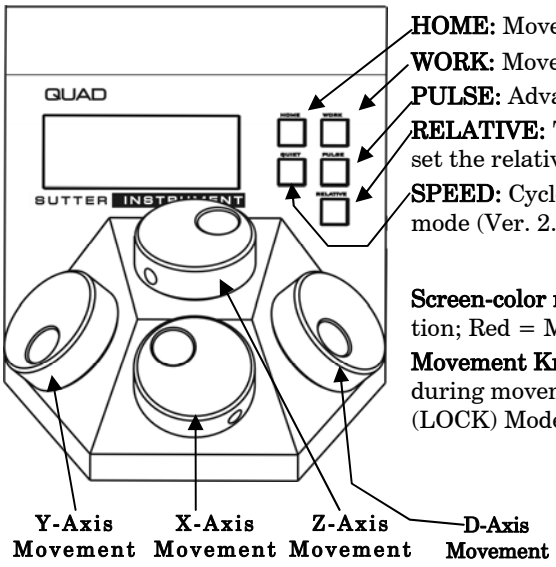
QUAD® Four-Axis Micromanipulator System
Quick Reference

Rev. 2.32 (20190131) (FW v2.3+)

Manual Operation



HOME: Move to defined home position. Press again to pause/resume.

WORK: Move to defined work position. Press again to pause/resume.

PULSE: Advances diagonal axis in 2.85 μm steps.

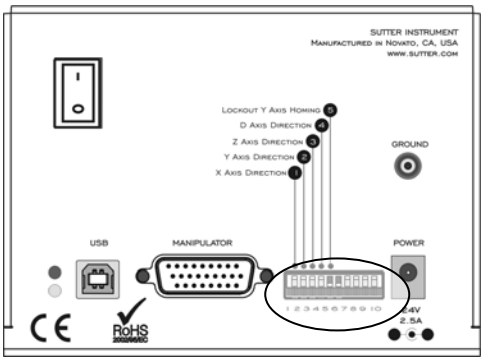
RELATIVE: Toggles between **Relative** and **Absolute** position moves. Hold 3-sec. to set the relative mode origin to the current absolute position.

SPEED: Cycles through Speed 0 (normal) through 3. Hold 3 sec. to enter **LOCK** mode (Ver. 2.3+).

Screen-color mode indications: Green = Absolute position; Blue = Relative position; Red = Movement in progress or in quiet (**LOCK**) mode; knobs disabled.

Movement Knobs Disabling and Quiet (LOCK) Mode: Movement knobs are disabled during movement to Home, Work, external movement command, or while in quiet (LOCK) Mode.

Configuration



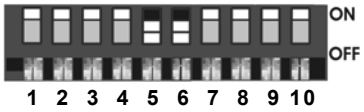


Table 1. Configuration Switches 1 through 10 definitions

Switch #	Definition	State	Setting	Position
1	**X –Axis Knob Directionality for Forward (+) Movement**	**Clockwise**	**OFF***	**UP***
		Counterclockwise	ON	DOWN
2	**Y –Axis Knob Directionality for Forward (+) Movement**	**Clockwise**	**OFF***	**UP***
		Counterclockwise	ON	DOWN
3	**Z –Axis Knob Directionality for Forward (+) Movement**	**Clockwise**	**OFF***	**UP***
		Counterclockwise	ON	DOWN
4	**D –Axis Knob Directionality for Forward (+) Movement**	**Clockwise**	**OFF***	**UP***
		Counterclockwise	ON	DOWN
5	**Y Axis Lock Out for Homing**	Enabled	OFF	UP
		Disabled	**ON***	**DOWN***
6	**Sensor Test** (see Caution)**	Enabled	OFF	UP
		Disabled	**ON***	**DOWN***
7 - 10	**Reserved**	**Disabled**	**OFF***	**UP***

*Normal operation (factory default).

CAUTION: To avoid damage to the micromanipulator/stage, DIP Switch 6 (Sensor Test) must always be set to ON (DOWN).

External Control

Controlling the QUAD externally via computer is accomplished by sending commands over the USB interface between the computer and the USB connector on the rear panel of the QUAD controller/ROE. The USB device driver for Windows is downloadable from Sutter Instrument's web site (www.sutter.com). The QUAD requires USB CDM (Combined Driver Model) Version 2.10.00 or higher. The CDM device driver for the QUAD consists of two device drivers: 1) USB device driver, and 2) VCP (Virtual COM Port) device driver. Install the USB device driver first, followed by the VCP device driver. The VCP device driver provides a serial RS-232 I/O interface between a Windows application and the QUAD. Although the VCP device driver is optional, its installation is recommended even if it is not going to be used. Once installed, the VCP can be enabled or disabled.

The CDM device driver package provides two I/O methodologies over which communications with the controller over USB can be conducted: 1) USB Di-

rect (D2XX mode), or 2) Serial RS-232 asynchronous via the VCP device driver (VCP mode). The first method requires that the VCP device driver not be installed, or if installed, that it be disabled. The second method requires that the VCP be installed and enabled.

Virtual COM Port (VCP) Serial Port Settings: The following table lists the required RS-232 serial settings for the COM port (COM3, COM5, etc.) generated by the installation or enabling of the VCP device driver.

Table 2. USB-VCP interface serial port settings.

Property	Setting
Data ("Baud") Rate (bits per second (bps))	57600
Data Bits	8
Stop Bits	1
Parity	None
Flow Control	None

The settings shown in the above table can be set in the device driver's properties (via the Device Manager if in Windows) and/or programmatically in your application.

Protocol and Handshaking: Command sequences do not have terminators. All commands return an ASCII CR (Carriage Return; 13 decimal, 0D hexadecimal) to indicate that the task associated with the command has completed. When the controller completes the task associated with a command, it sends ASCII CR back to the host computer indicating that it is ready to receive a new command. If a command returns data, the last byte returned is the task-completed indicator.

Command Sequence Formatting: Each command sequence consists of at least one byte, the first of which is the "command byte". Those commands that have parameters or arguments require a sequence of bytes that follow the command byte. No delimiters are used between command sequence arguments, and command sequence terminators are not used. Although most command bytes can be expressed as ASCII displayable/printable characters, the rest of a command sequence must generally be expressed as a sequence of unsigned byte values (0-255 decimal; 00 – FF hexadecimal, or 00000000 – 11111111 binary). Each byte in a command sequence being transmitted to the controller must contain an unsigned binary value. Attempting to code command sequences as "strings" is not advisable. Any command data being returned from the controller must also be received and initially treated as a sequence of unsigned byte values. Groups of contiguous bytes can later be combined to form larger values, as appropriate (e.g., 2 bytes into 16-bit "word", or 4 bytes into a 32-bit "long" or "double word"). For the QUAD, all axis position values (number of microsteps) are stored as "unsigned long" (32-bit) values, and each is transmitted and received to and from the controller as four contiguous bytes.

Axis Position Command Parameters: All axis positional information is exchanged between the controller and the host computer in terms of microsteps. Conversion between microsteps and microns (micrometers) is the responsibility of the software running on the host computer. The number of microsteps for any axis position must always be exchanged between the controller and the application running on an external computer as an unsigned 32-bit value ("unsigned long" for C/C++ or "U32" for LabVIEW). "Unsigned" means the value is always positive; negative values are not allowed. An unsigned long or U32 consists of four contiguous bytes, with a byte/bit-ordering format of Little Endian ("Intel") (most significant byte (MSB) in the first byte and least significant (LSB) in the last byte). If the platform on which your application is running is Little Endian, then no byte order reversal of axis position values is necessary. Examples of platforms using Little Endian formatting include any system using an Intel processor (including Microsoft Windows and Apple Mac OS X).

If the platform on which your application is running is Big Endian (e.g., Motorola PowerPC CPU), then these 32-bit position values must have their bytes reverse-ordered after receiving from, or before sending to, the controller. Examples of Big Endian platforms include many non-Intel-based systems, LabVIEW (regardless of operating system & CPU), and Java (programming language/environment). MATLAB adapts to the system on which it is running, so Little Endian may need to be enforced if running on a Big Endian system.

Microsteps and Microns (Micrometers): All coordinates sent to and received from the controller are in microsteps. To convert between microsteps and microns (micrometers), use the following conversion factors (multipliers):

Table 3. Microns/microsteps conversion.

System/Device	From/To Units	Conversion Factor (multiplier)
QUAD with QUAD/M micromanipulator	μsteps → μm	0.09375
	μm → μsteps	10.66666666667

For accuracy in your application, these conversion factors should be typed as double precision ("double"); "float" is not recommended. If the result is in microsteps, it can be typed as a 32-bit "long" integer; otherwise, it should be typed as floating point, preferably as double precision ("double").

Table 4. Ranges

Device	Axis	Millimeters	Microns	Microsteps
QUAD/M	X, Y, & Z	0 - 25	0 – 25,000	0 – 266,667
	D	0 - 30	0 – 30,000	0 – 320,000

Command Reference: The following table lists all the external-control commands for the QUAD.

Table 5. QUAD external control commands.

Command	Tx/-Delay/-Rx	Ver.	Total Bytes	Byte Offset (Len.)	Value: Dec.	Value: Hex.	Value: Binary	Alt-key-pad #	Ctrl-char	ASCII def./-char.	Description
Get Current Position and Angle ('c' or 'C')	**Tx**	All	1	0	99 or 67	63 or 43	0110 0011 or 0100 0011	0099 or 0043		'c' or 'C'	Returns the current positions (μsteps) of X, Y, Z, & D axes.
	Rx.	All	17		Four 4-byte (32-bit) values (current positions in μsteps of X, Y, Z, & D), + 1 byte for completion indicator. See Ranges table for minimum and maximum values.						
				0 (4)							X pos. in μsteps
				4 (4)							Y pos. in μsteps
				8 (4)							Z pos. in μsteps
				12(4)							D pos. in μsteps
				13	13	0D	0000 1101		^M	<CR>	Completion indicator
Move to HOME Position ('h')	**Tx**	All	1	0	104	68	0110 1000	0104		'h'	Moves to the position saved for the controller's HOME button. D mores first to the target, followed by Z, and then X & Y together.
	Rx	All	1	0	13	0D	0000 1101			<CR>	Completion indicator
Move to WORK Position ('w')	**Tx**	All	1	0	119	77	0111 0111	0119		'w'	Moves to the position saved for the controller's WORK button. X & Y move together to the target first, followed by Z, and then D.
	Rx	All	1	0	13	0D	0000 1101			<CR>	Completion indicator
Move to Specified "Home" Position ('H')	**Tx**	All	17	0	72	48	0100 1000	0072		'H'	Move all 4 axes to the specified position as if away from a work position. D moves first, then Z, and finally X & Y together (see Ranges table)
				1 (4)							X μsteps
				5 (4)							Y μsteps
				9 (4)							Z μsteps
				13(4)							D μsteps
	Rx	All	1	0	13	0D	0000 1101		^M	<CR>	Completion indicator
Move to Specified "Work" Position ('W')	**Tx**	All	17	0	87	57	0101 0111	0087		'W'	Move all 4 axes to the specified position, as if away from the home position and approaching a work position. X & Y move together first, then Z, and finally D (see Ranges table)
				1 (4)							X μsteps
				5 (4)							Y μsteps
				9 (4)							Z μsteps
				13(4)							D μsteps
	Rx	All	1	0	13	0D	0000 1101		^M	<CR>	Completion indicator
Move to specified X axis Position ('x' or 'X')	**Tx**		5	0	120 or 90	78 or 5A	0111 1000 or 0101 1010	0120 or 0090		'x' or 'X'	Move X axis to specified position (see Ranges table)

Command	Tx/-Delay/-Rx	Ver.	Total Bytes	Byte Offset (Len.)	Value Dec.	Value Hex.	Value Binary	Alt-key-pad #	Ctrl-char	ASCII def./-char.	Description
				1 (4)							X μsteps
	Rx	All	1	0	13	0D	0000 1101			<CR>	Completion indicator
Move to specified Y axis Position ('y' or 'Y')	**Tx**	All	5	0	121 or 91	79 or 5B	0111 1001 or 0101 1011	0121 or 0091		'y' or 'Y'	Move Y axis to specified position (see Ranges table)
				1 (4)							Y μsteps
	Rx	All	1	0	13	0D	0000 1101			<CR>	Completion indicator
Move to specified Z axis Position ('z' or 'Z')	**Tx**	All	5	0	122 or 92	7A or 5C	0111 1010 or 0101 1100	0122 or 0092		'z' or 'Z'	Move Z-axis to specified position (see Ranges table)
				1 (4)							Z μsteps
	Rx	All	1	0	13	0D	0000 1101			<CR>	Completion indicator
Move to specified D axis Position ('d' or 'D')	**Tx**	All	5	0	100 or 68	64 or 44	0110 0100 or 0100 0100	0100 or 0068		'd' or 'D'	Move D-axis to specified position (see Ranges table)
				1 (4)							D μsteps
	Rx	All	1	0	13	0D	0000 1101			<CR>	Completion indicator

NOTES:

1. All positions sent to and received from the controller are in microsteps. See **Microns/microsteps conversion** or conversion between microns (micrometers (μm)) and microsteps (μsteps).
2. See **Ranges** for exact minimum and maximum values for each axis of each compatible device that can be connected.
3. A short delay (usually around 2 ms) is recommended between commands (after the reception of one command and the sending of the next command).
4. All positions sent and received to and from the controller are in microsteps and consist of 32-bit signed integer values (four contiguous bytes). For C/C++, these are typed as "signed long" or just "long". For LabVIEW, these are typed as "I32".
5. All multibyte microstep values transmitted to, and received from, the controller consist of four bytes ordered in "Little Endian" (least significant byte last) format. The value stored in these four bytes is converted to and from 32-bit "[signed] long" (C/C++) or "I32" (LabVIEW) value storage entities. Although position values are unsigned, set the bounds for all positional storage values to appropriate range shown in **Ranges**.
6. "Little Endian" means that the least significant byte is last (last to send and last to receive). Byte-order reversal may be required on some platforms. Microsoft Windows, Intel-based Apple Macintosh systems running Mac OS X, and some Intel/AMD processor based Linux distributions handle byte storage in Little-Endian byte order so byte reordering is not necessary before converting to/from 32-bit "long" values. LabVIEW always handles "byte strings" in "Big Endian" byte order irrespective of operating system and CPU, requiring that the four bytes containing a microsteps value be reverse-ordered before/after conversion to/from a multibyte type value (I32, U32, etc.). MATLAB automatically adjusts the endianess of multibyte storage entities to that of the system on which it is running, so explicit byte reordering is generally unnecessary unless the underlying platform is Big Endian.
7. "Move" commands might have short to long distances of travel. If not polling for return data, an appropriate delay should be inserted between the sending of the command sequence and reception of return data so that the next command is sent only after the move is complete. This delay can be auto-calculated by determining the distance of travel (difference between current and target positions) and rate of travel. This delay is not needed if polling for return data. In either case, however, an appropriate timeout must be set for the reception of data so that the I/O does not time out before the move is made and/or the delay expires.

NOTES