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.

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: AFLIM (Requires Autofocus Hardware - See Autofocus Manual)

Command: AFOCUS (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* (expressed in millimeters). 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. It is possible to repeat the array using the **`RM F`** byte.

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 9.2g+)

Shortcut: BCA

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

Function: In version 9.2g+ the Button Function assignment has been rewritten to be more flexible. Every possible button function is now assigned a number. This function can be assigned to any button (@,**Home** and **Joystick Button**) and any press duration (*Normal*, *Long* and *Extra Long press*) thru the BCA commands X,Y,Z,F,T,R and M arguments.

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

Note: Behavior of this command is very different pre version 9.2g

Press Dur: When button is held down for an instant to 1 sec, it's a *Normal Press*

When button is held down for 1sec to 3sec, it's a *Long Press*

When button is held down for 3sec and more, it's an *Extra Long Press*

Table below lists and describes all possible button functions.

No	Function
0	No function performed
1	Smart Move related,
2	Toggles Knob between two axis, like Z and F
3	TRACKING,CRISP,LSRTRK related, short press functions. Steps from IDLE to Calibration states to lock and unlock state.
4	CLOCKED DEVICE related, moves clocked devices like Turret and slider to next position
5	ARRAY MODULE related, moves to next array position
6	RING BUFFER related, move to next ring buffer position
7	SCAN MODULE related, halts scan move
8	AUTOFOCUS related, performs autofocus routine.
9	CRIFF related, toggle CRIFF lock
10	AT_XYZ_KNOB related, changes xyz knob state. Cycles knob control between x,y,z axis.
11	AT_XYZF_KNOB related, changes xyzf knob state. Cycles knob control between x,y,z and f axis.
12	ZLOADER related, Moves Z to its lower limit and back.

13	CRISP related, performs CRISP very long press functions
14	ADEPT related, toggles between piezo external and internal input mode
15	CRISP related, performs CRISP long press functions. Stops current CRISP operation like stop dither, unlock etc.
16	ARRAY MODULE related, puts ARRAY Module in start state
17	CRIFF related, change CRIFF state.
18	RING BUFFER related, loads current stage position into ring buffer
19	AT RAMM LOAD related, moves Y and F axis to their upper and lower limits.
20	SCAN MODULE related, puts SCAN MODULE is start state
21	AUTOFOCUS related, performs autofocus calibration
22	ZOOM related, sets up zoom profile
23	PLANAR CORRECTION related, sets planar correction points
24	RING BUFFER related, clears ring buffer
25	TOGGLE ALL AXES related, swaps knob control between two axes.
26	TRACKING related, short press function
27	JS PULSE related, sets TTL in ready state
28	JS_FASTSLOW related, toggles joystick speed between fast and slow
29	PLANAR CORRECTION related, resets planar correction

Example: Lets take a firmware build like " ADEPT_XYZFP_5AXIS" . This firmware is packed with lot of modules packed. Like,

- ADEPT, ASI piezo
- Ring Buffer module
- JS_FASTSLOW, joystick button press toggles joystick speed between fast slow.
- AT_ZFPSWITCH, uses the @ button to toggle knob control between Z and F

In a built like this there are a lot of modules fighting for control of the buttons. There is a priority list that sets up the defaults button function . Lets Query the controller on what the default assignments ended up being.

```
BCA X? Y? Z? F? T? R? M?
X=2 Y=0 Z=14 F=0 T=0 R=28 M=18
```

```
X: @ Normal
Y: @ Long
Z: @ Ext Long
F: Home Long
T: Home Ext Long
R: Js btn Normal
M: Js btn Long
<LF>
```

So

@ *normal press*, swaps knob control between Z and F

@ *long press*, does nothing

@ *extra long press*, switches piezo between internal and external input mode.

Home long press, does nothing

Home extra long press, does nothing

Joystick button normal press, toggles joystick speed

Joystick button long press, loads current position into ring buffer.

If user wants the Ring Buffer to be more prominent, then following command can be issued.

```
BCA X=6 F=24 R=18 M=28
:A
<LF>
```

Now,

@ *normal press*, moves to next ring buffer position

Home long press, clear the ring buffer

Joystick button normal press, loads current stage position into ring buffer

Joystick button long press, toggles joystick speed

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	ZF Linear Encoder 5nm resolution	K	Ver 9.0h+
54	ZF Linear Encoder 2.5nm resolution	L	Ver 9.0h+
55	ZF Linear Encoder 50nm resolution	5	Ver 9.2f+
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.

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][F]

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]

`DU F` [Dumps Piezo History]

`DU F=999` [Resets Piezo History]

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:

0	NONE	
1	DEFAULT	
2	Joystick – X deflection	(X-axis default)
3	Joystick – Y deflection	(Y-axis default)
4	Standard Control Knob	(Z-axis default)
5	X-Wheel	(special hardware required)
6	Y-Wheel	"
7	ADC CH1 – For ADC_FOLLOW or ADC_LOCK operation.	
8	Foot switch	
9	JX and X-wheel combo	(special hardware required)
10	JY and Y-wheel combo	"
11	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*] [Y=*0 or 1*] [Z=*0 or 1*] [F=*0 or 1*]

LED X?

Function: X argument 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.

NOTE: If you encountering flickering in a live image, try adjusting your shutter speed to avoid aliasing with the LED PWM frequency.

Y, Z, and F arguments provide on/off control for additional LED lamp connectors on some controllers. Not PWM dimmable.

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. If a **+** is specified instead of a position, then the current position of the axis is stored in the ring buffer (as of firmware 9.2g). For example, the command **LOAD X+ Y+ Z=0.1** would store the current position of the X and Y axes and the Z position of 10nm to the ring buffer. The ring buffer contains 50 positions by default; contact ASI for the option of having 250 positions in the ring buffer (but this entails certain tradeoffs). 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) or accessed using **RM X?** (see the **RBMODE** command.). If positions for one or more axes are specified but others are not, the position of the unspecified axes during the ring buffer execution will not be well-defined. 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*] [F=*mode_byte*]

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.

RM X? returns the number of positions defined in the ring buffer (9.2g+).

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.

mode_byte: Lowest two bits are used to specify the mode:

0 – reserved.

1 – TTL triggered mode (default). A TTL pulse or RM command without arguments moves to the next position.

2 – One-shot autoplay mode. A TTL pulse or RM command without arguments plays the ring buffer from current position to end with delay between points set by RT Z (make sure delay is set appropriately; e.g. setting 1ms won't work with motorized stage).

3 – Repeat autoplay mode. Upon a TTL pulse or RM command without arguments, plays from current position continuously in a loop with delay set by RT Z (make sure delay is set appropriately; e.g. setting 1ms won't work with motorized stage). While running, another trigger causes autoplay to stop. Also enables repeat mode for ARRAY module.

Bits 2-6 are reserved.

MSB – read-only, set to 1 if ring buffer is autoplaying and 0 otherwise

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. When a qualifier is appended to the axis letter the behavior is different for version 9.2e and above. With **?**, a busy or not busy character is returned for that axis (***N*** or ***B***, just as in **STATUS**). With **-**, the left status character displayed on MS-2000 LCD of the axis is returned, including ***U*** or ***L*** for upper and lower limits, ***f*** or ***s*** for fast or slow joystick mode just selected, or a space for no event to report. With **+**, the right status character displayed on MS-2000 LCD is returned (with some additions), including ***M*** for move, ***B*** for commanded move (e.g. **HOME**), ***K*** for servo lock, ***S*** for spin move, ***P*** for pause, or a space for no event to report.

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 or ARRAY** 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 same axis settings also apply to the ARRAY module. 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

10 – output TTL OUT0 gated HIGH upon completion of video AUTOFOCUS function. AUTOFOCUS hardware and firmware required.

11 – generates TTL OUT0 pulse at end of commanded move providing CRISP is in ‘F’ state. Waits for CRISP ‘F’ state after move completion to send pulse.

12 -- TTL OUT0 high when CRISP is ‘F’ state, low otherwise.

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, unless the MAINTAIN behavior for the axis is set to code=3. 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.

For a piezo stage axis, the controller enters the *Pause* state as soon as the command is received and the voltage applied to the piezo. The controller remains BUSY until the *Pause* state times out. Typically used to allow for piezo stage settling time.

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. If the Z2B value of an axis is queried (e.g. **`Z2B Y?),`** the axis' index in the systemis returned (e.g. ***`:A Y=1`*** for the 2nd axis).

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.