

# Wishbone/Low Level Commands for TG1000

The TG-1000 controller can be interfaced through High Level commands(HL cmd) and W Commands(W cmd). HL cmd use ASCII characters, human readable, very verbose, easy to construct. W cmd are binary and ideal for hardware to hardware communication. In fact all internal communication between Tiger Comm card and Device cards are done in W cmd. No special operation is required to switch from HL cmd to W cmd, Tiger Comm automatically detects and processes them accordingly.

This document describes the packet structures of W commands and replies, gives examples, and lists the command set in detail. A complete and functional W command includes a card address and a W command packet.

The MS-2000 has its own set of commands, see the [MS-2000 Low Level Commands](#) documentation.

## Document Conventions

In this document, the terms device and card are defined such that devices are mechanical objects connected to cards by cables. Thus one or more devices, e.g., filterwheels or stage axis motors, may be connected to each card.

Non-readable data characters are represented in this document as hexadecimal values in the formats 0xFF and #FF, may be delimited by a following space when part of a character string. The space character is represented by its byte value: #20. Readable characters appear as themselves or enclosed in single quotes, e.g., A. For example, the following string includes the data characters 0xFF, 1, A, and 0xD:

#FF 1A#0D or #FF1A#0D

This string may also be represented this way: #FF #31 #41 #0D or #FF#31#41#0D

## W Command and Reply Structure

All W command packets are the structure below. They contain an Address byte, Command set ID, Command ID, Argument length. Argument optional and depend on the command itself.

**Table 1: W Command Structure**

Card Address 1 byte (0x30 for Comm, 0xFE for broadcast, 0x31.. for device cards)	Command Set ID 1 byte 0xD7 for W cmd	Command ID 1 byte	Argument length 1 byte	Argument 0-251 bytes
--	--------------------------------------	-------------------	------------------------	----------------------

**Table 2: TG-1000 addresses**

Addressee	Usage	Value
TG-1000 Comm	Hard coded, re-assignable	0x30 ('0')
Stage/ FW/Shutter	Unique device address	0x31 to 0x39

Addressee	Usage	Value
Reserved	Reserved for future use, including future broadcast commands	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
	Tiger bus address	0xFF

Address 0x30 to 0x39 and then 0x81 to 0xF5 are unique addresses. At any given time, only one card in the controller can have them. When a W command is sent, all cards receive it and parse it, but will not act unless the address matches. If the user likes to address multiple cards with the same Command, then he can use the Broadcast command.

Argument length denotes the number of bytes to follow in the packet. W command receivers use the argument length byte to count the remaining incoming characters. When that many characters have been received, processing begins.

## Reply Structure

The W reply packet has two elements: the outcome byte and the reply data. The outcome byte is one of the values specified in the table below.

Outcome	Reply data
1 byte	0...∞ bytes

**Table 3: W Reply Structure**

Character	Hex	Description
ENQ	0x05	The argument length byte does not match the specified value for that command.
ACK	0x06	The command is well-formed, and execution has begun.
BEL	0x07	The argument length byte value exceeds the capacity of TG-1000's input buffers.
NAK	0x15	One of the following has occurred: the cmd id byte is undefined; or an argument is not in the specified value range, or the command is not recognized by the addressed device.
CAN	0x18	An intercharacter timeout (2ms) expired before the expected number of bytes was received.

## Command Set

Table 4 W user Commands and Replies describes all W commands with Command Set ID 0xD7 . The commands described in Table 5 W Internal Command and Replies are for use on the backplane bus and not recommended for transmission from an external host. Table 6 Planned W commands, not implemented.

## Move Axis Absolute

<b>ID Hex</b>	0x01
<b>Normal recipient</b>	Stage
<b>Argument</b>	5 bytes
	<b>Byte 1</b> axis selector 0..3.
	<b>Byte 2-5</b> destination position given in 1/10 microns, an IEEE-754 single precision floating point number.
<b>Reply</b>	1 byte
	<b>Byte 1</b> ACK or NAK for out of range argument

Move axis to given position. On the XY controller, axis[0] is the X axis. On the ZF controller, axis[0] is the Z axis.

**Example:**

```
#31#D7#01#05#00#46#40#E4#01
#06
Moves first axis on card #1 by 1234.5 microns
```

## Move Axis Relative

<b>ID Hex</b>	0x02
<b>Normal recipient</b>	Stage
<b>Argument</b>	5 bytes
	<b>Byte 1</b> axis selector 0..3.
	<b>Byte 2-5</b> destination position given in 1/10 microns, an IEEE-754 single precision floating point number.
<b>Reply</b>	1 byte
	<b>Byte 1</b> ACK or NAK for out of range argument

Move axis to given position relative to its current position.

**Example:**

```
#31#D7#02#05#01#C6#40#E4#01
#06
Rel moves 2nd axis on card #1 by -1234.5 microns
```

## Spin Axis

<b>ID Hex</b>	0x03
<b>Normal recipient</b>	Stage

<b>Argument</b>	2 bytes
	<b>Byte 1</b> axis selector 0..3.
<b>Reply</b>	<b>Byte 2</b> motor power, a signed character in range -128 to 127, where 0=no power. Format is 2s complement.
	1 byte

Apply motor power to an axis

**Example:**

#31#D7#03#02#00#32

#06

Moves first axis on card #1 by at 50% total power in positive direction

#31#D7#03#02#00#CE

#06

Moves first axis on card #1 by at 50% total power in negative direction

## Set Axis Position

<b>ID Hex</b>	0x04
<b>Normal recipient</b>	Stage
<b>Argument</b>	5 bytes
	<b>Byte 1</b> axis selector 0..3.
<b>Reply</b>	<b>Byte 2-5</b> the position given in 1/10 microns, an IEEE-754 single precision floating point number.
	1 byte

Coerces current axis position

**Example:**

#31#D7#04#05#00#46#40#E4#01

#06

Sets first axis on card #1 position as 1234.5 microns

## Halt

<b>ID Hex</b>	0x08
<b>Normal recipient</b>	Stage
<b>Argument</b>	none

<b>Reply</b>	none
--------------	------

Halt all movement of all axes.

Since there is no reply, this command can be broadcast to all stages.

**Example:**

#31#D7#08#00

Halts all movement on all axis in just card #1

#FE#D7#08#00

Halts all movement on all stage class cards in controller

## Get Status and Position

<b>ID Hex</b>	0x0A
<b>Normal recipient</b>	Stage
<b>Argument</b>	1 bytes
	<b>Byte 1</b> axis selector 0..3.
	6 bytes
	<b>Byte 1</b> ACK
<b>Reply</b>	<b>Byte 2</b> Status (bits layout similar to RDSBYTE)
	<b>Byte 3-6</b> Position given in 1/10 Microns, an IEEE-754 single precision floating point number.

It's a function of the axis unit multiplier, set with the HL cmd "UM", default value is 10,000 or mm/10000.

**Note: Pre v2.7, units were millimeters.**

**Example:**

Card#1's 1st axis when idle

#31#D7#0A#01#00

#06#0A#00#00#00#00

Card#1's 1st axis while active

#31#D7#0A#01#00

#06#0F#46#40#E3#B4

## Get Status

<b>ID Hex</b>	0x0C
---------------	------

<b>Normal recipient</b>	Stage
<b>Argument</b>	None
<b>Reply</b>	1 byte
	<b>Byte 1</b> 'N' or 'B'

Same as STATUS command.

**Example:**

Card#1 is idle  
 #31#D7#0C#00  
 #4E (0x4E hex for N)

Card#1 is active  
 #31#D7#0C#00  
 #42 (0x42 hex for B)

## Set Resolution

<b>ID Hex</b>	0x0D
<b>Normal recipient</b>	Stage
<b>Argument</b>	1 bytes <b>Byte 1</b> 0..3 for number of decimal points.
<b>Reply</b>	1 byte <b>Byte 1</b> ACK or others

Selects decimal point of WHERE command in high level command set.

Default setting is tenths of a micron resolution.

**Example:**

#31#D7#0D#01#03  
 #06

```
w x
:A 12344.700
```

## Get Axis Names

<b>ID Hex</b>	0x0E
<b>Normal recipient</b>	Stage
<b>Argument</b>	none

<b>Reply</b>	4 to 6 byte
	<b>Byte 1</b> ACK
	<b>Byte 2</b> Number of valid axis names to follow
	<b>Byte 3</b> Axis 0 name
	<b>Byte 4</b> Axis 1 name
	<b>Byte 5</b> Axis 2 name
	<b>Byte 6</b> Axis 3 name

Requests names of the axes supported by a stage card. Also returns number of valid axes, given by Byte 1 of the reply.

**Example:**

STD XY card on address 1  
 #31#D7#0E#00  
 #06#02#58#59 (0x58 ASCII for X and 0x59 is Y)

A 4ch MicroMirror card on address 2  
 #32#D7#0E#00  
 #06#04#50#51#52#53 (0x50 to 0x53 is P,Q,R and S)

## Get Single Axis Position

<b>ID Hex</b>	0x0F
<b>Normal recipient</b>	Stage
<b>Argument</b>	1 bytes
	<b>Byte 1</b> axis selector 0..3.
<b>Reply</b>	4 byte
	<b>Byte 1-4</b> Axis position given in 1/10 microns, an IEEE-754 single precision floating point number.

It's a function of the axis unit multiplier, set with the HL cmd "UM", default value is 10,000 or mm/10000.

**Note: Pre v2.7, units were millimeters.**

**Example:**

Card#1 X is at 1234.4 microns and Y is at -1234.5 microns  
 #31#D7#0F#01#00  
 #46#40#E3#B4 (12344.92578125)  
 #31#D7#0F#01#01  
 #C6#40#E2#D2 (-12344.705078125)

## Get Device Class

<b>ID Hex</b>	0x14
<b>Normal recipient</b>	Any
<b>Argument</b>	none
	2 bytes
<b>Reply</b>	<b>Byte 1</b> ACK <b>Byte 2</b> : device type: 0=Comm; 1=Stage; 2=Filterwheel; 3=Shutter; 4=LCD; 255=no device (reply timeout)

Queries each card directly, about what class it is.

**Note: Piezo drive cards, micro mirror drive cards are also classed as stage cards.**

**Note: If no device is present at the given bus address, then no reply is sent.**

### Example:

In a Controller with 2 stage cards and 1 comm card

```
#30#D7#14#00
#06#30
#32#D7#14#00
#06#31
#31#D7#14#00
#06#31
#33#D7#14#00
(no reply)
```

## Get Device Map Element

<b>ID Hex</b>	0x16
<b>Normal recipient</b>	Comm
<b>Argument</b>	none
	1 byte
<b>Reply</b>	<b>Byte 1</b> ACK <b>Byte 2</b> bus address <b>Byte 3</b> device type ('0' (0x30)=Comm; '1' (0x31)=Axis; '2' (0x32)=Filterwheel; '3' (0x33)=Shutter; '4' (0x34)=LCD)

Queries the Comm card for its list of cards and their class present in the controller. Each time the command is given, Comm card moves down the list, prints the card address and its class. When it runs out of cards, it starts back at the top.

**Note: Piezo drive cards, micro mirror drive cards are also classed as stage cards.**

**Example:**

In a Controller with 2 stage cards and 1 comm card

#30#D7#16#00  
 #06#30#30 (first time, reports itself)  
 #30#D7#16#00 (note cmd is always directed to omm.. Card)  
 #06#31#31 (second time card#1 details are sent)  
 #30#D7#16#00  
 #06#32#31 (3rd time card#2 details are sent)  
 #30#D7#16#00  
 #06#30#30 (starts back at the top)  
 #30#D7#16#00  
 #06#31#31  
 #30#D7#16#00  
 #06#32#31  
 #30#D7#16#00  
 #06#30#30

---

**Get Number of Devices**

<b>ID Hex</b>	0x17
<b>Normal recipient</b>	Comm
<b>Argument</b>	none
<b>Reply</b>	2 bytes
	<b>Byte 1</b> ACK
	<b>Byte 2</b> number of devices listed in the device map.

Host command queries the Comm card for total number of card of all classes present in the controller.

**Example:**

In a Controller with 2 stage cards and 1 comm card

#30#D7#17#00  
 #06#03 (Answer 3 cards)

---

**Get Stage Axis Settings**

<b>ID Hex</b>	0x19
<b>Normal recipient</b>	Stage
<b>Argument</b>	1 bytes <b>Byte 1</b> axis selector 0..3.

<b>Reply</b>	23 bytes	
	<b>Byte 1</b>	ACK
	<b>Byte 2-5</b>	max speed in mm/sec, an IEEE-754 single precision floating point number.
	<b>Byte 6-9</b>	backlash in mm, an IEEE-754 single precision floating point number.
	<b>Byte 10-13</b>	drift error in mm, an IEEE-754 single precision floating point number
	<b>Byte 14-17</b>	finish error in mm, an IEEE-754 single precision floating point number
	<b>Byte 18-19</b>	ramp time in ms, a 16-bit unsigned integer.
	<b>Byte 20</b>	1=pointing device X movement controls this axis; 0=pointing device X movement does not control this axis.
	<b>Byte 21</b>	1=pointing device Y movement controls this axis; 0=pointing device Y movement does not control this axis.
	<b>Byte 22</b>	1=pointing device scroll wheel movement controls this axis; 0=pointing device scroll wheel movement does not control this axis.
	<b>Byte 23</b>	Encoder polarity. 0=negative (left side Z); 1=positive (right side Z)

Host command to stage controller. Gets Speed (S), Backlash (B), Drift error (E), Finish error (PC), Ramp time (AC), Mouse controls (X, Y, or Scroll) and encoder polarity

**Example:**

```
#31#D7#19#01#00
#06#40#B7#DE#93#3D#23#D7#0A#39#D1#B7#17#37#CB#42#4B#00#64#00#00#00#01
#31#D7#19#01#01
#06#40#B7#DE#93#3D#23#D7#0A#39#D1#B7#17#37#CB#42#4B#00#64#00#00#00#01
#31#D7#19#01#02
#06#15 (NACK as no 3rd axis)
```

Breaking down the reply and analysing

```
#06 (ACK)
#40#B7#DE#93 Speed is 5.74591970443726 mm/s
#3D#23#D7#0A Backlash is 0.0399999991059303 mm
#39#D1#B7#17 Drift Error is 0.00039999998989515 mm
#37#CB#42#4B Finish error is 2.42303558479762e-05 mm
#00#64 Ramp Time is 100 ms
#00
#00
#00
#01 Positive encoder polarity
```

---

## Move Filterwheel

<b>ID Hex</b>	0x1A
<b>Normal recipient</b>	Filterwheel

<b>Argument</b>	2 bytes	
	<b>Byte 1</b>	Wheel selector. Value: 0...1
	<b>Byte 2</b>	Filter selector. Values 0...7
<b>Reply</b>	1 byte	
	<b>Byte 1</b>	ACK

Host to Filterwheel command

**Example:**

#32#D7#1A#02#00#05

#06

Moves FW #0 to position 5

## Move Shutter

<b>ID Hex</b>	0x1B	
<b>Normal recipient</b>	Shutter	
<b>Argument</b>	2 bytes	
	<b>Byte 1</b>	Shutter selector. Value = 0...1
	<b>Byte 2</b>	Energize shutter=1, De-energize shutter = 0
<b>Reply</b>	1 byte	
	<b>Byte 1</b>	ACK

Host to shutter command

## Display Filterwheel Address

<b>ID Hex</b>	0x1C	
<b>Normal recipient</b>	Filterwheel	
<b>Argument</b>	none	
<b>Reply</b>	1 byte	
	<b>Byte 1</b>	ACK

Host to Filterwheel. Causes Filterwheel front panel 7 segment display to display device's bus address.

**Example:**

#FE#D7#1C#00

Causes all FW class cards to display their Address

## Restore Filterwheel Display

<b>ID Hex</b>	0x1D
<b>Normal recipient</b>	Filterwheel
<b>Argument</b>	none
	1 byte
<b>Reply</b>	<b>Byte 1</b> ACK

Host to Filterwheel. Used after Display Filterwheel Address command. Causes Filterwheel to restore its panel 7 segment display to the state it was in before the Display Filterwheel Address command was issued.

## Get Number of Axes

<b>ID Hex</b>	0x1E
<b>Normal recipient</b>	Stage
<b>Argument</b>	none
	2 bytes
<b>Reply</b>	<b>Byte 1</b> ACK
	<b>Byte 2</b> number of axes supported on this card

Host to stage controller. Returns number of axes supported on this card.

### Example:

#31#D7#1E#00  
#06#02 (2 axes)

## Save Filterwheel Settings

<b>ID Hex</b>	0x1F
<b>Normal recipient</b>	Filterwheel
<b>Argument</b>	none
	1 byte
<b>Reply</b>	<b>Byte 1</b> ACK

Host to Filterwheel. Causes Filterwheel controller to its write current settings to non-volatile memory.

## Write Filterwheel Settings to RAM

<b>ID Hex</b>	0x20
<b>Normal recipient</b>	Filterwheel
	12 bytes
<b>Argument</b>	<b>Byte 1-4</b> Channel 0 Filterwheel offset, a signed long integer <b>Byte 5</b> Channel 0 speed value <b>Byte 6-9</b> Channel 1 Filterwheel offset, a signed long integer
http://asiimaging.com/docs/	Printed on 2026/01/20 02:20

<b>Reply</b>	1 byte
	<b>Byte 1</b> ACK

Host to Filterwheel. Writes some current Filterwheel settings (see argument) to volatile memory.

## Read Filterwheel Settings from RAM

<b>ID Hex</b>	0x21	
<b>Normal recipient</b>	Filterwheel	
<b>Argument</b>	none	
	13 bytes	
	<b>Byte 0</b>	ACK
	<b>Byte 1-4</b>	Channel 0 Filterwheel offset, a signed long integer
	<b>Byte 5</b>	Channel 0 speed value
	<b>Byte 6-9</b>	Channel 1 Filterwheel offset, a signed long integer
	<b>Byte 10</b>	Channel 1 speed value
		Shutter normal state
<b>Reply</b>	<b>Byte 11</b>	<b>Bit 0</b> 0=Shutter 0 normally open; 1=Shutter 0 normally closed
		<b>Bit 1</b> 0=Shutter 1 normally open; 1=Shutter 1 normally closed
		<b>Bit 2-3</b> not used
		<b>Bit 4</b> 1=Shutter controller (SH2 card) is connected at this address; 0=No SH2 card is connected at this address.
		<b>Bit 5-7</b> not used
	<b>Byte 12</b>	number of currently operable wheels attached

Host to Filterwheel. Transmits settings from RAM to host.

## Confirm Halt

<b>ID Hex</b>	0x24
<b>Normal recipient</b>	Internal
<b>Argument</b>	none

<b>Reply</b>	1 byte	
	<b>Byte 1</b>	ACK = all axes halted, none were in motion when Halt command was given, or this is not the first Confirm Halt command issued since the most recent Halt command, or no Halt commands have been issued since the last system reset; NAK = at least one axis failed to halt; ENQ = a Halt command was issued prior to this command, and an axis was in motion when that Halt command was issued.

Comm to Stage Confirms that Halt command was executed successfully and stage is stopped. This command is invoked automatically when the H command HALT is transmitted from Host to Comm. If the result indicates that an axis failed to halt, then the system is automatically reset and all motors are shut down.

---

## Zero Axis

<b>ID Hex</b>	0x25
<b>Normal recipient</b>	Stage
<b>Argument</b>	1 bytes
	<b>Byte 1</b> axis selector 0..3.

  

<b>Reply</b>	1 byte
	<b>Byte 1</b> ACK or NAK for out of range argument

Comm to Stage Duplicates H command ZERO and MS-2000 Zero button function. Causes readjustment of limits.

### Example:

```
#31#D7#25#01#00
Zeros just 1st axis on card#1 #FE#D7#25#01#00
Zeros all 1st axes on all cards of stage class.
```

---

## Get Axis Types

<b>ID Hex</b>	0x26
<b>Normal recipient</b>	Stage
<b>Argument</b>	none

<b>Reply</b>	3 bytes	
	<b>Byte 1</b>	ACK
	<b>Byte 2</b>	Axis 0 type, where 0=no axis present; 1=XY; 2=motor-driven focus; 3=piezo-driven focus; 4=motor-driven zoom; 5=theta
	<b>Byte 3</b>	Axis 1 type, defined same as Byte 1

Comm to Stage

Reports whether each axis is an XY, motor-driven focus, or piezo-driven focus axis.

Users may also want to look at 0x4A Get Axis Kinds too. It is better implemented.

## Get Axis Kinds

<b>ID Hex</b>	0x4A
<b>Normal recipient</b>	Stage
<b>Argument</b>	none
<b>Reply</b>	4 to 6 bytes
	<b>Byte 1</b> ACK
	<b>Byte 2</b> Total axis on card (and number of bytes to follow)
	<b>Byte 3</b> Axis 0 type
	<b>Byte 4</b> Axis 1 type
	<b>Byte 5</b> Axis 2 type
	<b>Byte 6</b> Axis 3 type
<b>Axis Type Short</b>	<b>Description</b>
x	XY stage
z	Z focus motor drive. LS50s, Z scopes etc
p	Piezo Focus. ASIs ADEPT, Piezo DAC etc
o	Objective Turret
f	Filter Changer
t	Theta Stage
l	Generic linear motorized stage, TIRF, SISIYOU etc
a	Generic linear piezo stage
m	Zoom magnification motor axis
u	Micro Mirror, Scanner 75 etc
w	Filter Wheel
s	Shutter
u	Unknown axis type

Queries the Device card. Replies with total axes present, followed by ASCII codes for each axis representing which kind of axis.

**Example:**

```
#31#D7#4A#00
#06#02#78#78 (0x78 is x for xymotor)
```

```
#32#D7#4A#00
#06#04#75#75#75#75 (0x75 is u for MicroMirror)
```

## Get Axis Props

<b>ID Hex</b>	0x4B
<b>Normal recipient</b>	Stage
<b>Argument</b>	none
<b>Reply</b>	4 to 6 bytes
	<b>Byte 1</b> ACK
	<b>Byte 2</b> Total axis on card (and number of bytes to follow)
	<b>Byte 3</b> Axis 0 properties
	<b>Byte 4</b> Axis 1 properties
	<b>Byte 5</b> Axis 2 properties
	<b>Byte 6</b> Axis 3 properties
<b>Bit 0</b>	CRISP auto-focus firmware
<b>Bit 1</b>	RING BUFFER firmware
<b>Bit 2</b>	SCAN firmware
<b>Bit 3</b>	ARRAY firmware
<b>Bit 4</b>	SPIM firmware
<b>Bit 5</b>	SINGLEAXIS and/or MULTIAxis firmware
<b>Bits 6-7</b>	reserved

Queries the Device card. Replies with total axes present, followed by byte for each axis representing any special properties or capabilities (usually would be firmware module) such as CRISP or RING BUFFER.

**Example:**

```
#34#D7#4B#00
#06#02#0A#0A (xystage with RING BUFFER and ARRAY)
```

```
#33#D7#4B#00
#06#04#10#10#10#10 (Micromirror with SPIM)
```

## Set Stage Axis Settings

<b>ID Hex</b>	0x27
---------------	------

<b>Normal recipient</b>	Stage	
<b>Argument</b>	23 bytes	
	<b>Byte 1</b>	axis selector 0..3.
	<b>Byte 2-5</b>	max speed in mm/sec, an IEEE-754 single precision floating point number
	<b>Byte 6-9</b>	backlash in mm, an IEEE-754 single precision floating point number
	<b>Byte 10-13</b>	drift error in mm, an IEEE-754 single precision floating point number.
	<b>Byte 14-17</b>	finish error in mm, an IEEE-754 single precision floating point number.
	<b>Byte 18-19</b>	ramp time in ms, a 16-bit unsigned integer.
	<b>Byte 20</b>	1=pointing device X movement controls this axis; 0=pointing device X movement does not control this axis.
	<b>Byte 21</b>	1=pointing device Y movement controls this axis; 0=pointing device Y movement does not control this axis.
	<b>Byte 22</b>	1=pointing device scroll wheel movement controls this axis; 0=pointing device scroll wheel movement does not control this axis.
	<b>Byte 23</b>	Encoder polarity. 0=negative (left side Z); 1=positive (right side Z)
<b>Reply</b>	1 byte	
	<b>Byte 1</b>	ACK

Comm to Stage. Host command to stage controller. Counterpart to Get Axis Settings command. Sets Speed (S), Backlash (B), Drift error (E), Finish error (PC), Max lim (SU), Min lim (SL), Ramp time (AC), Mouse controls (X, Y, or Scroll), and Encoder polarity (EPOL).

**Example:**

To set 1st axis to 2mm/sec

```
#31#D7#27#17#00#40#00#00#00#3D#23#D7#0A#39#D1#B7#17#37#CB#42#4B#00#64#00
#00#00#01
#06
```

---

## Save Settings Stage

<b>ID Hex</b>	0x28
<b>Normal recipient</b>	Stage
<b>Argument</b>	none
<b>Reply</b>	1 byte

**Byte 1** ACK

Writes current stage settings to non-volatile memory

**Example:**

```
#31#D7#28#00
```

#06

saves settings of just card #1 to non volatile memory

#FE#D7#28#00

#06

saves settings of all cards in controller to nonvolatile memory.

## Get Saved Settings Stage

<b>ID Hex</b>	0x29
<b>Normal recipient</b>	Stage
<b>Argument</b>	none
<b>Reply</b>	1 byte <b>Byte 1</b>  ACK

Reads stage settings from non-volatile memory into stage volatile memory, overwriting current settings.

## Restore Stage Defaults

<b>ID Hex</b>	0x2A
<b>Normal recipient</b>	Stage
<b>Argument</b>	none
<b>Reply</b>	1 byte <b>Byte 1</b>  ACK

Marks stage non-volatile memory as unsaved. Next stage reset, the settings will be the original factory defaults.

## Restore Filterwheel Defaults to RAM

<b>ID Hex</b>	0x2B
<b>Normal recipient</b>	Filterwheel
<b>Argument</b>	none
<b>Reply</b>	1 byte <b>Byte 1</b>  ACK

Writes default settings to Filterwheel RAM.

## Read Filterwheel Settings to RAM

<b>ID Hex</b>	0x2C
<b>Normal recipient</b>	Filterwheel

<b>Argument</b>	none
<b>Reply</b>	1 byte
	<b>Byte 1</b> ACK

Host to Filterwheel. Reads settings from non-volatile memory to Filterwheel RAM.

---

## Reset Stage

<b>ID Hex</b>	0x2D
<b>Normal recipient</b>	Stage
<b>Argument</b>	none
<b>Reply</b>	1 byte

Host to stage. Invokes stage software reset.

### Example:

```
#31#D7#2D#00
resets just card #1.
#FE#D7#2D#00
resets all stage class cards in the controller
```

---

## Ping

<b>ID Hex</b>	0x2F
<b>Normal recipient</b>	All
<b>Argument</b>	none
<b>Reply</b>	1 byte

Replies ACK. Used to verify that sender has sent a command readable to the receiver. May be used to seek serial baud rate match.

---

## Set Clutch

<b>ID Hex</b>	0x31
<b>Normal recipient</b>	Stage
<b>Argument</b>	1 byte
	<b>Byte 1</b> 0=disengage; 1=engage
<b>Reply</b>	1 byte

Engages or disengages clutch.

---

## Get Stage Settings And Flags

<b>ID Hex</b>	0x32
<b>Normal recipient</b>	Stage
<b>Argument</b>	none
<b>Reply</b>	9 bytes
	<b>Byte 1</b> ACK
	<b>Byte 2</b> XY pitch
	<b>Byte 3</b> Z pitch
	<b>Byte 4</b> Where command format
	<b>Byte 5</b> X & Y encoder flag, where 'L' = linear, 'R' = rotary
	<b>Byte 6</b> Clutch engaged
	<b>Byte 7</b> 1st Axis, X or Z axis profile
	<b>Byte 8</b> 2st Axis, Y or F axis profile
	<b>Byte 9</b> Knob speed

Return states of system flags.

<b>Some common pitches</b>	
PITCH_A_FINE	'A', (0x41)
PITCH_B_COARSE	'B', (0x42)
PITCH_C_ULTRA_COARSE	'C', (0x43)
PITCH_25NM	'H', (0x48)
PITCH_NORM_Z	'N', (0x4E)
PITCH_D_ULTRA_FINE	'U', (0x55)
SD_ACTUATOR	'X', (0x58)
PITCH_ZEISS_Z	'Z', (0x5A)
GTS_A_FINE	'a', (0x61)
GTS_B_COARSE	'b', (0x62)
GTS_C_ULTRA_COARSE	'c', (0x63)

<b>Some common profiles</b>	
STANDARD_XY	0x00
STANDARD_Z	0x01
STD_CP_ROT	0x02
STD_FP_ROT	0x03
STD_CP_LIN	0x04
STD_FP_LIN	0x05
UCP_ROT	0x06
UUCP_ROT	0x07
UFP_ROT	0x08
UUFP_ROT	0x12
UFP_LIN	0x14
UCP_LIN	0x22

Some common profiles	
UUCP_LIN	0x23
SCOPE_STD_Z	0x0a
SCOPE_LIN_Z	0x0b
SD_XLATE	0x0f
PIEZ0_PROFILE	0x10
MM_PROFILE	0x2b

**Example:**

```
#31#D7#32#00
#06#42#46#97#52#00#02#02#05
```

Breaking down and analysing the reply.

```
#06 (ACK )
#42 (B pitch)
#46 (ignore, no z)
#97#52 (R for rotary)
#00#02 (#2 profile, std xy)
#02 (#2 profile, std xy)
#05 (knob speed)
```

## Set Joystick/Mouse Speeds

<b>ID Hex</b>	0x35
<b>Normal recipient</b>	Stage
<b>Argument</b>	3 bytes
	<b>Byte 1</b> Slow joystick speed
	<b>Byte 2</b> Fast joystick speed
	<b>Byte 3</b> Blank (used to be knob speed, however knob speed is handled differently now)
<b>Reply</b>	1 byte
	<b>Byte 1</b> ACK or others

Sets slow and fast joystick speed for XY. This command does the same things as the H cmd JS.

**Example:**

To set joystick slow at 20% and fast at 80%

```
#31#D7#35#03#14#50#00
#06
include a 3rd byte leave it at 0x00 to appease the controller
```

## Get Mouse Speeds

<b>ID Hex</b>	0x36								
<b>Normal recipient</b>	Stage								
<b>Argument</b>	none								
	4 byte								
<b>Reply</b>	<table border="1"> <tr> <td><b>Byte 1</b></td><td>ACK</td></tr> <tr> <td><b>Byte 2</b></td><td>Slow joystick speed</td></tr> <tr> <td><b>Byte 3</b></td><td>Fast joystick speed</td></tr> <tr> <td><b>Byte 4</b></td><td>Blank (used to be knob speed, however knob speed is handled differently now)</td></tr> </table>	<b>Byte 1</b>	ACK	<b>Byte 2</b>	Slow joystick speed	<b>Byte 3</b>	Fast joystick speed	<b>Byte 4</b>	Blank (used to be knob speed, however knob speed is handled differently now)
<b>Byte 1</b>	ACK								
<b>Byte 2</b>	Slow joystick speed								
<b>Byte 3</b>	Fast joystick speed								
<b>Byte 4</b>	Blank (used to be knob speed, however knob speed is handled differently now)								

Returns settings made by Set Stage Mouse Speeds command.

**Example:**

#31#D7#36#00  
#06#14#50#00 (slow at 20% and fast at 80%)

## Set Encoder Polarity

<b>ID Hex</b>	0x37				
<b>Normal recipient</b>	Stage				
<b>Argument</b>	2 bytes				
	<table border="1"> <tr> <td><b>Byte 1</b></td><td>axis selector 0..3.</td></tr> <tr> <td><b>Byte 2</b></td><td>values are 1 and -1. (2s complement)</td></tr> </table>	<b>Byte 1</b>	axis selector 0..3.	<b>Byte 2</b>	values are 1 and -1. (2s complement)
<b>Byte 1</b>	axis selector 0..3.				
<b>Byte 2</b>	values are 1 and -1. (2s complement)				
<b>Reply</b>	<table border="1"> <tr> <td>1 byte</td><td></td></tr> <tr> <td><b>Byte 1</b></td><td>ACK</td></tr> </table>	1 byte		<b>Byte 1</b>	ACK
1 byte					
<b>Byte 1</b>	ACK				

Default value is 1 for left-hand Z drive.

Sets encoder polarity. Left- or right-hand Z drives need this setting.

**Example:**

#31#D7#37#02#00#FF  
#06 (sets card#11st axis as epol as -1)  
#31#D7#37#02#00#01  
#06 (sets card#11st axis as epol as 1)

**Note:** Save settings to nonvolatile memory(0x28) and restart controller for settings to take affect.

## Get Encoder Polarity

<b>ID Hex</b>	0x38
<b>Normal recipient</b>	Stage
<b>Argument</b>	1 bytes
	<b>Byte 1</b> axis selector 0..3.
	2 bytes
<b>Reply</b>	<b>Byte 1</b> ACK
	<b>Byte 2</b> 0xFF for -1 , 0x01 for 1

Returns encoder polarity setting (see Set Encoder Polarity command)

**Example:**

#31#D7#38#01#00  
#06#FF (card#1 1st axis has negative encoder polarity)

## Set Encoder Type

<b>ID Hex</b>	0x39
<b>Normal recipient</b>	Stage
<b>Argument</b>	1 bytes
	<b>Byte 1</b> NUL =Rotary; anything else =Linear
<b>Reply</b>	1 byte
	<b>Byte 1</b> ACK or NAK for out of range argument

This setting informs the firmware about the hardware configuration with respect to encoders. Then turn the controller OFF/ON for settings to take effect.

**Example:**

#31#D7#39#01#01  
#06  
Sets all axis on card#1 to linear enc mode  
#31#D7#39#01#00  
#06  
Sets all axis on card#1 to rotary enc mode  
Note: Restart controller, for settings to take affect.

## Get Encoder Type

<b>ID Hex</b>	0x3A
---------------	------

<b>Normal recipient</b>	Stage
<b>Argument</b>	none
	2 bytes
<b>Reply</b>	<b>Byte 1</b> ACK
	<b>Byte 2</b> 0x52 , R for Rotary , 0x4C, L for Linear

Gets encoder type—linear or rotary

**Example:**

#31#D7#3A#00  
#06#52 (card#1 is in rotary encoder mode)

## Home Filterwheel

<b>ID Hex</b>	0x3D
<b>Normal recipient</b>	Filterwheel
<b>Argument</b>	1 byte
	<b>Byte 1</b> : 0 or 1, Filterwheel selector
<b>Reply</b>	1 byte
	<b>Byte 1</b> ACK or NAK for out of range argument

Corresponds to Filterwheel HO command.

## Get Firmware Version

<b>ID Hex</b>	0x3F
<b>Normal recipient</b>	Any
<b>Argument</b>	none
<b>Reply</b>	varies
	<b>String</b> ACK or NAK for out of range argument

Returns a String containing the version number

**Example:**

On a stage card #31#D7#3F#00  
#76#32#2E#37 (in ASCII is “v2.7”)  
On a filter wheel card  
#32#D7#3F#00  
#56#65#72#73#69#6F#6E#3A#20#76#31#2E#32#0A (in ascii is “Version: v1.2<CR>”)

## Set Default Manual Input Device

<b>ID Hex</b>	0x40
<b>Normal recipient</b>	Stage
<b>Argument</b>	<p>2 bytes</p> <p><b>Byte 1</b> axis selector 0..3.</p> <p>device selector, Some important ones            0x00 = NONE            0x02 = Joystick - X deflection            0x03 = Joystick - Y deflection            0x05 = X-Wheel            0x06 = Y-Wheel            0x09 = JX and X-wheel combo            0x0A = JY and Y-wheel combo            0x16 = Z-Wheel            0x17 = F-Wheel</p>
<b>Reply</b>	<p>1 byte</p> <p><b>Byte 1</b> ACK or NAK for out of range argument</p>

Sets type of device to be used for manual movements, i.e., analog joystick or Knobs. Writes settings to nonvolatile memory. Note: For safety, this command disables all movement. Also appropriate firmware modules must also be present on the card

### Example:

So switch axis to X and Y knobs.

```
#31#D7#40#02#00#05
#06
#31#D7#40#02#01#06
#06
```

## Get Default Manual Input Device

<b>ID Hex</b>	0x41
<b>Normal recipient</b>	Stage
<b>Argument</b>	<p>1 bytes</p> <p><b>Byte 1</b> axis selector 0..3.</p>
<b>Reply</b>	<p>2 bytes</p> <p><b>Byte 1</b> ACK or NAK for out of range argument</p> <p><b>Byte 2</b> see <a href="#">Set Manual Input Device</a></p>

Returns which kind of manual input device is used, i.e., analog joystick or knobs etc

**Example:**


---

```
#31#D7#41#01#00
#06#02 (1st axis is joystick x)
#31#D7#41#01#01
#06#03 (2nd axis is joystick y)
```

---

**Set Axis Speed**

<b>ID Hex</b>	0x43
<b>Normal recipient</b>	Stage
<b>Argument</b>	5 bytes
	<b>Byte 1</b> axis selector 0..3.
	<b>Byte 2-5</b> max speed in mm/sec, an IEEE-754 single precision floating point number.
<b>Reply</b>	1 byte
	<b>Byte 1</b> ACK or NAK for out of range argument

Sets speed of a single axis

**Example:**

```
set 1st axis to 2mm/sec
#31#D7#43#05#00#40#00#00#00
#06
```

---

**Set Encoder Counts Per Mm**

<b>ID Hex</b>	0x44
<b>Normal recipient</b>	Stage
<b>Argument</b>	8 bytes
	<b>Byte 1-4</b> for axis 0, counts per millimeter, an IEEE-754 single precision floating point number
	<b>Byte 5-8</b> for axis 1, counts per millimeter, an IEEE-754 single precision floating point number
<b>Reply</b>	1 byte
	<b>Byte 1</b> ACK or NAK for out of range argument

Sets both axes' encoder counts per millimeter.

**Example:**

Set Card#1 1st and 2nd axis to 60000 counts/mm

---

```
#31#D7#44#08#47#6A#60#00#47#6A#60#00
#06
```

---

## Get Encoder Counts Per Mm

<b>ID Hex</b>	0x45
<b>Normal recipient</b>	Stage
<b>Argument</b>	none
<b>Reply</b>	9 byte
	<b>Byte 1</b> ACK
	<b>Byte 2-5</b> encoder counts for axis 0, in IEEE 754 format
	<b>Byte 6-9</b> encoder counts for axis 1, in IEEE 754 format

Gets encoder counts per millimeter for the specified axis

**Example:**

---

```
#31#D7#45#00
#06#47#6A#60#00#47#6A#60#00 (#47#6A#60#00 is 60,000)
```

---

## Joystick XY data

<b>ID Hex</b>	0x46
<b>Normal recipient</b>	Stage
<b>Argument</b>	5 bytes
	<b>Byte 1</b> X joystick value, -127 to 127
	<b>Byte 2</b> Y joystick value, -127 to 127
	no reply

Sends current joystick X & Y values from Comm to Stage. Each value denotes the deflection of the joystick relative to its position at rest (center). Behaves like the spin command. Handy to simulate joystick/button in UI.

**Example:**

```
#FE#D7#46#02#40#40
```

All axis in controller that have joystick as manual input, start moving.

```
#FE#D7#46#02#CE#CE
```

All axis in controller that have joystick as manual input, now move in opposite direction

```
#FE#D7#46#02#00#00
```

All axis in controller that have joystick as manual input, stop moving.

## Button data

<b>ID Hex</b>	0x47
<b>Normal recipient</b>	Stage
<b>Argument</b>	2 bytes
	Button byte with bits defined as follows, its active low
	<b>Bits 0-3</b> undefined
	<b>Bits 4</b> Zero button state
	<b>Bits 5</b> Home button state
	<b>Bits 6</b> At (@) button state
	<b>Bits 7</b> Joystick button (fast/slow)
	Clutch byte with bits defined as follows:
	<b>Bits 0-5</b> undefined
	<b>Bits 6</b> Clutch switch state
	<b>Bits 7</b> undefined
<b>Reply</b>	no reply

Sends current button state from Comm to Stage.

Command has to be sent two twice, Once indicating press and the release. Based on the time interval between press and release commands, the stage card will conclude if it's a short press, or long press etc.

Note: For broadcast used Address 0xF6

## Example:

Zero button press and release for just Card #1

#31#D7#47#02#E0#00 #31#D7#47#02#F0#00

Home button press for all stage cards in controller

#F6#D7#47#02#D0#00 #F6#D7#47#02#F0#00

## Knob data

<b>ID Hex</b>	0x48
<b>Normal recipient</b>	Stage
<b>Argument</b>	4 bytes
	<b>Byte 1-2</b> Signed integer left knob value
	<b>Byte 3-4</b> Signed integer right knob value.
<b>Reply</b>	no reply

Sends left and right knob rotation values from Comm to Stage.

**Example:**

```
#FE#D7#48#04#00#FF#00#FF
Moves all axis that respond to knobs.
```

---

**Get Tiger Banner**

<b>ID Hex</b>	0x49
<b>Normal recipient</b>	All
<b>Argument</b>	none
<b>Reply</b>	varies
	<b>String</b> TIGER_BANNER string followed by ETX message terminator.

Gets TIGER\_BANNER string from device and relays it to host.

**Example:**

```
#32#D7#49#00
#41#74#20#33#32#3A#20#5A#3A#5A#4D#6F#74#6F#72#2C#46#3A#5A#4D#6F#74#6F#72
#20#76#32#2E#37#20#53#54#44#5F#5A#46#20#4A#75#6C#20#33#30#20#32#30#31#33
#3A#31#36#3A#30#39#3A#35#31#03
in ASCII its At 32: Z:Zmotor,F:Zmotor v2.7 STD_ZF Jul 30 2013:16:09:51<ETX>
```

---

**Set Axis Direction**

<b>ID Hex</b>	0x4C
<b>Normal recipient</b>	Stage
<b>Argument</b>	2 bytes
	<b>Byte 1</b> axis selector 0..3.
	<b>Byte 2</b> values are 1 and -1. (2s complement)
<b>Reply</b>	1 byte
	<b>Byte 1</b> ACK

Sets Axis direction. Setting is automatically saved into non volatile memory. Does not need a system reset. Default is 1 or positive direction.

**Example:**

```
#31#D7#4C#02#00#FF
#06 (sets card#1 1st axis direction as -1 or negative)
#31#D7#4C#02#00#01
#06 (sets card#1 1st axis direction as positive or 1)
```

## Get Axis Direction

<b>ID Hex</b>	0x4D
<b>Normal recipient</b>	Stage
<b>Argument</b>	1 byte
	<b>Byte 1</b> axis selector 0..3
	2 bytes
<b>Reply</b>	<b>Byte 1</b> ACK
	<b>Byte 2</b> 0xFF for -1 , 0x01 for 1

Returns axis direction setting

### Example:

#31#D7#4D#01#00  
#06#FF (card#1 1st axis direction is negative)

## W Internal Command and Replied

### Set Filterwheel Number

<b>ID Hex</b>	0xFA
<b>Normal recipient</b>	Filterwheel
<b>Argument</b>	2 bytes
	<b>Byte 1</b> Unsigned char, the assigned number for wheel 0
	<b>Byte 2</b> Unsigned char, the assigned number for wheel 1
<b>Reply</b>	no reply

After internally assigning each filterwheel its number for FW0, FW1,... FWn commands and before sending Get Tiger Banner command to filterwheel, Comm sends this command so that the filterwheel can reply to Get Tiger Banner command showing its assigned number.

## High Level Cmd

<b>ID Hex</b>	0xFC
<b>Normal recipient</b>	internal
<b>Argument</b>	11 bytes
	<b>Byte 1-2</b> gCmd [0x12 Move, 0x21 Zero, etc]
	<b>Byte 3</b> gOp [0x00 none, 0x01 read, 0x02 write, 0x06 plus, 0x07 minus]
	<b>Byte 4</b> gSubCmd [0x01 entry, 0x02 exit, 0x03 pseudoaxis, 0x04 non rad, 0x05 read]
	<b>Byte 5</b> gAxisChar [0x58 X, 0x59 Y, etc]
	<b>Byte 6-9</b> gNum , IEEE 754 format 4byte float

Reply	no reply
-------	----------

Internal ASI command: copies TG-1000 Comm parser state to TG-1000 stage card.

**Example:**

m x=123 generates

#31#D7#FC#0B#00#12#02#04#58#42#F6#00#00#04#01

---

## H Get gNum

<b>ID Hex</b>	0xFD
<b>Normal recipient</b>	unused

Internal ASI command: request gNum to be copies from TG-1000 stage card to TG-1000 command card

---

## Get Gerror

<b>ID Hex</b>	0x50
<b>Normal recipient</b>	Stage/Internal

Gets an error code following an High Level command. For inhouse use only

[serial](#), [tiger](#), [tech note](#)

From:

<http://asiimaging.com/docs/> - Applied Scientific Instrumentation



Permanent link:

[http://asiimaging.com/docs/tiger\\_w\\_commands](http://asiimaging.com/docs/tiger_w_commands)

Last update: **2025/05/22 15:44**