

# Serial Command

## Tech Note

- [Error Codes for MS2000, RM2000 and TG-1000 Diagnostics](#)
- [Quick Start on Serial Commands](#)
- [RS-232 Communication](#)
- [TG-1000 and MS-2000's instruction set differences](#)
- [Wishbone/Low Level Commands for TG1000](#)

## Advanced Feature

- [ARRAY MODULE for ARRAY Scanning](#)
- [Multi-axis function](#)
- [PLANAR CORRECTION Firmware Module](#)
- [Rapid Array Scanning with the MS2000 Stage](#)
- [RING BUFFER Multi-Point Save/Move](#)
- [SCAN MODULE Sync and Scanning Addendum](#)
- [SEQUENCER MODULE](#)
- [SERVOLOCK\\_TTL](#)
- [Single-axis functions](#)
- [Synchronized Z-axis Focus Sweeps](#)
- [Synchronous Encoder Reporting](#)
- [Z-Stacks Using the Array Module](#)

## Serial Command Directory

### Command:AALIGN (AA)

MS2000 or RM2000 syntax

<b>Shortcut</b>	AA
<b>Format</b>	AALIGN [axis] = ### ...
<b>Units</b>	0-99, hardware potentiometer value


Tiger syntax

<b>Shortcut</b>	AA
<b>Format</b>	AALIGN [axis] = ### ...
<b>Units</b>	0-99, hardware potentiometer value
<b>Type</b>	Axis-Specific

Adjusts the drive strength by writing to a non-volatile on-board potentiometer. Normally done once at the factory (to a very conservative value) and never adjusted again. If the AA is off, the stage may be

sluggish (too low) or it may oscillate, buzz, or sound like it's grinding (too high). After changing the AA value the **AZERO** command should be run until zeroed.

To optimize stage performance a high AA is desirable, but too high and there are big problems. AA can be increased until oscillations occur and then decreased by 1 or 2 as described at the page on [tuning stages to minimize move time](#).



**WARNING** - The stage may move when the AALIGN command is sent.

**Example**

```
AA X? Y? Z?
:A X=83 Y=78 Z=59
```

Queries the current AA parameters.

```
AA X=85
:A
```

Sets the X axis potentiometer to 85. The **AZERO** command should now be run until zeroed.

2016/03/14 17:32  
[commands](#), [tiger](#), [ms2000](#)

**Command:ACCEL (AC)**

MS2000 or RM2000 syntax

<b>Shortcut</b>	AC
<b>Format</b>	ACCEL [axis] = [time in msec]...
<b>Units</b>	millisecond
<b>Remembered</b>	Using SS Z

Tiger syntax

<b>Shortcut</b>	AC
<b>Format</b>	ACCEL [axis] = [time in msec]...
<b>Units</b>	millisecond
<b>Type</b>	Axis-Specific
<b>Remembered</b>	Using [addr#]SS Z

This command sets the amount of time in milliseconds that it takes an axis motor speed to go from stopped to the maximum speed (**S command**) during a commanded move. It is also the duration of the deceleration / ramp-down time at the end of the move.

At a minimum, this acceleration / deceleration time must be greater than TSTEP (the amount of time it takes for the controller to go through one loop of its main execution code; the [INFO command](#) shows TSTEP as Servo Lp Time). However, setting the acceleration time to less than the motor's intrinsic time constant (~7 ms for the most common motors) is generally foolish. Overly-aggressive acceleration times lead to performance degradation over millions of moves. 25 ms acceleration time is generally safe for short moves with small stages or when the speed setting is a small fraction of the maximum, but larger values are recommended for larger stages and/or long moves (where the speed is reached) with speed settings near the maximum.

### 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.

2016/03/14 17:25

[commands](#), [tiger](#), [ms2000](#)

## Command:AFADJ

MS2000 and RM2000 Syntax

<b>Shortcut</b>	AFADJ
<b>Format</b>	AFADJ [X= zero pot value] [Y= video amplitude value] [Z= value]
<b>Units</b>	integer
<b>Remembered</b>	Using SS Z

Tiger Syntax

<b>Shortcut</b>	AFADJ
<b>Format</b>	[Addr#]AFADJ [X= zero pot value] [Y= video amplitude value] [Z= value]
<b>Units</b>	integer
<b>Type</b>	Card-Addressed
<b>Remembered</b>	Using [Addr#]SS Z

**X & Y** values range between 0 and 100. Y determines amplitude of the video signal entering the system. Setting a 0 value attenuates a video signal completely, while a setting of 100 lets the full signal go through. Attenuating a video signal also reduces the noise in the signal. If the focus value on the LCD reads 2047, then the system is saturated with too much signal, try reducing the **Y** value.

**X** is the zeroing potentiometer; the value of X should be set such that, when **Y** is 0, the *focus value* is also 0000 (or as close as possible).

**Z** sets the gain of the final Analog to Digital Converter (ADC) in the auto-focus system. Range: 0 to 3. By adjusting this setting, a *focus value* for a sparse sample can be magnified to get better focus. If an incorrect value of gain is used, the ADC saturates and the focus value reaches 2047. Upon system restart, the setting returns to its default value of 0. Perform an [SS Z command](#) to save the current gain setting in non-volatile memory.

AFADJ	GAIN
Z=0	1x
Z=1	2x
Z=2	4x
Z=3	8x

### Response

:A or Error Reply

### Example

```
AFADJ X=15 Y=95
:A<CR><LF>
```

```
AFADJ
:N-3
```

(Error indicates missing arguments)

```
AFADJ X=1000 Y=-12 Z=4
:N-4
```

(Error indicates arguments out of range)

```
AFADJ X=20 Y=90
:N-5
```

(Error indicates operation failed, try entering one argument at a time)

```
AFADJ X? Y?
:A X=20 Y=95
```

2016/03/23 19:43

[commands](#), [ms2000](#), [autofocus](#)

## Command:AFCALIB (AFC)

MS2000 and RM2000 Syntax

<b>Shortcut</b>	AFC
<b>Format</b>	AFCALIB [X= contrast] [Y= frame offset] [F= switch axis]

<b>Units</b>	integer
<b>Remembered</b>	Using SS Z

## Tiger Syntax

<b>Shortcut</b>	AFC
<b>Format</b>	[Addr#]AFCALIB [X= contrast] [Y= frame offset] [F= switch axis]
<b>Units</b>	integer
<b>Type</b>	Card-Addressed
<b>Remembered</b>	Using [Addr#]SS Z

The command without arguments will initiate series of auto-focus scans and selects various internal parameters for best focus value. Parameters changed by AFCALIB are the *Highlighted Area* (AFLIM), *Zero Adjust* (AFADJ), and *ADC Gain* (AFADJ). On some focus controllers, the AFCALIB routine can also be activated by holding down the HOME button for longer than 3 seconds. The AFCALIB's auto-calibration scans use the same speed and travel distance value that were set by the AFOCUS command. **NOTE:** Please use the HALT command or the \ to cancel an auto calibration - any other method may stop the move but may corrupt your settings.

**X=** Sets the minimum *contrast* value. During an auto-focus run, if the controller finds the difference between the maximum and minimum *focus value* to be less than the contrast value, it declares the run a failure and returns to the starting position. Default value is 10.

**Y=** *Frame Offset*, a floating-point constant that maps to a time interval. Changing this number alters the sharpness of focus. The default values are 3.5 for motor driven focus drives, and 3.75 for piezo driven focus drives. This setting compensates for time lags inherent to the video processing.

**F=** *Switch Axis*; if your focus controller can control two focus axes, e.g., a motorized drive and a piezo drive, then you may have the option to choose which axis to use for auto-focusing. Every axis that the focus controller controls is assigned a number starting from zero. Check with ASI to determine if this option is available for your system and to get the number for each axis.

Executing AFC alone will begin the Auto Calibration routine. Using AFC with arguments will only set or read back those parameters.

**Response**

:A or Error Reply

If the **X**, **Y** and **F** arguments are omitted, then an A is returned after the calibration is complete.

**Example**

```
AFC
:A<CR><LF>
```

```
AFC X=8 Y=3.75
:A
```

AFC X?  
:X=8 A

### AFC

Returns an :A after operation is complete, an :N-5 if the operation failed, or an :N-50 if the focus drive’s clutch is disengaged (if applicable).

2016/03/23 19:50

[commands](#), [ms2000](#), [autofocus](#)

## Command:AFINFO

MS2000 and RM2000 Syntax

<b>Format</b>	AFINFO
<b>Minimum Firmware version Required</b>	v8.7+

Tiger Syntax

<b>Format</b>	[Addr#]AFINFO
<b>Type</b>	Card-Addressed
<b>Minimum Firmware version Required</b>	v3.26+

This command returns all the values of variables and constants that control Autofocus. It also returns maximum focus value found during last autofocus run, original location and location after frame offset was applied.

### Example

```

AFINFO
Best Focus:587
Position Preoffset:  0.0002 mm Afteroffset:  -0.0003 mm
Speed : 1  [AF X]
Travel:0.025000 [AF Y]
Frame Offset:3.500000 [AFC Y]
Hill Offset:70 [AF F]
Contrast:10 [AFC X]
Window Size X:98 Y:98 [AL X Y]
Zero ADJ X:50 Y:90 [AFADJ X Y]
ADC Gain:0 [AFADJ Z]
<LF>

```

Related commands are shown inside square brackets, which you can use to change parameters, where applicable

Sometimes because of limited storage space a lite version of the command may be available. It will only displays Best Focus, Positions preoffset and afteroffset.

Best Focus:587

Position Preoffset: 0.0002 mm Afteroffset: -0.0003 mm

2016/03/23 20:25

[commands](#), [ms2000](#), [autofocus](#)

## Command:AFLIM (AL)

For CRISP

Tiger Syntax

<b>Shortcut</b>	AL
<b>Format</b>	[Addr#]AL [X=Log_amp_AGC] [Y=LED_intesity_pot] [Z=in_focus_mm]
<b>Type</b>	Card-Addressed
<b>Remembered</b>	Using [Addr#]SS Z

MS2000 and RM2000 Syntax

<b>Shortcut</b>	AL
<b>Format</b>	AL [X=Log_amp_AGC] [Y=LED_intesity_pot] [Z=in_focus_mm]
<b>Remembered</b>	Using SS Z

X and Y arguments of this command to directly read and write values (0 to 255) to the CRISP electronics digital potentiometers. (Not recommended for use with host software.)

The Z-argument specifies the focus precision (in millimeters) when the lock state changes from K or k to F. Useful for automatic checking of desired focus stability. Also useful to enforce a tighter or looser focus state before indicating a lock condition. Note that this value is overwritten whenever the NA of the objective is specified via the LR Y command as of November 2015.

For Video Autofocus

Tiger Syntax


<b>Shortcut</b>	AL
<b>Format</b>	[Addr#]AL [X= x-axis highlight] [Y= y-axis highlight] [Z= safety limit enable]
<b>Type</b>	Card-Addressed
<b>Remembered</b>	Using [Addr#]SS Z

MS2000 and RM2000 Syntax

<b>Shortcut</b>	AL
<b>Format</b>	AL [X= x-axis highlight] [Y= y-axis highlight] [Z= safety limit enable]
<b>Remembered</b>	Using SS Z

The X and Y values set the length and breadth of the Sampled/Highlighted Video area. Range is 0 to 100, with the value of 0 covering 0% of the video frame and 100 covering 90% of video frame.

The Z value enables or disables the 200 µm safety limit described in the AUTOFOCUS OPERATION section on page 4. Setting safety limit enable = 1 enables the safety limit; safety limit enable = 0 disables the safety feature. The default value is 1.



Caution: Disabling the safety limit could result in damage to your optics, your sample, or your focus drive.

```
AL X=80 Y=50 Z=1
:A<CR><LF>
```

```
AL
:N-3
```

Error indicates missing arguments

```
AL X=1000 Y=- 12
:N-4
```

Error indicates arguments out of range

```
AL X=90 Y=90
:N-5
```

Error indicates operation failed, try entering one argument at a time

```
AL X? Y? Z?
:A X=80 Y=50 Z=1
```

2016/02/23 19:18

[commands](#), [tiger](#), [ms2000](#), [crisp](#), [autofocus](#)

### Command:AFMOVE (AM)

MS2000 and RM2000 Syntax

<b>Shortcut</b>	AM
<b>Format</b>	AFMOVE [X=0 or 1] [Y=0 or 1]
<b>Units</b>	integer code (0 or 1)
<b>Remembered</b>	Using SS Z

Tiger Syntax

<b>Shortcut</b>	AM
<b>Format</b>	[Addr#]AFMOVE [X=0 or 1] [Y=0 or 1]
<b>Units</b>	integer code (0 or 1)
<b>Type</b>	Card-Addressed



**Remembered** Using [Addr#]SS Z



At this time this Autofocus 2 feature isn't supported on Tiger controllers.

A mode command that influences subsequent commanded moves. If X=1, then upon completion of a commanded XY move (**MOVE** and **MOVREL**), for example, MOVE X=123 Y=456 , a multi-axis controller will then automatically initiate an auto-focus.

Y=1 enables the **SAFE\_TURRET** module. Y=0 disables it.

**Response**

:A or Error Reply.

**Example**

```
AM X=1
:A<CR><LF>
```

```
AM X?
:A X=1
```

2016/03/24 18:39

[commands](#), [autofocus](#), [ms2000](#)

**Command:AFOCUS (AF)**

MS2000 and RM2000 Syntax

<b>Shortcut</b>	AF
<b>Format</b>	AFOCUS X= [% of speed] Y= [travel distance in mm] Z= [Hill Detect enable] F= [Hill Offset]
<b>Remembered</b>	Using SS Z

Tiger Syntax

<b>Shortcut</b>	AF
<b>Format</b>	[Addr#] AFOCUS X= [% of speed] Y= [travel distance in mm] Z= [Hill Detect enable] F= [Hill Offset]
<b>Type</b>	Card-Addressed
<b>Remembered</b>	Using [Addr#]SS Z

The AFOCUS command will invoke an auto-focus routine using the *speed* and *travel* range specified by the previously set **X** and **Y** parameter settings. This routine can also be activated by pressing “@”

button on most controllers. Entering AF without arguments will initiate the auto-focus routine itself, whereas issuing an AF with arguments will only set the parameters. An auto-focus scan can be canceled with HALT command or by issuing the \ shortcut. When an AF command is issued, the controller only replies after the operation is complete. It returns a :A[###] if the operation was successful, or N-5 if there was an error.

**X=** Speed. Range is 0 to 100 denoting the percentage of the focus drive's maximum possible speed to travel during an auto-focus scan. This speed is also used by the [AFCALIB \(or AFC\) command](#).

**Y=** Total scan range in millimeters. The focus controller moves down one-half this travel distance, and then scans up the full travel distance. This range is also used by [AFCALIB \(or AFC\) command](#).

**Z=** Search type; either 0 or 1. A value of 0 enables *Normal* mode, while a 1 enables Hill Detect mode. (Z values 2 and 3 are reserved for future use.)

**F=** Hill Offset. Range is 0 to 100 denoting a percentage of a hill. If the Search Type is Hill Detect, then this setting determines when the scan will end. Once a hill peak is detected, the scan will terminate when past the peak by the *Hill Offset* percentage value.

### Response

:A or Error Reply.

### Example

```
AF
:A<CR><LF>
```

```
AF X=5 Y=0.1 Z=0
:A
```

```
AF X=10 Y=0.3 Z=1 F=10
:A
```

```
AF X=200 Z=2
:N-4
```

(Error indicates arguments out of range)

```
AF X?
:X=10 A
```

AF

Returns :A [quality#] after operation is complete, an :N-5 if the operation failed, or an :N-50 if the focus drive's clutch is disengaged (if applicable).

When all of the arguments are omitted, the A is transmitted after the focusing scan has completed. The number in brackets is the difference between *focus value* when in-focus compared to when out-of-focus. It indicates the quality of focus obtained.

**Caution:** To protect the optical assembly and sample, ensure that the sample is at least 200  $\mu\text{m}$  away from the optics and that the current position is zeroed, that is, the LCD display shows Z: 0.00000 > 0.00000 before sending the AF command to the controller (where Z is the focus axis).

2016/03/24 18:47

[commands](#), [autofocus](#), [ms2000](#)

## Command: AHOME (AH)

MS2000 or RM2000 syntax

<b>Shortcut</b>	AH
<b>Format</b>	AH [X=x0] [Y=y0]
<b>Units</b>	millimeters
<b>Required Firmware Module</b>	ARRAY
<b>Remembered</b>	Using SS Z

Tiger syntax

<b>Shortcut</b>	AH
<b>Format</b>	[addr#]AH [X=x0] [Y=y0]
<b>Units</b>	millimeters
<b>Type</b>	Card-Addressed
<b>Required Firmware Module</b>	ARRAY
<b>Remembered</b>	Using [addr#]SS Z

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.

2016/03/14 17:49

[commands](#), [tiger](#), [ms2000](#), [array](#)

## Command: AIJ

Array Module

MS2000 or RM2000 syntax

<b>Shortcut</b>	IJ
<b>Format</b>	AIJ [X=i] [Y=j]
<b>Units</b>	array location
<b>Required Firmware Module</b>	ARRAY

Tiger syntax

<b>Shortcut</b>	IJ
<b>Format</b>	[addr#]AIJ [X=i] [Y=j]
<b>Units</b>	array location
<b>Type</b>	Card-Addressed

<b>Required Firmware Module</b>	ARRAY
---------------------------------	-------

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). The query "AIJ X? Y?" will return the i and j locations where the stage is currently.

Tiger MicroMirror Phototargeting

<b>Shortcut</b>	IJ
<b>Format</b>	[addr#]AIJ [X=horiz_position] [Y=vert_position]
<b>Units</b>	axis units
<b>Type</b>	Card-Addressed
<b>Required Firmware Module</b>	MM_TARGET
<b>Remembered</b>	Using [addr#]SS Z

Moves to the specified location (horiz\_position, vert\_position) subsequently pulses the laser TTL signal. Positions are specified in axis units (the same as used by the WHERE or MOVE command). If the X and/or Y argument is omitted, the corresponding position from the last AIJ command will be used. Note that the position is changed as a side effect of this command. The WHERE or MOVE command will change the beam position without pulsing the laser TTL signal.

The TTL output used was the micromirror card itself (rarely wired to anything) until v3.35, but as of v3.36 the laser trigger backplane line is used instead so that the signal is more accessible.

The settling delay before turning on the laser and the laser pulse high time are specified using the WAIT and RTIME commands respectively.

The assignments of horizontal and vertical axes are done using the SCAN command (most users will never need to change the defaults: the horizontal or X axis is the card's first axis and the vertical or Y axis is the card's second axis).

2016/03/14 17:52

[commands](#), [tiger](#), [ms2000](#), [array](#), [phototargeting](#)

**Command:ARM**

<b>Shortcut</b>	ARM
<b>Format</b>	ARM [X] [Y=log] [Z]
<b>Remembered</b>	Using SS Z

Without arguments, this command establishes the **ARM\_COMMAND\_RCVD** event. Typically it is used to start the SEQUENCER via a single serial command.

**ARM X** re-initializes the SEQUENCER. This command is useful if the RESET conditions are changed or if the sequencer is hung in a non-IDLE state.

**ARM Y=1** turns on or ARM Y=0 turns off the serial log file creation.

**ARM Z** forces a "sequence stop" condition. The sequencer is re-initialized; all blocks with ALWAYS

start conditions are placed in the IDLE state, and all stage motion is halted.

2016/05/18 15:59

[commands](#), [ms2000](#), [sequencer](#)

## Command:ARRAY (AR)

MS2000 or RM2000 syntax

<b>Shortcut</b>	AR
<b>Format</b>	AR [X=Nx] [Y=Ny] [Z=Δx] [F=Δy] [T=θ]
<b>Units</b>	X and Y in integer, Z and F in mm, θ tilt degrees from X axis
<b>Required Firmware Module</b>	ARRAY
<b>Remembered</b>	Using SS Z

Tiger syntax

<b>Shortcut</b>	AR
<b>Format</b>	[addr#]AR [X=Nx] [Y=Ny] [Z=Δx] [F=Δy] [T=θ]
<b>Units</b>	X and Y in integer, Z and F in mm, θ tilt degrees from X axis
<b>Type</b>	Card-Addressed
<b>Required Firmware Module</b>	ARRAY
<b>Remembered</b>	Using [addr#]SS Z

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. Small tilt corrections can be included using the T=tilt angle in degrees from the X-axis. The actual Δx and Δy for each move will be modified to account for the tilt angle.

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.

Whether a raster or serpentine pattern is used is set using the [SCAN F](#) setting (default is serpentine).

2016/03/14 18:13

[commands](#), [tiger](#), [ms2000](#), [array](#)

## Command:AZERO (AZ)

MS2000 or RM2000 syntax

<b>Shortcut</b>	AZ
<b>Format</b>	AZERO [axis]...
<b>Units</b>	Integers 0-255

Tiger syntax

<b>Shortcut</b>	AZ
-----------------	----

<b>Format</b>	AZERO [axis]...
<b>Units</b>	none
<b>Type</b>	Axis-Specific

Automatically adjusts the zero balance of the motor drive card. The expected “zeroed” values of AZ are typically around 127, though acceptable values may fall between 90 and 164. If an axis is not zeroed, the stage may have very different performance in one direction compared to the other, e.g. it may have trouble landing from one direction.

If AZ replies with NOT Zerod , run it again. If its unable to zero, then you may need to change [AA setting](#).

**Example**

Queries the current AZ parameters.

```
AZ X? Y? Z?
:A X=83 Y=78 Z=59
```

**Example**

Runs the auto-zeroing algorithm on the X axis. Note: Multiple Axes can be set simultaneously (**AZ X Y Z**)

```
AZ X
:A Zero C:1
A C:1 H:100 L:0
B C:0 H:92 L:0
Bracket H:92 L:76
E C:0 H:92 L:76
E C:0 H:92 L:84
E C:1 H:92 L:88
E C:0 H:90 L:88
Zerod at:90
```

2016/03/14 17:35  
[commands](#), [tiger](#), [ms2000](#)

**Command:BACKLASH (B)**

Motorized Actuator

MS2000 or RM2000 syntax

<b>Shortcut</b>	B
<b>Format</b>	BACKLASH [axis] = [distance]...
<b>Units</b>	millimeter

<b>Remembered</b>	Using SS Z
-------------------	------------

Tiger syntax

<b>Shortcut</b>	B
<b>Format</b>	BACKLASH [axis] = [distance]...
<b>Units</b>	millimeter
<b>Type</b>	Axis-Specific
<b>Remembered</b>	Using [addr#]SS Z

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 which 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. Default depends on motor build but is 0.04 for most common 4 TPI leadscrew pitch, 0.01 for most common 16 TPI leadscrew pitch, and 0.02 for the x-axis of scan-optimized stages.

### Example:

```
B X=.05 Y=.05 Z=0
:A
B x?
:X=0.040000 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.

MicroMirror, Tiger Galvo and Tunable Lens Cards

<b>Shortcut</b>	B
<b>Format</b>	B [axis]=[0.1 to 650] ...
<b>Units</b>	Frequency in kHz
<b>Type</b>	Axis-Specific
<b>Remembered</b>	Using [addr#]SS Z

This command is "recycled" for a different use in MicroMirror axes than for motor axes. In the context of a MicroMirror axis this command is used to set the cut off frequency of the 5th order Bessel filter. Units are in KHz. The lowest acceptable value is 0.1 (100Hz) and highest is 650 (650kHz). For a typical micro-mirror to avoid mechanical resonance this should be set no higher than 0.8 kHz.

### Example:

```
B R=0.1 S=0.1
:A
```

Sets 100Hz filter cut off freq for R and S axes

```
B P? Q?
:P=0.4 B=0.4 A
```

Queries the filter cut off freq for P and Q axes

2016/03/14 18:19

[commands](#), [tiger](#), [ms2000](#), [micromirror](#), [tlens](#)

### Command:BCUSTOM (BCA)

MS2000 or RM2000 syntax

<b>Shortcut</b>	BCA
<b>Format</b>	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]
<b>Units</b>	Integer code (see table below)
<b>Firmware Version Required</b>	9.2g+
<b>Remembered</b>	Automatically on powerdown. Not affected by SS Z or SS X. Before version 9.2m, SS Z used to save settings.

Tiger syntax

<b>Shortcut</b>	BCA
<b>Format</b>	[addr#]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]
<b>Units</b>	Integer code (see table below)
<b>Type</b>	Card-Addressed
<b>Firmware Version Required</b>	3+
<b>Remembered</b>	Automatically on powerdown. Not affected by [addr]SS Z or [addr]SS X. Before version 3.3, [addr]SS Z used to save settings.

In Tiger version 3.0+ and MS2000 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) through 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.

As of Tiger version 3.18 and MS2000 version 9.2l a button function can be initiated over serial using the BE F command ([BENABLE](#)). The function doesn't need to be assigned to a particular button for this to work. Note: this function does not interact with the `button_flag_byte`, use [EXTRA M=#](#) if you need that functionality.

Note: Behavior of this command is very different pre-Tiger version 3.0 and MS2000 version 9.2g



As of MS000 version 9.2m+ and Tiger v3.35+, it's possible to assign button functions to Home Normal Press and Joystick Extra Long Press using the **BENABLE** command. The parameters **R** and **T** are used, and follow the same behavior as BCUSTOM. The parameter **M** to sets the Zero/Halt Normal Press button function with **BENABLE M=#**.

Note: The Zero/Halt button will always halt all axes on a button press unless you set the button function to 0 - No Function Performed, which disables the halting routine. The halting routine happens as soon as you press the button, not in the release handler.

Example: **#BENABLE R=0** to disable the home button normal press, where **#** is the Tiger card address.

#### Button Press Duration:

- \* 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.

#### Button Function Table

The 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, LSRTK 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. Note, if there are multiple clocked devices on the same card or controller, all of them are moved. For more specificity see 36 and 37 below
5	ARRAY MODULE related, same as <a href="#">Command:ARRAY (AR)</a> . When <b>TTL X=7</b> does move to next array position.
6	RING BUFFER related, move to next ring buffer position; On TGLED card, button press mimics TTL pulse verion 3.24+.
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 away from the sample, to its upper limit, and back. For loading samples.
13	CRISP and TRACKING related, performs CRISP very long press functions
14	ADEPT related, toggles between piezo external and internal input mode
15	CRISP and TRACKING 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

No	Function
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
30	CLOCKED DEVICE related, moves clocked devices like Turret and slider to previous position. Note, if there are multiple clocked devices on the same card or controller, all of them are moved. For more specificity see 38 and 39 below
31	JS_FASTSLOW related, sets joystick speed to fast
32	JS_FASTSLOW related, sets joystick speed to slow
33	LED related, increase LED intensity
34	LED related, decrease LED intensity
35	LED related, toggle LED ON and OFF
36	CLOCKED DEVICE related, moves the FIRST clocked devices like Turret and slider to next position. Similar to option 4
37	CLOCKED DEVICE related, moves the SECOND clocked devices like Turret and slider to next position. Similar to option 4
38	CLOCKED DEVICE related, moves the FIRST clocked devices like Turret and slider to previous position. Similar to option 30
39	CLOCKED DEVICE related, moves the SECOND clocked devices like Turret and slider to previous position. Similar to option 30
40	HOME related, issues a <a href="#">Command:HOME</a> on all axis motors (MS2000 9.2m+ and Tiger v3.35+ firmware)
41	ZERO related, issues a <a href="#">Command:ZERO</a> on all axes. (MS2000 9.2m+ firmware and Tiger v3.35+ firmware)

**Example:**

Take a firmware build like STD\_XY. This firmware is packed with modules like,

- Ring Buffer module
- JS\_FASTSLOW, joystick button press toggles joystick speed between fast slow.

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.

Tiger Example

```
1BCA X? Y? Z? F? T? R? M?
```

```
X=0 Y=0 Z=0 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>
```

### MS2000 Example

```
BCA X? Y? Z? F? T? R? M?
X=0 Y=0 Z=0 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, does nothing @ long press, does nothing @ extra long press, does nothing 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.

### Tiger Example

```
1BCA X=6 F=24 R=18 M=28
:A
<LF>
```

### MS2000 Example

```
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, and toggles joystick speed.

**Tiger example scenario**

```
30: Comm v3.26 TIGER_COMM Apr 09 2019:15:52:41
31: X:XYMotor,Y:XYMotor v3.25 STD_XY_LED Mar 18 2019:15:56:27
32: Z:ZMotor,S:Slider v3.26 STDZ_SLDR8 Apr 30 2019:08:40:35
33: 0:Tur v3.26 TURRET_SINGLE Apr 30 2019:08:41:10
```

A user with a Tiger system in the above configuration has two clocked position devices. A filter slider on card #2 and turret on card#3. They would like to move these devices with button presses. Lets set it up so that on short '@' button press (less than 1-sec press) Filter slider moves, then on long '@' button press (1 to 3-sec press) the turret moves. The button function that does this is '# 4' (see the table above). '@' short press assigned with 'X' variable, and '@' long press is assigned with 'Y' variable.

2BCA X=4 Y=0 , this tells card#2 which drives the filter slider to respond to short @ press with slider move, and ignore any long @ button presses.

3BCA X=0 Y=4 , this tells card#3 which drives the turret to respond to long @press with turret move and ignore any short @ button presses.

1BCA X=0 Y=0 , this tells card#1 which drives XY stage to ignore any short or long @ button presses.

Try it out. If you like the behavior, commit it to systems onboard memory with serial commands

```
1SS Z
2SS Z
3SS Z
```

More info on SS Z command is here, <http://www.asiimaging.com/docs/commands/saveset>

2016/03/14 18:34  
[commands](#), [tiger](#), [ms2000](#)

**Command:BENABLE (BE)**

MS2000 or RM2000 syntax

<b>Shortcut</b>	BE
<b>Format</b>	BENABLE [X=Toggle] [Z=Enable_Byte] [F=Button_Function] [R=Home Normal Press] [T=]S Extra Long Press] [M=Zero/Halt Normal Press]
<b>Units</b>	None
<b>Remembered</b>	Using SS Z

Tiger syntax

<b>Shortcut</b>	BE
<b>Format</b>	[Addr#]BENABLE [X=Toggle] [Y?] [Z=Enable_Byte] [F=Button_Function] [R=Home Normal Press] [T=]S Extra Long Press] [M=Zero/Halt Normal Press]
<b>Units</b>	None
<b>Type</b>	Card-Addressed
<b>Remembered</b>	Using [addr#]SS Z

Enables or disables button functions for the specified card and specified buttons, either all/none (X, or Toggle) or with finer granularity (Z, or Enable\_byte). Toggle=0 disables all buttons and pulses; i.e. BE X=0 is equivalent to BE Z=0. Toggle=1 enables all buttons and pulses (default settings); i.e. BE X=1 is equivalent to BE Z=15. Querying X returns the same as querying Z. 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:

**Z** parameter Enable\_Byte is bit-mapped number that determines which buttons are enabled or disabled as tabulated below. Bit5 is a special case; it changes the behavior of the physical button to zero the Z axis only and has been removed from Tiger code after v3.20.

Bit#	Button
0	"Zero" Button
1	"Home" Button
2	"@" Button
3	Joystick Button
4	Reserved
5	Zero Z Only (removed from Tiger after v3.20)
6	Reserved
7	Reserved

**F** parameter Button\_Function is a positive integer code for the button function that will be executed. See [documentation of BCUSTOM](#) for a list of available button functions. This allows a button press to be simulated using a serial command. This is available from Tiger v3.19 onward and in middle of Whizkid v9.2I.

If you need to call button functions that interact with the button\_flag\_byte use **EXTRA M=#** instead.

### Button Function Assignment Parameters

As of MS2000 9.2m+ and Tiger v3.35+ you can assign functions from from the button function table in [BCUSTOM](#) to additional Home and Joystick button press durations.

**R** - assign a button function to Home Normal Press.

**T** - assign a button function to Joystick Extra Long Press .

**M** - assign a button function to Zero/Halt Normal Press.

Note: The Zero/Halt button will always halt all axes on a button press unless you set the button function to 0 - No Function Performed, which disables the halting routine. The halting routine happens as soon as you press the button, not in the release handler.

#### Tiger Example

To make Axis on Card#1 ignore the zero and home button, and only respond to Joystick and @ buttons. Bit 3 and Bit 2 set to '1' , Bit 1 and Bit 0 set to '0'. Binary '1100' is Decimal '12'

```
1BE Z=12
:A
```

#### MS2000 Example

To make all the axis on controller ignore the zero and home button, and only respond to Joystick and @ buttons. Bit 3 and Bit 2 set to '1' , Bit 1 and Bit 0 set to '0'. Binary '1100' is Decimal '12'

```
BE Z=12
:A
```

#### Additional Feature in Tiger Version 3.14+

Addressing the COMM card specifically (or omitting the address on the command) results in a different behavior. Disabling a button on the COMM card will result in disabling that functionality for all cards in the rack. This disabling happens at the COMM card so the other cards in the rack never receive notification of a change in the button's state. In this way, individual cards may be set to the desired functionality and a layer of control over all cards may be applied without affecting the settings on individual cards.

The `0BE Y?` query command provides a report of past button states. Every time one of the inputs is activated, the COMM card notes the activation by setting a bit in a status byte. That bit will remain set until the status byte is queried again on the COMM card with `0BE Y?`. The result of the query will return the current status byte to the host, then clear the status byte (set all bits representing input sources to 0), thereby preparing the status byte for further button press detection. The format of the bits in the returned status byte is the same as the table above. A one 1 on the specified bit represents the input has been activated since the last query.

Starting with v3.21, a button that is held down during a serial query of button state will be reported as activated every query up through the one immediately following release. This can be used by high-level software to time button presses, e.g. if querying happens every second and the button is held down for 1.001 seconds then it will be reported as being held down twice. If the button is held down for 0.1 seconds then most of the time it will only be reported once, unless the query happens to occur during that 0.1 seconds in which case it will be reported twice.

2016/03/14 18:49

[commands](#), [tiger](#), [ms2000](#)

### Command:BUILD (BU)

MS2000 or RM2000 syntax

<b>Shortcut</b>	BU
<b>Format</b>	BUILD [X] [Y]
<b>Units</b>	none

Tiger syntax

<b>Shortcut</b>	BU
<b>Format</b>	[Addr#]BUILD [X] [Y]
<b>Units</b>	none
<b>Type</b>	Card-Addressed(defaults to COMM)

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

The BU Y command is used to store a user defined string which can be saved using the [SAVESET Z](#) command. The maximum length of the user string is 20 characters. Remember to use the card address if you are storing a string on Tiger. This command is available on MS2000 9.2m+ and Tiger v3.39 firmware, you will also need TIGER\_COMM v3.39 firmware .

**BU Y=#** writes a character to the user string at the write position. The write position starts at 0 on system power up and increase by 1 every time BU Y=# is sent to the controller. You specify the characters in their decimal ASCII form, for example lower-case 'a' is 97, so you would send BU Y=97. The valid range of ASCII values is 32-126 inclusive.

**BU Y-** clears the user string and resets the write position to 0.

**BU Y?** reads the user string.

To make things easier on the user, here is script that takes a Python string and enters the serial commands for you:

#### Python User String Script

```
import serial # pip install pyserial (pyserial version 3.5 were used here)

def main() -> None:
    # the input string to send to the controller
    user_string = "abcdefghij1234567890"
    save_settings = True

    # use an empty string for MS2000 (card_address = "")
    card_address = "2"

    # error checking
    if len(user_string) > 20:
        raise Exception("Max stored string length is 20.")

    # open the serial port and send characters to the controller
    with serial.Serial("COM4", 115200, timeout=1) as serial_port:
        # clear the current stored string
        serial_port.write(bytes(f"{card_address}BU Y-\r", encoding="ascii"))
        serial_port.readline()

        # send the input string to the controller
        for character in user_string:
            serial_port.write(bytes(f"{card_address}BU Y={ord(character)}\r", encoding="ascii"))
            serial_port.readline()

        # print the stored string and check to see if it's the same as the input string
```

```

        serial_port.write(bytes(f"{card_address}BU Y?\r",
encoding="ascii"))
        response = serial_port.readline().decode().rstrip("\r\n")
        if response == user_string:
            print(f"Successfully stored the input string =>
{response}")
            if save_settings:
                serial_port.write(bytes(f"{card_address}SAVESET Z\r",
encoding="ascii"))
                serial_port.readline()
                print("Settings saved to the controller using SAVESET
Z.")
        else:
            print(f"Error: expected {user_string} but got {response}
instead!")

if __name__ == "__main__":
    main()

```

### Additional Feature on Tiger

Adding the card address is optional. If no address is given, then Tiger Comm replies. If an address is present, then the specified card replies.

```

BU
TIGER_COMM

```

```

BU X
TIGER_COMM
Motor Axes: X Y A B C C 0 1
Axis Types: x x u u u u w w
Axis Addr: 1 1 2 2 2 2 3 3
Hex Addr: 31 31 32 32 32 32 33 33
Axis Props:  0  0  0  0  0  0  0  0

```

As no address was given, Tiger Comm replies. It replies with its build name, all axis names present in the system (axes will always be A-Z, filterwheels 0-9). For each axis the type is given (see table in section "Identifying Controller Configuration") and the card address in both character and hex formats. Finally, an integer is given to indicate the presence of special properties or capabilities of the axis, such as CRISP auto-focus or RING BUFFER module for TTL positioning; these can be interpreted using the information below. This command is useful to quickly identify all the axes names and types present in the system.

### Example

#### Tiger Example

```

1bu

```



STD\_XY

```

1bu x
STD_XY
Motor Axes: X Y
Axis Types: x x
Axis Addr: 1 1
Hex Addr: 31 31
Axis Props: 0 0
CMDS: XY
BootLdr V:NONE
LL COMMANDS
SEARCH INDEX
NO CLTCH SW
SHUTDOWN_TASK
    
```

With an address 1 given, the specified card #1 replies. This reply just contains axis name and types present on the card. However it goes into more detail, printing all the firmware modules present on the card.

The values listed for axis properties are decimal integer representations of a binary code which represents any special properties of the axis. Usually these could also be identified by doing a BU X query of each card and interpreting the response, but they are listed separately on the axis property line for convenience.

MS2000 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 ...
INO_INT ...
DAC OUT
    
```

**Axis Properties**

Bit 0:	CRISP auto-focus firmware
Bit 1:	RING BUFFER firmware
Bit 2:	SCAN firmware
Bit 3:	ARRAY firmware or MM_TARGET firmware
Bit 4:	SPIM firmware (v2.81+)
Bit 5:	SINGLEAXIS and/or MULTIAXIS firmware (v2.81+)
Bit 6:	LED illumination (v2.87+)

Bit 7: Reserved

2016/02/24 15:48 · asiadmin

### Axis Type List

Axis Type Short	Axis Type Long	Description
x	XYMotor	XY stage
z	ZMotor	Z focus motor drive. LS50s, Z scopes etc
p	Piezo	Piezo Focus. ASIs ADEPT, Piezo DAC etc
o	Tur	Objective Turret
f	Slider	Filter Changer
t	Theta	Theta Stage
l	Motor	Generic linear motorized stage, TIRF, SISKIYOU etc
a	PiezoL	Generic linear piezo stage
m	Zoom	Zoom magnification motor axis
u	MMirror	Micro Mirror, Scanner 75 etc
w	FW Filter	Wheel
s	Shutter	Shutter
g	Logic	Programmable logic card
i	LED card	Multi LED Driver card
b	Lens	Tunable Lens
d	DAC	Digital to Analog converter(DAC)
u	Unknown	Unknown axis type

2016/02/24 15:48 · asiadmin

2016/03/14 19:18

[commands](#), [tiger](#), [ms2000](#)

### Command:CDATE (CD)

MS2000 or RM2000 syntax

<b>Shortcut</b>	CD
<b>Format</b>	CDATE
<b>Type</b>	Card-Addressed

Tiger syntax

<b>Shortcut</b>	CD
<b>Format</b>	[Addr#]CDATE
<b>Type</b>	Card-Addressed

This command returns the date and time the current firmware were compiled

Tiger Example

```
1CD
Dec 19 2008:16:19:59
```

## MS2000 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.

2016/03/14 19:51

[commands](#), [tiger](#), [ms2000](#)

**Command:CNTS (C)**

MS2000 or RM2000 syntax

<b>Shortcut</b>	C
<b>Format</b>	CNTS [axis] = [encoder counts per mm]...
<b>Units</b>	encoder counts per mm
<b>Remembered</b>	Using SS Z

Tiger syntax

<b>Shortcut</b>	C
<b>Format</b>	CNTS [axis] = [encoder counts per mm]...
<b>Units</b>	encoder counts per mm
<b>Type</b>	Axis-Specific
<b>Remembered</b>	Using [addr#]SS Z

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!** If your stage is not moving as expected try making sure the leadscrew pitch is set appropriately with the [CUSTOMA](#) command.

$$\begin{equation} Cnts = \frac{(6.5536 * 107)}{d} \end{equation}$$

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: For a piezo device, always set CNTS first, then limits (SL and SU) afterward.

**Example:**

```
C X=13490.4
:A
```

```
C x?  
: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.

2019/04/18 19:33  
[commands](#), [tiger](#), [ms2000](#)


### Command:CUSTOMA (CCA)

#### MS2000 Syntax

<b>Shortcut</b>	CCA,CA
<b>Format</b>	CCA X=n Y=m Z=o
<b>Remembered</b>	X automatically saved, Y and Z need SS Z

#### Tiger Syntax


<b>Shortcut</b>	CCA,CA
<b>Format</b>	[Addr#]CCA X=n Y=m Z=o
<b>Type</b>	Card-Addressed
<b>Remembered</b>	X automatically saved, Y and Z need [Addr#]SS Z



For the Tiger programmable logic card this command is used very differently; for that see [Tiger Programmable Logic Card \(TGPLC\)](#) documentation and ignore this page.

### X parameter

X sets the 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 or reset via the RESET command.



The controller should be reset immediately after setting any desired CCA X flags. Executing a SS Z command before resetting can cause the firmware to get confused.

CCA X=	Description	Display	MS2K specific Comments	TG-1000 specific Comments
1	XY Linear Encoders Used	L	Use DIP SW 3 (See Note 1)	

CCA X=	Description	Display	MS2K specific Comments	TG-1000 specific Comments
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)	
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		Not implemented
43	XY Maxon Direct-Drive (3.18 mm)	e		Not implemented
44	XY Maxon Direct-Drive (6.35 mm)	X		Not implemented
48	XY LS25 Fine Pitch (1.59 mm)	G	Ver 9.2l+	
56	XY LS25 Coarse Pitch (6.35 mm)	H	Ver 9.2m+	
58	XY Leadscrew Ultra Coarse (25.4 mm) with 76:1 Gear motor	E	Ver 9.2l+	
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 um/rev.	1	Firmware default	
10	Z Scope Drive 200 um/rev.	2		
19	Z Scope Drive 400 um/rev.	4		
11	Z Leadscrew Coarse Pitch	B		
12	Z Leadscrew Fine Pitch	A		
13	Z Leadscrew Super Coarse Pitch	C		
14	Z Leadscrew Ultra Fine Pitch	U		
47	Z Leadscrew Ultra Coarse	D		
29	Z SISKIYOU Motor/Leadscrew	S		
49	Z LS25 Fine Pitch (1.59 mm)	G	Ver 9.2l+	
57	Z LS25 Coarse Pitch (6.35 mm)	H	Ver 9.2m+	

CCA X=	Description	Display	MS2K specific Comments	TG-1000 specific Comments
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 um	f or Pf		
65	Piezo Range 70 um	g or Pg	not implemented	
23	Piezo Range 100 um	1 or P1		
35	Piezo Range 150 um	S or PS	Firmware default	
24	Piezo Range 200 um	2 or P2		
36	Piezo Range 300 um	3 or P3		
25	Piezo Range 350 um	t or P4		
37	Piezo Range 500 um	5 or P5		
66	Piezo Range 1000 um	W	Experimental	Not Implemented
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 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 9.0f and later.	
	Fixed Profile	F		Place holder profile
	MicroMirror 6 degrees	U6		TG-1000 only
	MicroMirror 8 degrees	U8		TG-1000 only
	MicroMirror 10 degrees	UA		TG-1000 only

**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
A:
```

Sets to XY stage for 1.59mm pitch lead screws.

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.

```
A: XY:RBJ Z:RN PF:2

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
```

## Y parameter

Y 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  $m > 0$ , 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. Requires MOVETASK firmware module which is standard for XY builds but not most others.

### Example

```
CCA Y=3
:A
```

All moves will be initiated four times.

## Z parameter

Z sets system configuration flags according to following table. Parameter changes must be saved in non-volatile memory by the "SS Z" command.

CCA Z=	Description	Display	MS2K Comment	TG-1000
1	X axis movement direction is positive (default).	+	Firmware 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)	+	Firmware Default	
4	Y axis movement is negative.	-		
5	Z axis movement is positive	+	Firmware Default	
6	Z axis movement is negative.	-		
7	F axis movement is positive.	+		
8	F axis movement is negative.	-		
9	Disengage clutch	D		
10	Engage clutch	E		
11	Enable LCD display	O		not implemented
12	Disable LCD display	F		not implemented
13	CLOCKED DEVICES on 1st axis take shortest path	S		Applied to both axes until 3.10
14	CLOCKED DEVICES on 1st axis do not take shortest path	L		Applied to both axes until 3.10
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	Firmware default
17	CLOCKED DEVICES on 2nd axis take shortest path	S	Works in 9.2m and above	
18	CLOCKED DEVICES on 2nd axis do not take shortest path	L	Works in 9.2m and above	
20	The joystick and knob can be disabled, and the device assignments can be changed. The JOYSTICK command works normally.	J	Firmware default Version 9.0f and later.	Firmware default
21	The joystick and knob are always enabled, and the device assignments cannot be changed. The JOYSTICK command has no effect..	j	Version 9.0f and later.	
22	Reverses joystick polarity of the card's first axis	r	TG-1000 only V3.05 and later. Re-issue J command or restart after SS Z to take effect.	
23	Joystick polarity of card's first axis set to default	l	TG-1000 only, Firmware Default V3.05 and later. Re-issue J command or restart after SS Z to take effect.	
24	Reverses joystick polarity of the card's second axis	r	TG-1000 only V3.05 and later. Re-issue J command or restart after SS Z to take effect.	
25	Joystick polarity of card's second axis set to default	l	TG-1000 only, Firmware Default V3.05 and later. Re-issue J command or restart after SS Z to take effect.	
26	Enable Encoder E flag check expressed as error 110+ in dump buffer		MS2000 only v9.2j and later	




CCA Z=	Description	Display	MS2K Comment	TG-1000
27	Disable Encoder E flag check		MS2000 only, Firmware Default v9.2j and later	

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

2016/03/11 18:17

[commands](#), [tiger](#), [ms2000](#)

### Command: CUSTOMB (CCB)



Not implemented in Tiger except for programmable logic card, for that see [Tiger Programmable Logic Card \(TGPLC\)](#) documentation and ignore this page.

<b>Shortcut</b>	CCB
<b>Format</b>	CCB Z=n
<b>Units</b>	none
<b>Remembered</b>	Using SS Z

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
```

2016/03/14 20:20

[commands](#), [ms2000](#), [planar](#)

## Command:DACK (D)

Motorized Actuator

MS2000 or RM2000 syntax

<b>Shortcut</b>	D
<b>Format</b>	D [axis]=[ratio in mm/sec] ...
<b>Units</b>	ratio in mm/sec
<b>Remembered</b>	Using SS Z

Tiger syntax

<b>Shortcut</b>	D
<b>Format</b>	D [axis]=[ratio in mm/sec] ...
<b>Units</b>	ratio in mm/sec
<b>Type</b>	Axis-Specific
<b>Remembered</b>	Using [addr#]SS Z

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
D X?
:A X=0.055000
```

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

MicroMirror

<b>Shortcut</b>	D
<b>Format</b>	D [axis]=[0 to 1] ...
<b>Units</b>	Unitless float between 0 and 1
<b>Type</b>	Axis-Specific
<b>Remembered</b>	Using [addr#]SS Z

This command is “recycled” for a different use in MicroMirror axes than for motor axes. For MicroMirror axes it sets up a calibration constant or scale factor that is used to attenuate the scanner motion when used in internal input mode (as the implementation is in software it does not apply in external input mode). In tests we found that the two axis of the MEMS MicroMirror does not tilt by the same amount when similar inputs are applied. This may be critical for some applications. This parameter sets an attenuation, so both axes can be made to tilt the same amount. For example, if the S axis was found to be doing 85% travel of R axis then we could attenuate R to compensate by setting D R=0.85 S=1. Note that behavior is undefined if

changed in the middle of a single-axis move or SPIM move on the same axis. **N.B: This command was retired in v3.14 of the firmware because it wasn't being used and the internal implementation was complex.**

### Example:

```
D R=0.85 S=1
:A
```

Attenuates the travels of R axis by 15%.

2016/03/14 20:27

[commands](#), [tiger](#), [ms2000](#), [micromirror](#)

## Command:DUMP (DU)

MS2000 or RM2000 syntax

<b>Shortcut</b>	DU
<b>Format</b>	DUMP [X] [Y][F]
<b>Units</b>	none

Tiger syntax

<b>Shortcut</b>	DU
<b>Format</b>	[addr#]DUMP [X][Y][F][R=mode][T=interval]
<b>Units</b>	none
<b>Type</b>	Card-Addressed
<b>Remembered</b>	not possible

Dump internal buffers to terminal screen.

The Tiger and 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 to 500 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.

DU, without arguments, dumps the Trajectory Buffer.

DU X clears Trajectory Buffer.

DU Y dumps Error Buffer. See [Error Codes for MS2000,RM2000 and TG-1000 Diagnostics](#)

DU F Prints controller log, it has information like total time controller was ON, Total distance XY stage travelled and others. The log first has to be initialized with command DU F=999 . This is done in the factory, and we suggest users not reset the log as it may wipe data that is useful for ASIs long term reliability study.

DU R=mode sets the trajectory buffer mode (introduced in Tiger v3.19). Mode 0 (default) means the trajectory buffer, once full, will not be changed further. Mode 1 continuously overwrites the trajectory buffer information so that the most recent move information is always present.

DU T=interval sets the sampling interval for the trajectory buffer (introduced in Tiger v3.19). Default is 1 which adds to the trajectory buffer on every axis loop. Setting to e.g. 10 will add to the buffer once and then skip the next 9 axis loops. Handy for looking at longer-term trends.

Tiger Example

- 1DU X Clears the dump buffers on card#1
- 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: 1DU Dumps Trajectory Buffer on card #1
- 2DU Y Dumps Error Buffer on card #2
- 4DU F Dumps Card#4's Piezo History
- 4DU F=999 Resets Card#4's Piezo History

MS2000 Example

- 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

2016/03/14 20:37

[commands](#), [tiger](#), [ms2000](#)

### Command:ENSYNC (ES)

MS2000 or RM2000 syntax

<b>Shortcut</b>	ES
<b>Format</b>	ENSYNC [axis]=[position in mm]...
<b>Units</b>	position in millimeter
<b>Remembered</b>	Using SS Z

Tiger syntax

<b>Shortcut</b>	ES
<b>Format</b>	ENSYNC [axis]=[position in mm]...
<b>Units</b>	position in millimeter
<b>Type</b>	Axis-Specific
<b>Remembered</b>	Using [addr#]SS Z

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.

**On MS2000**

(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

**On Tiger**

Need Dual Axis card Rev F3 and above. Pin is routed to the backplane. Additional hardware is needed to expose the signal. Contact ASI for additional details on these modifications



Warning—units of the position info is millimeters rather than tenths of microns

2016/03/15 19:36

[commands](#), [tiger](#), [ms2000](#)

**Command:EPOLARITY (EP)**

MS2000 or RM2000 syntax

<b>Shortcut</b>	EP
<b>Format</b>	EP [axis]=[-1 or 1]...
<b>Units</b>	only -1 or +1 accepted
<b>Remembered</b>	Using SS Z

Tiger syntax

<b>Shortcut</b>	EP
<b>Format</b>	EP [axis]=[-1 or 1]...
<b>Units</b>	only -1 or +1 accepted
<b>Type</b>	Axis-Specific
<b>Remembered</b>	Using [addr#]SS Z

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.

2016/03/15 19:43

[commands](#), [tiger](#), [ms2000](#)

**Command:ERROR (E)**

Motorized axis

MS2000 or RM2000 syntax

<b>Shortcut</b>	E
-----------------	---

<b>Format</b>	ERROR [axis]=[position in mm]...
<b>Units</b>	millimeter
<b>Remembered</b>	Using SS Z

Tiger syntax

<b>Shortcut</b>	E
<b>Format</b>	ERROR [axis]=[position in mm]...
<b>Units</b>	millimeter
<b>Type</b>	Axis-Specific
<b>Remembered</b>	Using [addr#]SS Z

This command sets the Drift Error setting. This setting controls the crossover position error (in millimeters) between the target and position at which the MS 2000 and TG-1000 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 are ignored.

**Example**

```
E X=.0004
:A
e x?
:X=0.000400 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.

TGPMT card

<b>Shortcut</b>	E
<b>Format</b>	ERROR [axis]=[0 to 5]...
<b>Units</b>	integer code
<b>Type</b>	Axis-Specific
<b>Remembered</b>	Using [addr#]SS Z

On a TGPMT card , ERROR command has a different purpose and function. ERROR command sets the [ADC averaging's sample size](#). Its a card wide setting, that affects both the PMT tubes. This only affects the value reported with the [RDADC](#) command. The Analog signal output on the BNC isn't affected.

ERROR [axis]=	Simple Moving Average sample size
0	1 Avg routine disabled, reports RAW reading
1	2
2	4

<b>ERROR [axis]=</b>	<b>Simple Moving Average sample size</b>
3	8
4	16
5	32

### Example

If TGPMT card axis is **m**

```
e m=2
:A
```

Sets the ADC averaging routine sample size to 4.

```
e m?
:M=2.000000 A
```

Query the TGPMT card for ADC averaging routine sample size, 2 is reported, hence sample size is 4.

### MicroMirror card

<b>Shortcut</b>	E
<b>Format</b>	ERROR [axis]=[correction in %]...
<b>Units</b>	integer code
<b>Type</b>	Axis-Specific
<b>Remembered</b>	Using [addr#]SS Z

On a Micromirror card ERROR command has a different purpose and function. It sets the amount of bow correction applied to the axis. It is specified in units of percent deflection of the paired axis when the specified axis is at full deflection. The sign indicates whether a positive or negative deflection will be applied to the paired axis. It was introduced in Tiger firmware v3.18 along with the bow correction described below.

Bow is an optical effect where the motion of a scanned beam takes on a slightly curved shape whenever the axis of the moving mirror's rotation is not orthogonal to the plane formed by input and output beams. Because the MEMS mirror inside the ASI scanner tilts in two dimensions, at least one axis will be subject to this effect. In a typical ASI light sheet scanner, the motion of the "fast axis" (e.g. the axis that is scanned to make a light sheet) will deflect slightly in the slow axis during scanning, whereas the motion of the slow axis is unaffected by the bow phenomenon. To counteract this optical effect, starting in firmware v3.18 the firmware applies a small opposite correction to the slow axis as the fast axis moves, and the magnitude of this correction is specified by the ERROR command. The correction is quadratic relative to the displacement from the center of the mirror range (generally position 0), so maximum correction is applied at the edge of the mirror range.

The default is -2% for the "fast axis" and 0.0 for the "slow axis" in each axis pair because this seems to be about right for a typical ASI scanner. This means that the slow axis position will be adjusted as the fast axis moves but not vice versa. The bow correction is applied during regular commanded moves, during single axis commands, and during the SPIM state machine. In the

SPIM state machine it is assumed that only the fast axis has a non-zero bow correction coefficient.

The allowable range of correction is between -6.22% to +6.22%.

### Tunable Lens Card

<b>Shortcut</b>	E
<b>Format</b>	ERROR [axis]=[DPT to DAC coefficient]...
<b>Units</b>	float
<b>Type</b>	Axis-Specific
<b>Remembered</b>	Automatically

On a Tunable card ERROR command has a different purpose and function. It sets the relation between the DAC(current) and Tunable lens's Diopter. Its used to compensate for temperature drift. [More info here](#)

2016/03/15 19:46

[commands](#), [tiger](#), [ms2000](#), [tgpmt](#), [micromirror](#), [tlens](#)

### Command:EXTRA

#### Tiger Syntax

<b>Format</b>	[Addr#]EXTRA [X?] [Y?] [Z=ki_lock] [M?]
<b>Type</b>	Card-Addressed
<b>Remembered</b>	Using [Addr#]SS Z

#### MS2000 and RM2000 Syntax

<b>Format</b>	EXTRA [X?] [Y?] [Z=ki_lock] [M?] [M=button_code]
<b>Remembered</b>	Using SS Z

**X?** Provides the CRISP bottom line string as is shown on the LCD display.

**Y?** Returns the SNR value shown on the LCD after log amp calibration.

The **Z** argument sets the integral error servo parameter. The default is 1. Higher values may speed settling but can also generate instability. Use sparingly.

**M?** Returns the `button_flag_byte` and resets the value to 0. (MS2000 9.2m+ and Tiger v3.35+)

**M=#** Modify the `button_flag_byte` with the button code and call the button functions associated with that code. (MS2000 9.2m+)

This command differs from **BE F** which does not modify the `button_flag_byte` and only calls a single button function.

#### Additional Details About M?

The `button_flag_byte` stores the state of the last detected button press for each button.



When the controller is powered on, the value is initialized to 0. As the user presses buttons the value of the `button_flag_byte` changes, it is important to note that this value only changes when you **release** the button.

After receiving the EXTRA M? command, the internal value on the controller is reset to 0, enabling you to detect new button presses.

If a button has already been pressed, and then is pressed again, the new state overwrites the old state for that button. Example: if you do a Normal Press and then a Long Press on the Joystick Button, the next time you send the "EXTRA M?" command the state of the Joystick Button will be Long Press.

### Zero/Halt button presses only have the states 0 and 1. (Not Pressed and Normal Press)

The `button_flag_byte` is divided into four 2-bit sections that each contain the state of a button:

Bits	Button
1-2	@ Button
3-4	Home Button
5-6	Joystick Button
7-8	Zero/Halt Button

Each 2-bit section can take on the values 0-3, these codes represent the state of the button.

Code	Type
0	Not Pressed
1	Normal Press
2	Long Press
3	Extra Long Press

Example:

1. @ Button Normal Press
2. Home Button Long Press
3. Joystick Extra Long Press
4. Zero/Halt Normal Press
5. Send serial command EXTRA M?

Results of steps 1-5 in binary:

1. `button_flag_byte` = 00 00 00 01
2. `button_flag_byte` = 00 00 10 01
3. `button_flag_byte` = 00 11 10 01
4. `button_flag_byte` = 01 11 10 01
5. `button_flag_byte` = 00 00 00 00 (serial command reset)

Example Python code for extracting button states from the `button_flag_byte`

```
# the value returned from EXTRA M?
button_flag_byte = 127

# bit masks
mask_at   = 0x03 # 00000011
mask_home = 0x0C # 00001100
mask_js   = 0x30 # 00110000
mask_zero = 0xC0 # 11000000

# get the button states from button_flag_byte
btn_at   = button_flag_byte & mask_at
btn_home = (button_flag_byte & mask_home) >> 2
btn_js   = (button_flag_byte & mask_js) >> 4
btn_zero = (button_flag_byte & mask_zero) >> 6

# show the results in decimal and binary
print(f"{button_flag_byte = } (binary {button_flag_byte :08b})")
print(f"{btn_at = } (binary {btn_at:02b})")
print(f"{btn_home = } (binary {btn_home:02b})")
print(f"{btn_js = } (binary {btn_js:02b})")
print(f"{btn_zero = } (binary {btn_zero:02b})")

# console output:
# button_flag_byte = 127 (binary 01111111)
# btn_at = 3 (binary 11)
# btn_home = 3 (binary 11)
# btn_js = 3 (binary 11)
# btn_zero = 1 (binary 01)
```

### Additional Details About M=button\_code

This function allows you to simulate button presses programmatically through a serial command.

This command modifies the `button_flag_byte` and calls the button functions associated with that button code.

The button codes are the same values that are returned by EXTRA M?. The input value is clamped to the range: 0-127.

If a button code represents multiple button presses then the button functions will be called in the order ⇒

@, Home, Joystick, Zero/Halt (LSB ⇒ MSB)

You can expect the same behavior as if you were pressing physical buttons ⇒

1. Send the command EXTRA M=3: `button_flag_byte = 3`, @ Extra Long Press button function called.
2. Send the command EXTRA M=1: `button_flag_byte = 1`, @ Normal Press button function called.

3. Send the command EXTRA M=5: button\_flag\_byte = 5, @ Normal Press and Home Normal Press button functions called.

This demonstrates that button presses are overwritten as if you were interacting with the physical controller pressing buttons.

Example Python code for creating a button\_code

```
def create_button_code(at: int = 0, home: int = 0, joystick: int = 0, zero_halt: int = 0) -> int:
    assert at in range(4), "Must be in the range 0-3."
    assert home in range(4), "Must be in the range 0-3."
    assert joystick in range(4), "Must be in the range 0-3."
    assert zero_halt == 0 or zero_halt == 1, "Must be 0 or 1."

    button_code = 0

    # bit masks
    BITMASK_AT = 0x03 # 00000011
    BITMASK_HOME = 0x0C # 00001100
    BITMASK_JS = 0x30 # 00110000
    BITMASK_ZERO = 0xC0 # 11000000

    # set bits
    button_code &= ~BITMASK_AT
    button_code |= at & BITMASK_AT

    button_code &= ~BITMASK_HOME
    button_code |= (home << 2) & BITMASK_HOME

    button_code &= ~BITMASK_JS
    button_code |= (joystick << 4) & BITMASK_JS

    button_code &= ~BITMASK_ZERO
    button_code |= (zero_halt << 6) & BITMASK_ZERO

    return button_code

def main():
    button_code = create_button_code(at=1, home=1)
    print(button_code)
    # prints 5

if __name__ == "__main__":
    main()
```

[commands](#), [tiger](#), [ms2000](#), [crisp](#)

## Command:HALT (\)

MS2000 or RM2000 syntax

<b>Shortcut</b>	\ (the backslash character)
<b>Format</b>	HALT

Tiger syntax

<b>Shortcut</b>	\ (the backslash character)
<b>Format</b>	HALT
<b>Type</b>	Broadcast or Card-Addressed command

This command will stop all active motors and other actuators too. 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.

Additional Notes regarding usage on Tiger

It's usually a Broadcast command but can be used as a Card-Addressed Command as well. When addressed to a specific card, it stops motion on that card only. Note that to use as a Card-Addressed Command the full command HALT must be used instead of the shortcut \, because \ is handled quickly in the command parser.

2016/03/15 19:52

[commands](#), [tiger](#), [ms2000](#)

## Command:HERE (H)

The 'here' command differs on a TGPM T card from our other systems.

Actuators, Piezos, MicroMirror etc

MS2000 or RM2000 syntax

The HERE command sets the current reported position(s) of the axis(es) provided.

<b>Shortcut</b>	H
<b>Format</b>	HERE [axis]=[position in 1/10 microns]...
<b>Units</b>	1/10 microns
<b>Remembered</b>	Automatically

Tiger syntax

<b>Shortcut</b>	H
<b>Format</b>	HERE [axis]=[position in 1/10 microns] or see below for clocked devices
<b>Units</b>	1/10 microns
<b>Type</b>	Axis-Specific
<b>Remembered</b>	Automatically

Assign the specified number to the axis's current position buffer. If no position is specified, 0 is assumed. For non-clocked devices, the unit of measurement is in tenths of microns. This defines the current position to be a specific distance from the origin (0).

### Clocked Devices:

On clocked devices (filter slider, turret, mirrored port switch, etc), you can *change position #1* or you can *change the spacing between positions*.

Change position #1"

1. Remove all axes from either the joystick or a knob (for example, the Z-axis from the Z-knob J Z=0).
2. Assign the clocked device axis to that joystick or knob (for example, the M-axis to the Z-knob J M=22).
3. Use the joystick or knob to adjust the axis to the desired position.
4. Issue the 'here' command to set the position (for example, the M-axis H M=1).
5. Restore (assign and unassign) the desired axes to the joystick / knobs as desired (for example, the M-axis from and the Z-axis to the Z-knob J M=0 Z=22).

Change the spacing between positions:

1. Move to position #2 (for example, the M-axis M M=2).
2. Remove all axes from either the joystick or a knob (for example, the Z-axis from the Z-knob J Z=0).
3. Assign the clocked device axis to that joystick or knob (for example, the M-axis to the Z-knob J M=22).
4. Use the joystick or knob to adjust the axis to the desired position.
5. Issue the 'here' command to set the spacing (for example, the M-axis H M+).
6. Restore (assign and unassign) the desired axes to the joystick / knobs as desired (for example, the M-axis from and the Z-axis to the Z-knob J M=0 Z=22).

Only available on MS2000 version 9.2m and above, TG1000 version 3.28 and above.

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



DOES NOT WORK FOR PIEZOS AND MICROMIRRORS.  
DOES NOT WORK FOR "CLOCKED DEVICES" SUCH AS  
FILTER SLIDERS AND TURRETS.

### 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. *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.

### TGPMT usage

<b>Shortcut</b>	H
<b>Format</b>	HERE [axis] or HERE [axis]=0
<b>Type</b>	Axis-Specific
<b>Remembered</b>	Automatically

On a TGPMT card , this command this used to Zero the PMT signal reported by the [RDADC command](#). When this command is issue , the current PMT signal is saved , and the controller begins subtracting it from the current PMT signal. Users can use this as a background subtract or offset feature.

User can cancel the zeroing by issuing `HERE [axis]=0` . Readings reported by [RDADC command](#) will not be altered anymore.

Only the readings reported by the [RDADC command](#) are altered. 0-4V Analog PMT signal on the BNC connector is not altered.

This is a card wide-settings , both PMT0 and PMT1's reading are altered.

More details [here](#).

### Example

```
h m
:A
```

If the TGPMT card axis char is **m**. Saves the current PMTs readings and starts subtracting them from RDADC commands reported readings.

```
h m=0
:A
```

If the TGPMT card axis char is **m**. Clears the saved PMTs readings and stops subtracting them from RDADC commands reported readings.

2016/03/15 19:56

[commands](#), [tiger](#), [ms2000](#), [tgpmt](#)

## Command:HOME (!)

MS2000 or RM2000 syntax

<b>Shortcut</b>	! (the exclamation point character)
<b>Format</b>	HOME axis [axis] [axis] ...

Tiger syntax

<b>Shortcut</b>	! (the exclamation point character)
<b>Format</b>	HOME axis [axis] [axis] ...
<b>Type</b>	Axis-Specific

Executes a halt and then moves 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.

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

### Example

```
! X Y Z
:A
```

The X, Y and Z 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.

2016/03/15 20:00

[commands](#), [tiger](#), [ms2000](#)

## Command:INFO (I)

MS2000 or RM2000 syntax

<b>Shortcut</b>	I
<b>Format</b>	I axis [Axis] ...

Tiger syntax

<b>Shortcut</b>	I
<b>Format</b>	I axis [Axis] ...
<b>Type</b>	Axis-Specific

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.

2016/03/15 20:03  
[commands](#), [tiger](#), [ms2000](#)

### Command:JOYSTICK (J)

MS2000 or RM2000 syntax

<b>Shortcut</b>	J
<b>Format</b>	JOYSTICK [axis]± or JOYSTICK [axis] = [Manual Input #]
<b>Units</b>	Unitless integer , see table below
<b>Remembered</b>	Using SS Z

Tiger syntax

<b>Shortcut</b>	J
<b>Format</b>	JOYSTICK [axis]± or JOYSTICK [axis] = [Manual Input #]
<b>Units</b>	Unitless integer , see table below
<b>Type</b>	Axis-Specific
<b>Remembered</b>	Using [addr#]SS Z

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



**Table 1: Valid device assignments**

Code	Manual Input Device	Description
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, right side of joypod)
23	F-Wheel	(TG-1000 only, left side of joypod)
100	Add to any other code to make the setting savable with SS Z command	

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.

```
J X? Y?
:A X=2 Y=3
```

The above query and reply shows that the X & Y axes use the X & Y joystick driver.

```
J *?
```

The above command queries all axes.

```
J *
```

The above command clears all axes from the joystick and knobs (TG-1000 only, firmware version 3.10 and later)

**To set a default value that can be saved in non-volatile memory, add 100 to the argument and execute a SAVESET command.**

```
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.

```
J X?
:A X=2
J X=1
:A
RESET
J X?
:A X=2
J X=105
:A
J X?
:A X=5
1SS Z
:A
RESET
J X?
:A X=5
```

In this session, the default manual input device is changed to X-Wheel. Sets the manual input device to whatever the default value is, which is now X-Wheel (5). Saves the settings. After the reset , the manual input device is set on startup its new saved default value, X-Wheel.

2016/03/15 20:13  
[commands](#), [tiger](#), [ms2000](#)

### Command: JSSPD (JS)

MS2000 or RM2000 syntax

<b>Shortcut</b>	JS
<b>Format</b>	JSSPD [X=fast] [Y=slow] [Z=knob_speed] [F=xy_knobs_fast] [T=xy_knobs_slow]
<b>Units</b>	integer
<b>Remembered</b>	Using SS Z

Tiger syntax

<b>Shortcut</b>	JS
<b>Format</b>	[Addr#]JSSPD [X=fast] [Y=slow] [Z=knob_speed] [F=xy_knobs_fast] [T=xy_knobs_slow]
<b>Units</b>	integer
<b>Type</b>	Card-Addressed
<b>Remembered</b>	Using [addr#]SS Z

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, or -100 to -0.1 (negative setting reverses the direction). 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 (not

commonly used on TG-1000).

xy\_knobs\_high and xy\_knobs\_low are used to set the fast and slow speeds for XY\_KNOBS and ZF\_KNOB, which respond to input from the two knobs on the side of the TG-1000 joystick. Note this is different from MS-2000 where the F parameter sets the slow speed and the fast speed is equal to the low speed multiplied by the T parameter raised to the third power. Note that prior to stage firmware version 2.87 the implementation for F and T was like MS-2000.

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

#### Tiger Example

```
1JS X? Y?
:A X=80.000000 Y=3.000000
```

#### MS2000 Example

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

2016/03/15 20:21

[commands](#), [tiger](#), [ms2000](#)

## Command:KA (KA)

#### Tiger Syntax

<b>Shortcut</b>	KA
<b>Format</b>	KA [axis]=n...
<b>Units</b>	unitless integer
<b>Type</b>	Axis Specific
<b>Remembered</b>	Using [Addr#]SS Z

#### MS2000 and RM2000 Syntax

<b>Shortcut</b>	KA
<b>Format</b>	KA [axis]=n...
<b>Units</b>	unitless integer
<b>Remembered</b>	Using SS Z

Adjusts acceleration gain parameter in the servo loop where n is a signed integer. Default n=0. **MOST USERS DO NOT NEED TO USE THIS FUNCTION!**

Note: Before Tiger v3.20 and Whizkid 9.21, the KA or KADC commands changed a variable an overall servo gain paramter, Kadc, which was primarily used for CRISP. This parameter is now set using the command "LR T=Kadc". The KADC command no longer exists. The KA command sets the servo acceleration gain feed forward constant.

```
KA Z?
```

:A Z=0

2016/02/23 19:37

[commands](#), [tiger](#), [ms2000](#)

### Command:KADC (KADC)

Before Tiger v3.20 and Whizkid 9.2l, the KA or KADC commands changed a variable an overall servo gain parameter, KADC, which was primarily used for CRISP. The KADC parameter is now set using the command LR T and the KADC command no longer exists (as of early 2018)

The KA command sets the servo acceleration gain feed forward constant. For a time these two commands overlapped.

2019/04/18 15:51

[commands](#), [led](#), [tiger](#), [ms2000](#), [crisp](#)

### Command:KD

Motorized Axis

MS2000 or RM2000 syntax

<b>Shortcut</b>	KD
<b>Format</b>	KD [Axis]=###
<b>Units</b>	integer
<b>Remembered</b>	Using SS Z

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!**

Tiger syntax

<b>Shortcut</b>	KD
<b>Format</b>	KD [Axis]=###
<b>Units</b>	integer
<b>Type</b>	Axis-Specific
<b>Remembered</b>	Using [addr#]SS Z

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!**

2016/03/15 20:26

[commands](#), [tiger](#), [ms2000](#)

## Command:KI

MS2000 or RM2000 syntax

<b>Shortcut</b>	KI
<b>Format</b>	KI [Axis]=### ...
<b>Units</b>	integer
<b>Remembered</b>	Using SS Z

Tiger syntax

<b>Shortcut</b>	KI
<b>Format</b>	KI [Axis]=### ...
<b>Units</b>	integer
<b>Type</b>	Axis-Specific
<b>Remembered</b>	Using [addr#]SS Z

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!**

2016/03/15 20:27

[commands](#), [tiger](#), [ms2000](#)

## Command:KP

MS2000 or RM2000 syntax

<b>Shortcut</b>	KP
<b>Format</b>	KP [Axis]=### ...
<b>Units</b>	integer
<b>Remembered</b>	Using SS Z

Tiger syntax

<b>Shortcut</b>	KP
<b>Format</b>	KP [Axis]=### ...
<b>Units</b>	integer
<b>Type</b>	Axis-Specific
<b>Remembered</b>	Using [addr#]SS Z

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!**

2016/03/15 20:29

[commands](#), [tiger](#), [ms2000](#)

## Command:KV (KV)

### Tiger Syntax

<b>Shortcut</b>	KV
<b>Format</b>	KV [axis]=n...
<b>Units</b>	unitless integer
<b>Type</b>	Axis Specific
<b>Remembered</b>	Using [Addr#]SS Z

### MS2000 and RM2000 Syntax

<b>Shortcut</b>	KV
<b>Format</b>	KV [axis]=n...
<b>Units</b>	unitless integer
<b>Remembered</b>	Using SS Z

Adjusts motor gain parameter in the servo loop where n is a signed integer. Default depends on motor and leadscrew pitch but is generally within 10% of the ideal value. Ideal value will change with speed setting, especially when AA is far from maximum.

### **MOST USERS DO NOT NEED TO USE THIS FUNCTION!**

```
KV Z?
:A Z=39
```

```
KV Z=40
:A
```

2020/10/01 12:37

[commands](#), [tiger](#), [ms2000](#)

## Command:LCD

### **Supported on MS2000 and RM2000 only**

<b>Shortcut</b>	LCD
<b>Format</b>	LCD "string"
<b>Units</b>	ASCII Characters

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

The quotes are syntactically required. If the quotes are not present, or one is missing, then the command will clear the LCD line. This command is write-only and will not return the contents of the LCD line if queried.

LCD "my super string" causes the last line of the LCD to be updated to: my super string

However, if issued this command: LCD my serial string or LCD ? or just LCD , Then the last line of the

LCD display will be cleared (with no characters present).

2016/03/16 13:22

[commands](#), [ms2000](#)

## Command:LED

On Tiger with Two Axis Card

<b>Format</b>	[Addr#]LED X=[0 to 100]
<b>Type</b>	Percentage between 0 and 100
<b>Remembered</b>	Using [Addr#]SS Z
<b>Firmware Required</b>	STD_XY_LED

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 is 1.3KHz. It's a Card-Addressed command.

The LED command has a slightly different format on a TGLED card. Refer to TGLED card user guide for more details.



**NOTE:** If you are encountering flickering, try adjusting your shutter speed to integer multiples of the PWM period (0.77 ms).

On Tiger and TGLED card

<b>Format</b>	[Addr#]LED X=[1 to 100] Y=[1 or 100] Z=[1 or 100] F=[1 or 100]
<b>Type</b>	Percentage between 0 and 100
<b>Remembered</b>	Using [Addr#]SS Z

This command is "recycled" for a slightly different use in TGLED than for other cards . In the context of a TGLED card this command is used to set the individual brightness percentage of the LEDs connected to the card. Setting the brightness to 0% results in LED to be off. Setting the brightness to 50% results the in the LED being driven with PWM with 50% duty cycle. Setting the percentage to be 100% results in the LEDs to turn on fully.

X sets the brightness for LED connected to channel 1, Y sets the brightness for LED connected to channel 2, Z sets the brightness for LED connected to channel 3, and F sets the brightness for LED connected to channel 4.

Default is 50.

### Example

```
1LED X=10 Y=50 F=0
:A
```

Sets the Brightness of LED connected to Channel #1 to be 10% , Channel #2 to be 50% and Channel #4 to OFF. Brightness of Channel #3 will be unchanged.

```
1LED X? Y? Z? F?
X=10 Y=50 Z=50 F=0 :A
```

Queries the card for Brightness of LEDs connected to Channel #1,#2,#3 and #4

On Tiger with Micro-mirror card

<b>Format</b>	[Addr#]LED X=[output] Y=[switch_time] Z=[laser_mode] R=[Side0_state] T=[Side1_state]
<b>Type</b>	Integer (see below)
<b>Remembered</b>	Using [Addr#]SS Z
<b>Firmware Required</b>	MM_LASER_TTL

The LED commands are used to control the laser outputs of the card (Micro-mirror card backplane connectors, usually output by the Programmable Logic Card or TTL card in early versions). The origin of these commands was for MM\_SPIM firmware, but a separate define MM\_LASER\_TTL was created so that it could be used in other situations as well. This documentation for v2.88+, though the Y parameter was added ~v2.86. In general these only apply when the auxiliary TTL output mode is set to 1 (TTL T) and when the SPIM state machine is not running.

*output (X)*: selects which logical laser is on when the SPIM state machine is not running. The exact hardware output depends on bits 0-2 of the LED Z setting. X=0 means neither laser is on, X=1 means the Side0 laser is on, X=2 means the Side1 laser is on, and X=3 means both lasers are on (LED Y setting applies in this case). Requires TTL T (aux\_IO\_mode) setting to be 1. Setting is overridden during SPIM state machine operation.

*switch\_time (Y)*: sets the switching time in ms between the laser outputs when the SPIM state machine is not running and when both lasers are active per the LED X command. Used to simulate the effect of a passive 50/50 beam splitter, which is particularly useful during alignment. Default is 10ms, cannot be set less than 1ms.

*laser\_mode (Z)*: determines the behavior of the laser TTL outputs, both when using the LED settings and also during the SPIM state machine.

- Bits 0-2 form a code 0-7 selecting how the logical laser outputs are converted to physical TTL outputs. Setting should correspond to the physical hardware connected to the laser TTL outputs. Specifically
  - code 0 for individual laser shutters for the two sides (default until v3.01)
  - code 1 for single laser with side switch (Side0 is laser shutter for both sides and Side1 is high when the second side is active) (default v3.01+)
  - code 2 for Side0 high when the first side of SPIM is active, Side1 high when the second side of SPIM is active
  - code 3 for single laser (Side0) automatically turned on during a FAST\_CIRCLES move
  - codes 4-7 are reserved for future use
- Bits 3-7 are reserved for future use and currently cannot be set



**Side0\_state (R):** can be set to 1 or 0 as shorthand for turning on/off the Side0 laser without affecting the state of the Side1 laser. It is equivalent to querying the logical laser output state (X), changing the LSB, and then setting the output state (X). Introduced in v3.11.

**Side1\_state (T):** can be set to 1 or 0 as shorthand for turning on/off the Side1 laser without affecting the state of the Side0 laser. It is equivalent to querying the logical laser output state (X), changing the 2nd bit, and then setting the output state (X). Introduced in v3.11.

#### On MS2000 or RM2000

<b>Format</b>	LED X=[1 to 100] Y=[0 or 1] Z=[0 or 1] F=[0 or 1]
<b>Type</b>	Percentage between 0 and 100
<b>Remembered</b>	Using SS Z
<b>Firmware Required</b>	LED DIMMER

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.



If you encountering flickering in a live image, try adjusting your shutter speed to avoid aliasing with the LED PWM frequency. The PWM frequency is 1 KHz.

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

#### On MS2000 with Dual LED

<b>Format</b>	LED X=[0 to 100] Y=[0 or 100] R=[0 or 100] T=[0 or 100]
<b>Type</b>	Percentage between 0 and 100
<b>Remembered</b>	Using SS Z
<b>Firmware Required</b>	DUAL_LED

Dual LED is a new two LED driver for MS2000. Instead of using PWM to dim the LED, DUAL LED varies the current applied to dim the LEDs resulting in very little flicker even at high shutter speeds. The brightness of the two LEDs can be independently set using the X and Y arguments of the LED command. At 100% brightness about 900ma is applied. Dual LED have a current limiting feature, user can set the maximum allowed brightness of the two LEDs with R and T arguments. This setting is saved in a non volatile memory on the PCB , so this setting is preserved even after firmware upgrade. Say a LED with maximum rated current is 100ma attached to channel 1. Then setting LED R=10 will limit the maximum applied current to 90ma.

X sets the brightness for LED connected to channel 1,  
 Y sets the brightness for LED connected to channel 2,  
 R sets the maximum brightness for LED connected to channel 1, and  
 T sets the maximum brightness for LED connected to channel 2.

### Example

```
LED X=10 Y=50
:A
```

Sets the Brightness of LED connected to Channel #1 to be 10% , Channel #2 to be 50% .

```
LED X? Y?
X=10 Y=50 :A
```

Queries the card for Brightness of LEDs connected to Channel #1,and #2.

```
LED R=10
:A
LED X=50
:A
LED X?
X=10 :A
```

User sets the maximum brightness of LED#1 to 10%. If the user tries to increase the brightness beyond 10%, its overwritten by maximum brightness settings and limited to 10%.

2016/02/22 17:57

[commands](#), [led](#), [tiger](#), [ms2000](#), [tgled](#), [dual led](#)

### Command:LLADDR (LL)

<b>Shortcut</b>	LL
<b>Format</b>	LLADDR X=xaddr Y=yaddr Z=zaddr LLADDR X? Y? Z?
<b>Remembered</b>	Using SS Z

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.

2016/03/16 13:29

[commands](#), [lowlevel](#), [ms2000](#)

### Command:LOAD (LD)

MS2000 or RM2000 syntax

<b>Shortcut</b>	LD
<b>Format</b>	LOAD [Axis]=### ...
<b>Units</b>	position in 1/10 microns
<b>Remembered</b>	not saved
<b>Firmware Module Required</b>	RING BUFFER

## Tiger syntax

<b>Shortcut</b>	LD
<b>Format</b>	LOAD [Axis]=### ...
<b>Units</b>	position in 1/10 microns
<b>Type</b>	Axis-Specific
<b>Remembered</b>	not saved
<b>Firmware Module Required</b>	RING BUFFER

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 =###, then the current position of the axis is stored in the ring buffer (as of Tiger firmware v2.81 and MS2000 firmware v9.2g). 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 command](#) (see below), or by using a front panel button. The LOAD operation increments the number-of-positions counter accessed using RM X? (see the [RBMODE command](#)). In TG-1000 the ring-buffer is stored and executed on a per-card basis. If positions for one or more axes on one card 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.

The current stage position (for all axes with RING\_BUFFER firmware) may be loaded into the ring-buffer by pressing the Joystick button for 3 seconds and releasing.

CAUTION: If you are using TTL mode 12 (see the [TTL command](#)), the values entered into the ring buffer using the LOAD command represent RELATIVE coordinates, not ABSOLUTE coordinates. You must drive the stage to the appropriate starting position before triggering any ring buffer sequence.

2016/03/16 13:36

[commands](#), [tiger](#), [ms2000](#), [ringbuffer](#)

## Command:LOCK (LK)

This command has slightly different usage for CRISP, Phototrack, and SERVOLOCK\_TTL, and very different use for TGPMT card.

### CRISP usage

#### Tiger Syntax

<b>Shortcut</b>	LK
<b>Format</b>	[Addr#]LK [X] [Y] [Z=lock_offset] [F=code]
<b>Type</b>	Card-Addressed
<b>Remembered</b>	Using [Addr#]SS Z

#### MS2000 and RM2000 Syntax

<b>Shortcut</b>	LK
<b>Format</b>	LK [X] [Y] [Z=lock_offset] [F=code]
<b>Remembered</b>	Using SS Z

The LOCK command without any arguments X, Y, or Z advances to the next system state just as would a short-press of the @ button.

LK X? returns the single character indicating the current CRISP system state as described in the table [CRISP System States](#). For historical reasons, do not use LK X to set the current state, instead use LK F.

LK Y? returns the present value of the focus error which is also shown on the LCD display.

LK Z? returns the present value of the focus error lock\_offset. The offset is automatically determined during calibration and is modified when the command wheel on the controller is used to focus a locked system. The offset is also reset with a >10 sec. press of the @ button. A particular value of lock\_offset may be set using LK Z= lock\_offset.

LK M? will query the value set by the calibration routine and you can use LK M=# to set it manually. See [Saving Calibration and Offsets](#) for more details. This is available on Tiger v3.38+ and MS2000 9.2m+ firmware.

LK F=code will unconditionally set the focus state. Code is the ASCII decimal equivalent for the 'state' character that is displayed on the LCD. For example, to unconditionally enter the 'B' state the command would be LK F=66. Not all states are best entered directly. See the [CRISP system state table](#) for the appropriate ASCII code to enter a particular state gracefully.

```
LK X?
:A W
```

Shows the system is in the WAIT state.

### SERVOLOCK\_TTL usage

#### Tiger Syntax

<b>Shortcut</b>	LK
<b>Format</b>	[Addr#]LK [X] [F=code]
<b>Type</b>	Card-Addressed
<b>Remembered</b>	Using [Addr#]SS Z

The LOCK command without any arguments toggles the SERVOLOCK\_TTL function from active to inactive. As short-press of the @ button will also do unless the firmware build also has CRISP, in which case CRISP takes priority. See [full documentation of SERVOLOCK\\_TTL](#) firmware module.

LK X? returns the single character indicating the current state, which for SERVOLOCK\_TTL is the letter T for enabled and Z for disabled. If CRISP is also present in the firmware module then those states will also appear. LK F=code will unconditionally set the focus state. Use LK F=84 (ASCII letter T) to enable SERVOLOCK\_TTL control and LK F=90 (ASCII letter Z) to disable it when done

## TGPMT usage

<b>Shortcut</b>	LK
<b>Format</b>	[Addr#]LK [X] [Y] or [Addr#]LK [X?] [Y?]
<b>Type</b>	Card-Addressed
<b>Remembered</b>	Using [Addr#]SS Z

This command has a different function on a TGPMT card. Here its used to check the status of PMT (overloaded or not). Then if it is overloaded, issue a reset pulse to clear the overload. Duration of the reset pulse is set with the [RT Y command](#)

[addr#] LK X? , Query **PMT0**'s status. Controller return a **0** if Overloaded , and **1** if not overloaded.

[addr#] LK Y? , Query **PMT1**'s status. Controller return a **0** if Overloaded , and **1** if not overloaded.

[addr#] LK X , Issue a reset pulse to **PMT0**.

[addr#] LK Y , Issue a reset pulse to **PMT1**.

Alternately , the status of the PMTs is also indicated by the LEDs on the TGPMT card (Green is ok , Red is overloaded). And the reset button can be pressed to clear the overload state.

Example

```
7\lock x?
:A 0
```

Query the status of **PMT0** on TGPMT card at address **7** for status. **0** is returned, indication PMT0 is overloaded

```
7\lock x
:A
```

Issue reset pulse to PMT0 on TGPMT card at address 7.

```
7\lock x?
:A 1
```

Overloaded was cleared, Query the status of **PMT0** on TGPMT card at address 7 for status again. **1** is returned, indication PMT0 is NOT overloaded

## On Phototrack systems

<b>Shortcut</b>	LK
<b>Format</b>	LK [X] [Y] [Z=sum_min] [F=quad_order]
<b>Remembered</b>	Using SS Z

LK with no argument performs same action as "@" short press.

LK X performs same action as “@” long press.

LK Y performs same action as “HOME” very long press.

Use *sum\_min* to set the minimum sum-signal level required for tracking the sample. If the sum signal is less than *sum\_min*, tracking will PAUSE.

The *quad\_order* is the relative orientation of the PMT assembly and is normally set during calibration.

2016/02/23 20:02

[commands](#), [tiger](#), [ms2000](#), [crisp](#), [tgpmt](#), [phototrack](#), [servolock ttl](#)

### Command:LOCKRG (LR)

This commands function changes if the system has a Phototrack module or CRISP/SERVOLOCK\_TTL module.

On CRISP systems

Tiger Syntax

<b>Shortcut</b>	LR
<b>Format</b>	[Addr#]LR [X=cal_gain] [Y= objective lens NA] [Z=lock_range] [F=cal_range] [T=loop_gain]
<b>Type</b>	Card-Addressed
<b>Remembered</b>	Using [Addr#]SS Z

MS2000 and RM2000 Syntax

<b>Shortcut</b>	LR
<b>Format</b>	LR [X=cal_gain] [Y= objective lens NA] [Z=lock_range] [F=cal_range] [T=loop_gain]
<b>Remembered</b>	Using SS Z

The LOCKRG command allows the user to control of several system variables. The X parameter, *cal\_gain*, is the gain variable normally obtained from running the calibration sequence. Although not recommended, it can be changed with this command, but it will be reset upon running the calibration sequence.

The Y parameter sets both the *cal\_range* focus depth (LR F) and also the *in\_focus\_mm* range (AFLIM Z) appropriately for the specified numerical aperture of the objective. The computed values can be read and/or overridden using the LR F and AFLIM Z commands respectively.

The Z parameter controls the maximum excursion (in either direction) of the stage from the position where the Lock state was initiated before the system generates an error condition and unlocks. The value *lock\_range* is in units of millimeters. The default value is 1.0 mm.

The F parameter controls the excursion of the stage when going through the calibration sequence. The default value for *cal\_range* is 0.005 mm. Setting the objective's NA using LR Y

will override this value.

The T parameter controls the gain multiplier or loop gain. The default value is 4.

Note: Firmware with a compile date prior to November 2016 used the KADC command to set loop gain. Firmware builds from November 2016 to March 2018 have both KADC and LR T commands which have an identical effect. With LR T the axis character does not need to be specified.

With SERVOLOCK\_TTL

Tiger Syntax

<b>Shortcut</b>	LR
<b>Format</b>	[Addr#]LR [Z=lock_range]
<b>Type</b>	Card-Addressed
<b>Remembered</b>	Using [Addr#]SS Z

The SERVOLOCK\_TTL module uses the LOCKRG Z command to set the maximum excursion. If CRISP is also present then the same setting is shared by both modules.

The Z parameter controls the maximum excursion of the stage from the position where it was initially locked before the system generates an error condition and unlocks. The value lock\_range is in units of millimeters. The default value is 1.0 mm.

On Phototrack system

<b>Shortcut</b>	LR
<b>Format</b>	LR [X=cal_value] [Y=xy_lock_range] [Z=z_lock_range] [F=cal_range]
<b>Remembered</b>	Using SS Z

This command sets range limits for tracking and autofocus systems. For XY tracking systems, the excursion from the point of lock for both the X and Y axes in millimeters is set with the lock\_range value using the Y parameter. If the system encounters a lock\_range or focus\_range limit, servo tracking is terminated.

Cal\_range is the distance in millimeters of the stage movement for automatic calibration of the Tracking or Focus system, set using the F parameter. The result of such a calibration is the cal\_value, which can be set explicitly with the X parameter or queried using LR X?. The tracking parameters can be displayed on the serial terminal using LR Z.

Query: LR X? Y? F? returns the current value of the parameters.

2016/02/23 19:42

[commands](#), [tiger](#), [ms2000](#), [crisp](#), [phototrack](#), [servolock ttl](#)

## Command:LOCKSET (LS)

<b>Shortcut</b>	LS
-----------------	----

<b>Format</b>	LS Z=[focus_trim]
<b>Required Module</b>	AF-DUAL system

The command directly sets the focus\_trim value normally adjusted with the control knob after locking.  
:A is returned upon receipt of the command.

**Example**

```
LS Z?
:A Z=-48
```

returns the current reference value.

2016/03/16 13:47  
[commands](#), [ms2000](#), [afdual](#)

**Command:MAINTAIN (MA)**

The Maintain command has a different function in case of a piezo actuator then a motorized actuator

MS2000 or RM2000 syntax

<b>Shortcut</b>	MA
<b>Format</b>	MAINTAIN [Axis] = [0 to 3]...
<b>Units</b>	integer codes, see table below
<b>Remembered</b>	Using SS Z

Tiger syntax

<b>Shortcut</b>	MA
<b>Format</b>	MAINTAIN [Axis] = [0 to 3]...
<b>Units</b>	integer codes, see table below
<b>Type</b>	Axis-Specific
<b>Remembered</b>	Using [addr#]SS Z

Function for Motorized Actuator (xy stage, z drive etc)

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](#)" or "[PC](#)" command). Another relevant setting is the drift error ("[ERROR](#)" or "[E](#)" command). The actions for various code values are:


<b>code = 0</b>	[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.
<b>code = 1</b>	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.
<b>code = 2</b>	The motor drivers remain on and the servo loop remains active. NB: making a manual move (e.g. joystick) may turn the motors off.



<b>code = 3</b>	Drivers remain on and servos active for the post-move time set by the "WAIT" or "WT" 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.
<b>code = 4</b>	Reserved
<b>code = 5</b>	Servos Disengage. For example with motorized turrets this will allow the user to manually move the turret.

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

Function for Piezo actuator



This feature is only available on Tiger system


This command is “recycled” for a different use in piezo axes than for motor axes. For ADEPT piezo cards sets the maintain code. Currently 2 modes are implemented with the following codes:

MA [axis] =	Mode
0	default
1	Overshoot algorithm

**Mode 0** is the default where the piezo DAC is set and the piezo drive electronics apply a voltage to the piezo to move the piezo to the correct position as measured by the strain gauge.

**Mode 1** is the overshoot algorithm which may reduce the setting time in many circumstances where speed is critical and the user is willing to do some tuning. The piezo DAC is set as if the move were traveling past the actual target according to an overshoot percentage set by the PZ T setting. The idea is to slew more quickly initially. When an exit condition is reached the piezo DAC is set to the desired final position. There are two exit conditions, meeting either one or the other is sufficient:

- (1) the strain gauge indicates that the piezo is at least halfway to the final position or
- (2) the maximum time for the overshoot to be applied set by the PZ R is reached.



Note: At one point this behavior was controlled using the PM command. It was shifted over to MA starting in v3.06 but not in its current form until v3.11. Using this anything besides the default setting with firmware between v3.06 and v3.10 is not recommended.

## Command:MOTCTRL (MC)

MS2000 or RM2000 syntax

<b>Shortcut</b>	MC
<b>Format</b>	MOTOCTRL [Axis]±

Tiger syntax

<b>Shortcut</b>	MC
<b>Format</b>	MOTOCTRL [Axis]±
<b>Type</b>	Axis-Specific

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. When queried, the controller returns values of 1 or 0 representing enabled and disabled respectively.

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.

2016/03/16 14:00

[commands](#), [tiger](#), [ms2000](#)

## Command:MOVE (M)

MS2000 or RM2000 syntax

<b>Shortcut</b>	M
<b>Format</b>	MOVE [Axis]=[units 1/10 microns]...
<b>Units</b>	1/10 microns

Tiger syntax

<b>Shortcut</b>	M
<b>Format</b>	MOVE [Axis]=[units 1/10 microns]...
<b>Units</b>	1/10 microns
<b>Type</b>	Axis-Specific

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.

For rotary THETA stages, the position is in the milli-degrees, so M T=1000 will move to the 1 degree position.

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.

### Example

```
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.

2016/03/16 14:04

[commands](#), [tiger](#), [ms2000](#), [speed](#), [vector](#)

## Command:MOVREL (R)

MS2000 or RM2000 syntax

<b>Shortcut</b>	R
<b>Format</b>	MOVREL [Axis]= [units 1/10 microns]...
<b>Units</b>	1/10 microns

Tiger syntax

<b>Shortcut</b>	R
<b>Format</b>	MOVREL [Axis]= [units 1/10 microns]...
<b>Units</b>	1/10 microns
<b>Type</b>	Axis-Specific
<b>Remembered</b>	Using [addr#]SS Z

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

If the previous move was commanded by serial or TTL then the target position is based on the position that the previous move would ideally have ended up, which might differ slightly from the actual position. This behavior prevents the possibility of accumulating small positioning errors when making sequential relative moves. Events such as manual moves (joystick/wheel), halted moves (by user or via motor error conditions), and automated drift correction moves will change the target

position to the actual position.

The increment to the target position is quantized by encoder counts. For example, with 16 TPI and rotary encoders there are 181590.4 encoder counts per millimeter. If you request 1 um move the encoder target will change by 182 encoder counts. Repeat 600 times and you get a total target move of 109200 encoder counts which is 601.3 um, not 600.0 um. However, if you request for 2um moves then the target will move by 363 encoder counts. Repeated 300 times gives 599.7 um move in total, not 600.0 um.

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.

**Example**

```
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.

2016/03/16 14:09

[commands](#), [tiger](#), [ms2000](#)

**Command: MULTIMV (MM)**

MS2000 or RM2000 syntax

<b>Shortcut</b>	MM
<b>Format</b>	MM [X=radius] [Y= speed] [Z= width] [F=mode_byte]
<b>Remembered</b>	Using SS Z
<b>Firmware Version Required</b>	v8.7+
<b>Firmware Module Required</b>	MULTIAXIS_FUNCTION

Tiger syntax

<b>Shortcut</b>	MM
<b>Format</b>	[addr#]MM [X=radius] [Y= speed] [Z= width] [F=mode_byte] [R?]
<b>Type</b>	Axis-Specific
<b>Remembered</b>	Using [addr#]SS Z
<b>Firmware Module Required</b>	MULTIAXIS_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. Commanded and manual (joystick) moves are not allowed while a multi-axis move is occurring.

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 FAST_CIRCLES	
7	01 Circle 10 Helix 11 Spiral	

This above settings can be saved into non-volatile memory by issuing the `SAVESET` command.

Specifying an argument for the pseudoaxis R in decimal sets the state directly (see table below; the value is simply the decimal representation of the corresponding state character). Note that the firmware expects only certain states to be set by the user (marked as "OK to set" in the table); setting to a different state may yield unpredictable results. Querying the pseudoaxis R value returns the decimal associated with the current state (currently expressed as a float; discard anything after the decimal).

<b>Multi-axis move states (MULTIAXIS_FUNCTION firmware)</b>			
Char	Dec	OK to set?	State
I	73	No	Idle/disabled
S	83	Yes	Starts state machine
P	80	Yes	Stop (goes to idle state after cleanup)
L	76	No	Lead-in move
A	65	No	Accelerating
M	77	No	Main move
m	109	No	Back move (unused?)
F	70	No	Fast circle move
R	82	Yes	Restart move (fast circles only)

## FAST\_CIRCLES

In FAST\_CIRCLES mode, a lookup table is generated internally so the circles can occur very fast assuming the hardware allows fast motion. Using this mode changes the behavior of some parameters. There is no way for the user to enable or disable the FAST\_CIRCLES modifications; the firmware build either includes FAST\_CIRCLES or not (tell by using the BUILD X command).

X parameter (radius) remains the same. Note that the units are millimeters (or degrees for micro-mirror).

Y parameter (speed) is now specified in circles per second. Valid values range from 2 to 1000 with a default of 100 (equivalently a 10 ms period). Most hardware cannot support speeds faster than a few hundred cycles per second.

Z parameter is now the asymmetry ratio. If set to 1.0 (the default) a perfect circle will be generated, if set to 2.0 then the Y axis will move twice as far as the X axis to form an ellipse with major axis twice that of the minor axis. At present there is no way to tilt the ellipse (e.g. phase) but this can be added if needed.

In FAST\_CIRCLES mode the circle is repeated indefinitely regardless of Bit2 of the mode\_byte.

In FAST\_CIRCLES mode the circle is centered at the position whenever it was initiated. To change the offset of the circle, stop the multi-axis move if needed, move to the desired center, and initiate the move (again). (Commanded and manual (joystick) moves are not allowed during any multi-axis moves.)

Changes to settings during a fast circles move will not take effect until the fast circles move is re-initiated.

During a fast circles move the position reported by the WHERE command is the center of the circle, not the instantaneous position.

### Circles

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

### Spirals

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

### Example

```
3MM x=0.02 y=5 z=0.02 f=68
```

```
:A
```

This command sets up the parameters to do a circle pattern for axes on card with address 3 (for MS-2000 omit the initial character 3)

```
3MM x=0.02 y=2 z=0.002 f=196
:A
```

This command sets up the parameters to do a spiral pattern for axes on card with address 3 (for MS-2000 omit the initial character 3)

3MM Initiates the patterns (for MS-2000 omit the initial character 3)

3MM The second time disables the routine (for MS-2000 omit the initial character 3)

2016/03/16 14:36

[commands](#), [tiger](#), [ms2000](#), [multiaxis](#)

## Command:OS

MS2000 or RM2000 syntax

<b>Shortcut</b>	OS
<b>Format</b>	OS [axis] = [distance]...
<b>Units</b>	millimeter
<b>Remembered</b>	Using SS Z

Tiger syntax

<b>Shortcut</b>	OS
<b>Format</b>	OS [axis] = [distance]...
<b>Units</b>	millimeter
<b>Type</b>	Axis-Specific
<b>Remembered</b>	Using [addr#]SS Z

This command sets (or displays) the amount of distance in millimeters to travel to absorb the backlash in the axis' gearing. Analogous to backlash, but will always overshoot the desired position by the set amount and then come back towards the move's origination, whereas backlash always approaches from the same direction. Backlash move, if any, occurs before the overshoot move. Default is 0, which turns the overshoot routines off. When queried will return the actual value instead of the user-set target value

### Example:

```
OS X=.05 Y=0
:A
OS x?
:X=0.049981 A
```

The command in this example will make the controller overshoot any X moves by location 50 microns before moving to the final target, while the overshoot algorithm for the Y axis is disabled.

2016/12/23 14:21

[commands](#), [tiger](#), [ms2000](#)

## Command:PCROS (PC)

Motorized Axis - Drift Error

MS2000 or RM2000 syntax

<b>Shortcut</b>	PC
<b>Format</b>	PCROS [Axis]= [units mm]...
<b>Units</b>	millimeter
<b>Remembered</b>	Using SS Z

Tiger syntax

<b>Shortcut</b>	PC
<b>Format</b>	PCROS [Axis]= [units mm]...
<b>Units</b>	millimeter
<b>Type</b>	Axis-Specific
<b>Remembered</b>	Using [addr#]SS Z


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 TG-1000 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 PCROS value is extremely small, moves may take an excessively long time to complete.

Tunable Lens



<b>Shortcut</b>	PC
<b>Format</b>	PCROS [Axis]= ###...
<b>Units</b>	float
<b>Type</b>	Axis-Specific
<b>Remembered</b>	Automatically

On the Tunable Lens card this command gets and sets a different parameter. Tunable lens diopter changes with temperature , this change varies with current being applied to the lens. So we built a model equation to help us calculate this Diopter per Celsius change. PC command sets the constant in this model equation. [More info here](#)

2016/03/16 14:21

[commands](#), [tiger](#), [ms2000](#), [tlens](#)

### Command:PEDAL (PD)


<b>Shortcut</b>	PD
<b>Format</b>	PEDAL [X=distance in mm] [Y=rate in mm/sec] [Z=multiplier] PEDAL X? Y? Z?
<b>Remembered</b>	Using SS Z
<b>Hardware/Firmware Required</b>	Foot Pedal

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.

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

#### Example

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

2016/03/16 14:58

[commands](#), [ms2000](#)

## Command:PG

### Piezo

<b>Shortcut</b>	PG
<b>Format</b>	PG [axis]=[25 to 5101] (pre v2.84) PG [axis]=[1 to 255] (v2.84 and above)
<b>Units</b>	integer code
<b>Type</b>	Axis-Specific
<b>Remembered</b>	Automatically

PG command is used to set the gain of the feedback stage. 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. It is an Axis specific command. Setting is automatically saved in the non-volatile memory.

The settings set with this command can also be done with PZ commands. One does not have an advantage over another; usage is left to user preference.

**Pre v2.83** a formula was used to convert 25-5101 to 8bit 255. We had rounding off issues and such. We removed the formula so now user can enter the setting directly and thus have more control.

### Tunable Lens

<b>Shortcut</b>	PG
<b>Format</b>	PG [axis]=[1 to 255]
<b>Units</b>	integer code
<b>Type</b>	Axis-Specific
<b>Remembered</b>	Automatically

On the Tunable Lens card , this PG is used to get and set a different parameter. Tunable lens diopter changes with temperature , this change varies with current being applied to the lens. So we built a model equation to help us calculate Diopter per Celsius change as a function of current. PG command sets the coefficient in this model equation. [More info here](#)

2016/03/18 13:41

[commands](#), [tiger](#), [piezo](#), [tlens](#)

## Command:PM

**This is a Tiger only command** , that has different usage for micro-mirrors and piezos.

### Micro-mirror

<b>Shortcut</b>	PM
<b>Format</b>	PM [axis]=[0 or 1] ...
<b>Units</b>	Integer code, 0-1
<b>Type</b>	Axis-Specific

<b>Remembered</b>	Using [addr#]SS Z
<b>Firmware Required</b>	v2.8+

This command is “recycled” for a different use in MicroMirror axes than for piezo. In the context of a MicroMirror axis this command is used to put the axis in internal or external mode.

**1** is external input mode. Mirror is positioned based on analog input voltage. Default mode until v3.10, except for SPIM-enabled systems.

**0** is internal input mode. Mirror can be positioned thru serial command or onboard routines. Default mode for SPIM-enabled systems and v3.10+.

**Example**

```
PM A=1 B=1
:A
```

Puts the axes in external input mode

```
PM A? B?
A=1 B=1 :A
```

Queries the mode of axes

Piezo

<b>Shortcut</b>	PM
<b>Format</b>	PM [axis]=[0 to 3,+,-]
<b>Units</b>	Integer code, 0 to 3 and + and - sign
<b>Type</b>	Axis-Specific
<b>Remembered</b>	Using [addr#]SS Z

PM command sets the ADEPT card in various modes, Open Loop, Closed loop, Internal Input and External Input. In MS-2000 this is set by the "PZ Z" command; in TG-1000 either PM or PZ Z can be used.

PM [axis] =	Mode of Operation
0	TG-1000 input, Closed loop (default)
1	External input, Closed Loop
2	TG-1000 input, Open loop
3	External input, open loop
PM [axis]+	Fast Mode
PM [axis]-	Slow Mode

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 gauges. This is the default mode of operation.

TG-1000 input, in this mode the TG-1000 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 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.

PM [axis]+ : Requires v3.10+ firmware and Rev M5 or later ADEPT card. Switches in a faster more responsive piezo position controller. However it is less stable and prone to oscillation. Suitable for 150um piezos.

PM [axis]- : Requires v3.10+ firmware and Rev M5 or later ADEPT card. Switches in a slower but more stable piezo position controller. Ideal for 300um or 500um piezos and when using heavier payload or samples.

The modes will revert back to default state, i.e. TG-1000 input with Closed Loop when system is powered off. Use the [#Addr]ss z [command](#) to save your preference.

The settings set with this command can also be done with PZ commands. One does not have an advantage over another; usage is left to user preference.

### Tunable Lens

<b>Shortcut</b>	PM
<b>Format</b>	PM [axis]=[0 or 3] ...
<b>Units</b>	Integer code, 0-3
<b>Type</b>	Axis-Specific
<b>Remembered</b>	Using [addr#]SS Z
<b>Firmware Required</b>	v3.19+

This command is “recycled” for a different use in Tunable Lens axes than for piezo. In the context of a Tunable Lens axis this command is used to put the axis in internal or external mode, and also to enable or disable Temperature Compensation. Temperature compensation is only available when tunable lens is in TG-1000 input mode. When using external input(s), make certain that JP1 and/or JP2 have been changed. These jumpers determine whether the BNC connectors are used as TTL Output/Input or as analog inputs. With *Axis Mode 1* and *JP1:2-3 shorted*, the left BNC labeled **TTL OUT** becomes a 0-5v analog input. With *Axis Mode 3* and *JP2:2-3 shorted*, the right BNC labeled **TTL IN** becomes a 0-5v analog input.

PM [axis] =	Mode of Operation
0	TG-1000 input, Temperature Compensation disabled (default)
1	External input, Temperature Compensation disabled (Requires JP1: 2-3 shorted)
2	TG-1000 input, Temperature Compensation enabled
3	External input, Temperature Compensation disabled (Requires JP2: 2-3 shorted)

### Example

```
PM R=1 S=1
:A
```

Puts the axes in external input mode

```
PM R? S?
R=1 S=1 :A
```

Queries the mode of axes

TGGALVO / TGDAC4

<b>Shortcut</b>	PM
<b>Format</b>	PM [axis]=[0-7] ...
<b>Units</b>	Integer code, 0-7
<b>Type</b>	Axis-Specific
<b>Remembered</b>	Using [addr#]SS Z
<b>Firmware Required</b>	v3.2+

For TGGALVO card, the bits 0-2 of the specified code sets the output range (values 0-7 in decimal). Bits 3-7 are reserved for future use.

**Note: in case of SIGNAL\_DAC firmware, this operation is performed with [Command:PR](#) command instead.**

Code (Decimal)	Code (Binary)	Output range
0	000	0V to 2.048V
1	001	0V to 4.096V
2	010	0V to 10.24V
3	011	not supported
4	100	-1.024V to 1.024V
5	101	-2.048V to 2.048V
6	110	-5.128V to 5.128V
7	111	-10.24V to 10.24V

**Example**

```
PM A=2 B=7
:A
```

Puts the output A in 0-10V mode and output B in +/- 10V mode.

```
PM A? B?
A=2 B=7 :A
```

Queries the mode of axes

2016/03/17 19:27

[commands](#), [tiger](#), [piezo](#), [micromirror](#), [tlens](#)

## Command:PR

This is a Tiger only command , that has different usage for MicroMirrors and Piezos

### MicroMirror

<b>Shortcut</b>	PR
<b>Format</b>	PR [axis]=[5 to 10] ...
<b>Units</b>	Integer code, 5 to 10
<b>Type</b>	Axis-Specific
<b>Remembered</b>	Automatically
<b>Firmware Required</b>	v2.83+

This command is “recycled” for a different use in MicroMirror axes than for piezo. In the context of a MicroMirror axis this command is used to set the MicroMirror travel range. Settings is automatically saved into non-volatile memory, however controller needs a system RESET or RESTART for setting to take effect.

<b>PR[Axis Name] =</b>	<b>MicroMirror Range in degrees</b>
5	5
6	6
8 (default)	8
10	10

### Example

```
PR A=5
:A
```

Sets range of A axis as 5 degrees

```
PR A? B?
A=5 B=8 :A
```

Queries the range of axes

### Piezo

<b>Shortcut</b>	PR
<b>Format</b>	PR [axis]=[0 to 8] ...
<b>Units</b>	Integer code, 0 to 8
<b>Type</b>	Axis-Specific
<b>Remembered</b>	Automatically

PR is used to set the piezo travel range. It is an Axis specific command. Setting is automatically saved in the non-volatile memory. Will need a system RESET or RESTART for setting to take effect.

<b>PR [Axis Name] =</b>	<b>Piezo Range in microns</b>
1	50
2	100
3	150
4	200
5	300
6	350
7	500
8	70

### Tunable Lens

<b>Shortcut</b>	PR
<b>Format</b>	PR [axis]=[0 to 3] ...
<b>Units</b>	Integer code, 0 to 3
<b>Type</b>	Axis-Specific
<b>Remembered</b>	Automatically

PR is used to set the Tunable Lens units or user input. It is an Axis specific command. Setting is automatically saved in the non-volatile memory. Will need a system RESET or RESTART for setting to take effect. For more info refer to [Units/Resolution](#) section on TGTLIC card page.

<b>PR [Axis Name] =</b>	<b>Units</b>
1 (default)	[0 to 290mA] with coordinates -32768 to +32768
2	[0 to 290mA] 1/1000 of dpt, range usually between 26000 to 7000
3	[-290mA to +290mA] with coordinates -32768 to +32768

### SIGNAL\_DAC for TGDAC4 / TGGALVO

<b>Shortcut</b>	PR
<b>Format</b>	PR [axis]=[0-7] ...
<b>Units</b>	Integer code, 0-7
<b>Type</b>	Axis-Specific
<b>Remembered</b>	automatically, no SS Z required
<b>Firmware Required</b>	v3.3+

For TGDAC4 card with SIGNAL\_DAC firmware, the bits 0-2 of the specified code sets the output range (values 0-7 in decimal). Bits 3-7 are reserved for future use. **Controller reset or restart is needed for setting to take effect.**

For the SIGNAL\_DAC firmware, the axis units are always in millivolts (0.001V), regardless of the PR setting. For example, when PR H=2, the maximum axis value of H is 10240. When PR H=0, then the maximum axis value of H is 2048.

<b>Code (Decimal)</b>	<b>Code (Binary)</b>	<b>Output range</b>
0	000	0V to 2.048V
1	001	0V to 4.096V
2	010	0V to 10.24V
3	011	not supported

Code (Decimal)	Code (Binary)	Output range
4	100	-1.024V to 1.024V
5	101	-2.048V to 2.048V
6	110	-5.128V to 5.128V
7	111	-10.24V to 10.24V

### Example

```
PR A=2 B=7
:A
```

Puts the output A in 0-10V mode and output B in +/- 10V mode.

```
PR A? B?
:A A=2 B=7
```

Queries the mode of axes

2016/03/17 19:41

[commands](#), [tiger](#), [piezo](#), [micromirror](#), [tlens](#)

## Command:PSG

Piezo

<b>Shortcut</b>	PSG
<b>Format</b>	PSG [axis]=[1 to 255]
<b>Units</b>	integer code, 1 to 255
<b>Type</b>	Axis-Specific
<b>Remembered</b>	Automatically

PSG command argument sets the zero adjust potentiometer of the ADEPT card. Only values between 1 and 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.

The settings set with this command can also be done with PZ commands. One does not have an advantage over another; usage is left to user preference.

Tunable Lens

<b>Shortcut</b>	PSG
<b>Format</b>	PSG [axis]=[1 to 255]
<b>Units</b>	integer code, 1 to 255
<b>Type</b>	Axis-Specific
<b>Remembered</b>	Automatically

On the Tunable Lens card , this command gets and sets a different parameter. Tunable lens



diopter changes with temperature , this change varies with current being applied to the lens. So we built a model equation to help us calculate Diopter per Celsius change as a function of current. PSG command sets the reference temperature in this model equation. [More info here](#)

2016/03/18 13:43

[commands](#), [tiger](#), [piezo](#), [tlens](#)

### Command:PZ

MS2000 or RM2000 syntax

<b>Format</b>	PZ X=[1 to 255] Y=[20 to 5101] Z=[0 to 3] (pre 9.2f) PZ X=[1 to 255] Y=[1 to 255] Z=[0 to 3,+,-] F=[0 to 65000] (9.2f and above)
<b>Units</b>	integer codes
<b>Remembered</b>	X and Y automatically , Z and F Using SS Z

Tiger syntax

<b>Format</b>	[#Addr]PZ X=[1 to 255] Y=[25 to 5101] Z=[0 to 3] F=[0 to 100] T=[0 to 500] (pre v2.83) [#Addr]PZ X=[1 to 255] Y=[1 to 255] Z=[0 to 3,+,-] (v3.11 and above: F=[0 to 65000] R=[0 to 100] T=[0 to 500]) (v2.83-3.10: F=[0 to 100] T=[0 to 500])
<b>Units</b>	integer codes
<b>Type</b>	Card-Addressed
<b>Remembered</b>	Using [addr#]SS Z

The **X** argument sets the zero adjust potentiometer of the ADEPT card. 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. On Tiger controller [PSG command](#) is equivalent. One does not have an advantage over another; usage is left to user preference.

The **Y** argument sets the gain of the feedback stage. 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. Pre MS2000 v9.2f and Tiger v2.83 a formula was used to convert 25-5101 to 8 bit 255. We had rounding off issues and such. We removed the formula so now user can enter the setting directly and thus have more control. On Tiger [PG command](#) is equivalent. One does not have an advantage over another; usage is left to user preference.

The **Z** argument sets the board in various modes. On Tiger controller [PM command](#) is equivalent in firmware v2.8+. One does not have an advantage over another; usage is left to user preference.

<b>PZ Z =</b>	<b>Mode of Operation</b>
0	Controller controlled, Closed loop (default)
1	External input, Closed Loop
2	Controller controlled, Open loop
3	External input, open loop
PZ Z+	Fast Mode
PZ Z-	Slow Mode

The **F** argument (requires Tiger v3.11+ and MS2000 v9.2f) sets the value of the timer for Auto Sleep feature. Units are in minutes. To maximize piezo actuators' lifetime they should be turned off when not in use. Every time the piezo is moved (e.g. commanded move, TTL-triggered move, or with a manual input device like the wheel) the auto sleep timer is reset to 0. When the timer reaches the value set by the **F** argument the sleep state is entered. In the sleep state piezos are moved to the sleep position and the code returned by the **RS+ command** (equivalent to the right status character on MS-2000 LCD screens) changes to **E**. However, the position returned by the **WHERE command** is not changed during sleep. Further, any move will proceed from the pre-sleep position. To disable the auto sleep feature, set the **F** argument to 0. On most firmware builds the default value is 0, i.e. disabled. However on SPIM builds the default value is 5 minutes. To exit sleep without affecting anything else, send the **R <axis> command** to execute a relative move of distance 0. Setting the **F** argument clears the timer but does not exit the sleep state.

#### Additional Tiger Only function

The **R** argument only applies when the piezo maintain code is set to 1. (In firmware between v2.83 and v3.10 it was the **F** argument instead.) It sets the maximum time to move towards the overshoot position, expressed in milliseconds. Refer to the documentation under [MA](#).

The **T** argument only applies when the piezo maintain code is set to 1. It sets the overshoot amount, expressed as a percentage. For example, when set to 100 the piezo will begin the move as if the target position is twice as far away as it really is. Refer to the documentation under [MA](#).

2016/03/18 14:14

[commands](#), [tiger](#), [ms2000](#), [piezo](#)

### Command:PZC

MS2000 or RM2000 syntax

<b>Format</b>	PZC X=[0 or 1] Y=[0,1,2,3] Z=[1 to 100] F=[1 to 100] ,or PZC
<b>Units</b>	integer codes
<b>Remembered</b>	Using SS Z

Tiger syntax

<b>Format</b>	[addr#]PZC X=[0 or 1] Y=[0,1,2,3] Z=[1 to 100] F=[1 to 100] ,or [addr#]PZC
<b>Units</b>	integer codes
<b>Type</b>	Card-Addressed
<b>Remembered</b>	Using [addr#]SS Z

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. Note: Long calibration is

not implemented for ADEPT card with TG-1000. Usage will end in an error.

**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.

2016/03/18 13:50

[commands](#), [tiger](#), [ms2000](#), [piezo](#)

## Command:PZINFO

MS2000 or RM2000 syntax

<b>Format</b>	PZINFO
---------------	--------

Tiger syntax

<b>Format</b>	[addr#]PZINFO
<b>Type</b>	Card-Addressed

PZINFO is a diagnostic command. ASI reserves the right to change the format of the PZINFO command at any point as more diagnostic features are found to be useful.

MicroMirror Example on Tiger

```
3PZINFO
Hdwr REV.E
V0 :24.3 V
HV :143.5 V
V1 :63.7 V
V2 :63.2 V
V3 :63.2 V
V4 :64.9 V
V5 :64.3 V
V6 :65.4 V
I2C Check> DAC[OK] OSC1[OK] OSC2[OK] EEPROM[OK]
Mode> A[IN] B[IN] C[IN] D[IN]
```

Piezo Example on Tiger

```
1PZINFO
Voltage @ Pos1>
```

```
HV   : 147 V
Sout : 4 V
Pzout: 65 V
I2C Check> DAC[OK] SWITCH[OK] DigPot[OK]
ADEPT Rev 0
DigPot> Sgoffset: 110 Gain: 96
Closed Loop
TG1000 IN
HV ENABLE
FAST MODE
SG Offset [OK]
```

### Piezo Example on MS2000

```
PZINFO
Voltage @ Pos1>
HV   : 147 V
Sout : 4 V
Pzout: 65 V
I2C Check> DAC[OK] SWITCH[OK] DigPot[OK]
ADEPT Rev 0
DigPot> Sgoffset: 110 Gain: 96
Closed Loop
TG1000 IN
HV ENABLE
FAST MODE
SG Offset [OK]
```

### PMT example on TGPMT

if TGPMT card address is 7.

```
7pzinfo
Hdwr REV.0
V0 :24.0 V
V1 :15.0 V
Avg: 2
I2C FRAM: OK
PMT0> Gain:  0 , ADC:  13 , BG:  0 , Status: ENABLED
PMT1> Gain:  0 , ADC:  13 , BG:  0 , Status: ENABLED
<LF>
```

2016/03/17 19:33

[commands](#), [tiger](#), [ms2000](#), [piezo](#), [micromirror](#), [tgpmt](#)

## Command:RBMODE (RM)

MS2000 or RM2000 syntax

<b>Shortcut</b>	RM
<b>Format</b>	RBMODE [X=control] [Y=axis_byte] [Z=buffer_pointer] [F=mode_byte]
<b>Remembered</b>	Using SS Z
<b>Firmware Module Required</b>	RING BUFFER

Tiger syntax

<b>Shortcut</b>	RM
<b>Format</b>	[addr#]RBMODE [X=control] [Y=axis_byte] [Z=buffer_pointer] [F=mode_byte]
<b>Type</b>	Card-Addressed
<b>Remembered</b>	Using [addr#]SS Z
<b>Firmware Module Required</b>	RING BUFFER

Provides control of movement and save operations involving the controller's internal 50-position ring-buffer (optionally to 250 positions, contact ASI). The [LOAD command](#) is used to fill the ring buffer.

The command, without any arguments, sets the TTL input interrupt flag and 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:

- a TTL pulse when the appropriate IN0\_mode is selected (See [TTL command](#), IN0\_INT Firmware Module Required).
- a short press and release of the @ button (as long as other special functions are not utilizing the @ button).
- by the RM command without arguments.

RM X? returns the number of positions defined in the ring buffer (Tiger firmware v2.89+ or MS2000 firmware v9.2g+).

**The argument variables are defined as follows**

#### **control**

- 0 - Clears the ring buffer (RING\_BUFFER firmware required)
- 1 - Starts array scan (ARRAY\_MODULE firmware required)

#### **axis\_byte**

1-7: Binary value determines which axes are commanded to move; the same axes' positions are reported using IN0\_mode=5. The ordering of axes is generally the order on the card/controller, with the first axis getting the LSB. The axis ordering can also be ascertained using the Z2B command; axis number 0 gets the LSB, axis 1 gets the 2s position in the binary value, and so on. The axis\_byte setting also applies to the SERVLOCK\_TTL module and TTL-triggered moves.

### buffer\_pointer

sets or reads the pointer to the buffer position for the next move. The buffer pointer is zero-indexed, so its maximum value is the one less than the number of positions in the ring buffer.

### mode\_byte

#### On Tiger v2.81-2.88 these were on X pseudoaxis instead of F

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 the current position to end with a 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. Starting v3.24, this mode works on TGLED card too. When TTL X=21, TGLED cycles thru all LED channels without waiting for a TTL trigger.

Bits 3-7 are reserved.

MSB - read-only, set to 1 if ring buffer is auto playing and 0 otherwise

CAUTION: If you are using TTL mode 12 (see the [TTL command](#)), the values entered into the ring buffer using the [LOAD command](#) represent RELATIVE coordinates, not ABSOLUTE coordinates. You must drive the stage to the appropriate starting position before triggering a ring buffer sequence.

2016/03/16 15:18

[commands](#), [tiger](#), [ms2000](#), [ringbuffer](#), [array](#), [servolock ttl](#)

### Command:RDADC (RA)

#### MS2000 Syntax and Function

<b>Shortcut</b>	RA
<b>Format</b>	RA [X] [Y] [Z] [F]

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.

If the system has [Video Autofocus](#) user can query the focus score with the Z parameter ie RDADC Z?

If the system has a temperature sensor, the user can query the temperature in 1/100 degrees celcius with the T parameter ie RDADC T? If there are two temperature sensors present, one connected to channel-1 and the other connected to channel-2 of an ADEPT Hub (I2C breakout

board), then the M parameter also becomes active, E.G. RDADC T? M? would answer with :A 2565 2389 meaning sensor-1 has read 25.65C and sensor-2 has read 23.89C.

**Example**

```
RA X Y
:A 128 128
```

Shows typical ADC values for a centered joystick.

Tiger and TGPMT Syntax and Function

<b>Shortcut</b>	RA
<b>Format</b>	[addr#]RA [X?] [Y?]
<b>Units</b>	integer
<b>Type</b>	Card-Addressed
<b>Remembered</b>	Not Applicable

On a TGPMT card in a Tiger Controller , this is a Read Only command. It reports PMT signal read thru an ADC onboard the TGPMT card.

**X?** Returns the ADC reading of PMT0

**Y?** Returns the ADC reading of PMT1

[ADC Specification can be found here.](#)

**Example**

If the TGPMT card address was **7**,

```
7 rdadc x? y?
:A 2 1
```

“2” is the ADC reading from PMT0 , and “1” is the ADC reading from PMT1

```
7 ra x? y?
:A 2 1
```

2016/03/16 15:27

[commands](#), [ms2000](#), [tiger](#), [tgpmt](#)

**Command:RDSBYTE (RB)**

MS2000 or RM2000 syntax

<b>Shortcut</b>	RB
<b>Format</b>	RDSBYTE axis [axis] [axis]...

Tiger syntax

<b>Shortcut</b>	RB
<b>Format</b>	RDSBYTE axis [axis] [axis]...
<b>Type</b>	Axis-Specific

Requests the TG-1000 and 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:

<b>Bit 0</b>	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.
<b>Bit 1</b>	0 = The axis is disabled. It can be reenabled 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. 1 = The axis is enabled.
<b>Bit 2</b>	0 = Motor is inactive (off), 1 = Motor is active (on).
<b>Bit 3</b>	0 = Joystick/Knob disabled, 1 = Joystick/Knob enabled
<b>Bit 4</b>	0 = Motor not ramping, 1 = Motor ramping
<b>Bit 5</b>	0 = Ramping down, 1 = Ramping up
<b>Bit 6</b>	Upper limit switch: 0 = open, 1 = closed
<b>Bit 7</b>	Lower limit switch: 0 = open, 1 = closed

**Reply**

: <byte as hexadecimal>

**Example**


```
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.

2016/03/16 15:37  
[commands](#), [tiger](#), [ms2000](#)

### Command:RDSTAT (RS)

MS2000 or RM2000 syntax

<b>Shortcut</b>	RS
<b>Format</b>	RDSTAT axis [axis] [axis]...

Tiger syntax

<b>Shortcut</b>	RS
<b>Format</b>	RDSTAT axis [axis] [axis]...
<b>Type</b>	Axis-Specific

Without any additional characters this is the same as [RDSBYTE](#), except the data is returned in ASCII decimal format.

Beginning with Tiger v2.8+ and MS2000 v9.2e a qualifier can be added to the axis name to change the information reported.

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, D for motor disabled, 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, A for single axis move, T for multi-axis move, P for pause, E for piezo sleep, or a space for no event to report.

#### Example

```
RS X
:A 138
```

```
RS X?
:A N
RS X-
:A S
RS X+
:A M
```

2016/03/16 16:02

[commands](#), [tiger](#), [ms2000](#)

### Command:RELOCK (RL)

<b>Shortcut</b>	RL
<b>Format</b>	RL
<b>Firmware Module Required</b>	CRIFF or AF-DUAL

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.

2016/03/16 17:11

[commands](#), [ms2000](#), [criff](#), [afdual](#)

### Command:RESET (~)

<b>Shortcut</b>	~
<b>Format</b>	RESET

#### Additional Notes about usage on Tiger

RESET is usually a Broadcast command but can be used as a Card-Addressed Command as well. When addressed to a specific card, it resets that card and then the Comm card (required to communicate with the reset card) without affecting other installed cards.

This command causes the controller to do a software reset. A software reset re-initializes all variables back to their pre-assigned values and initializes all positions to 0. Saved settings and system flags are preserved across resets, but not saved positions. Therefore commands like [HM](#) require a power cycle instead of a reset. To re-initialize settings to default, use [SS X](#) before the reset (see documentation for the [SS command](#); note that on Tiger SS is a card-addressed command).

#### Example

```
RESET
:A
```

2016/03/16 17:14

[commands](#), [tiger](#), [ms2000](#)

## Command:RTIME (RT)

### General Usage

#### On Tiger

<b>Shortcut</b>	RT
<b>Format</b>	[Addr#]RT [X=report_time] [Y=pulse_length in ms] [Z=delay_time in ms] [F=num_aves] [T=finish_error_time in ms]
<b>Type</b>	Card-Addressed
<b>Remembered</b>	Using [Addr#]SS Z

#### On MS2000 and RM2000

<b>Shortcut</b>	RT
<b>Format</b>	RT [X=report_time] [Y=pulse_length in ms] [Z=delay_time] [F=num_aves]
<b>Remembered</b>	Using SS Z

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 Y arguments command has a slightly different usage on a TGLED card. Refer to TGLED card user guide for more details.)

The Z argument sets the post-move delay time in ms for sequenced arrays, and/or the delay between ring buffer moves when RB X is set to autoplay (mode 2 or 3). Note that for ring buffer moves the delay time specifies the interval between attempted moves, whereas for sequenced arrays the delay specifies the time between arriving at the desired position and initiating movement to the next position. For ring buffer if the delay time is set to be 0 then the actual time between moves will be the axis loop time (generally 0.25ms times the number of axes, e.g. 1ms for a four axis card).

The F argument sets num\_aves, the power-of-two exponent for the number of samples to be averaged. Used with the CRISP system.

The T argument sets finish\_error\_time, which is the total amount of time that a motorized stage has to spend within the finish error of the target position at the end of a commanded move (PC setting) before the busy flag is cleared and the move is considered complete. Added to Tiger firmware as of v3.34 (Oct 2020). Defaults to 3ms (previous to v3.34 firmware the setting didn't exist but its effective value was 0.5ms).

#### On Tiger with Micro-mirror for SPIM

<b>Shortcut</b>	RT
<b>Format</b>	RT [Z=delay_time] [F=scan_duration] [R=laser_duration] [T=camera_duration]

<b>Units</b>	Milliseconds
<b>Type</b>	Card-Addressed
<b>Remembered</b>	Using SS Z
<b>Firmware Required</b>	MM_SPIM

Sets up timings used in the high-level operation of SPIM state machine coordinated by Micro-mirror card. They are specified in ms with 0.25ms resolution. These commands augment, not replace, the other RTIME parameters applicable to all TG-1000 cards. For SPIM state machine triggered by TTL after arming, these values are usually “locked in” during the arming step (i.e. upon SN X=97).

*delay\_time* is the delay between ring buffer moves just as normal, however it serves an added function: it is also the delay between the receipt of the trigger and the start of the SPIM state machine operation, allowing a systematic delay to be added. Note that resolution is 1 ms for this setting.

*scan\_duration*: sets the duration of each beam scan during SPIM operation. Total beam scan time will be multiplied by the number of scans (NR X). Introduced in v3.14; in v3.13 and earlier the value for SAF <axis> was used instead. Cannot be less than 1 ms.

*laser\_duration*: sets the duration that the laser control output stays high. Cannot be less than 0.25 ms.

*camera\_duration*: sets the duration that the camera trigger output stays high. Cannot be less than 0.25 ms.

All units in milliseconds and are currently rounded to the nearest 0.25ms

On Tiger with MicroMirror and Phototargeting

<b>Shortcut</b>	RT
<b>Format</b>	[Addr#]RTIME [Y=laser_duration] [Z=delay_time]
<b>Units</b>	Milliseconds
<b>Type</b>	Card-Addressed
<b>Remembered</b>	Using [Addr#]SS Z

The **Y** parameter *laser\_duration* sets the time that the laser is turned on in milliseconds, essentially the same as TTL pulse length as described in the main TG-1000 programming manual. The setting applies to both moves initiated by **AJ** as well as to ring buffer moves. Normal moves using **MOVE** or **MOVREL** commands will not turn on the laser. Its value should be between 1 and 65000.

The **Z** parameter *delay\_time* is the delay between ring buffer moves just as normal. However if *delay\_time* is less than (*laser\_duration* + *settle\_delay*) then the ring buffer behavior is unspecified. Its value should be between 1 and 16000.

On Tiger with TGLED

<b>Shortcut</b>	RT
<b>Format</b>	[Addr#]RT Y=[LED ON time on TTL trigger in ms]
<b>Units</b>	Time in millisec between 1 to 65000

<b>Type</b>	Card-Addressed
<b>Remembered</b>	Using [Addr#]SS Z

The RT command's Y argument is "recycled" for a different purpose for the TGLED cards. Here it is used to set the duration the LEDs stay on after a TTL trigger.

Other Behavior and function of RT command have been left unchanged. Refer to the TG-1000 programming manual for more info.

On Tiger with TGPMT

<b>Shortcut</b>	RT
<b>Format</b>	[Addr#]RT Y=[PMT overload reset pulse duration]
<b>Units</b>	Time in millisec between 1 to 65000
<b>Type</b>	Card-Addressed
<b>Remembered</b>	Using [Addr#]SS Z

The RT command's Y argument is "recycled" for a different purpose for the TGPMT cards. Here it is used to set the duration the Reset pulse to clear the PMT from Overload state. Overload reset pulse is generated when LOCK command is issued.

Example

Assuming TGPMT card address is **7**

```
7rt y=100
:A
```

```
7rt y?
:A Y=100.000000
```

On Phototrack systems

<b>Shortcut</b>	RT
<b>Format</b>	RT [X=report_time]
<b>Remembered</b>	Using SS Z

Sets the time interval between report events when using TTL X=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.

To turn ON/OFF serial position logging first set the ttl\_function to serial logging using TTL X=5. Then either RM command without any arguments, or a TTL pulse on the INPUT BNC will toggle the serial reporting function ON or OFF. To change the reporting time interval use RT X=report\_time. Save any changes you wish to keep using SS Z.

With SERVOLOCK\_TTL Function

<b>Shortcut</b>	RT
<b>Format</b>	RT [R= duration threshold]

<b>Remembered</b>	Using SS Z
-------------------	------------

Sets the trigger duration threshold for the SERVOLOCK\_TTL functionality (pulses longer than the threshold are considered “long” = negative move and shorter are considered “short” = positive move. Restricted to units of 0.25 milliseconds. Defaults to 0.75 ms.

2016/02/22 19:30

[commands](#), [led](#), [tiger](#), [ms2000](#), [tgled](#), [crisp](#), [ttl](#), [micromirror](#), [phototargeting](#), [tgpmt](#), [spim](#), [servolock ttl](#)

### Command:RUNAWAY (RU)

MS2000 or RM2000 syntax

<b>Shortcut</b>	RU
<b>Format</b>	RU [axis] = [distance]... (9.2m and later, earlier RU X=n)
<b>Units</b>	millimeter
<b>Remembered</b>	Using SS Z

Tiger syntax

<b>Shortcut</b>	RU
<b>Format</b>	RU [axis] = [distance]...
<b>Units</b>	millimeter
<b>Type</b>	Axis-Specific
<b>Remembered</b>	Using [addr#]SS Z

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. Prior to MS-2000 version 9.2f and TG-1000 version 3.20 a single value applied to all axes (Tiger card-addressed) but afterwards it is axis-specific. Prior to MS-2000 version 9.2m and TG-1000 version 3.20 only integer values were allowed, but afterwards floating point numbers are allowed and reported.

#### Reply

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

MS2000 Example

```
RU X=5
:A
```

Sets runaway sensitivity to 5mm on the X axis for 9.2m and later, sets the sensitivity on all axis for 9.2l and earlier.

Tiger Example

```
1RU X=5
:A
```

Sets runaway sensitivity to 5mm on all axes in card #1 in v3.19 and earlier, would only set the X axis (assuming it's on card #1) on v3.20 and later.

2016/03/16 17:25

[commands](#), [tiger](#), [ms2000](#)

## Command:SAA

MS2000 or RM2000 syntax

<b>Shortcut</b>	SAA
<b>Format</b>	SAA [axis]=### ...
<b>Units</b>	Axis units
<b>Remembered</b>	Using SS Z

Tiger syntax

<b>Shortcut</b>	SAA
<b>Format</b>	SAA [axis]=### ...
<b>Units</b>	Axis units
<b>Type</b>	Axis-Specific
<b>Remembered</b>	Using [addr#]SS Z

This command sets the peak-to-peak amplitude of the pattern. Negative numbers will reverse the direction of the ramp pattern and make the first triangle sweep negative instead of positive.

### Example

```
SAA A=1000
:A
```

Sets 1 degree as the total (peak-to-peak) amplitude for (MicroMirror) axis A.

2016/03/17 19:57

[commands](#), [tiger](#), [ms2000](#), [singleaxis](#)

## Command:SAF

MS2000 or RM2000 syntax

<b>Shortcut</b>	SAF
<b>Format</b>	SAF [axis]=### ...
<b>Units</b>	milliseconds or clock edges
<b>Remembered</b>	Using SS Z

## Tiger syntax

<b>Shortcut</b>	SAF
<b>Format</b>	SAF [axis]=### ...
<b>Units</b>	milliseconds or clock edges
<b>Type</b>	Axis-Specific
<b>Remembered</b>	Using [addr#]SS Z

This command sets the period of the pattern, in units of milliseconds in case of internal clock (default, see SAP) or in number of clock edges for external clock. Note that the triangle pattern and square wave pattern will automatically force themselves to have a period of an even number of milliseconds, even if the user specifies a period of an odd number of milliseconds. When period is set to be 1msec, the resulting behavior is undefined.

**Note:** On MicroMirror the actual waveform output is modified by the filter (tunable with BACKLASH), especially when the period is set to be less than 10msec.

## Example

```
SAF A=10
:A
```

Sets up the period of the pattern at 10msec or 100Hz.

2016/03/17 20:03

[commands](#), [tiger](#), [ms2000](#), [singleaxis](#)

## Command:SAM

## MS2000 or RM2000 syntax

<b>Shortcut</b>	SAM
<b>Format</b>	SAM [axis]=### ...
<b>Units</b>	Integer code, 0-3 (see below)
<b>Remembered</b>	Using SS Z

## Tiger syntax

<b>Shortcut</b>	SAM
<b>Format</b>	SAM [axis]=### ...
<b>Units</b>	Integer code, 0-3 (see below)
<b>Type</b>	Axis-Specific
<b>Remembered</b>	Using [addr#]SS Z

Sets the single-axis mode according to the integer code.

Code	Meaning
0	Puts single-axis mode in idle state (i.e. stops it if running)
1	Puts the single-axis mode in active state (i.e. starts generating the pattern)



Code	Meaning
2	Arms the trigger; version 3.29 and prior the routine is free running after the TTL trigger. Version 3.31 and later, the routine only cycles once, then waits again for another TTL trigger.
3	Makes the single-axis mode active and restarts the pattern of any other axis on the same card so they will be synchronized

**Example**

```
SAM A=0
:A
```

Disables the routine for the A axis

```
SAM A=1
:A
```

Enables the routine for the A axis

```
SAM B=2
:A
```

Arms the routine for the B axis. The routine for B will start running on receipt of a TTL pulse. Version 3.29 and prior the routine is free running after the TTL trigger. Version 3.31 and later, the routine only cycles once, then waits again for another TTL trigger.

Works with [TTL X=30](#) mode.

```
SAM B=3
:A
```

Enables the routine for the B axis. Also the routine for A axis is reset, so they will both run in sync

2016/03/17 20:05

[commands](#), [tiger](#), [ms2000](#), [singleaxis](#)

**Command:SAO**

MS2000 or RM2000 syntax

<b>Shortcut</b>	SAO
<b>Format</b>	SAO [axis]=### ...
<b>Units</b>	Axis units
<b>Remembered</b>	Using SS Z

Tiger syntax

<b>Shortcut</b>	SAO
<b>Format</b>	SAO [axis]=### ...
<b>Units</b>	Axis units
<b>Type</b>	Axis-Specific

**Remembered** Using [addr#]SS Z

This command sets the position of the center position of the single-axis pattern. For example, if the offset is 1000 and the peak-to-peak amplitude is set to 1000, the pattern will go between positions 500 and 1500.

Note that manual moves (e.g. with joystick or wheels) while a single-axis pattern is being generated automatically adjust the single-axis offset value, such that this command may not be needed.

As of Tiger firmware v2.82, using the “+” operator instead of specifying a position will store the current position to the offset for the specified axis. For example, SA0 A+ set the position of axis A to the center position of the single-axis pattern

**Example**

```
SA0 A=1000
:A
```

Sets offset of the offset of the pattern to be 1 degree for (MicroMirror) axis A.

2016/03/17 20:18

[commands](#), [tiger](#), [ms2000](#), [singleaxis](#)

**Command:SAP**

MS2000 or RM2000 syntax

<b>Shortcut</b>	SAP
<b>Format</b>	SAP [axis]=### ...
<b>Units</b>	Integer code, 0-255 (see below)
<b>Remembered</b>	Using SS Z

Tiger syntax

<b>Shortcut</b>	SAP
<b>Format</b>	SAP [axis]=### ...
<b>Units</b>	Integer code, 0-255 (see below)
<b>Type</b>	Axis-Specific
<b>Remembered</b>	Using [addr#]SS Z

This command sets the type of pattern to generate and configures the clocks. The parameter is a bit-mapped number that determines the characteristics of the motion, with the lowest bits determining the type of pattern. The code is interpreted according to the following table:

Bit	Clear	Set
7	Internal Clock	External clock on TTL input
6	Polarity of Clk, positive edge	Polarity of Clk, negative edge
5	No TTL out	TTL out
4	Polarity of TTL out, active high	Polarity of TTL out, active low

Bit	Clear	Set
3	reserved	Reserved
2-0	000 Ramp/sawtooth (code 0) 001 Triangle (code 1) (period always even number of msec) 010 Square wave (code 2) (period always even number of msec) 011 Sine wave (code 3)	

**Example**

```
SAP r=0
:A
```


Sets up for ramp (sawtooth) pattern, running off internal clock with no TTL outs

```
SAP r=129
:A
```

Sets up for triangle pattern, running off positive edge external TTL clock with no TTL outs.

```
SAP r=161
:A
```

Sets up for triangle pattern, running off positive edge external TTL clock with TTL outs. A 250usec pulse is put out at the start of the pattern.



Serial command `<Card Addr#>TTL Y=22` will route the TTL pulses generated when BIT5 is set, to the TTL OUT0 port. Available only in firmware version 3.17 and above.

2016/03/17 20:14

[commands](#), [tiger](#), [ms2000](#), [singleaxis](#)

**Command:SAVEPOS (SP)**

MS2000 or RM2000 syntax

<b>Shortcut</b>	SP
<b>Format</b>	SP [X=inhibit]
<b>Units</b>	0 or 1 only
<b>Remembered</b>	Using SS Z

Tiger syntax

<b>Shortcut</b>	SP
<b>Format</b>	[addr#]SP [X=inhibit]
<b>Units</b>	0 or 1 only
<b>Type</b>	Card-Addressed

**Remembered** Using [addr#]SS Z

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:** In MS2000 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.

**Note 3:** OK is not transmitted in TG-1000

2016/03/16 18:05  
[commands](#), [tiger](#), [ms2000](#)

**Command:SAVESET (SS)**

MS2000 or RM2000 syntax

<b>Shortcut</b>	SS
<b>Format</b>	SAVESET [X][Y][Z]

Tiger syntax

<b>Shortcut</b>	SS
<b>Format</b>	[addr#]SAVESET [X][Y][Z]
<b>Type</b>	Card-Addressed

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

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

**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 MS2000 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

#### MS2000 example

```
SS Z
:A
```

Saves current settings to flash memory.

#### Tiger example

```
1SS Z
:A
```

Saves Card#1 's current settings to flash memory.

2016/03/16 17:35

[commands](#), [tiger](#), [ms2000](#), [sequencer](#)

## Command:SCAN (SN)

### MS2000 or RM2000 syntax and Function

<b>Shortcut</b>	SN
<b>Format</b>	SCAN [X=scan_axis for X] [Y=scan_axis for Y] [Z=scan_axis for Z] [F=pattern]
<b>Units</b>	Integer
<b>Remembered</b>	Using SS Z
<b>Firmware Required</b>	SCAN or ARRAY

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. Note this is different from the Tiger meaning of pseudoaxes X/Y/Z.

The scan pattern may be set to 0 for RASTER scans or 1 for SERPENTINE scans. This setting defaults to RASTER for firmwares with the SCAN MODULE included and SERPENTINE for firmwares with the ARRAY MODULE included. When the number of lines (SCANV Z) is set to the default of 1 then the behavior is the same for both raster and serpentine.

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

Tiger (micro-mirror) syntax

<b>Shortcut</b>	SN
<b>Format</b>	[addr#]SCAN [X=state]
<b>Units</b>	Integer
<b>Type</b>	Card-Addressed
<b>Remembered</b>	Using [addr#]SS Z
<b>Firmware Required</b>	MM_SPIM

Without arguments, starts or arms the SPIM state machine or terminates it if running or armed (starts state machine execution for Micro-mirror, puts in arm state for piezo). By so doing, any active single-axis functions will be stopped and the SPIM positions/steps will be calculated according to the active parameters (e.g. SAA, SAO, NR, NV, RT). Specifying an argument for the pseudo-axis X in decimal sets the state directly (see table below; the value is simply the decimal representation of the corresponding state character). Note that the firmware expects only certain states to be set by the user (marked as "OK to set" in the table); setting to a different state may yield unpredictable results. Querying the pseudo-axis X value returns the character associated with the current state.

<b>Micro-mirror SPIM states (MM_SPIM firmware)</b>			
<b>Char</b>	<b>Dec</b>	<b>OK to set?</b>	<b>State</b>
I		No	Idle/disabled
S	83	Yes	Starts main acquisition state machine
a	97	Yes	Arm for trigger (goes to state 'A')
A		No	Armed and waiting
T	84	Yes	Trigger from state 'A' (v3.37+)
P	80	Yes	Stop (goes to state 'I')
M		No	In middle of sheet (executing per-sheet scan/camera/laser state machines)
s		No	Starting sheet
c		No	Incrementing sheet
R		No	Starting side
y		No	Delay between sides
Y		No	Delay between repeats

Note: Other undocumented states may be used during SPIM state machine execution.

Tiger (motorized) syntax

<b>Shortcut</b>	SN
<b>Format</b>	[addr#]SCAN [X?] [Y=fast_axis_id] [Z=slow_axis_id] [F=pattern]
<b>Units</b>	Integer
<b>Type</b>	Card-Addressed
<b>Remembered</b>	Using [addr#]SS Z
<b>Firmware Required</b>	SCAN or ARRAY

Note multiple small changes in usage from MS-2000 command set.

Specifying an argument for the pseudoaxis X in decimal sets the state directly (see table below; the value is simply the decimal representation of the corresponding state character). Note that the firmware expects only certain states to be set by the user (marked as "OK to set" in the table); setting to a different state may yield unpredictable results. Querying the pseudoaxis X value returns the character associated with the current state.

Specify the cards axis ID as the `fast_axis_id` or `slow_axis_id` (axis IDs are obtainable using Z2B query; they are 0 and 1 for X and Y axes respectively on a two-axis motor card). If `slow_axis_id` is set to 9 then a true single-axis scan will be executed (not even anti-backlash moves on the slow axis); if `slow_axis_id` is specified but the slow axis start and stop positions (NV X and NV Y) are equal then a single-axis scan will execute but the slow axis position will be checked at the scan turnaround points.

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.

Scan states (SCAN_MODULE firmware)			
Char	Dec	OK to set?	State
I		No	Idle/disabled
S	83	Yes	Starts state machine
P	80	Yes	Stop (goes to idle state after cleanup)
a		No	Waits until slow axis move complete
b		No	Starts move along fast axis
c		No	Waits for fast axis move to finish
d		No	Starts serpentine slow axis move
e		No	Waits until serepentine move complete
f		No	Used in XF_SPIM only
g		No	Waits until retrace is complete
h		No	Scan complete, cleans up

2016/03/16 18:26

[commands](#), [tiger](#), [ms2000](#), [scan](#), [array](#)

### Command:SCANR (NR)

MS2000 or RM2000 syntax

<b>Shortcut</b>	NR
<b>Format</b>	SCANR [X=start] [Y=stop] [Z=enc_divide] [F= #_pixels]
<b>Units</b>	X and Y in mm, Z and F as integer
<b>Remembered</b>	Using SS Z
<b>Firmware Required</b>	SCAN

Tiger micro-mirror syntax

<b>Shortcut</b>	NR
-----------------	----

<b>Format</b>	[addr#]SCANR [X=scans_per_slice] [Y=slices_per_volume] [Z=SPIM_mode] [F=volume_repeats] [R=slice_repeats]
<b>Units</b>	Integer
<b>Type</b>	Card-Addressed
<b>Remembered</b>	Using [addr#]SS Z
<b>Firmware Required</b>	MM_SPIM

Sets up the high-level operation of the SPIM state machine coordinated by the Micro-mirror card

*scans\_per\_slice* (**X**): sets the number of one-way beam scans in each slice (recall the slice corresponds to one image). Minimum value is 1.

*slices\_per\_volume* (**Y**): sets the number of slices (or images) in each volume. No facility exists to make it different for the two sides, though in principle it is possible. Minimum value is 1.

*SPIM\_mode* (**Z**): sets a byte (by assigning a decimal) with the functions below. The default value is 2 (usual diSPIM, no special functionality).

- 2 LSBs correspond to single-sided vs. double-sided and the specified start side according to the following
  - 3 for diSPIM starting on opposite side
  - 2 for usual diSPIM (default)
  - 1 for usual iSPIM
  - 0 for iSPIM on opposite side
- Bits 2-3 were laser output mode in v2.85-v2.87; for v2.88+ this functionality is instead controlled by LED Z laser mode, bits 0-2.
- Bit 2 is set to disable micro-mirror moving to home position when other side is active during the SPIM state machine (i.e. rely completely on laser-based blanking while reducing micro-mirror movements). Default is unset (home move enabled). (v2.89+)
- Bit 3 is set to disable piezo moving to illumination position (home). Default is unset (piezo home enabled). (v2.89+)
- Bit 4 is set to alternate sides after each piezo/slice position (for interleaved stage scan). Note that piezo trigger signals will continue, but this is OK for the stage scan situation when the piezos' SAA value is 0 and either the piezo's SAO position is the same as the offset or else bit 3 is set. Default is unset (not alternating sides). (v3.09+).
- Bit 5 is set to alternate the beam scan direction between sweeps (either between slices or within same slice if the number of line scans per slice is more than 1). Default is unset (not alternating direction). Before v3.14 this was set using the LSB of the SAP setting. (v3.14+).
- Planned but not yet implemented: Bit 6 is set to add one extra camera trigger at the end of each side. Use this to accommodate "synchronous" or "overlap" camera mode without requiring an entire additional slice. Default is unset (no extra camera trigger). Proper operation requires the side delay (NV Y) be longer than the sum of the camera delay (NV T) and the camera duration (RT T). Because this occurs during the side switch time the total acquisition time is only increased by the time required for the final camera trigger.
- Bit 7 is reserved for future use

*volume\_repeats* (**F**): sets the number of volumes to be collected per trigger event (two sides count as a single volume). Minimum value is 1.



## Tiger (motorized) syntax

<b>Shortcut</b>	NR
<b>Format</b>	[addr#]SCANR [X=start] [Y=stop] [Z=enc_divide] [F= #_pixels] [R=retrace_speed]
<b>Units</b>	X and Y in mm, Z and F as integer, R as percentage (0-100)
<b>Type</b>	Card-Addressed
<b>Remembered</b>	Using [addr#]SS Z
<b>Firmware Required</b>	SCAN

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. Applicable to ENC\_INT only.

On TG-1000 v3.30 and later use retrace\_speed to specify the speed of the retrace move as a percentage of the max speed (decimal value between 0 and 100). The default value of 67 was the hardcoded value previously. This feature can be added to MS-2000 on customer request.

2016/03/16 18:29

[commands](#), [tiger](#), [ms2000](#), [scan](#)

**Command:SCANV (NV)**

## MS2000 or RM2000 syntax

<b>Shortcut</b>	NV
<b>Format</b>	SCANV [X=start] [Y=stop] [Z=number_of_lines] [F=overshoot_factor]
<b>Units</b>	X and Y in mm, Z in integer, F in positive real
<b>Remembered</b>	Using SS Z
<b>Firmware Required</b>	SCAN

## Tiger micro-mirror SPIM syntax

<b>Shortcut</b>	NV
<b>Format</b>	[addr#]SCANV [X=scan_delay] [Y=side_delay] [Z=repeat_delay] [F=scan_settle_time] [R=laser_delay] [T=camera_delay]
<b>Units</b>	X, Y, Z, F, R and T are in milliseconds
<b>Type</b>	Card-Addressed
<b>Remembered</b>	Using [addr#]SS Z
<b>Firmware Required</b>	MM_SPIM

Sets up various delays used in the high-level operation of SPIM state machine coordinated by Micro-mirror card. The delays are specified in ms with 0.25ms resolution. The lower limit is 0.0ms and the upper limit is a bit more than 16 seconds for all except repeat\_delay which can

be over a day.

*scan\_delay* (**X**): sets the delay between the start of the slice and when the beam scan begins.

*side\_delay* (**Y**): sets the delay between the start of a side and when slices start. Defaults to 50 ms in v3.14+ (in v3.13- default was 0). In v3.14+ cannot be less than 2.0 ms. It is highly recommended to use a value of at least 10 ms; the signal for the piezo to move to illumination position takes 2.5 ms to send and the piezo has a mechanical response time (typ. 10 ms for 90% settling). In most cases even more time should be allowed for any vibrations resulting from the piezo move to settle, e.g. a typical value of *side\_delay* is 50 ms or 100 ms.

*repeat\_delay* (**Z**): sets the delay after one volume (either one or two sides) before the next one begins. In v3.14+ cannot be less than 1 ms.

*scan\_settle\_time* (**F**): (v3.14+) sets the amount of time before the scan start that the scanned axis will reach its initial position; before that it will ramp smoothly from the previous position to the initial position. Defaults to 1 ms. If the value of *scan\_settle\_time* is equal to or greater than the value of *scan\_delay* there will be an abrupt transition at the corresponding point. Such an abrupt transition can lead to undesired scanner ringing and happened in all cases prior to firmware v3.14.

*laser\_delay* (**R**): sets the delay between the start of the slice and when the laser control output goes high.

*camera\_delay* (**T**): sets the delay between the start of the slice and when the camera trigger output goes high.

### Tiger motorized stage syntax

<b>Shortcut</b>	NV
<b>Format</b>	[addr#]SCANV [X=start] [Y=stop] [Z=number_of_lines] [F=overshoot_time] [T=scan_overshoot]
<b>Units</b>	X and Y in mm, Z in integer, F in ms
<b>Type</b>	Card-Addressed
<b>Remembered</b>	Using [addr#]SS Z
<b>Firmware Required</b>	SCAN

X, Y, and Z parameters set up the slow-scan (vertical) start and stop positions, with the position values expressed in millimeters, and the number of lines. 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. If the start and stop values are identical then a 1-D scan will occur, repeated *number\_of\_lines* times.

The F and T parameters pertain to the fast-scan (horizontal scan) motion, and there is a difference between the behavior of the F parameter on TG-1000 vs. MS-2000.

On MS-2000, *overshoot\_factor* (F) sets the amount of extra motion to account for the acceleration ramp at the start and stop of the trace. An *overshoot\_factor*=1.0 (default) 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.

On TG-1000, overshoot\_time (F) sets an additional settling time in ms for the stage velocity to settle before reaching the start position (beyond the always-required ramp time set by the AC command). Thus the time required between scan line initiation and reaching the start position is given by summing the AC time and the NV F time. The same delay occurs after the stop position except for raster scans in firmware v3.20 and higher in which case the after-stop overshoot time is capped at 10ms. The default value is 50ms.

The T parameter was partially implemented for TG-1000 firmware versions 3.17 and 3.18, absent in 3.19, and then present in 3.20 and greater. It is intended mostly for scan-optimized stages that have a significant amount of physical backlash. The default value is 0.02 when the SCAN\_OPTIMIZED define is enabled and 0 otherwise. If the value is non-zero there are several changes to the scan operation: (1) There is an extra overshoot move performed (with amplitude specified by the parameter) before any scan move in either direction, which ensures that the physical backlash is removed correctly before beginning each scan pass. (2) Before the scan moves begin, an initialization move to the center of the range is made to ensure that the overshoot move happens correctly. (3) When the scan moves are complete, the stage moves to the center position (otherwise behavior is to move to the start position).

2016/03/16 18:33

[commands](#), [tiger](#), [ms2000](#), [scan](#)

### Command:SECURE

This command is used to lock and unlock the MicroServo(U\_SERVO\_LK) and the Solenoid based (SOL\_LK) lock inserts. SECURE command has a bit more functionality in case of Solenoid Lock insert.

For Micro Servo Lock Insert

<b>Shortcut</b>	SECURE
<b>Format</b>	SECURE [X= p]
<b>Remembered</b>	Using SS Z
<b>Hardware/Firmware Module Required</b>	Micro Servo Lock Insert and U_SERVO_LK

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
:A
```

fully opens lever

```
SECURE X=0.25
:A
```

Closes lever for typical well plate

SECURE  
:N-3

Error at axis required

SECURE Y=0  
:N-2

invalid axis

SECURE X?  
:N-2

invalid operation

For Solenoid lock insert

MS2000 or RM2000 syntax and Function

<b>Shortcut</b>	SECURE
<b>Format</b>	SECURE [X= 0 or 1] [Y=0 to 99] [Z=0 to 99] [F=0 to 255] [T=0 to 65000]
<b>Remembered</b>	Using SS Z
<b>Hardware/Firmware Module Required</b>	Solenoid Lock Insert and SOL_LK

←-

Tiger syntax and Function

<b>Shortcut</b>	[addr#]SECURE
<b>Format</b>	[addr#]SECURE [X= 0 or 1] [Y=0 to 99] [Z=0 to 99] [F=0 to 255] [T=0 to 65000]
<b>Remembered</b>	Using [addr#]SS Z
<b>Hardware/Firmware Module Required</b>	Solenoid Lock Insert and SOL_LK

←-

With inserts equipped with Solenoid lock mechanism, this command is used to lock or unlock samples on the stage.

**X** argument accepts either “0” or “1”. “0” is the locking command , and “1” is the unlocking command. The Solenoid use no power when in “0” or lock position , so this is the default and the controller's initial state.

**Y** arguments is a percentage of power briefly applied to the solenoid to pull the lever back and unlock the wellplate. Set by factory, we recommend that this setting not be adjusted unless suggested by ASI support.

**Z** arguments is a percentage of power applied to the solenoid to keep it unlock. After unlocking, the solenoid needs very little power to keep the lever pulled back

and keep the well plate unlocked. Set by factory, we recommend that this setting not be adjusted unless suggested by ASI support.

**F** argument sets the auto lock time, units are in minutes. When in unlock position , the solenoid is consuming power, over time solenoid will heat up and may damage it. There is a auto locking timer , Y sets the maximum time the solenoid stays unlocked , after which the controller auto locks. Default is 5 min . This feature can be disabled by setting Y as "0", this is not recommended.

**T** argument, units are in millisecc. This arguments sets the amount of time higher power (Y arguments) needs to be applied to unlock the wellplate. After that lower power (Z arguments) is applied to keep the wellplate unlocked. Set by factory, we recommend that this setting not be adjusted unless suggested by ASI support.



Note 1: Solenoid only consumes and dissipates power when in unlock state. Over time the heat generated by this power dissipation may damage the solenoid. So only unlock when needed.

Note 2: TTL Out mode must be set to 9 ie [TTI Y=9](#) . This give control of the TTL out connector to Secure command .

### MS2000 Example:

```
SECURE X=1
:A
```

fully opens lever, unlock state

```
SECURE X=0
:A
```

Closes lever, lock state

```
SECURE
:N-3
```

Error at axis required

```
SECURE X=1 F=2
:A
```

lever unlocks, and will auto lock after 2 mins

```
SECURE X?
X=1 :A
```

reply 1 indicates lever is in unlock state.

2016/03/16 19:21

[commands](#), [ms2000](#), [tiger](#)

## Command:SETHOME (HM)

MS2000 or RM2000 syntax

<b>Shortcut</b>	HM
<b>Format</b>	HM [axis]=[position in mm]...
<b>Units</b>	millimeter
<b>Remembered</b>	Automatically
<b>Required MS2000 Firmware Version</b>	8.0+

Tiger syntax

<b>Shortcut</b>	HM
<b>Format</b>	HM [axis]=[position in mm]...
<b>Units</b>	millimeter
<b>Type</b>	Axis-Specific
<b>Remembered</b>	Automatically

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](#). The home position defaults to a large positive number far exceeding the mechanical limits of the system, or else with the upper limit for DAC devices including piezos, micro-mirror, and tunable lenses.

HM [axis]+ will set the home position to be the current position. Restore the default home position by executing HM [axis]-.

### 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.

2016/03/16 19:32

[commands](#), [tiger](#), [ms2000](#)**Command:SETLOW (SL)**

MS2000 or RM2000 syntax

<b>Shortcut</b>	SL
<b>Format</b>	SETLOW [axis]=[position in mm]...
<b>Units</b>	millimeter
<b>Remembered</b>	Automatically

Tiger syntax

<b>Shortcut</b>	SL
<b>Format</b>	SETLOW [axis]=[position in mm]...
<b>Units</b>	millimeter
<b>Type</b>	Axis-Specific
<b>Remembered</b>	Automatically

This command sets/displays the lower firmware limit for an axis. The limit is considered a fixed hardware locations and are adjusted properly when the controller's coordinate system is altered with the [HERE](#) or [ZERO](#) commands. The limit positions are automatically remembered and recalled through a power cycle and do not need to be saved using the [SAVESET](#) command.

SL [axis]+ will set the lower limit to be the current position. Restore the default limit by executing SL [axis]-. The +/- operand syntax is supported as of Tiger firmware v2.8 and MS-2000 firmware as of roughly 2013.

**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 greater than the value for SETUP, then the controller will operate incorrectly. See also Note 2.

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

**Note 3:** For clocked devices (e.g. filter sliders, turrets) the lower limit is always 1.

2016/03/16 19:37

[commands](#), [tiger](#), [ms2000](#)

### Command:SETUP (SU)

MS2000 or RM2000 syntax

<b>Shortcut</b>	SU
<b>Format</b>	SU [axis]=[position in mm]...
<b>Units</b>	millimeter
<b>Remembered</b>	Automatically

Tiger syntax

<b>Shortcut</b>	SU
<b>Format</b>	SU [axis]=[position in mm]...
<b>Units</b>	millimeter
<b>Type</b>	Axis-Specific
<b>Remembered</b>	Automatically

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=###](#)), upper limit settings must be negative values, and lower limit settings must be positive values.

**Note 3:** For clocked devices (e.g. filter sliders, turrets) the upper limit is always the number of positions. Querying the value is useful for determining how many positions.

**Note 4:** As of Tiger firmware v2.8, SU [axis]+ will set the upper limit to be the current position. Restore the default limit by executing SU [axis]-.

2016/03/16 19:43

[commands](#), [tiger](#), [ms2000](#)

### Command:SI

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

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

Linear Encoder and SEARCH INDEX

MS2000 or RM2000 syntax

<b>Shortcut</b>	SI
<b>Format</b>	SI [axis]=[position in 1/10 microns]...
<b>Units</b>	1/10 microns
<b>Firmware Version Required</b>	v8.4+



## Tiger syntax

<b>Shortcut</b>	SI
<b>Format</b>	SI [axis]=[position in 1/10 microns]...
<b>Units</b>	1/10 microns
<b>Type</b>	Axis-Specific

This command searches for the physical centers of the stage and marks it with a user inputted value. Software limits are reset to default. Note, if the command is rerun again it will fail and print the "N-5" error. To avoid this, move the axis off-center, zero the position and then issue the command.

**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.

```
SI X? Y?
:A X=0 Y=0
```

In this example the X and Y axes are being queried for the current setting of the axes centers. The response is what they have previously been set to (not necessarily 0).

## Rotary Encoder and SEEK LIMITS

## MS2000 or RM2000 syntax

<b>Shortcut</b>	SI
<b>Format</b>	SI [axis] = [1 or -1]...
<b>Units</b>	1/10 microns
<b>Firmware Version Required</b>	v8.8e+

## Tiger syntax

<b>Shortcut</b>	SI
-----------------	----

<b>Format</b>	SI [axis] = [1 or -1]...
<b>Units</b>	-
<b>Type</b>	Axis-Specific

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

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 after issuing the command. If an error occurs during execution it will be reported then.

**Example**

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

In this example the command is issued to seek the X axis positive limit and the Y axis negative limit.

```
SI X? Y?
:A X=0 Y=0
```

In the example the X and Y axes are being queried for the current setting for the direction to seek the limits.

**Auto Homing For Clocked Devices like Sliders**

As of firmware 9.21 (for MS2000) and 3.18 (for Tiger) Seek Limit routine performs an additional step for clocked devices. After finding the Limit it moves a set distance (distance is specified as Home Position [Command:SETHOME](#)) and zeros itself there. By default, the distance between Upper limit and Slot 1 is saved in Home position , so when Seek Limit is run , the controller is able to move the slider to position 1 by finding the limit and moving a set distance from it.

2016/03/16 20:01

[commands](#), [tiger](#), [ms2000](#)

**Command:SPEED (S)**

MS2000 or RM2000 syntax

<b>Shortcut</b>	S
<b>Format</b>	SPEED [axis]=[max speed in mm/sec]...

<b>Units</b>	mm/sec
<b>Remembered</b>	Using SS Z

Tiger syntax

<b>Shortcut</b>	S
<b>Format</b>	SPEED [axis]=[max speed in mm/sec]...
<b>Units</b>	mm/sec
<b>Type</b>	Axis-Specific
<b>Remembered</b>	Using [addr#]SS Z

Sets the speed at which the stage will move during the middle of a commanded move (e.g. using MOVE, MOVEREL, or the home joystick button; speed during joystick moves is separate and set using the [JSSPD command](#)). The start of the move has a ramp-up period and the end of the move has a ramp-down period plus a separate landing phase. Duration of the ramps is set by the [AC command](#), and the landing phase is discussed on the [tuning page](#).

Speed is set in millimeters per second. Maximum speed setting is is ~7.68 mm/s for standard 6.35 mm pitch leadscrews (4 TPI). See the page on [lead screw pitch options](#). Default speed is ~67% of the max speed.

The stage might not be able to keep up with the firmware-set maximum speed depending on mechanical load and internal friction, and for that reason the advertised maximum speeds are a bit less than the maximum firmware setting (but advertised max speeds are always attainable with typical loads).

The maximum possible speed setting can be determined by setting the speed to a very high number, querying the resulting speed, and then restoring the original setting.

### Reply

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

### Example

```
s x? y?
:A X=5.745920 Y=5.745920
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.

### Special use for MMIRROR\_SLAVE on Micro-mirror card

On a Micromirror card with the MMIRROR\_SLAVE define the SPEED command has a special meaning (it is otherwise unused with micro-mirror cards) starting in v3.18. It sets the scaling of the slave axis. The default scale is 0.5, meaning the electrical outputs corresponding to the invisible slaved axis will move half as far as the main axis.

2016/03/16 20:04

[commands](#), [tiger](#), [ms2000](#), [micromirror](#), [speed](#), [vector](#)

### Command:SPIN (@)

MS2000 or RM2000 syntax

<b>Shortcut</b>	@
<b>Format</b>	SPIN [axis]=[-128 to 128]...
<b>Units</b>	integer -128 to 128

Tiger syntax

<b>Shortcut</b>	@
<b>Format</b>	SPIN [axis]=[-128 to 128]...
<b>Units</b>	integer -128 to 128
<b>Type</b>	Axis-Specific

Tells controller to ‘spin’ the motor of specified axis at a rate expressed as its DAC value, a bit value from -128 to 128. This causes the motor to run in open-loop mode with no position or speed feedback.

#### 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 1:** 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 2:** The HALT command will not return an :N-21 when stopping a SPIN command.

2016/03/16 20:08

[commands](#), [tiger](#), [ms2000](#), [spin](#), [vector](#)

### Command:STATUS (/)

MS2000 or RM2000 syntax

<b>Shortcut</b>	/
-----------------	---

<b>Format</b>	STATUS
---------------	--------

Tiger syntax

<b>Shortcut</b>	/
<b>Format</b>	STATUS
<b>Type</b>	Broadcast command

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.

2016/03/16 20:13

[commands](#), [tiger](#), [ms2000](#)

### Command:STOPBITS (SB)

<b>Shortcut</b>	SB
<b>Format</b>	STOPBITS X=n STOPBITS X?
<b>Units</b>	integer, 1 or 2
<b>Remembered</b>	Using SS Z

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.

2016/03/16 20:16

[commands, ms2000](#)

## Command:TTL

TTL functionality is somewhat different depending on whether the controller is a Tiger (TG-1000) or MS2000/RM2000 controller, stemming from hardware differences. Some TTL modes are only available with certain firmware modules.

The MS2000 controller electronics has a buffered TTL input (IN0) and output (OUT0) port that are usually connected to the IN and OUT BNC connectors on the back of the controller. (It also has several unbuffered I/O ports on the motherboard that are occasionally exposed for special purposes.) The behavior of these connectors are determined by the IN0\_mode and OUT0\_mode parameters set by the TTL X and TTL Y commands respectively.

On Tiger TG-1000 controllers, some cards have buffered TTL input (IN0) and output (OUT0) ports exposed, in which case the behavior is determined by the IN0\_mode and OUT0\_mode parameters set by the TTL X and TTL Y commands respectively. The TTL command is Card-Addressed, meaning that on Tiger it applies to each card separately. A few Tiger cards have extended functionality using the TTL\_AUXILIARY firmware module affected by the TTL R and TTL T commands.

MS2000 or RM2000 syntax

<b>Format</b>	TTL [X=IN0_mode] [Y=OUT0_mode] [Z=aux_IO_mode] [F=OUT0_polarity] TTL (firmware v9.2k+)
<b>Remembered</b>	Using SS Z
<b>Required Version</b>	8.5+

Unless otherwise specified, the TTL commands used for Tiger apply to all WK2000-based systems as well.

Tiger syntax

<b>Format</b>	[Addr#]TTL [X=IN0_mode] [Y=OUT0_mode] [Z=aux_IO_state] [F=OUT0_polarity] [R=aux_IO_mask] [T=aux_IO_mode] [Addr#]TTL (firmware v3.16+)
<b>Type</b>	Card-Addressed
<b>Remembered</b>	Using [Addr#]SS Z

### 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](#) commands.

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 TTL trigger 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 for the specified focus axis but not for other axes.

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 (set by the [RM command](#)). 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 the system is in the READY state. TTL IN0 LOW will cause the system to UNLOCK is locked already.

10 - Toggle TTL OUT0. If TTL OUT0 is set either LOW or HIGH, an input pulse on the TTL IN0 will cause the output to toggle to the other state.

11 - SERVOLOCK\_TTL mode. Requires firmware support; as of early 2020 it is only implemented in Tiger but could be ported to MS2000/RM2000. To engage the SERVOLOCK\_TTL mode use the [LK](#) command which will then automatically change the TTL IN0 mode and normally will restore it afterwards. When the SERVOLOCK\_TTL mode is engaged do not change the TTL X setting. For further documentation see the [SERVOLOCK\\_TTL documentation](#).

12 - Behavior is exactly the same as mode 1 above except the moves are relative rather than absolute. As of early 2020 it is only implemented in Tiger but could be ported to MS2000/RM2000.

CAUTION: If you are using TTL X mode 12, the values entered into the ring buffer using the [LOAD](#) command represent RELATIVE coordinates, not ABSOLUTE coordinates. You must drive the stage to the appropriate starting position before triggering a ring buffer sequence.

20 - The TTL IN0 pulse turns on the TTL OUT0 for a fixed duration set by the "[RT Y](#)" command.

Can be used with a LED illumination to act as a flash of defined duration. For the TGLED card and MS2000/RM2000 with a Dual LED card (DLED), all LED channels remain off until a TTL pulse is received, and the TTL rising edge turns all LED channels ON and they remain ON for the duration set by the [RT Y](#) command. <sup>1)</sup> LED intensity is set by the [LED](#) command.

Added in v3.14 for Tiger and version 9.2I for MS2000/RM2000.

21 - Similar to mode 20 except TTL pulses cycle through the channels of the TGLED card or Dual LED card. On each TTL pulse, only ONE LED channel turns ON for a fixed time. The next TTL pulse turns on the next LED channel for a fixed time, and so on.

For Tiger v3.24 and later, when [RB F=3](#) the TGLED card cycles thru all channels without waiting for TTL pulses. If an intensity is set to 0 using the LED command then that channel is

automatically skipped.

Added in v3.14 for Tiger and version 9.2l for MS2000/RM2000.

22 - Similar as mode 20 except waits for TTL pulse to go low before turning off the LED. Setting RT Y isn't required.

Added in v3.30 for Tiger and version 9.2n for MS2000/RM2000.

23 - Same as mode 21 except waits for TTL pulse to go low before turning off the LED. Setting RT Y isn't required.

Added in v3.30 for Tiger and version 9.2n for MS2000/RM2000.

30 - Used with Single Axis function `SAM [Axis]=2`. On rising edge of TTL pulse, the routine is performed once.

Available in firmware 3.30 and above; not yet implemented for MS2000/RM2000.

103 - Repeating Autofocus. This mode does not use TTL IN. When set, the controller will automatically attempt an autofocus routine every *K* milliseconds indefinitely until TTL X is changed. The repetition time, *K*, is set using `RTIME X=`. Example: `RT X=5000` will set the repetition rate to 5 seconds. This is effectively the same as using `TTL X=3` with an oscillator connected to TTL IN. Note: *MS2000 Only*

#### 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, MOVREL, move via ring buffer, or via array module). The pulse duration is set with command `RT Y`.

3 - ~~output TTL OUT0 gated HIGH during axis index 0 (X) constant speed move~~ **not implemented but hardware-level TTL signal is available.**

4 - ~~output TTL OUT0 gated HIGH during axis index 1 (Y) constant speed move~~ **not implemented but hardware-level TTL signal is available.**

5 - ~~output TTL OUT0 gated HIGH during axis index 2 (Z) constant speed move~~ **not implemented but hardware-level TTL signal is available.**

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

9 - TTL OUT0 PWM and MicroServo or Solenoid output. See the LED or the SECURE command. Requires LED\_DIMMER or USERVO or SOL\_LK firmware module. On WK, with LED\_DIMMER, the PWM frequency is 1KHz.

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 a pulse. Also enables CRISP status busy polling that will not return 'N' until CRISP is in the In-Focus 'F' state. Also enables CRISP status busy polling that will not return 'N' until CRISP is in the In-Focus 'F' state.<sup>2)</sup>

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

20 - TTL OUT0 set during SPIM state machine operation. Requires MM\_SPIM firmware module (TG-1000 only). Was code 10 until v3.12..

21 - TTL OUT0 set at the end of a ring buffer move or All-initiated move (for laser trigger). Requires MM\_TARGET firmware module (TG-1000 only). Was code 11 until v3.12. As of firmware v3.36 outputs to the TTL1 backplane signal instead of to the usual TTL output.

22 - In this mode, TTL OUT0 is controlled by [Single-axis function](#) module. With the [SAP](#) command the user can generate a TTL pulse that is synchronized with [Single-axis function](#) actuator motion. TG-1000 only, available in firmware version 3.17 and above.

30 - TTL OUT1 high when the position of first axis is more than the PC setting away from its target position. Tiger v3.38 and above (not yet in MS2000 but could be.)

31 - TTL OUT1 high when the position of second axis is more than the PC setting away from its target position. Tiger v3.38 and above (not yet in MS2000 but could be.)

32 - TTL OUT1 high when the position of third axis is more than the PC setting away from its target position. Tiger v3.38 and above (not yet in MS2000 but could be.)

33 - TTL OUT1 high when the position of fourth axis is more than the PC setting away from its target position. Tiger v3.38 and above (not yet in MS2000 but could be.)

#### OUT0\_polarity

1 - default polarity, -1 inverts the polarity of TTL OUT0.

#### aux\_IO\_state

Tiger only. Requires TTL\_AUXILIARY firmware module; behavior depends on the firmware build and hardware present. Sets the state of the auxiliary TTL output according to the `aux_IO_mask`. Input and output as a decimal number representing the binary pattern desired. The following uses have been defined so far:

For MM\_SPIM firmware with SPIM TTL card: Bit0 = Side0/Laser0 output, Bit1 = Side1/Laser1 output

#### aux\_IO\_mask

Requires TTL\_AUXILIARY firmware module; behavior depends on the firmware build and hardware present. Controls how the `aux_IO_state` bits are used, or how the backplane is used when `aux_IO_mode` is set to 2. Input and output as a decimal number representing a binary mask. If the corresponding mask bit is set to 1 then the state bit will be reflected at the output, but if the mask bit is 0 then the state bit has no effect. The following uses have been defined so far:

For MM\_SPIM firmware: Defaults to 3 = 0b00000011.

### aux\_IO\_mode

Requires TTL\_AUXILIARY firmware module; behavior depends on the firmware build and hardware present. The SPIM state machine overrides these setting during its operation.

0 - TTL outputs determined by aux\_IO\_state/mask.

1 - TTL outputs determined by the LED command (requires MM\_LASER\_TTL module). The default setting for MM\_SPIM firmware.

2 - Simulates a TTL input from the backplane. The backplane value is masked by aux\_IO\_mask and the binary value is considered. If a 0-1 transition occurs then a TTL input pulse is simulated and action will be taken depending on the setting of IN0\_mode. The default setting for TTL\_AUXILIARY on piezo firmware.

### Without any Argument

In firmware version 3.16 and above on TG-1000 and version 9.2k and above on MS2000/RM2000, when the TTL command is issued without any arguments, like [Card Addr#]TTL , the controller reports the state of TTL IN.

For Tiger/TG-1000, the controller replies with :A 0 when signal is low. and :A 1 when TTL IN sees a high signal.

For MS2000/RM2000, the controller replies with the inverse polarity, :A 1 when the signal is low and :A 0 when the signal is high. This is a logical bug in the firmware, but because it was in the field so long before being noticed we made the intentional decision to keep the behavior unchanged.

2017/07/26 13:03

[commands](#), [led](#), [tiger](#), [ms2000](#), [tglyed](#), [crisp](#), [ttl](#), [spim](#), [array](#), [sequencer](#), [servolock](#) [ttl](#)

## **Command:Unit Multiplier(UM)**

MS2000 or RM2000 syntax

<b>Format</b>	UM [axis]= ### ...
<b>Units</b>	integer
<b>Remembered</b>	Using SS Z

Tiger syntax

<b>Format</b>	UM [axis]= ### ...
<b>Units</b>	integer
<b>Type</b>	Axis-Specific
<b>Remembered</b>	Using [addr#]SS Z

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.

**For Clocked Position devices**

UM command has no effect if the axis is a Clocked Device like Filter Slider , Objective slider or Objective Turret.

As of Firmware version 9.2l (for MS2000) and 3.18 (for Tiger) , when UM for a Clocked Device is set to "1" , [Where command](#) prints the Axis position in MilliMeters instead of Slot Position. Move command continues to work, while Relative move doesn't. This feature is mean for troubleshoot and diagnostic usage only and not meant for regular operation.

2016/03/16 20:21

[commands](#), [tiger](#), [ms2000](#)

**Command:UNITS (UN)**

<b>Shortcut</b>	UN
<b>Format</b>	UNITS
<b>Remembered</b>	Using SS Z

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.

2016/03/16 20:23

[commands](#), [ms2000](#)

**Command:UNLOCK (UL)**

For CRISP or ZS

Tiger Syntax

<b>Shortcut</b>	UL
<b>Format</b>	[Addr#]UL [X=LED_Intensity] [Y=Update_Rate] [Z=Rel_LK_knob_spd][F=focus_index]
<b>Type</b>	Card-Addressed
<b>Remembered</b>	Using [Addr#]SS Z

MS2000 and RM2000 Syntax

<b>Shortcut</b>	UL
-----------------	----

<b>Format</b>	UL [X=LED_Intensity] [Y=Update_Rate] [Z=Rel_LK_knob_spd][F=focus_index]
<b>Remembered</b>	Using SS Z

Without arguments, this command unlocks the servo from the focus system and returns control to encoder feedback from the Z-axis drive. The focus error offset is not changed.

**X:** The LED\_Intensity may be set from 0 to 100 (%) of full power using the X argument. The default value is 99.

**Y:** Update\_Rate is used to reduce the update rate of the CRISP system to the motor drive system. Increase Update\_Rate to improve stability when using piezo Z-axis drive systems. As of Tiger firmware version 3.38, the value of Update\_Rate is given in milliseconds. The default update rate for piezos is 5ms and everything else has an update rate of 10ms as of version 3.38.

**Z:** Rel\_LK\_knob\_spd controls the sensitivity of the control focus knob when the system is locked. This will vary depending on the calibration factor that the system finds, so don't be alarmed if you find large sensitivity differences with conditions. The default value is 2.

**F:** To select active Z-focus axis: If the controller can handle more than one Z-axis focus device, you can specify the focus\_index to select which one is active for the CRISP, TRACKING or ZS functions. Specify as a 0-indexed number; find the axis index from the letter using Z2B command or else parse the output of the BU X command and figure out what number in order the axis letter is). Save the parameter change (SS Z) and reset the controller for setting to take effect.

For TRACKING

<b>Shortcut</b>	UL
<b>Format</b>	UL [X=focus_enable ] [Y= z_cal_value ] [Z=closeness ] [F=focus_index]
<b>Remembered</b>	Using SS Z

Without arguments, this command unlocks the servo from the focus system and returns control to encoder feedback from the Z-axis drive. The focus error offset is not changed.

focus\_enable: X=0 autofocus off; X=1 autofocus on.

z\_cal\_value: is the gain constant for the focus servo.

closeness: is a relative value describing the precision of XY centering before Z focus tracking is activated. closeness should be set to roughly the acceptable  $xerr^2 + yerr^2$ , where xerr and yerr are typical "well tracked" errors numbers seen on the controller LCD display.

To select active Z-focus axis: If the controller can handle more than one Z-axis focus device, you can specify the focus\_index to select which one is active for the CRISP, TRACKING or ZS functions. Save the parameter change (SS Z) and reset the controller for setting to take effect.

2016/02/23 19:58

[commands](#), [tiger](#), [ms2000](#), [crisp](#), [tracking](#), [phototrack](#)

### Command:VB

This command has a slightly different usage on Tiger then in MS2000 and RM2000.

MS2000 or RM2000 syntax and function

<b>Shortcut</b>	VB
<b>Format</b>	VB [X=binary_code] [Y=TTL IN1 state (read only)] [Z=read_decimal_places] [T=CMD_code]
<b>Units</b>	integer
<b>Remembered</b>	Using SS Z
<b>Firmware Required</b>	v8.5+

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.

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

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

CMD\_code can be specified on firmware with the NO\_CHANGE\_SETTINGS module included in the firmware. If you need this function and do not have the module in the firmware, contact ASI. This feature allows some commands to be "turned off" for modification of settings by host software.

Serial commands are enumerated according to the following table with a command number CMD:

AA, AALIGN	0	PC, PCROS	20	MC, MOTCTRL	40	MA, MAINTAIN	60	LCD	80
AC, ACCEL	1	RM, RBMODE	21	PD, PEDAL	41	Z2B	61	WRDAC	81
B, BACKLASH	2	RB, RDSBYTE	22	AF, AFOCUS	42	AM, AFMOVE	62	AR, ARRAY	82
BE, BENABLE	3	RS, RDSTAT	23	WT, WAIT	43	BU, BUILD	63	AH, AHOME	83
CD, CDATE	4	~, RESET	24	AZ, AZERO	44	LL, LLADR	64	AIJ	84
C, CNTS	5	SL, SETLOW	25	SS, SAVESET	45	AL, AFLIM	65	AFINFO	85
CR, CREST,	6	SU, SETUP	26	SN, SCAN	46	RU, RUNAWAY	66	EXTRA	86
D, DACK	7	S, SPEED	27	LK, LOCK	47	UM	67	PZ	87
E, ERROR	8	@, SPIN	28	UN, UNITS	48	ZS	68	PZC	88

\, HALT	9	/, STATUS	29	MT, MTIME	49	HM, SETHOME	69	PZINFO	89
TTL	10	V, VERSION	30	VE, VECTOR	50	OS	70	ARM	90
H, HERE	11	W, WHERE	31	KA	51	CCA, CUSTOMA	71	BCA,BCUSTOM	95
!, HOME	12	N, WHO	32	RDADC	52	CCB, CUSTOMB	72	LED	96
J, JOYSTICK	13	Z, ZERO	33	NR, SCANR	53	TEST	73	SECURE	97
KD, KDP	14	JS, JSSPD	34	NV, SCANV	54	EP, EPOL	74	MM, MULTIMV	98
KI, KIP	15	ES, ENSYNC	35	UL, UNLOCK	55	RT, RTIME	75	TSLOCK	99
KP, KPP	16	I, INFO	36	RL, RELOCK	56	AFADJ	76	SAA	100
KV, KVP	17	SP, SAVEPOS	37	LR LOCKRG	57	AFC, AFCALIB	77	SAM	101
M, MOVE	18	LD, LOAD	38	SB, STOPBITS	58	AFHOLD	78	SAP	102
R, MOVREL	19	DU, DUMP	39	VB, VBMODE	59	SI	79	SAF	103
								SAO	104

To disable the write function of a command, use VB T=(1000+CMD).

**Example** VB T=1027 will disable changing the SPEED command.

The command is explicitly enabled by using VB T=CMD.

**Example** VB T=27 will allow the SPEED command to work again.

### Tiger syntax and function

<b>Shortcut</b>	VB
<b>Format</b>	[addr#]VB [X=binary_code] [Z=read_decimal_places] [F=###]
<b>Units</b>	integer
<b>Type</b>	Card-Addressed
<b>Remembered</b>	Using [addr#]SS Z

The **Z** argument sets the number of decimal places for the WHERE command. This is card-addressed so that different cards can be set differently. It is saved to non-volatile memory using the [SS Z command](#).

The **F** argument sets the reply syntax. Setting to 0 (default) is the MS-2000 syntax, and setting to 1 gives the Tiger syntax (see above section for details). The syntax state does not persist when power is turned off.

**Reply** Note that this command does NOT return :A or other acknowledgement.

The **X** argument implemented in firmware version 3.17 and above . When set with the Binary bit shown in the table below, performs the corresponding action

<b>Bit 4</b>	16	Move and Move Rel will print the new Target Position.Vector command will print the current position
--------------	----	---

```
1vb x=16
<LF>
ve x=1 y=-1
```

```

:A -0 -0
<LF>
ve x=2 y=-2
:A 66562 -66567
<LF>
ve x=0 y=0
:A 156651 -156663
<LF>

```

In the above example , XY stage is on Card Addr#1. Because VB X=16 , BIT4 was set. This makes the Vector command reply with the axis's current position.

2016/03/17 14:18

[commands](#), [tiger](#), [ms2000](#)

## Command:VECTOR (VE)

MS2000 or RM2000 syntax

<b>Shortcut</b>	VE
<b>Format</b>	VE [axis]=[speed in mm/sec]...
<b>Units</b>	mm/sec
<b>Firmware Required</b>	v8.5+

Tiger syntax

<b>Shortcut</b>	VE
<b>Format</b>	VE [axis]=[speed in mm/sec]...
<b>Units</b>	mm/sec
<b>Type</b>	Axis-Specific

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.

### Example

```

ve x=10
:A
<LF>
ve x?
:A X=9.999151

```

```
<LF>
ve x=0
:A
<LF>
```

2016/03/17 14:25

[commands](#), [tiger](#), [ms2000](#), [spin](#), [speed](#), [vector](#)

### Command:VERSION (V)

MS2000 or RM2000 syntax

<b>Shortcut</b>	V
<b>Format</b>	VERSION

Tiger syntax

<b>Shortcut</b>	V
<b>Format</b>	[addr#]VERSION
<b>Type</b>	Card-Addressed

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

Tiger Example

```
1V
:A v2.4
```

Firmware version of card#1

MS2000 example

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

2016/03/17 14:32

[commands](#), [tiger](#), [ms2000](#)

### Command:WAIT (WT)

MS2000 or RM2000 syntax

<b>Shortcut</b>	WAIT
<b>Format</b>	WAIT [axis]=[time in msec]...
<b>Units</b>	millisecond
<b>Remembered</b>	Using SS Z

Tiger syntax



<b>Shortcut</b>	WAIT
<b>Format</b>	WAIT [axis]=[time in msec]...
<b>Units</b>	millisecond
<b>Type</b>	Axis-Specific
<b>Remembered</b>	Using [addr#]SS Z

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.

For Phototargeting or MM\_TARGET firmware , this can be used to specify the amount of time between the move initiation (either a ring buffer move or from the [AlJ command](#)) and when the laser pulse turns on. Thus it would normally be non-zero for at least one of the micro-mirror axes involved in phototargeting. Default for all axes in MM\_TARGET firmware is 5 ms (usual default is 0 ms).

**Example**

```
wt x?
:X=0 A
WT X=20
:A
```

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

2016/03/17 14:36  
[commands](#), [tiger](#), [ms2000](#), [phototargeting](#)

**Command:WHERE (W)**

MS2000 or RM2000 syntax

<b>Shortcut</b>	W
<b>Format</b>	WHERE axis [axis] [axis]...

Tiger syntax

<b>Shortcut</b>	W
<b>Format</b>	WHERE axis [axis] [axis]...
<b>Type</b>	Axis-Specific

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

The reporting precision of the **WHERE** command can be changed with the [VB Z command](#).


**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 of the underlying hardware. Hence, the ordering of the responses does not necessarily follow the order of the query or alphabetical order. The order of the underlying hardware is first by card address (for Tiger only) and then by the ordering of the electronics on each card. The ordering of electronics can be determined by using the Z2B query and also the output of the BU X command under the line beginning with "Motor Axes:".

2016/03/17 14:45  
[commands](#), [tiger](#), [ms2000](#)

**Command:WHO**

MS2000 or RM2000 syntax

<b>Shortcut</b>	N
<b>Format</b>	WHO
<b>Firmware Version</b>	Tiger v1.6+

Tiger syntax

<b>Shortcut</b>	N
<b>Format</b>	WHO
<b>Type</b>	Comm-default command
<b>Firmware Version</b>	Tiger v1.6+

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.

In TG-1000 it prints all the card address, axis characters, build name and compile date, of all cards installed in the system. The card addresses are printed in the hex representation of the character, e.g. "32" is printed for card address "2".

MS2000 example

```
N
:A ASI-MS2000-XYBR-Zs-USB
```

Tiger example

```
N
At 30: Comm v1.6 TIGER_COMM Jul 02 2013:17:19:34
At 31: X:XYMotor,Y:XYMotor v2.4 STD_XY Jun 11 2013:10:24:35
At 32: P:MMirror,Q:MMirror,R:MMirror,S:MMirror v2.4 MMIRROR_4CH
May 10 2013:16:22:55
```

2016/03/17 14:49

[commands](#), [tiger](#), [ms2000](#)

### Command:WRDAC

On Tiger with TGLED

<b>Format</b>	[Addr#]WRDAC X=[1 to 100]
<b>Units</b>	Percentage between 0 and 100
<b>Type</b>	Card-Addressed
<b>Remembered</b>	Using [Addr#]SS Z

This command is "recycled" for a slightly different use in TGLED than for other cards. In the context of a TGLED card this command is used to set the maximum amount of current for all the LED channels.

The Maximum amount of current a TGLED Rev A card can output on each LED channel is 1.2Amps. When the X argument is set to 75, then maximum current each channel will output is reduced to 75% of 1.2Amps i.e. 0.9Amps.

This command can be used as a quick way to adjust the brightness of all LED channels. Default is 75%, ASI recommends not exceeding this limit.

#### Example

```
1WRDAC X=50
:A
```

Limits the maximum current output on each channel to 50% or 0.6Amps

```
1WRDAC X?
X=50 :A
```

Queries the card for maximum current percentage.

On MS2000 and RM2000

<b>Format</b>	WRDAC X=[0 to 10]
<b>Units</b>	Voltage, 0 to 10V
<b>Firmware Required</b>	8.4f+

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. If using a a firmware build with "PZ" in the name, such as PZ\_CRISP, commanded moves of the Z axis will be scaled using the [CCA X command](#) (minimum=0v, maximum=10.00v) and output on the analog 0-10V output BNC (SV1 Pin 5).

*Example*

```
WRDAC X=1.1
:A
```

Voltage on PIN SV1-5 is 1.1Volts

```
WRDAC X=20
:N-4
WRDAC X=- 1
:N-4
```

Parameter out of range

On Tiger with TGPMT

<b>Format</b>	[Addr#]WRDAC X=[0 to 1000] Y=[1 to 1000]
<b>Units</b>	integer, between 0 and 1000
<b>Type</b>	Card-Addressed
<b>Remembered</b>	Using [Addr#]SS Z

This command is "recycled" for a slightly different use on TGPMT card than for other cards . In the context of a TGPMT card this command is used to set the PMT's control voltage or gain. When set to 0 , PMT output signal drops to 0Volts , turning it off. When set to 1000 , 100% of control signal (1 Volts incase of H10722) is applied to the PMT.

**X** sets the gain for PMT0

**Y** sets the gain for PMT1

This function does the same function as the Dials on TGPMT cards faceplate.

### Example

```
7 wrdac x? y?
X=500.000000 Y=0.000000 :A
```

Queries the TGPMT card at Address 7 for PMT gain settings. PMT0 is at 50% gain , PMT1 is at 0% gain and so is off.

```
7wrdac x=505
:A
```

Sets the gain of PMT0 at 50.5% on TGPMT card at address 7

2016/02/22 20:03

[commands](#), [led](#), [tiger](#), [ms2000](#), [tgled](#), [tgpmt](#), [dac](#), [0-10v](#)

## Command:Z2B

MS2000 or RM2000 syntax

<b>Format</b>	Z2B axis=[new axis letter ascii code]...
<b>Units</b>	ASCII code
<b>Remembered</b>	Using SS Z
<b>Firmware Required</b>	v8.6d+

Tiger syntax

<b>Format</b>	Z2B axis=[new axis letter ascii code]...
<b>Units</b>	ASCII code
<b>Type</b>	Axis-Specific
<b>Remembered</b>	Using [addr#]SS Z

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 on the card is returned (e.g. :A Y=1 for the 2nd axis on the card).

### Reply

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

**Example**

```
Z2B Z=66
:A
```

change to **B** axis name

```
2SS Z
:A
```

Required to save new name setting to flash. 2 because Z is on card#2

2016/03/17 15:24

[commands](#), [tiger](#), [ms2000](#)

**Command:ZERO (Z)**

MS2000 or RM2000 syntax

<b>Shortcut</b>	Z
<b>Format</b>	ZERO

Tiger syntax

<b>Shortcut</b>	Z
<b>Format</b>	ZERO
<b>Type</b>	Broadcast Command

Writes a zero to the position buffer of **all** axes. Allows the user to set current position as the origin. It's a Broadcast Command.

For setting the position of a single axis to zero instead, use the HERE command.

**Reply**

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

**Note**

This command has no effect on clocked devices

**Example**

```
Z
:A
```

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

2016/03/17 15:16

[commands](#), [tiger](#), [ms2000](#)

## Command:ZFLOCK (ZF)

Not implemented. Command not present on MS2000 and unused on Tiger. Instead we use SLAVE\_F\_TO\_Z module that doesn't have the ability to be controlled via a command, though if needed the ability to turn on and off or change which axis is the master then it could be added.

2016/03/29 19:51

[commands](#), [tiger](#), [zflock](#)

## Command:ZS

MS2000 or RM2000 syntax

<b>Shortcut</b>	ZS
<b>Format</b>	ZS [X=dZ] [Y=n] [Z=mode] [F= stack_timeout]
<b>Remembered</b>	Using SS Z

Tiger syntax

<b>Shortcut</b>	ZS
<b>Format</b>	[addr#]ZS [X=dZ] [Y=n] [Z=mode] [F= stack_timeout]
<b>Type</b>	Card-Addressed
<b>Remembered</b>	Using [addr#]SS Z

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 10th 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.

The axis moved by the TTL is the designated "focus index" (also the axis used for CRISP among other things). Use [UNLOCK F](#) to read or set the axis letter corresponding to "focus index". Note the setting has to be changed, settings saved, and the controller reset or power cycled for the new setting to take effect. If the controller has a piezo but no motorized focus drive then the piezo axis should be set as the "focus index". If both are present the "focus index" normally defaults to the piezo.



**Backlash** : ZS routine also performs the backlash compensation move for each step. For step size smaller than 10 microns, this might result in issues like irregular



step sizes. Consider disabling [Command:BACKLASH](#) for smaller step sizes.



**Stack Timeout** : If the TTL frequency is less than 2Hz , then the controller might consider it a stack time out condition. Consider increasing the Stack timeout to accommodate the slower TTL frequency, to avoid any issue.

### Reply

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

#### Tiger Example

Focus Drive axis is on Card#2

```
ZZS X=10 Y=20 Z=1  
:A
```

#### MS2000 Example

```
ZS X=10 Y=20 Z=1  
:A
```

Setup to do twenty 1 micron slices with triangular pattern.

2016/03/17 15:37

[commands](#), [tiger](#), [ms2000](#)

## Quick Start on Serial Commands

This section is intended to quickly bring the users up to speed on the most common serial commands. Find separately [an index of all commands](#) and a [compiled list of commands](#); separate documentation for each command are linked there.

There is a separate page detailing the [communication details](#) (also includes notes on various softwares that can be used to send serial commands directly or indirectly). Serial commands are not case-sensitive. For commands that require a numeric input, if none is specified then the value is taken to be 0.

The most common commands are identical across both the MS-2000/RM-2000 controller and the TG-1000 "Tiger" controller. Any "axis-specific" commands that specifies a physical axis that the command applies to will be identical (e.g. M, R, W for absolute and relative moves along with



position query) along with the busy query /. Commands that use a letter to specify what the setting is (e.g. TTL X sets TTL input mode and TTL Y sets TTL output mode) are slightly different on Tiger in that the card address must be prepended. There is a full description of [TG-1000 and MS-2000's instruction set differences](#).

### Most common commands

Full name	Short name	Use	Default units	Specify axis or address?	Example
<b>MOVE</b>	<b>M</b>	Initiates move to absolute position	10ths of microns	Axis	M X=10000 moves to position 1.000 mm per current origin
<b>MOVREL</b>	<b>R</b>	Initiates move a specified distance from current position	10ths of microns	Axis	R X=- 1000 moves 100um in negative direction
<b>WHERE</b>	<b>W</b>	Queries the absolute position of the axis	10ths of microns	Axis	W X returns :A 1000 if the absolute position is 100um relative to the origin
<b>HERE</b>	<b>H</b>	Sets the origin of the coordinate system for the specified axis	10ths of microns	Axis	H X sets the current position of the X axis to be 0
<b>STATUS</b>	<b>/</b>	Queries status of all axes on controller (MS-2000) or on card (TG-1000 if address is specified)	-	Address optionally (TG-1000 only)	/ returns N if no axis is busy or B if one is
<b>RDSTAT</b>	<b>RS</b>	When used with ? qualifier, queries status specified axis	-	Axis	RS X? returns N if X axis is busy or B if it is

### Command Syntax

The controller's instruction set is implemented using the following general syntax. Details are given on the page of each command.

```
<command name> (<axis letter> [=<value>] | [?] | [+] | [-])* <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. Most commands have abbreviated names that help cut down on typing time and serial bus traffic.

Next comes one or more sets of strings that contain an axis name. When setting a value the axis name is followed immediately by an equal sign and the corresponding 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 usually assumed to mean =0 (the behavior isn't universal but it works for the commonly-used MOVE and MOVREL commands). Truncation or rounding of values may occur depending on the command. Set parameter values with an equal sign, and query them using a question mark. Some commands do not require a parameter value (e.g., INFO X) or use the + or - signs (e.g. MC X-).

The axis name is a single character. All 26 alphabet characters A-Z can potentially be used as axis names, and the special character \* means “all axes” (not including filterwheels) in recent Tiger firmware. Axis letters X and Y are typically used for a 2-axis motorized stage, Z and F are commonly used for focus axes, and A-D are the default letters for micro-mirror scanner axes. Filter Wheels are designated by numbers, TG-1000 can accommodate up to 10 wheels numbered 0-9.

For Tiger only: When [Addr#] appears in the format, then the intended card address must be prepended to the serial command, as the command is Card-Addressed. ... indicates more arguments can be sent with the same command.

All commands are completed with a Carriage Return (ASCII hex code: 0D). The 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 ~ and halt /, the controller will not process a command in the memory buffer until the Carriage Return <CR> has been received.

## Reply Syntax

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

### MS-2000 Reply Syntax

When a command is recognized, the controller will 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. This programming manual gives examples in the MS-2000 reply syntax unless otherwise specified.

### Examples

Typed commands are in THIS TYPEFACE  
Controller replies are in THIS TYPEFACE

```
MOVE X=1234 Z=1234.5 <CR>
:A <CR><LF>
MOVE X Y Z <CR>
:A <CR><LF>
WHERE X <CR>
:A 0 <CR><LF>
MOVE X=4 Y=3 Z=1.5 <CR>
:A <CR><LF>
WHERE X Y Z <CR>
:A 4 3 1.5 <CR><LF>
WHERE Z Y X <CR>
```

```
:A 4 3 1.5 <CR><LF>
```

### Tiger Reply Syntax

The TG-1000 has two reply syntaxes; the active one is set using the [commands:vb|VB F]] command. The default syntax is backwards compatible with the MS-2000 controller, including all the quirks and inconsistencies between commands. The Tiger syntax is more self-consistent and in some cases more explanatory (e.g. with WHERE), but is not backwards compatible. Choice of reply syntax is completely arbitrary and does not affect operation.

In the Tiger syntax no :A is sent back. Furthermore, whenever an axis position or command value is returned (i.e. whenever the command is a query), the axis letter is always specified. Consequently, when no information needs to be sent back and there is no error the controller simply replies with <CR><LF> only.

The above examples in Tiger reply syntax are as follows:

```
MOVE X=1234 Z=1234.5 <CR>
:A <CR><LF>
MOVE X Y Z <CR>
:A <CR><LF>
WHERE X <CR>
:A 0 <CR><LF>
MOVE X=4 Y=3 Z=1.5 <CR>
:A <CR><LF>
WHERE X Y Z <CR>
:A 4 3 1.5 <CR><LF>
WHERE Z Y X <CR>
:A 4 3 1.5 <CR><LF>
```

### Query of Parameters

Most commands used to set parameter values – for example the setting S for maximum speed or B for backlash move distance – 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. The exact details of the reply syntax depend on the exact command but are self-evident when you try it.

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.

Note that the position query W is a command by itself and does not use the question-mark syntax; simply use W <axis>.

### Command Syntax Error Codes

When a command is received that the controller cannot interpret, for one reason or another, an

error is returned in the following format:

```
:N-<error code>
```

The error codes are as follows:

<b>:N-1</b>	Unknown Command (Not Issued in TG-1000)
<b>:N-2</b>	Unrecognized Axis Parameter (valid axes are dependent on the controller)
<b>:N-3</b>	Missing parameters (command received requires an axis parameter such as x=1234)
<b>:N-4</b>	Parameter Out of Range
<b>:N-5</b>	Operation failed
<b>:N-6</b>	Undefined Error (command is incorrect, but for none of the above reasons)
<b>:N-7</b>	Invalid card address
<b>:N-8..:N-10</b>	Reserved
<b>:N-11..:N-20</b>	Reserved for filterwheel
<b>:N-21</b>	Serial Command halted by the HALT command
<b>:N-30..:N-39</b>	Reserved

### A Note Regarding Units

The most common commands including MOVE, MOVEREL, HERE, and WHERE use axis units, which can be changed using the UM command. However, some commands such as SETUP, SETLOW, and PCROS always use units of mm. By default, axis units where position is given in distance (e.g. for motorized stages and piezo stages) are represented in 0.1um increments, or 10,000 units per millimeter of travel. For mirror scanner axes, default axis units are 0.001 degrees (uncalibrated), or 1000 units per degree of travel. When a time is required, the unit is generally milliseconds. Some commands use integer codes or other input units as described in the per-command documentation.

2020/11/07 16:04

[serial](#), [commands](#), [manual](#), [tech note](#)

### TG-1000 and MS-2000's instruction set differences

This section is intended to quickly bring the users up to speed on the major differences in the serial commands between TG-1000 and MS-2000 controller.

Where possible the TG-1000 commands were kept close to the MS-2000 controller commands. The most common commands are identical because they are Axis-Specific commands. A special [Tiger reply syntax](#) was added to Tiger starting in COMM card firmware v1.92. However the default syntax closely mirrors the MS-2000 syntax.

### Categories of Commands

#### Axis-Specific Commands

Commands where the axis letter of the stage is specified work just like MS-2000 on the Tiger, e.g. MOVE, MOVREL, and WHERE. For example, X and Y axis reside on card with address 1, and Z resides on card with address 2. When M X=## Y=### Z=### is issued the COMM card in TG-1000 controller parses the command, and redirects each sub-command to the appropriate card automatically. These commands are called Axis-Specific Commands. A card address should not be specified with an Axis-Specific command, otherwise undefined behavior may result from any mismatch between the address specified and the card with the axis as determined by the COMM card.

### **Card-Addressed Commands**

We were forced to alter handling of commands which do not contain the letter of a stage axis on Tiger. In those cases, the user must specify which card the command is intended for, and the COMM card will relay the command appropriately. For example, in MS-2000 the SS Z command saves settings into non-volatile memory, but the user may not want all the settings on all the cards to be saved. For the TG-1000 the user has to add the card address in front of the command to make it work. For example, if the card that drives XY axis has the address 1, to only save setting on this card user should issue 1SS Z. These commands are called Card-Addressed Commands. How to determine the address of a specific card is described below.

Importantly, for any settings selected by Card-Addressed Commands the setting applies to all axes on the card. For example, the ring buffer delay can be set independently for the XY stage on one card and a piezo stage on a different card. However, if a card contains more than one independent axis (e.g. a ZF card for two linear stages, or a Micro mirror card for 2 scanners) then the same setting applies to all axes. Continuing our example, the ring buffer on the ZF card would control both Z and F axes (though one of these axes could be disconnected from the ring buffer) so Z and F could not have separate ring buffer delay times.

Some commands default to a particular card if an address is not specified, though they can be addressed to specific cards as well. There are sub-categories of commands sent to the COMM card unless otherwise specified (e.g. BUILD and WHO), commands sent to whatever card has the X axis or else all the stage cards if the X axis doesn't exist (e.g. JSSPD and LED), and commands sent to whatever card has the Z axis or all the stage cards if the Z axis doesn't exist (e.g. most CRISP and autofocus commands). These sub-categories may be referred to as Comm-Default, Stage-Default, and Focus-Default commands respectively.

### **Broadcast Commands**

A third category is Broadcast commands, like STATUS, HALT, RESET, and ZERO. They are routed to all cards in a TG-1000 system by default, just like the MS-2000. However, in most cases they can also be directed at a single card by simply adding the card address, so they could be considered as special type of Card-Addressed Commands sent to all cards if an address is not specified.

### **Other differences**

Note that a few commands have minor behavioral differences between MS-2000 and TG-1000.

Most often these are due to hardware differences, e.g. different buttons or different TTL capabilities. Deviations between MS-2000 and TG-1000 command set are noted in the documentation of those commands. Note that TG-1000 offers a wide variety of different types of cards driving different devices.

### Special "All-Axis" Letter

Beginning with COMM card firmware version 3.10 a special axis letter \* was added to Tiger. When this character is specified in an Axis-specific command then it applies to all the axes in the controller. For instance, all axes can be moved to zero position using M \*=0 (or equivalently M \* because numbers not specified default to 0). The \* axis letter can also be combined with card addressing to apply the command to all the axes on the same card, e.g. 3M \*=0 will move all axes on card address 3 to their zero position, and 3! \* will home all axes on card 3.

### How does the user find out a card address?

When the TG-1000 is turned on or when WHO serial command is issued, the controller prints out card address, axis names, firmware version and firmware build date of all the cards installed in the system (see section Tiger Banner below for more details). Also, more complete information about the cards and corresponding axes can be accessed using the BUILD X command which is described in the section Build Command below.

As an illustrative example, the controller may have the following configuration as reported on startup:

```

.....
At 30: Comm v1.5 TIGER_COMM May 07 2013:15:42:05
At 31: X:XYMotor,Y:XYMotor v2.4 STD_XY Jun 11 2013:17:00:12
At 32: Z:Piezo v2.4 ADEPT_PIEZO Jun 11 2013:17:05:00
Joystick ready.
System ready.
    
```

The card addresses are shown in ASCII code. At 31 indicates a card with X and Y axes has the address 1 (ASCII code of 1 is 31).

<b>To save settings in non volatile memory:</b>	1SS Z	Card-Addressed
<b>To print its error dump buffer:</b>	1DU Y	Card-Addressed
<b>To move X and Y axis:</b>	M X=### Y=###	Axis-Specific

Similarly the card with Z axis has address 2 (ASCII code of 2 is 32).

<b>To move Z axis:</b>	M Z=###	Axis-Specific
<b>To issue pzinfo command on this card:</b>	2PZINFO	Card-Addressed
<b>To put in closed loop external input mode:</b>	2PZ Z=1	Card-Addressed
<b>To save settings into non-volatile memory:</b>	2SS Z	Card-Addressed
<b>To run short calibration on the piezo:</b>	2PZC	Card-Addressed

**Table 1 TG-1000 addresses**

Addressee	Usage	Value
TG-1000 Comm	Hard coded, re-assignable	0x30 ('0')
Stage/ FW/Shutter	Unique Address	0x31 to 0x39, then 0x81 to 0xF5
Stage Broadcast	Recognized by all stage controllers	0xF6
Filterwheel Broadcast	Recognized by all FW controllers	0xF7
Shutter Broadcast	Recognized by all shutter controllers	0xF8
LCD Broadcast	Recognized by all LCD controllers	0xF9
Broadcast	Recognized by all cards	0xFD
Broadcast except Comm	Recognized by all cards except TG-1000 Comm	0xFE

## Identifying Controller Configuration

### Build Command

In Controller with Tiger Comm version v1.8 and above, the BUILD X command directed at the COMM card (or without any address) can be used to query axis names, axis types and axis address. (When the build command is addressed to a specific card the build information for that card is returned.)

### Example:

```
build x
TIGER_COMM
Motor Axes: X Y P Q R S 0 1
Axis Types: x x u u u u w w
Axis Addr: 1 1 2 2 2 2 3 3
Hex Addr: 31 31 32 32 32 32 33 33
Axis Props: 0 0 0 0 0 0 0 0
```

The above system has card address 1 with an XY stage with axes named X and Y. Card number 2 has micro mirror with axes P,Q,R,S. Then card address 3 with filter wheel IDs 0 and 1.

The next line contains the axis type short code. For example, x means xy stage, u means micro mirror and w means filter wheel. A complete listing of axis types with designations is as follows:

The next two lines contain the addresses in two forms, first the form that is used to prefix Card-Addressed commands and second in hex format.

Finally, any special axis properties are printed (e.g. CRISP or SCAN capabilities) starting with firmware version 2.8. The decimal equivalent of a byte (0-255) is printed with the bits of the byte representing the following

**Axis Type List**

Axis Type Short	Axis Type Long	Description
x	XYMotor	XY stage
z	ZMotor	Z focus motor drive. LS50s, Z scopes etc
p	Piezo	Piezo Focus. ASIs ADEPT, Piezo DAC etc
o	Tur	Objective Turret
f	Slider	Filter Changer
t	Theta	Theta Stage
l	Motor	Generic linear motorized stage, TIRF, SISKIYOU etc
a	PiezoL	Generic linear piezo stage
m	Zoom	Zoom magnification motor axis
u	MMirror	Micro Mirror, Scanner 75 etc
w	FW Filter	Wheel
s	Shutter	Shutter
g	Logic	Programmable logic card
i	LED card	Multi LED Driver card
b	Lens	Tunable Lens
d	DAC	Digital to Analog converter(DAC)
u	Unknown	Unknown axis type

**Axis Properties**

Bit 0:	CRISP auto-focus firmware
Bit 1:	RING BUFFER firmware
Bit 2:	SCAN firmware
Bit 3:	ARRAY firmware or MM_TARGET firmware
Bit 4:	SPIM firmware (v2.81+)
Bit 5:	SINGLEAXIS and/or MULTIAXIS firmware (v2.81+)
Bit 6:	LED illumination (v2.87+)
Bit 7:	Reserved

**Tiger Banner**

A banner is printed by the controller on Startup and by the WHO command. It tells the user (and also to any scripts and programs) all the available cards in the system, with their axis characters, axis types etc.

```
At 30: Comm v1.5 TIGER_COMM May 07 2013:15:42:05
At 31: X:XYMotor,Y:XYMotor v2.4 STD_XY Jun 11 2013:17:00:12
At 32: Z:Piezo v2.4 ADEPT_PIEZO Jun 11 2013:17:05:00
```

It is a multiline reply, each line is terminated by a <Carriage Return> and final line by a <Carriage Return> +<Line Feed> to designate end of Transmission.

Each line can be sub divided into strings using a white space as delimiter. Then the 2nd string is the card address in Hex. 0x30 is 0 in ascii, 0x31 is 1 and 0x32 is 2.All possible address are 0x30 to 0x39 and then 0x81 to 0xF5. 0x81 is ü, 0xF5 is J.



The second string can be further subdivided with comma as delimiter. Then the first character of the string is the axis character, colon and what kind of an axis it is. A to Z are all possible axis names, system is case insensitive. Except in case of filter wheels, IDs are integers 0 to 9

**Example:**

X:XYMotor,Y:XYMotor . X and Y are axis names of a XY stage. Z:Piezo . Z is the axis names of a Piezo focus.

Then the 4th string is the firmware version number. 5th String is our build name, which is an internal designation. The last set of strings are the firmware compile date and time.

**A Note Regarding Units**

The most common commands including MOVE, MOVEREL, HERE, and WHERE use axis units, which can be changed using the UM command. However, some commands such as SETUP, SETLOW, and PCROS always use units of mm. By default, axis units where position is given in distance (e.g. for motorized stages and piezo stages) are represented in 0.1um increments, or 10,000 units per millimeter of travel. For mirror scanner axes, default axis units are 0.001 degrees (uncalibrated), or 1000 units per degree of travel. When a time is required, the unit is generally milliseconds. Some commands use integer codes or other input units as described below.

2016/02/24 15:48

[serial, commands, manual, tiger, tech note products](#)

1)

Old text, unknown meaning: TTL Y must be in 0 mode (TTL Y=0) or it might cause an issue.

2)

Tiger only?

From:

<https://www.asiimaging.com/docs/> - **Applied Scientific Instrumentation**

Permanent link:

[https://www.asiimaging.com/docs/products/serial\\_commands](https://www.asiimaging.com/docs/products/serial_commands)

Last update: **2021/09/23 17:16**

