor for an axis specified. Does not give or receive data so the data field is omitted and the end character ':' follows directly. Starting with firmware version 3.3, a disabled axis / function will reply to the Status command with a not busy 'b' even if the current position and target position do not match.

Example: **Command:** 24 71 58
Response: There is no response

Command: Write Motor Position

Dec: 65 **Hex:** 0x41 **Keyboard:** A **Data Size:** 3

Description: Requests the MS-2000 to write the given position to the current position count buffer. The position is given in two's compliment form using 3 bytes. The number represents the position in tenths of microns.

Example: Command: 24 65 03 160 134 01 58
Reply: There is no reply

The above is an example of a stream of bytes that a PC would send serially to the controller. The 24 represents the X axis, the 65 represents the Write Motor Position command. The 3 means that the controller should read three bytes of data. The three bytes are: lsb:160, the mb:134, and the msb:01. The 58 is the colon which signifies the end of the command.

Conversion: 160+(134*256)+(1*256*256)=100000 tenths of a micron or 10 millimeters from the origin.

Reverse Conversion:

10 millimeters
*10,000 to get tenths of microns
=100,000

lsb = remainder of 100,000 / 256 = 160
mb = remainder of 100,000 / 256 / 256 = 134
msb = remainder of 100,000 / 256 / 256 / 256 =1

Command: Write Target Position (move)

Dec: 84 **Hex:** 0x54 **Keyboard:** T **Data Size:** 3

Description: Requests the MS-2000 to write the given position to the target position buffer. The position is given in two's compliment form using 3 bytes. The number represents the position in tenths of microns.

Example: Command: 24 84 03 160 134 01 58
Reply: There is no reply

The above is an example of a stream of bytes that a PC would send serially to the controller. The 24 represents the X axis, the 84 represents the Write Target Position command. The 3 means that the controller should read three bytes of data. The three bytes are: lsb:160, the mb:134, and the msb:01. The 58 is the colon which signifies the end of the command.

Conversion: 160+(134*256)+(1*256*256)=100000 tenths of a micron or 10 millimeters from the origin.

Reverse Conversion:

10 millimeters
*10,000 to get tenths of microns
=100,000

lsb = remainder of 100,000 / 256 = 160
mb = remainder of 100,000 / 256 / 256 = 134
msb = remainder of 100,000 / 256 / 256 / 256 =1

Command: Increment Move Up

Dec: 43 **Hex:** 0x2B **Keyboard: +** **Data Size:** 0

Description: Requests the MS-2000 to add the Increment Value to the Current Position Value and place the result in the Target Position Buffer. There is no data or response.

Example: Command: 24 43 0 58
Reply in Dec: No reply

The above is an example of a stream of bytes that a PC would send serially to the controller. In the above example the 24 represents the X axis, the 43 represents the Increment Move Up command. The 0 means that there is no data. The 58 is the end of command character.

Command: Increment Move Down

Dec: 45 **Hex:** 0x2D **Keyboard:** - **Data Size:** 0

Description: Requests the MS-2000 to subtract the Increment Value to the Current Position Value and place the result in the Target Position Buffer. There is no data or response.

Example: Command: 24 45 0 58
Reply in Dec: No reply

The above is an example of a stream of bytes that a PC would send serially to the controller. In the above example the 24 represents the X axis, the 45 represents the Increment Move Down command. The 0 means that there is no data. The 58 is the end of command character.

Command: Write Increment Value

Dec: 68 **Hex:** 0x44 **Keyboard:** D **Data Size:** 3

Description: Requests the MS-2000 to write the given position to the Increment Value buffer. The position is given in two's compliment form using 3 bytes. The number represents the position in tenths of microns. The Increment Value is used for making successive Relative Moves.

Example: Command: 24 68 03 160 134 01 58
Reply: There is no reply

The above is an example of a stream of bytes that a PC would send serially to the controller. The 24 represents the X axis, the 68 represents the Write Increment Value command. The 3 means that the controller should read three bytes of data. The three bytes are: lsb:160, the mb:134, and the msb:01. The 58 is the colon which signifies the end of the command.

Conversion: 160+(134*256)+(1*256*256)=100000 tenths of a micron or 10 millimeters from the origin.

Reverse Conversion:

10 millimeters
*10,000 to get tenths of microns
=100,000
lsb = remainder of 100,000 / 256 = 160
mb = remainder of 100,000 / 256 / 256 = 134
msb = remainder of 100,000 / 256 / 256 / 256 =1

Command: Write Ramping Time

Dec: 81 **Hex:** 0x51 **Keyboard:** Q **Data Size:** 1

Description: Requests the MS-2000 to write the given byte to the Ramping Time buffer. Value Range is from 0 to 256 in the unit of milliseconds. The ramp time sets the stage acceleration at Max_Speed / Ramp_Time. For short moves the acceleration will be at the same rate as for long moves, but the duration of the ramp will be less than the full ramp time. *To minimize damage to the servo motors it is recommended that the ramp time always be greater than 50ms when ramping to full motor speed.*

Example: Command: 24 81 01 45 58
Reply: There is no reply

The above is an example of a stream of bytes that a PC would send serially to the controller. The 24 represents the X axis, the 81 represents the Write Ramping Time command. The 01 means that the controller should read one byte of data. The 45 is the byte of data which means the ramp time will be set to 45 milliseconds. The 58 is the colon which signifies the end of the command.

Command: Write Start Speed
Dec: 82 **Hex:** 0x 52 **Keyboard:** R **Data Size:** 2
Description: Dummy function – do not use.

Command: Write Top Speed

Dec: 83 **Hex:** 0x53 **Keyboard:** S **Data Size:** 2

Description: Requests the MS-2000 to write the given speed to the Top Speed buffer. The speed is divided down into two 8-bit bytes by dividing the number down by 256. The number represents the speed in μm/s.

Example: Command: 24 83 2 112 23 58
Reply: There is no reply

The above is an example of a stream of bytes that a PC would send serially to the controller. The 24 represents the X axis, the 83 represents the Write Top Speed command. The 2 means that the controller should read two bytes of data. The two bytes are: lsb:112 and the msb:23. This means the top speed the axis will travel is at 6000 μm/s. The 58 is the colon which signifies the end of the command.

Conversion: 112+(23*256)=6000 μm/s or 6 mm/s.

Reverse Conversion:

6 millimeters/second
*1000 to get microns
=6000 microns/second

lsb = remainder of 6000 / 256 = 112
msb = remainder of (6000 / 256)* / 256 = 23

*drop remainder of first division and take remainder of second division

Command: Write Vector Speed

Dec: 94 **Hex:** 0x5E **Keyboard:** ^ **Data Size:** 2

Description: Instructs the MS-2000 to immediately ramp motors to given velocity value and continue at that speed until instructed otherwise. The velocity is a two-byte value. The binary number represents the velocity in μm/s. The acceleration rate is set by the Write-Ramping-Time and Write-Top-Speed settings. (see Write-Ramping-Time command).

Example: Command: 24 94 2 112 23 58
Reply: There is no reply

The above is an example of a stream of bytes that a PC would send serially to the controller. The 24 represents the X axis, the 94 represents the Write Vector Speed command. The 2 means that the controller should read two bytes of data. The two bytes are: lsb:112 and the msb:23. The stage ramps to the speed 6000 μm/s. The 58 is the colon which signifies the end of the command.

Conversion: 112+(23*256)=6000 μm/s or 6 mm/s.

Command: Joystick / Control Device Enable

Dec: 74 **Hex:** 0x4A **Keyboard:** J **Data Size:** 0

Description: Enables the control device function. Allows enabling a control device such as a Joystick or Command Knob to be re-enabled.

Example: **Command:** 24 74 58 **OR** 24 74 0 58

The above is an example of a stream of bytes that a PC would send serially to the controller. The 24 represents the X axis, the 74 represents the Enable Joystick command. The data size is 0 and can either be included or left off on the MS-2000. The 58 is the colon which signifies the end of the command.

Command: Joystick / Controller Disable

Dec: 75 **Hex:** 0x 4B **Keyboard: K** **Data Size:** 0

Description: Disables the control device function. Allows disabling a control device such as a Joystick or Command Knob so that no external signals are allowed to affect move functions during PC control.

Example: **Command:** 24 75 58 **OR** 24 75 0 58

The above is an example of a stream of bytes that a PC would send serially to the controller. The 24 represents the X axis, the 74 represents the Disable Joystick command. The data size is 0 and can either be included or left off on the MS-2000. The 58 is the colon which signifies the end of the command.

ASI's Five Year Warranty on Automated DC Servomotor Stages

Applied Scientific Instrumentation, Inc., hereafter referred to as ASI, guarantees its automated XY stages & control electronics against all defects in materials and workmanship to the original purchaser for a period of five (5) years from the date of shipment. ASI's responsibility to this warranty shall not arise until the buyer returns the defective product, freight prepaid, to ASI's facility. After the product is returned, ASI at its option, will replace or repair free of charge any defective component or device that it has manufactured. The warranty set forth above does not extend to damaged equipment resulting from alteration, misuse, negligence, abuse, or as outlined below:

1.) Equipment not manufactured by ASI that is offered as part of complete system carries the original equipment manufacturer's warranty.

2.) The PZ-2150, PZ-2300, PZ-2500 units that are manufactured by ASI have a one year warranty.

3.) The DC servomotors used in our automated stages have a three-year warranty for biological applications in routine research.

4.) The linear encoder option has a two-year warranty.

5.) Damage from corrosive materials such as saline solution or other extreme contamination within the bearings and lead-screw assemblies voids the warranty

THE WARRANTY AND REMEDIES SET FORTH ABOVE ARE IN LIEU OF ALL OTHER WARRANTIES. APPLIED SCIENTIFIC INSTRUMENTATION, INC. EXPRESSLY DISCLAIMS ALL OTHER WARRANTIES WHETHER EXPRESSED, IMPLIED OR STATUTORY, INCLUDING THE WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE AND AGAINST INFRINGEMENT.

In no event will ASI be liable for incidental or consequential damages, even if ASI has been advised of the possibility of such damages howsoever, arising out of the sale or use of the products described herein.