# Contents

# TG-1000 PROGRAMMING MANUAL

The following section describes the RS-232 serial command set that the TG-1000 controller uses when communicating with a host computer. It closely follows the command set for the MS-2000. If you don't need to know everything, just use the quick reference below to get started. Details of each command, including examples, follow.

## *Quick Reference – Main Operating Commands*

| Command | Shortcut | Description |
|---|---|---|
| <DEL> , <BS> | | Abort current command and flush input buffer |
| HALT | \ | Halts all moves being executed |
| HERE | H | Writes a position to specified axis position buffer |
| HOME | ! | Moves specified axes to physical limit switches |
| INFO | I | Returns a screen full of information about specified axis |
| MOTCTRL | MC | Enables/disables motor control for specified axis |
| MOVE | M | Writes a position to an axis target buffer |
| MOVREL | R | Writes a relative position to specified axis target buffer |
| RDSBYTE | RB | Returns a status byte for specified axis |
| RDSTAT | RS | Same as RDSBYTE, in decimal ASCII format |
| RESET | ~ | Resets the controller |
| SPEED | S | Sets the maximum velocity/speed of specified axis |
| SPIN | @ | Causes specified axis to spin motor at given DAC rate |
| STATUS | / | Returns B-Busy, N-Not Busy |
| WHERE | W | Returns current position of specified axis |
| WHO | WHO | Prints list of cards with address and firmware information |
| ZERO | Z | Sets all axes to zero position (same as HERE <axis>=0 for all axes) |
| | | |

## *Quick Reference – Customization Commands*

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

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

## *RS-232 Communication*

The TG-1000 utilizes an RS-232 serial link to connect with any computer with an RS-232 serial port in order to utilize all of the controller's abilities. The current setup for the TG-1000 serial link is: ***115200 baud***, no parity, eight data bits, one stop bit, and no flow control ***(115200: 8 : N : 1 : None)***. This serial control feature can be accessed through terminal programs such as TeraTerm, and HyperTerminal. Software such as LabView can be used to send serial commands. Finally, high-level microscope control software (e.g. Micro-manager, Molecular Devices' Metamorph, Nikon Elements) uses serial commands to communicate with the controller.

### Command

The controller's instruction set is implemented using the following format:

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

The **`COMMAND`** is a string of ASCII characters such as **`MOVE`** or **`HOME`**, which must be followed by a space. All commands are case-insensitive. Many commands have abbreviated versions that help cut down on typing time and serial bus traffic.

Next are the axis parameters. (Bracketed "[ ]" parameters are optional.) The axis name is given, followed immediately by an equal sign and the axis parameter value. Each axis must be separated from the one before by one blank space. One or more axes may be specified on a single command line. An axis symbol typed without an "=" assignment is often assumed to mean "=0", but that behavior isn't guaranteed in general (it does, however, work for the commonly-used MOVE and MOVREL commands). Sometimes the command format may not require a parameter value (e.g., **`INFO X`**). Commands will accept integer or decimal numbers; internal truncation or rounding will occur if fractional decimals are of no meaning to the command.

**`axis`** or **`[axis]`** is the placeholder for the axis name, which is a single character. All 26 alphabet characters A-Z can be used as axis names, and the special character * means "all axes" (not including filterwheels). For example, X and Y are typically used for a sample translation stage, Z and F are commonly used for focus axes, and A-D are the default letters for scanner axes. Filterwheels are designated by numbers, TG-1000 can accommodate up to 10 wheels numbered 0-9.

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 (~), the controller will not process a command in the memory buffer until the Carriage Return **`(<CR>)`** has been received.[1]

[1] ASI's Control Character Bracketed Command Set, e.g., <Ctrl G><Ctrl H>, cause the memory command buffer to be emptied. This allows the daisy-chaining of other peripherals, such as ASI's SC-2 shutter controller, on the RS-232 line without causing unrecognized command errors to be reported back by the TG-1000 and MS-2000. [does this apply to TG-1000?]

### Reply

Upon receiving a Carriage Return **<CR>**, the TG-1000 will process the command stored in its command buffer, clear the command buffer, and return a reply.

The TG-1000 has two reply syntaxes; the active one is set using the **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.

### 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**; computer replies are in ***THIS TYPEFACE***.)

```
MOVE X=1234 Z=1234.5
:A
MOVE X Y Z
:A
WHERE X
:A 0
MOVE X=4 Y=3 Z=1.5
:A
WHERE X Y Z
:A 4 3 1.5
WHERE Z Y X
:A 4 3 1.5
```

### Tiger Reply Syntax

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

MOVE X Y Z
```

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

WHERE X Y Z
X=4 Y=3 Z=1.5
WHERE Z Y X
X=4 Y=3 Z=1.5
```

**Error Codes**

When a command is received that the TG-1000 cannot interpret, for one reason or another, an error message is returned in the following format:

**:N<error code>**

The error codes are as follows:

| | |
|---|---|
| ***:N-1*** | Unknown Command (Not Issued in TG-1000) |
| ***:N-2*** | Unrecognized Axis Parameter (valid axes are dependent on the controller) |
| ***:N-3*** | Missing parameters (command received requires an axis parameter such as x=1234) |
| ***:N-4*** | Parameter Out of Range |
| ***:N-5*** | Operation failed |
| ***:N-6*** | Undefined Error (command is incorrect, but for none of the above reasons) |
| ***:N-7*** | Invalid card address |
| ***:N-8..:N-10*** | Reserved |
| ***:N-11..:N-20*** | Reserved for filterwheel |
| ***:N-21*** | Serial Command halted by the HALT command |
| ***:N-30..:N-39*** | Reserved |

**Query of Parameters**

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

**CMND X? Y? Z? F?**

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

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

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

## *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 Tiger reply syntax, described above, was added to Tiger starting in COMM card firmware v1.92.

Commands where the axis letter is specified work just like MS-2000, 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 the 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.

The TG-1000 is able to support multiple add-ons like CRISP and Piezo stages whereas MS-2000 was only able to support one, so we were forced to alter the command set to handle commands which are intended for a certain card but do not contain the axis name. 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, the **SS Z** command saves the 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***.[2]

Importantly, for ***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.

Various methods of determining the address of a specific card are given below.

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.

A final category includes commands that default to a particular card, 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.

Beginning with COMM card firmware version 3.10 a special axis letter * was added. 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**.[3] The * axis letter can also be

[2] Previously Card-addressed commands they were called "Non-Addressed Commands," referring to the fact that the command itself does not carry address information and thus it needed to be separately specified.

[3] Actually the **=0** can be omitted, since numeric arguments for **MOVE** default to 0.

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).

- To save settings in non volatile memory: **1SS Z** (Card-Addressed)
- To print its error dump buffer: **1DU Y** (Card-Addressed)
- To move X and Y axis: **M X=### Y=###** (Axis-Specific)

Similarly the card with Z axis has address "2" (ASCII code of 2 is 32).

- To move Z axis: **M Z=###** (Axis-Specific)
- To issue pzinfo command on this card: **2PZINFO** (Card-Addressed)
- To put in closed loop external input mode: **2PZ Z=1** (Card-Addressed)
- To save settings into non-volatile memory: **2SS Z** (Card-Addressed)
- To run short calibration on the piezo: **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:

| 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 |
| **u** | **Unknown** | Unknown axis type |

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:

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 **⌡.**

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.

## *TG-1000 Command Set*

**Command:** **ACCEL**

Shortcut: **AC**

Format: **ACCEL [axis] = [time in msec]**…

Units: millisecond

Type: Axis-Specific

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

Example:

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

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

**Command:** **AALIGN**

Shortcut: **AA**

Format: **AALIGN [axis] = ###** …

Units: 0-255, hardware potentiometer value

Type: Axis-Specific

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

Example:

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

**Command: AFCONT (Requires Video Autofocus Hardware - See Autofocus Manual)**
**Command: AFLIM (Requires Video Autofocus Hardware - See Autofocus Manual)**
**Command: AFOCUS (Requires Video Autofocus Hardware - See Autofocus Manual)**
**Command: AFSET (Requires Video Autofocus Hardware - See Autofocus Manual)**
**Command: AFMOVE (Requires Video Autofocus Hardware - See Autofocus Manual)**

**Command: AHOME (Requires ARRAY firmware module, not yet tested)**
Shortcut: **AH**

**Command: AIJ (Requires ARRAY firmware module, not yet tested)**
Shortcut: **IJ**

**Command: ARRAY (Requires ARRAY firmware module, not yet tested)**
Shortcut: **AR**

**Command: AZERO**
Shortcut: **AZ**
Format: **AZERO [axis]**…
Units: none
Type: Axis-Specific
Function: Automatically adjusts the zero balance of the motor drive card.
Example: **AA 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
```

**Command:** **BACKLASH**

Shortcut: **B**

Format: **BACKLASH [axis] = [distance]**…

Units: millimeter

Type: Axis-Specific

Function: This command sets (or displays) the amount of distance in millimeters to travel to absorb the backlash in the axis' gearing. This backlash value works with an anti-backlash routine 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; zero is default.

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.

**Command:** **BCUSTOM** **(Version 3+)**

Shortcut: **BCA**

Format: [**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]**

Units: Integer code (see table below)

Type: Card-Addressed

Function: In version 3.0+ the Button Function assignment has been rewritten to be more flexible. Every possible button function is now assigned a number. This function can be assigned to any button (@,**Home** and **Joystick Button**) and any press duration (*Normal, Long* and *Extra Long press*) thru the BCA commands X,Y,Z,F,T,R and M arguments.

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

Note: Behavior of this command is very different pre version 3.0

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

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

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

Table below lists and describes all possible button functions

| No | Function |
|---|---|
| **0** | No function performed |
| **1** | Smart Move related, |

| | |
|---|---|
| **2** | Toggles Knob between two axis, like Z and F |
| **3** | TRACKING,CRISP,LSRTRK related, short press functions. Steps from IDLE to Calibration states to lock and unlock state. |
| **4** | CLOCKED DEVICE related, moves clocked devices like Turret and slider to next position |
| **5** | ARRAY MODULE related, moves to next array position |
| **6** | RING BUFFER related, move to next ring buffer position |
| **7** | SCAN MODULE related, halts scan move |
| **8** | AUTOFOCUS related, performs autofocus routine. |
| **9** | CRIFF related, toggle CRIFF lock |
| **10** | AT_XYZ_KNOB related, changes xyz knob state. Cycles knob control between x,y,z axis. |
| **11** | AT_XYZF_KNOB related, changes xyzf knob state. Cycles knob control between x,y,z and f axis. |
| **12** | ZLOADER related, Moves Z to its lower limit and back. |
| **13** | CRISP related, performs CRISP very long press functions |
| **14** | ADEPT related, toggles between piezo external and internal input mode |
| **15** | CRISP related, performs CRISP long press functions. Stops current CRISP operation like stop dither, unlock etc. |
| **16** | ARRAY MODULE related, puts ARRAY Module in start state |
| **17** | CRIFF related, change CRIFF state. |
| **18** | RING BUFFER related, loads current stage position into ring buffer |
| **19** | AT RAMM LOAD related, moves Y and F axis to their upper and lower limits. |
| **20** | SCAN MODULE related, puts SCAN MODULE is start state |
| **21** | AUTOFOCUS related, performs autofocus calibration |
| **22** | ZOOM related, sets up zoom profile |
| **23** | PLANAR CORRECTION related, sets planar correction points |
| **24** | RING BUFFER related, clears ring buffer |
| **25** | TOGGLE ALL AXES related, swaps knob control between two axes. |
| **26** | TRACKING related, short press function |
| **27** | JS PULSE related, sets TTL in ready state |
| **28** | JS_FASTSLOW related, toggles joystick speed between fast and slow |
| **29** | PLANAR CORRECTION related, resets planar correction |

Example: 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.

**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>***
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.

**1BCA X=6 F=24 R=18 M=28**
***:A***
***<LF>***
Now,
*@ normal press*, moves to next ring buffer position
*Home long press*, clear the ring buffer
*Joystick button normal press*, loads current stage position into ring buffer
*Joystick button long press*, toggles joystick speed

**Command:** **BENABLE**
Shortcut: **`BE`**
Format: **`[Addr#]BENABLE [X=Toggle] [Z=Enable_Byte]`**
Units: none
Type: Card-Addressed
Function: 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:

| Bit | Button |
|---|---|
| **0** | "Zero" Button |
| **1** | "Home" Button |
| **2** | "@" Button |
| **3** | Joystick Button |
| **4** | Reserved |
| **5** | Zero button zeros Z axis only |
| **6** | Reserved |
| **7** | Reserved |

**Command:** **BUILD**

Shortcut: **BU**

Format: **[Addr#]BUILD [X]**

Units: none

Type: Card-Addressed (defaults to COMM)

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

Example:

```
BU
TIGER_COMM

BU 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
```

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.

```
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.

| | |
|---|---|
| Bit 0: | CRISP auto-focus firmware |
| Bit 1: | RING BUFFER firmware |
| Bit 2: | SCAN firmware |
| Bit 3: | ARRAY firmware |
| Bit 4: | SPIM firmware |
| Bit 5: | SINGLEAXIS and/or MULTIAXIS firmware |
| Bits 6-7: | reserved |

**Command:** **CDATE**

Shortcut: **CD**

Format: **[Addr#]CDATE**

Function: This command returns the date and time the current firmware were compiled. It's a ***Card-Addressed command.***

Example: **1CD**

***Dec 19 2008:16:19:59***

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

**Command:** **CNTS**

Shortcut: **C**

Format: **CNTS [axis] = [encoder counts per mm]**…

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

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.

**Command:** **CUSTOMA**

Shortcut: **`CCA, CA`**

Format: **`[Addr#]CCA X = ###`**

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

| CCA X= | Description | Displa | Comment |
|---|---|---|---|
| 5 | XY Leadscrew Coarse Pitch (6.35 mm - Standard) | B | Firmware default |
| 6 | XY Leadscrew Fine Pitch (1.59 mm) | A | |
| 7 | XY Leadscrew Super Coarse (12.7 mm) | C | |
| 8 | XY Leadscrew Ultra Fine (0.635 mm) | U | |
| 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 | |
| 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 | |
| 52 | XY Linear Encoder 2.5nm resolution | L | |
| 30 | XY Limit Polarity – Normally Open | o | Firmware default |
| 31 | XY Limit Polarity – Normally Closed | c | |
| 9 | Z Scope Drive 100 µm/rev. (50 nm enc. resolution) | N | Firmware default |
| 10 | Z Scope Drive 200 µm/rev. (50 nm enc. resolution) | Z | |
| 19 | Z Scope Drive 100 µm/rev. (25 nm enc. resolution) | H | |
| 11 | Z Leadscrew Coarse Pitch | B | |
| 12 | Z Leadscrew Fine Pitch | A | |
| 13 | Z Leadscrew Super Coarse Pitch | C | |

| CCA X= | Description | Displa | Comment |
|---|---|---|---|
| 14 | Z Leadscrew Ultra Fine Pitch | U | |
| 29 | Z SISKIYOU Motor/Leadscrew | S | |
| 26 | ZF Linear Encoder 10 nm resolution | 1 | Leadscrew devices only. |
| 27 | ZF Linear Encoder 20 nm resolution | 2 | |
| 53 | XY Linear Encoder 5nm resolution | K | |
| 54 | XY Linear Encoder 2.5nm resolution | L | |
| 32 | ZF Limit Polarity – Normally Open | o | Firmware default |
| 33 | ZF Limit Polarity – Normally Closed | c | |
| 34 | Piezo Range 50 µm | Pf | |
| 41 | Piezo Range 70 µm | Pg | |
| 23 | Piezo Range 100 µm | P1 | |
| 35 | Piezo Range 150 µm | PS | Firmware default |
| 24 | Piezo Range 200 µm | P2 | |
| 36 | Piezo Range 300 µm | P3 | |
| 25 | Piezo Range 350 µm | P4 | |
| 37 | Piezo Range 500 µm | P5 | |
| 1 | XY Linear Encoders Used | L | |
| 2 | XY Rotary Encoders Used | R | |
| 3 | Z Linear Encoders Used | L | |
| 4 | Z Rotary Encoders Used | R | |
| 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 | |
| 71 | The joystick and knob can be disabled, and the device assignments can be changed. The JOYSTICK command works normally. | j | Firmware default. |
| | Fixed Profile | F | place holder profile |

| CCA X= | Description | Displa | Comment |
|---|---|---|---|
| | MicroMirror 6 degrees | U6 | |
| | MicroMirror 8 degrees | U8 | |
| | MicroMirror 10 degrees | UA | |

Example: **1CCA X=6** Sets to card #1's XY stage to 1.59mm pitch lead screws.
***:A***

Query: **1CCA X?**

***:A XY:RA*** Shows card #1's XY stage is rotary encoded, lead screw pitch A (1.59mm).

**2CCA X?**

***:A Z:RN*** Shows card #2's Z stage is rotary encoded and setup for 100µm/turn scope motor drive.

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

Example:

```
: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
```

***25 P 3 350um RANGE***

Format: **[Addr#]CCA Y = ###**

Function: Requires MOVETASK firmware, which is typically only included on XY stages. Sets number of extra move repetitions for all axes on the card. Default value is zero, so a MOVE command causes the system to initiate a single move to the given position. If *n > 0*, then the move will be initiated more than once as a means to achieve fine adjustment and a more stable landing. This parameter is saved in non-volatile memory by the **SS Z** command. A system restart is required in most cases for changes to take effect .It's a ***Card-Addressed command.***

Example: 1**CCA Y=3** All moves will be initiated four times.
***:A***

Format: **[Addr#]CCA Z = ###**

Function: Sets system configuration flags according to following table. It's a ***Card-Addressed command.***

| CCA Z= | Description | Display | Comment |
|---|---|---|---|
| 1 | X axis movement direction is positive (default). [+] | + | |
| 2 | X axis movement direction is negative. [-] | - | |
| 3 | Y axis movement is positive (default)[+] (Note: In the MS-4000, the default direction value for the Y axis is -1) | + | |
| 4 | Y axis movement is negative. [-] | - | |
| 5 | Z axis movement is positive (default)[+] | + | |
| 6 | Z axis movement is negative. [-] | - | |
| 7 | F axis movement is positive. [+] | + | |
| 8 | F axis movement is negative. [-] | - | |
| 9 | Disengage clutch [D] | D | |
| 10 | Engage clutch[E] | E | |
| 11 | Enable LCD display[O] | O | |
| 12 | Disable LCD display[F] | F | |
| 13 | Clocked rotation devices on 1st axis take shortest path, including wrapping between last and first position | S | Applied to both axes until 3.10 |
| 14 | Clocked rotation devices on 1st axis not allowed to wrap between last and first position | L | Applied to both axes until 3.10 |

| CCA Z= | Description | Display | Comment |
|---|---|---|---|
| 15 | Disable ADEPT piezo self test on startup | N | Version 2.5 and later |
| 16 | Enable ADEPT piezo self test on startup | C | Firmware default, V2.5 and later |
| 17 | Clocked rotation devices on 2nd axis take shortest path, including wrapping between last and first position | S | |
| 18 | Clocked rotation devices on 2nd axis not allowed to wrap between last and first position | L | |
| 20 | The joystick and knob are always enabled, and the device assignments cannot be changed. The JOYSTICK command has no effect. | J | |
| 21 | The joystick and knob can be disabled, and the device assignments can be changed. The JOYSTICK command works normally. | j | Firmware default. |
| 22 | Reverses joystick polarity of the card's first axis | r | V3.05 and later |
| 23 | Joystick polarity of card's first axis set to default | l | Firmware Default,V3.05 and later |
| 24 | Reverses joystick polarity of the card's second axis | r | V3.05 and later |
| 25 | Joystick polarity of card's first axis set to default | l | Firmware Default,V3.05 and later |

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

Format: **[Addr#]CCA Z?**

Function: Transmits string of sign characters for all active axes, then clutch and display status characters.
***Note 1:*** When the direction of an axis is negative, upper limit settings must be negative values, and lower limit settings must be positive values.

**Command: CUSTOMB (PLANAR_CORRECTION firmware module required)**
Shortcut: **CCB**
Format: **[Addr#]CCB Z=n**
Function: Planar correction functions. It's a ***Card-Addressed command.***
**1CCB Z=1**
**1CCB Z=2**
**1CCB Z=3**
Store current xyz location values x1,y1,z1…x3,y3,z3 respectively.

**1CCB Z=4** Calculate coefficients for planar correction function, and enable planar correction
**1CCB Z=5** Disable planar correction.
**1CCB 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.
**1CCB Z=7** Re-initialize to zero all Planar variables including x1,y1,z1…x3,y3,z3 and planar correction function coefficients. Disable planar correction.

**Command: DACK**

Shortcut: **D**

Format: **DACK [Axis]=[ratio in mm/sec]**

Function: Sets motor speed control ratio, in mm/sec, of movement per DAC count. A DAC count is a value change of one (1) in the 8-bit integer written to the motor speed control register. It's an ***Axis-Specific Command.*** 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.

**Command: DUMP**

Shortcut: **DU**

Format: **[Addr#]DUMP [X] [Y]**

Function: Dump internal buffers to terminal screen. **DU**, without arguments, dumps the Trajectory Buffer. **DU X** clears Trajectory Buffer. **DU Y** dumps Error Buffer. See the *Error Codes for TG-1000 and MS-2000 Diagnostics* section below. It's a ***Card-Addressed command.***

The TG-1000 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 move steps. For best results, restrict testing to a single axis at a time; otherwise information from multiple axes will be interleaved in the dump buffer. Any motion from any axis will write information into the dump buffer until it is full.

Examples: 1**DU 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

**1DU Y** Dumps Error Buffer on card #2

**Command: ENSYNC**

Shortcut: **ES**

Format: **ENSYNC [axis]=[position in mm]**…

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

**Command: EPOLARITY**

Shortcut: **EP**

Format: **EP [axis]=[-1 or 1]**…

Function: 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. It's an ***Axis-Specific Command.***

**Command: ERROR**

Shortcut: **E**

Format: **ERROR [axis]=[position in mm]**…

Function: This command sets the Drift Error setting. This setting controls the crossover position error (in millimeters) between the target and position at which the 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. It's an ***Axis-Specific Command.***

Examples: **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.

**Command:** **HALT**
Shortcut: **** (the backslash character)
Format: **HALT**
Function: This command will stop all active motors and other actuators too. It's usually a ***Broadcast command*** but can be used as a ***Non-Addressed Command*** as well. When addressed to a specific card, it stops motion on that card only. Note that to use as a Non-Addressed Command the full command **HALT** must be used instead of the shortcut ****, because **** is handled quickly in the command parser.

**Command:** **HERE**
Shortcut: **H**

Format: **HERE [axis]=[position in 1/10 microns]**…
Function: Assign the specified number to the axis's current position buffer. The unit of measurement is in tenths of microns. This defines the current position to be a specific distance from the origin (0), i.e., the origin may change. It's an ***Axis-Specific Command. DOES NOT WORK FOR PIEZOS AND MICROMIRRORS. DOES NOT WORK FOR "CLOCKED DEVICES" SUCH AS FILTER SLIDERS AND TURRETS.***

Reply: If there are no errors, the positive reply ***:A*** will be sent back from the controller.
Example: **H X=1234 Y=4321 Z**
***:A***
The X position will change to 123.4 microns from the origin, Y will change to 432.1 microns, and the Z will be zeroed.

**Command:** **HOME**
Shortcut: **!** (the exclamation point character)
Format: **HOME axis [axis] [axis]** …
Function: 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. It's an ***Axis-Specific Command.***

Reply: 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**.

**Command:** **INFO**
Shortcut: **`I`**
Format: **`I axis [Axis]`** …
Function: This command returns the current values of various variables and constants that control the way the specified axis performs, as well as its current status. It's an ***Axis-Specific Command.***

Example:
**`I X`**

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

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

**Command:** **JOYSTICK**
Shortcut: **`J`**
Format: **`JOYSTICK [axis]±`** or
**`JOYSTICK [axis] = [Manual Input #]`**
Function: This command enables (+) or disables (–) the input from the default manual control device for the axis (joystick or knob). If you specify an input device number *dev*, the axis specified will be connected to that input device. It's an ***Axis-Specific Command.***
The table below shows the valid device assignments:

| | | |
|---|---|---|
| 0 | NONE | |
| 1 | DEFAULT | |
| 2 | Joystick – X deflection | (X-axis default) |
| 3 | Joystick – Y deflection | (Y-axis default) |
| 4 | Standard Control Knob | (Z-axis default) |
| 5 | X-Wheel | (special hardware required) |
| 6 | Y-Wheel | " |
| 7 | ADC CH1 – For ADC_FOLLOW or ADC_LOCK operation. | |
| 8 | Foot switch | |
| 9 | JX and X-wheel combo | (special hardware required) |
| 10 | JY and Y-wheel combo | " |
| 11 | CRIFF knob | (used for CRIFF system) |
| 22 | Z-Wheel | (TG-1000 only, right side of joypod) |
| 23 | F-Wheel | (TG-1000 only, left side of joypod) |

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

Example:
```
J X+ Y+ Z-
:A
```
The above command enables the default X and Y joystick control and disables the Z control knob.

Example:
```
J X? Y?
:A X=2 Y=3
```
Here the query shows that the X & Y axes use the X & Y joystick driver.

To set a default value that can be saved in nonvolatile memory, add 100 to the argument. Note that doing so only enables you to save the setting to nonvolatile memory, but you must execute a **SAVESET** command to do so.

Example:
```
J X=105
:A
```
This makes X-Wheel the default X axis manual control device. This is a setting that can be saved with the **SAVESET** command.

Example:
```
J X?
:A X=2
J X=1
:A
System restart
J X?
:A X=2
J X=105
:A
J X?
:A X=5
1SS Z
:A
System restart
J X?
:A X=5
```

**Command:** **JSSPD**
Shortcut: **JS**
Format: [**Addr#]JSSPD [X=high] [Y=low] [Z=knob_speed] [F=xy_knobs_high] [T=xy_knobs_low]**
Function: This command sets the relative motor speed for maximum deflection of the joystick to the values specified. Values between 0.1 and 100 (%) are acceptable, 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 set the low speed and the high speed was given by 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.
It's a ***Card-Addressed command.***
Reply: If there are no errors, the positive reply "***:A***" will be sent back from the controller.
Example: **1JS X? Y?**
***:A X=80.000000 Y=3.000000***

**Command:** **KADC** **(For CRIFF and AF-DUAL Systems)**
Shortcut: **KA**
Format: **KA [Axis]=###**
Function: Adjusts a gain parameter in the servo loop where *n* is a signed integer. Use to change the polarity and gain of the feedback. Default *n=1*, (use *n=5* for PZ-2000 CRIFF). It's an ***Axis-Specific Command.***

Reply: ***:A*** is returned upon receipt of the command.
Example: **KA Z?** returns the current value.
***:A Z=1***

**Command:** **KD**

Shortcut: **KD**
Format: **KD [Axis]=###**
Function: Sets the servo derivative error term constant, the integer value kd. Usually set to zero (0). Especially useful when inertia is a factor to improve settling time and stability. It's an ***Axis-Specific Command.*** MOST USERS DO NOT NEED TO USE THIS FUNCTION!

**Command:** **KI**

Shortcut: **KI**
Format: **KI [Axis]=###** …

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

**Command: KP**

Shortcut: **KP**
Format: **KP [Axis]=###** …
Function: Sets the servo proportional error term constant, the integer value *kp*. Larger values of kp increase the stiffness of the response to loss of position, but decreases stability if set too large. MOST USERS DO NOT NEED TO USE THIS FUNCTION! It's an ***Axis-Specific Command.***

**Command: LCD (not supported in TG-1000)**

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

Format: **[Addr#]LED [X=0 to 100]**
Function: Sets the brightness of ASIs LED illuminator by generating PWM thru TTL out. TTL out mode should be set to '9' (i.e. `TTL y=9`). Enable out from the LED illuminator should be connected to TTL out on controller. This setting can be saved in non-volatile memory using the **SAVESET** command. The PWM frequency is 1.3KHz. It's a ***Card-Addressed command.***

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

**Command: LLADR**

Shortcut: **LL**

Format: **LLADR [Axis]=###** …
Function: Sets the address of the axis used by the low-level command set. The default values are **X=24**, **Y=25**, and **Z=26**. Some systems require **X=1**, **Y=2**, and **Z=3**. This setting can be saved in non-volatile memory using the **SAVESET** command. It's an ***Axis-Specific Command.***

**Command: LOAD (RING BUFFER Firmware Module Required)**
Shortcut: **LD**
Format: **LOAD [Axis]=###** …
Function: The **LOAD** function places a set of position coordinates in the next available internal ring-buffer memory location. The position values are expressed as floating point numbers representing tenths of a micron, the same as the MOVE command. If a **+** is specified instead of **=###**, then the current position of the

axis is stored in the ring buffer (as of firmware v2.81). 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** (see the **RBMODE** command). It's an ***Axis-Specific Command.***

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.

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

Shortcut: **LK**

Format: [**Addr#]LK [X] [Y] [Z] [F]**

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

**LK X** returns the current system state code.

**LK Y** returns current focus error signal.

**LK Z** Unconditionally advances to the next system state.

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

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

Shortcut: **LR**

Format: **[Addr#]LR [Z=lock_range]**

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

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

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

***:A 0.05***

**Command:** **MAINTAIN**

Shortcut: **MA**

Format: **MAINTAIN [Axis] = [0 to 3]**…

Function: The maintain command specifies the behavior of the controller after move

completion. Move commands complete when the stage moves to within the *finish error* tolerance of the target position (PCROS command). The actions for various *code* values are:
*code* = 0 [default] Post-move, when the controller detects drift from target specified by the *drift error* value, it will return the stage axis to the target several times (18) within a timeout period (~0.5 sec.) before declaring a move error code 60 and giving up further attempts.
*code* = 1 Post-move, the controller will indefinitely continue to try to reach target when drifts greater than the *drift error* are detected.
With *codes* 0 and 1, the motor drivers are turned off when the stage reaches the *finish error* tolerance.
*code* = 2 The motor drivers remain on and the servo loop remains active. (Version 8.5+)
*code* = 3 Drivers remain on and servos active for the post-move time set by the WAIT command. The system BUSY is released when the *finish error* tolerance is first achieved. Setting the WAIT time sufficiently long can stabilize post-move drifts during data recording, but then allow for less power consumption of the driver amplifiers when waiting between moves.
*code = 4* , is a Piezo only mode where a tunable speedup algorithm is applied to moves. The tuning is done by the user setting two parameters using the **PZ** command. The user sets up an intentional overshoot amount and a maximum time for the overshoot to be applied. When a move is initiated, the piezo moves towards the overshoot position until the maximum time is reached or else until the measured position (using the strain gauge) has reached the halfway point between the previous position and the intended (non-overshoot) position. Subsequently the command signal to the intended position is applied. Experimentally this can reduce settling time by 10-60%. The overshoot amount is set using **[#Addr]PZ T**, expressed in percent. The maximum time to move towards the overshoot position is set using **[#Addr]PZ F**, expressed in milliseconds. Mode 4 does not function with CRISP-enabled firmware.

This is an ***Axis-Specific Command.***

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

**Command:** **MOTCTRL**
Shortcut: **MC**
Format: **MOTOCTRL [Axis]±**
Function: This command enables (+) or disables (-) the controller's ability to control the motor of a certain axis. The motor control voltage is set to zero and the position feedback control is not monitored when the motor is in disable (-) mode. The electronics of the controller will attempt to keep the motor from moving while disabled, however, it should be noted that this is an open-loop brake control only, and any movement or drift is not corrected. When queried, the controller returns values of 1 or 0 representing enabled and disabled respectively.
Reply: If there are no errors, the positive reply **:A** will be sent back from the controller.
Example: **MC X+ Y+ Z-**

***:A***

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

**Command:** **MOVE**
Shortcut: **M**
Format: **MOVE [Axis]=[units 1/10 microns]**…
Function: Move one or more axis motors to an absolute *position*. The unit of measurement is in tenths of microns. If no *position* is specified, 0 (the origin) is assumed.
For devices with **CLOCKED POSITIONS** (turrets and filter wheels), the *position* is an integer value between one and the number-of-positions. It's an ***Axis-Specific Command.***

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

**Command:** **MOVREL**
Shortcut: **R**
Format: **MOVREL [Axis]= [units 1/10 microns]**…
Function: Move one or more axis motor a distance relative from its current position. This command is very similar to the **MOVE** command. The unit of measurement is also in tenths of microns. It's an ***Axis-Specific Command.***
Reply: A positive reply of ***:A*** is sent back when the command is received correctly. Reception of the reply does not mean the end of execution, and the command **STATUS** can be used to determine if the move has been completed.
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.

**Command:** **PCROS**
Shortcut: **PC**
Format: **PCROS [Axis]= [units mm]**…
Function: This command sets/displays the Finish Error setting, which controls when the motor algorithm routines will turn off. The setting controls the crossover position error (in millimeters) between the target and position at which the MFC-2000 and 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. It's an ***Axis-Specific Command.***
Example: **PC X=.00005 Y=.00002 Z=.00005**
***:A***

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

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

**Command:** **MULTIMV**
Shortcut: **MM**
Format**:** **[Addr#]MM [X=radius] [Y= speed] [Z= width] [F=mode] [T?]**
Function: The **MULTIMV** command allows several common multi-axis move patterns to be executed. Presently the patterns supported include circles and spirals. If users have other special requirements, they should contact ASI for assistance.
The command, without any arguments, initiates the multi-axis pattern move.
The patterns are initiated from the current stage position. The movement is parameterized in terms of the speed (feed rate) in mm/sec and pattern parameters. For circles, the radius in millimeters is the only required parameter. For spirals the width per spiral turn in millimeters is required as well as the maximum radius.
The mode is a bit-mapped character that determines the characteristics of the motion. The mode bits are used according to the following table.

| Bit | Set | Clear |
|---|---|---|
| 0 | Lead-in Move Used | No Lead-in Move |
| 1 | Controlled acceleration along path, set by ACCEL command, to programmed speed. | No controlled acceleration |
| 2 | Move pattern repeated indefinitely | Only single cycle of move pattern executed |

| | | |
|---|---|---|
| 3 | Reserved | |
| 4 | Reserved | |
| 5 | Reserved | |
| 6 & 7 | Motion pattern selector bits 7 & 6:<br>00 Reserved<br>01 Circle<br>10 Reserved<br>11 Spiral | |

This setting can be saved into non-volatile memory by issuing the **[Addr#]SSZ** command

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

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

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.

**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.

**3MM**
Initiates the patterns

**3MM**
The second time disables the routine

**Command: PEDAL (Requires Foot Pedal Hardware/Firmware)**
Shortcut: **PD**
Format: **[Addr#]PEDAL [X=distance] [Y=rate] [Z=multiplier]**
Function: This command sets/displays the dual-pedal footswitch controls for controllers with this feature. The command is set up as follows: X = Pedal Step Increment size, in millimeters. Y = Rate when pedal is held down, as an integer proportional to a speed in millimeters per second, Z = an integer multiplier used when the pedal controls a zoom axis. It's a ***Card-Addressed command.***

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

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

Example:
```
1PD X=0.02 Y=8 Z=5
:A
1PD X? Y?
:A X=0.02000 Y=8.00000
```

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

Shortcut: **RM**

Format: **[Addr#]RBMODE [X=control] [Y=axis_byte] [Z=buffer_pointer] [F=mode_byte]**

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

The command, without any arguments, performs the same operation that a TTL IN0 input pulse would control as determined by the current *IN0_mode*. See TTL command. It's a ***Card-Addressed 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 (v2.89+).

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)
- *mode_byte* (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 current position to end with delay between points set by **RT Z** (make sure delay is set appropriately; e.g. setting 1ms won't work with motorized stage).
    - 3 – Repeat autoplay mode. Upon a TTL pulse or **RM** command without arguments, plays from current position continuously in a loop with delay set by **RT Z** (make sure delay is set appropriately; e.g. setting 1ms won't work with

motorized stage). While running, another trigger causes autoplay to stop. Also enables repeat mode for ARRAY module.

- Bits 2-6 are reserved.
- MSB – read-only, set to 1 if ring buffer is autoplaying and 0 otherwise

- *axis_byte*: 1-7: Binary value determines which axes are commanded to move; the same axes' positions are reported using *IN0_mode*=5. Ordering of axes is generally the order on the card, with the first axis getting the LSB.
- *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.

**Command:** **RDADC**
Shortcut: **RA**
Format: **[Addr#]RA [X] [Y] [Z] [F]**
Function: Returns the present values on the MS2000's 4-channel ADC. The X and Y channels are used for the joystick. The Z and F channels may be used for special applications, e.g. Autofocus or **ADC_LOCK** and **ADC_FOLLOW** modes of controlling the stage. Special firmware is required for these applications. It's a ***Card-Addressed command.***
Example: 1**RA X Y**
***:A 128 128***

**Command:** **RDSBYTE**
Shortcut: **RB**
Format: **RDSBYTE axis [axis] [axis]**…
Function: Requests the TG-1000 and MS-2000 to respond with the Status Byte. It's an ***Axis-Specific Command.***
The number is one byte, which can be broken down into 8 bits that represent the following internal flags:
Bit 0: 0 = No commanded move is in progress. 1 = A commanded move is in progress. This bit is synonymous with the STATUS command. If the bit is set, then STATUS returns '**B**', otherwise STATUS returns '**N**'.
Bit 1: 0 = The axis is disabled. It can be renabled by one of the following: High Level command **MC <axis>+**, cycling the clutch switch for the Z-axis, Low Level StartMotor command (hex 47), or a system reset. This feature is available in versions 6.2c and later; 1 = The axis is enabled.
Bit 2: 0 = Motor is inactive (off), 1 = Motor is active (on).
Bit 3: 0 = Joystick/Knob disabled, 1 = Joystick/Knob enabled
Bit 4: 0 = Motor not ramping, 1 = Motor ramping
Bit 5: 0 = Ramping down, 1= Ramping up
Bit 6: Upper limit switch: 0 = open, 1 = closed
Bit 7: Lower limit switch: 0 = open, 1 = closed
Reply: ***:<byte as hexadecimal>***

Examples: **RB X**
***:<0x8A>***
**RB X Y**
***:<0x8A><0x02>***
The X-axis example value of 0x8A means the following:
B7: 1 -X Axis is at its lower limit
B6: 0 -X Axis upper limit switch open
B5: 0 -Ramping down, if ramping
B4: 0 -Motor not ramping
B3: 1 -Joystick/Knob is enabled
B2: 0 -Motor power is off.
B1: 1 -X Axis is enabled
B0: 0 -No commanded move is in progress

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

**Command:** **RDSTAT**
Shortcut: **RS**
Format: **RDSTAT axis [axis] [axis]**…
Function: Without any additional characters this is the same as **RDSBYTE**, except the data is returned in ASCII decimal format. It's an ***Axis-Specific Command.*** When a qualifier is appended to the axis letter the behavior is different for v2.8+. With **?**, a busy or not busy character is returned for that axis (N or B, just as in **STATUS**). With -, the left status character displayed on MS-2000 LCD of the axis is returned, including ***U*** or ***L*** for upper and lower limits, ***f*** or ***s*** for fast or slow joystick mode just selected, or a space for no event to report. With **+**, the right status character displayed on MS-2000 LCD is returned (with some additions), including ***M*** for move, ***B*** for commanded move (e.g. **HOME**), ***K*** for servo lock, ***S*** for spin move, ***A*** for single axis move, ***P*** for pause, or a space for no event to report.
Examples: **RS X**
***:A 138***
**RS X?**
***:A N***
**RS X-**
***:A s***
**RS X+**
***:A N***

**Command:** **RELOCK** **(For CRIFF and AF-DUAL Systems)**
Shortcut: **RL**
Format: **[Addr#]RL**

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

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

**Command:** **RESET**

Shortcut: **~**

Format: **RESET**

Function: This command causes the controller to do a software reset. A software reset reinitializes all variables back to their pre-assigned values. It's usually a ***Broadcast command*** but can be used as a ***Non-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.

Example: **RESET**
***:R***

**Command:** **RTIME**

Shortcut: **RT**

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

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

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

The Z argument sets the post-move delay time 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 CRIFF system.

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

**Command:** **RUNAWAY**

Shortcut: **RU**

Format: **[Addr#]RU X=n**

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

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

Example: 1**RU X=5** Sets runaway sensitivity to 5mm on all axes in card#1
***:A***

**Command: SAVESET**

Shortcut: **SS**

Format: **[Addr#]SAVESET Z** (saves settings to flash memory)
**[Addr#]SAVESET Y** (restores last saved settings, overwriting any changes since power-on)
**[Addr#]SAVESET X (**will reload factory defaults upon next power-up)

Function: SAVESET allows the user to save current parameters settings to non-volatile memory, where they will be restored after a power cycle. Examples of settings which are saved include motor control settings (speed, error) and single-axis settings. It's a ***Card-Addressed command.***

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.

Example: **2SS Z** Saves current settings to non-volatile memory on card#2.
***:A***

**Command: SAVEPOS**

Shortcut: **SP**

Format: **[Addr#]SP [X=inhibit]**

Function: 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. It's a ***Card-Addressed command.***

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:*** ***OK*** is not transmitted in TG-1000.

**Command:** **SCAN** **(SCAN firmware required)**
Shortcut: **SN**
Format: **`[Addr#]SCAN [X?] [Y=fast_axis_id] [Z=slow_axis_id] [F=pattern]`**
Function: Note multiple small changes in usage from MS-2000 command set. Without parameters, starts (or stops) the scan state machine using parameters set with the **`SCANR`** and **`SCANV`** commands and described there. 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.
It's a ***Card-Addressed command.***

| 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 |

**Command:** **SCANR** **(SCAN firmware required)**
Shortcut: **`NR`**
Format: **`[Addr#]SCANR [X=start] [Y=stop] [Z=enc_divide] [F=#_pixels]`**
Function: Sets up raster scan *start* and *stop* positions, with the position values expressed in millimeters. During scanning, the stage will move past both of these positions slightly, so that when scanning within the range specified, the scan proceeds with

uniform speed (set by the SPEED command). On units equipped with hardware position Sync, the output pulse goes high as the stage crosses the *start* position and low again when the stage crosses the *end* 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. It's a ***Card-Addressed command.***

**Command:** **SCANV (SCAN firmware required)**
Shortcut: **NV**
Format: **[Addr#]SCANV [X=start] [Y=stop] [Z=number_of_lines] [F=settle_time]**
Function: Sets up the slow-scan (vertical) *start* and *stop* positions, with the position values expressed in millimeters. The stage will move to the start position before beginning the scan. The scan range will be divided into *number_of_lines* lines. Following a completed horizontal scan, the stage will move vertically to the next scan line. The processes will conclude when the stage has moved to the vertical *stop* position and completed the last horizontal scan. If the start and stop values are identical then a 1-D scan will occur, repeated *number_of_lines* times.
The *settle_time* parameter is different in TG-1000 from the MS-2000. In TG-1000 it defines 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. The default value is 50ms.
It's a ***Card-Addressed command.***

**Command:** **SECURE (special hardware and U_SERVO_LK firmware module needed)**
Shortcut: **SECURE**
Format: **[Addr#]SECURE [X= p]**
Function: With stages equipped with Micro Servo lock mechanism, this command is used to lock or unlock samples on the stage. The value of **p** determines the position of the lever arm and can be any decimal number between 0.0 and 1.0. A value of 1.0 fully retracts the lever. The best value for a particular well plate model may vary and can be determined experimentally. It's a ***Card-Addressed command.***
Example: 1**SECURE X=1.0** (fully opens lever)
1**SECURE X=0.25** (closes lever for typical well plate)
Reply: ***:A***
**1SECURE**
***:N-3*** (Error at axis required )
**1SECURE Y=0**
***:N-2*** (invalid axis)
**1SECURE X?**
***:N-2*** (invalid operation )

**Command:** **SETHOME**
Shortcut: **HM**
Format: **HM [axis]=[position in mm]**…
Function: This command sets/displays a fixed hardware *HOME* location for an axis in units of millimeters. The *HOME* position is considered a fixed hardware location and is adjusted properly when the controller's coordinate system is altered with the **HERE** or **ZERO** function. The *HOME* position is automatically remembered and recalled through a power cycle and does not need to be saved using the **SAVESET** command. It's an ***Axis-Specific Command.***

Reply: If there are no errors, a reply of "***:A***" is returned.
Example: **HM X?**
***:A X=1000.000***
In the above example the default location for the *HOME* position for the X-axis is returned.

**Command:** **SETLOW**
Shortcut: **SL**
Format: **SETLOW [axis]=[position in mm]**…
Function: This command sets/displays the lower firmware limit switch for an axis. The *Limit* positions are considered fixed hardware locations and are adjusted properly when the controller's coordinate system is altered with the **HERE** or **ZERO** function. The *Limit* positions are automatically remembered and recalled through a power cycle and do not need to be saved using the **SAVESET** command. It's an ***Axis-Specific Command.***

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.
***Note 4:*** As of firmware v2.8, **SL [axis]+** will set the lower limit to be the current position. Restore the default limit by executing **SL [axis]-**.

**Command:** **SETUP**
Shortcut: **SU**
Format: **SU [axis]=[position in mm]**…

Function: Same as **SETLOW** command (see above) but for upper firmware limit switch. It's an ***Axis-Specific Command.***

***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 firmware v2.8, **SU [axis]+** will set the upper limit to be the current position. Restore the default limit by executing **SU [axis]-**.

**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.

Shortcut: **SI**

Format: **SI [axis]=[position in 1/10 microns]**…

Function: This Command searches for the physical centers of the stage and marks it with a user inputted value. Software limits are reset to default. It's an ***Axis-Specific 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.

(For rotary encoders: **SEEK LIMITS** firmware required -- Rotary Encoder Stages)

Format: **SI [axis] = [1 or -1]**…

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

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

**Send SI command.**

**Poll with STATUS command until 'N' is received.**

**Send HERE command with desired real world position.**

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

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

**Command:** **SPEED**
Shortcut: **S**
Format: **SPEED [axis]=[max speed in mm/sec]**…
Function: Sets the maximum speed at which the stage will move during 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). Speed is set in millimeters per second. Maximum speed is = 7.5 mm/s for standard 6.5 mm pitch leadscrews.
Reply: If there are no errors, a positive reply of "***:A***" is sent back. It's an ***Axis-Specific Command.***
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.

**Command:** **SPIN**
Shortcut: **@**
Format: **SPIN [axis]=[-128 to 128]**…
Function: Tells controller to 'spin' the motor of specified axis at a rate expressed as its DAC value, a bit value from -128 to 128. It's an ***Axis-Specific Command.***
Reply: If there are no errors, a positive reply of "***:A***" is sent back.
Example:
```
@ X=100 Y=-100 Z
:A
```
This example shows a command that will instruct the X-axis turn at a motor rate of 100 DAC bits in one direction, the Y-axis at the same rate but in the other direction, and stop any rotation or motion of the Z-axis.

***Note:*** To stop rotation, give a value of zero, or just the type the axis letter without an assignment as shown in the example above, or use the **HALT** ( **** ) command.
The **HALT** command will not return an ***:N-21*** when stopping a **SPIN** command.

**Command:** **STATUS**
Shortcut: **/**
Format: **STATUS**
Function: Inquires regarding the motor status of all axes. Queries the controller whether or not any of the motors are still busy moving following a serial command. Using the shortcut / is the preferred method for rapid polling of the controller for a busy state. The / is handled quickly in the command parser. It's a ***Broadcast Command.***
Reply: The positive reply can come in two forms:
***N*** - there are no motors running from a serial command
***B*** - there is a motor running from a serial command
Example:
```
MOVE X=12345
:A
STATUS
B
/
N
```

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

**Command:** **STOPBITS (not supported in TG-1000)**
Shortcut: **SB**
Format: **[Addr#]STOPBITS X=n**
Function: Sets the number of stop bits, *n*, to be used for RS232 serial communication. The default is one (1) stop bit; the other option is two (2) stop bits. Use the **SAVESET Z** command to retain the new stop bit setting after power off. Not supported in TG-1000.

**Command:** **TTL (only partly implemented in TG-1000)**
Format: **[Addr#]TTL [X=IN0_mode] [Y=OUT0_mode] [Z=aux_IO_state] [F=OUT0_polarity] [R=aux_IO_mask] [T=aux_IO_mode]**
Function: The MS2000 controller has a buffered TTL input (*IN0*) and output (*OUT0*) port as well as several unbuffered I/O ports. The signals *IN0* and *OUT0* are found on the board connector SV1 pin1 and 2 respectively. On many controllers these signals are connected to the IN and OUT BNC connectors on the back of the controller. The *IN0_mode* and *OUT0_mode* parameters set with this command determine the character of the I/O pins. It's a ***Card-Addressed command.***
*IN0_mode*: **0** - turns off TTL IN0 controlled functions; TTL interrupt DISABLED.
**1** – TTL IN0 initiates a Move-to-Next-Position of the stored positions in the Ring Buffer pointed to by the *buffer_pointer*. When the *buffer_pointer* reaches a value equal to the number of saved positions, it resets to the first position, allowing cyclic repetitions to the saved locations. See RBMODE and LOAD command.

**2 -** TTL IN0 repeats most recent relative move (See **MOVREL**) For example, begin a session by issuing the command **MOVREL X=0 Y=0 Z=0.5**, and each subsequent move to Next Position will cause the Z axis to move 0.05 micron. This function can be used for repetitive relative moves of any axis or combination of axes. You may directly set the *dZ* value with the **ZS** command's X parameter.
**3** – TTL IN0 initiates an autofocus operation on systems with autofocus installed.
**4** – enables TTL IN0 controlled Z-stacks. (See **ZS** command).
**5** – enables TTL IN0-started position reporting via the serial interface. Information is asynchronously sent out the serial interface every *report_time* interval, where *report_time* is set with the RT command. Data returned in the serial stream are the elapsed time in milliseconds since the TTL trigger, followed by the position of each axis enable by the *axis_byte*. On TRACKING systems, the PMT sum signal is also reported. Reporting is toggled on and off by the TTL input pulse.
**6** – TTL interrupt ENABLED; use with TTL triggered position reporting.
**7** – TTL commanded ARRAY move to next position.
**9** – Used with CRISP focus lock. TTL IN0 HIGH engages lock if system is in READY state. TTL IN0 LOW will cause system to UNLOCK is locked already.

*OUT0_mode*: **0** – TTL OUT0 unconditionally set LOW.
**1** – TTL OUT0 unconditionally set HIGH.
**2** – generates TTL pulse at end of a commanded move (**MOVE** or **MOVREL**). The pulse duration is set with command **RT Y=???.**
**3** – output TTL OUT0 gated HIGH during axis index 0 (X) constant speed move.
**4** – output TTL OUT0 gated HIGH during axis index 1 (Y) constant speed move.
**5** – output TTL OUT0 gated HIGH during axis index 2 (Z) constant speed move.
**8** – TTL OUT0 timed arrival pre-pulse output. See RT command. Requires PREPULSE firmware module.
**9** – TTL OUT0 PWM and MicroServo Output. See the LED or the SECURE command. Requires LED_DIMMER or USERVO firmware module
10 – TTL OUT0 set during SPIM state machine operation. Requires MM_SPIM firmware module.
**11 –** TTL OUT0 set at the end of a ring buffer move or **AIJ**-initiated move (for laser trigger). Requires MM_TARGET firmware module.

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

*aux_IO_state*: Requires TTL_AUXILIARY firmware module; behavior depends on the firmware build and hardware present. Sets the state of the auxiliary TTL output according 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). 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 a TTL input pulse is simulated and action will be taken depending on the setting of *IN0_mode*. Default setting for TTL_AUXILIARY on piezo firmware.

**Command:** **UM**
Shortcut: **`UM`**
Format: **`UM [axis]= ###`** …
Function: Specifies the multiplier for most serial commands such as **`MOVE`** and **`WHERE`**. Default values are 10000 (/mm), setting the default input scaling to 0.1μm/count. The sign of the Units Multiplier can be used to change the relative direction of motion for commanded moves. However, using the "**`CCA Z`**" command is the recommended procedure for changing the stage direction. The Units Multiplier can be saved with the "**`SS Z`**" command. It's an ***Axis-Specific Command.***

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

**Command:** **UNITS** **(not supported in TG-1000)**

**Command:** **UNLOCK** **(For CRIFF or AF-DUAL Systems)**
Shortcut: **`UL`**
Format: **`[addr#]UL`**
Function: This command unlocks the servo from the focus system and returns control to encoder feedback from the Z-axis drive. The CRIFF laser is turned off and the CRIFF system is placed in the **`Laser_OFF`** state. Current CRIFF lock reference values are saved for eventual use by the **`RELOCK`** command. It's a ***Card-Addressed command.***
Reply: ***`:A`*** is returned upon receipt of the command.

**Command:** **VB** **(Z and F arguments only for TG-1000)**
Shortcut: **`VB`**
Format: **`[addr#]VB [Z=read_decimal_places] [F=###]`**
Function: The Z argument sets the number of decimal places for the **`WHERE`** 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). It only applies to the COMM card (and defaults to the COMM card address, so the address is not needed). The syntax state does not persist when power is turned off.

**Command:** **VECTOR**
Shortcut: **VE**
Format: **VE [axis]=[speed in mm/sec]**…
Function: The **VECTOR** command causes the stage to immediately ramp up to the velocity value specified by the command. The command arguments are expressed in units of mm/sec. The stage will continue indefinitely at the commanded velocity until the controller receives another command. A value of zero for the velocity component will halt motion on that axis. The controller will accelerate the stage to the commanded velocity at the rate specified by the ACCEL and SPEED commands until the commanded velocity is obtained. It's an ***Axis-Specific Command.***

**Command:** **VERSION**
Shortcut: **V**
Format: **[addr#]VERSION**
Function: Requests controller to report which firmware version it is currently using. It's a ***Card-Addressed command.***
Example: **1V**
***:A v2.4***

**Command:** **WAIT**
Shortcut: **WT**
Format: **WAIT [axis]=[time in msec]**…
Function: Sets the length of time *msec*, in milliseconds, the controller will pause at the end of a move. The busy status is not cleared during this pause state, unless the MAINTAIN behavior for the axis is set to code=3. Additionally, a "**P**" is displayed on the LCD display when in the Pause state. During the Pause state, the servo loop remains actively attempting to position the axis on target. It's an ***Axis-Specific Command.***
Example: **wt x?**
***:X=0 A***
**WT X=20**
***:A***

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

**Command:** **WHERE**
Shortcut: **W**
Format: **WHERE axis [axis] [axis]**…

Function: Returns the current position of the device for the axis specified. It's an ***Axis-Specific 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 card address. Hence, the ordering of the responses does not necessarily follow the order of the query.

The reporting precision of the **WHERE** command can be changed with the **VB Z** command.

**Command: WHO (TG-1000 v1.6+)**

Shortcut **N**

Format: **WHO**

Function: In TG-1000 it prints all the card address, axis characters, build name and compile date, of all cards installed in the system. It's a ***Comm-default Command.***

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
```

**Command: WRDAC (not supported on TG-1000)**

Format: **[addr#]WRDAC [axis]=###**

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

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

Example
```
1WRDAC X=1.1
:A          (Voltage on PIN SV1-5 is 1.1Volts)

WRDAC X=20 OR -1
:N-4        (Parameter out of range)
```

**Command: ZERO**

Shortcut: **Z**

Format: **ZERO**

Function: Writes a zero to the position buffer of all axes. Allows the user to set current position as the origin. It's a ***Broadcast 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**

**Command:** **Z2B**
Format: **Z2B axis=[new axis letter ascii code]**…
Function: Allows the user to change the axis name for a motor axis. The *current_axis_letter* must be one of the motor axes names listed with the "**BU X**" command. The *new_axis_letter_ascii_code* must be the decimal ASCII code for the desired axis name for letters between upper case 'A'(65) and 'Z'(90). For the change to take effect, the new setting must be saved to flash memory using "**SS Z**", followed by a hardware reset. The new axis name will remain in effect unless default settings are restored to the controller. It's an ***Axis-Specific Command.***
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** (change to "**B**" axis name)
**:A**
**2SS Z** (required to save new name setting to flash. 2 because Z is on card#2)
**:A**

**Command:** **ZF** **(requires ZFLOCK module)**
Shortcut: **ZF**
Format: **[addr#]ZF [X=0 or 1] [Y=0 or 1]**
Function: When enabled the controller slaves one of the axis to another, resulting in the slave axis mirroring all of the master axis's moves. Issuing ZF serial command enables or disabling the slaving. The designation of the master and slave can be swapped with the Y argument. Setting Y to 0, makes Z axis the master and F the slave. Setting Y to 1, makes F axis the slave and F the master. It's a ***Card-Addressed command.***

Reply: If there are no errors, a positive reply of **:A** will be returned.
Example: **2ZF X=1** (Enables the slaving; slave axis will mirror all of the master axis's moves)
**:A**
**2ZF X=0** (Disables the slaving; slave axis will not mirror master axis's moves, and behave as an independent axis.)
**:A**
**2ZF Y=0** (Makes Z axis the master and F the slave.)
**:A**
**2ZF Y=1** (Makes F axis the slave and F the master.)
**:A**

**Command:** **ZS**

Shortcut: **`ZS`**

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

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

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

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

Example: **`1ZS X=10 Y=20 Z=1`**

**`:A`**

Setup to do twenty 1 micron slices with triangular pattern.

## *Error Codes for TG-1000 Diagnostics*

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

| Error Number* | Error Description |
|---|---|
| 1-9 | OVERTIME – **RECOVERABLE.** Error caused by competing tasks using the microprocessor. |
| 10-12 | OVERSHOT – Move overshot the target; happens frequently, not really an error. |
| 15 | NEGATIVE LOG – Negative number for Log conversion. |
| 20-22 | AXIS DEAD – **FATAL.** No movement for 100 cycles; axis halted. |
| 24 | ENCODER_ERROR |
| 30-32 | EMERGENCY STOP – **FATAL.** Getting further from the target; axis halted. |
| 34 | UPPER LIMIT – Upper Limit reached. (axis unspecific) |
| 35 | LOWER LIMIT – Lower Limit reached. (axis unspecific) |
| 40-42 | PULSE PARAMETER VALUES OUT OF RANGE – code error. |
| 44 | FINISH SPEED CLAMP – Reached the maximum allowed move-finishing speed. |
| 45 | ADC_LOCK_OOR – Out-of-range error on ADC input. |
| 46 | ADC_FOLLOW_ERR – Error attempting to follow an analog ADC input. |
| 50-52 | ENCODER ERROR OVERFLOW – **FATAL.** Error term so large that move intent is indiscernible; axis halted. |
| 55 | EPROM NO LOAD – Saved-settings on EPROM not loaded, compile date mismatch. |
| 60-62 | ADJUST-MOVE ERROR – Failed to clear '**M**' soon enough. **FATAL** |
| 85 | SCAN LOST PULSES – During a scan, missing pulses were detected. |
| 86 | SCAN INCOMPLETE – During a scan, terminated before completing the row. |
| 90-92 | ERROR_LARGE – **RECOVERABLE.** Error large. Motor set to FULL SPEED; hope to catch up. |
| 100-102 | INDEX NOT FOUND |
| 140 | PIEZO WRITE DAC – Error writing to the piezo DAC. |
| 141 | PIEZO READ DAC – Error reading from piezo DAC |
| 142 | PIEZO READ POS |
| 143 | PIEZO WRITE POS |
| 144 | PIEZO MOVE ERR |
| 145 | PIEZO READ POS1 |
| 146 | PIEZO INIT |
| 147 | PIEZO POS ERROR |
| 148 | Autofocus 200um safety limit Encountered |
| 149 | I2C_BAD_BUSY ERROR |
| | |
| 173 | I2C_AXIS_ENABLE_ERR1 |
| 174 | I2C_AXIS_ENABLE_ERR2 |
| 175 | I2C_AXIS_MUTE1_ERR |
| 176 | I2C_AXIS_MUTE2_ERR |

| | |
|---|---|
| 203 | I2C_NACK_ERROR |
| 205 | ERR_TTL_MISMATCH  I2C bus error. |
| 255 | 10 MINUTE CLOCK – Provides time reference for error dump list. |
| | |
| 300 | Autofocus Scan failed due to insufficient contrast |
| 302 | Clutch Disengaged, Engage clutch to do Autofocus |

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

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

**RECOVERABLE** errors do not stop the controller from attempting to complete a commanded move. Large numbers of recoverable errors should be taken as a warning. Frequent servo errors (numbers 90-92) often mean that the speed is near or exceeding the stage maximum. Frequent overtime errors (numbers 1-9) often mean that competing processes, such as over-frequent serial status requests, are using too much CPU time.

## *Change log*

| | | |
|---|---|---|
| 5/20/13 | Branched from MS2000 manual | Vik |
| 5/21/13 | TG-1000 Prog started guide, integrated into this doc | Vik |
| 6/20/2013 | New options to **CCA Z** | Vik |
| 7/2/2013 | Major edits to commands to better represent TG-1000s expectations | Vik |
| 7/26/2013 | Expanded on Tiger banner, and other minor edits | Vik |
| 7/30/2013 | Build command improved | Vik |
| 3/18/2014 | Adjusted **RDSTAT** cmd function desp a bit | Vik |
| 4/4/2014 | Added TG-1000 addresses table | Vik |
| 6/26/2014 | Elaborated on CCA X? | Vik |