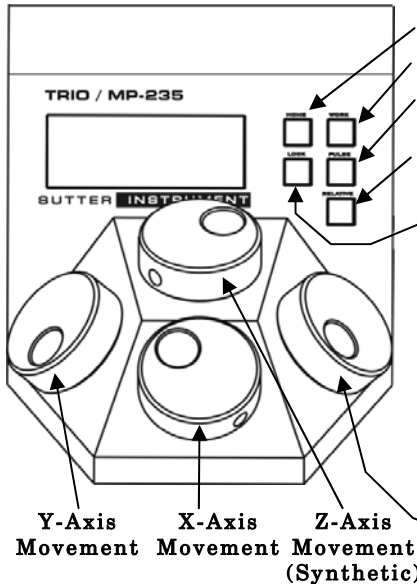


# TRIO™ MP-235 QUICK REFERENCE

REV. 2.23 (20190130) (FW v2.2 & 2.3)

## 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  $\mu\text{m}$  steps. Hold 3-sec. to enter **ANGLE SET** mode (active for 10 sec.) – use Knob D to change angle (0 – 90 degrees).
- 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 (slowest). Hold 3 sec. to enter **LOCK** mode.
- Setting Home/Work Pos., Relative Mode Origin Pos., & Angle:** To set position, hold down HOME, WORK, & RELATIVE buttons for 3 seconds until beep sounds. Likewise, set angle with LOCK.
- 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.
- D-Axis Movement;** selects angle in degrees when in **ANGLE SET** mode.

## Configuration

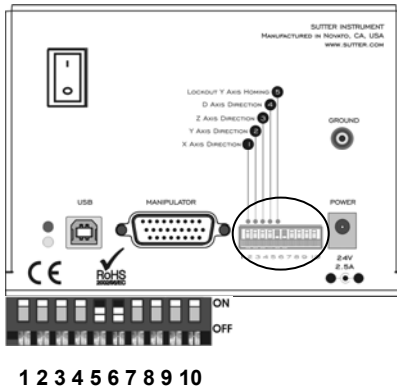


Table 1. Configuration Switches 1 – 5.

Sw #	Definition	State	Setting	Position
1	X-Axis Knob Rotation for Forward (+) Movement	Clockwise	Off*	Up*
		Counter	On	Down
2	Y-Axis Knob Rotation for Forward (+) Movement	Clockwise	Off*	Up*
		Counter	On	Down
3	Z-Axis Knob Rotation for Forward (+) Movement	Clockwise	Off*	Up*
		Counter	On	Down
4	D-Axis Knob Rotation for Forward (+) Movement	Clockwise	Off*	Up*
		Counter	On	Down
5	Y Axis Lock Out for Homing	Enabled	Off	Up
		Disabled	On*	Down*

\* Normal operation (factory default).

Table 2. Configuration Switches 6 – 10 (Ver. <2.2)

Sw #	Definition	State	Setting	Position
6	Sensor Test** (see Caution)	Enabled	Off	Up
		Disabled	On*	Down*
7 - 10	Reserved		Off*	Up*

\*\*CAUTION: To avoid damage to the micromanipulator or stage, DIP Switch 6 (Sensor Test) must always be set to ON (down).

Table 3. Config. Switches 6 - 10 (Ver. 2.2 & 2.3+)

Sw #	Definition	State	Setting	Position
6	Calibration Homing on Power On	None	Off	Up
		Calibrate	On	Down*
7	SPEED Select or PULSE button mode	Speed Select	Off*	Up*
		Pulse	On	Down
8	Reserved (Ver = <2.2) ROE Axis knobs active – all vs. D only (Ver =>2.3)	All Active	Off*	Up*
		D only	On	Down
9 ***	Y-axis travel length (25 or 12.5 mm)	25mm	Off*	Up*
		12.5mm	On	Down
10 ***	X-axis travel length (25 or 12.5 mm)	25mm	Off*	Up*
		12.5mm	On	Down

\*\*\*CAUTION: To avoid possible equipment damage, always be certain that Switch 9 and 10 settings correctly match the physical length of travel of the Y & X axes on the connected device.

## External Control

Controlling the TRIO MP-235 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 TRIO MP-235 controller/ROE. The USB device driver for Windows is downloadable from Sutter Instrument's web site ([www.sutter.com](http://www.sutter.com)). The TRIO MP-235 requires USB CDM (Combined Driver Model) Version 2.10.00 or higher. The CDM device driver for the TRIO MP-235 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 TRIO MP-235. 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 Direct (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 4. 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 TRIO MP-235, 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 5. Microns/microsteps conversion.

System/Device	From/To Units	Conversion Factor (multiplier)
TRIO MP-235 with TRIO MP-235/M micromanipulator	$\mu\text{steps} \rightarrow \mu\text{m}$	0.09375
	$\mu\text{m} \rightarrow \mu\text{steps}$	10.66666666667

For accuracy in your application, these conversion factors should be typed as double precision (“dou-

ble”); “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 6. Ranges.

Device	Axis	Millimeters	Microns	Microsteps
TRIO MP-235/M	X & Y	25	0 – 25,000	0 – 266,667
	D	50	0 – 50,000	0 – 533,334

**Command Reference:** The following table lists all the external-control commands for the TRIO MP-235.

Table 7. TRIO MP-235 external control commands.

Command	Tx/ Delay/ Rx	Ver.	Total Bytes	Byte Offset (Len.)	Value			Alt-key-pad #	Ctrl-char	ASCII def./-char.	Description						
					Dec.	Hex.	Binary										
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 ( $\mu\text{steps}$ ) of X, Y, & D axes						
	Rx	All	14		Three 4-byte (32-bit) values (current positions in $\mu\text{steps}$ of X, Y, & D), + 1 byte for completion indicator. See Ranges table for minimum and maximum values.												
				0 (4)	X pos. in $\mu\text{steps}$												
				4 (4)	Y pos. in $\mu\text{steps}$												
				8 (4)	D pos. in $\mu\text{steps}$												
			12	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.						
	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.						
	Rx	All	1	0	13	0D	0000 1101			<CR>	Completion indicator						
Move to Specified "Home" Position ('H')	Tx	All	13	0	72	48	0100 1000	0072		'H'	Move all 3 axes to specified position, moving D before X & Y together (see Ranges table)						
											1 (4)	X $\mu\text{steps}$					
											5 (4)	Y $\mu\text{steps}$					
											9 (4)	D $\mu\text{steps}$					
	Rx	All	1	0	13	0D	0000 1101		^M	<CR>	Completion indicator						
Move to Specified "Work" Position ('W')	Tx	All	13	0	87	57	0101 0111	0087		'W'	Move all 3 axes to specified position, moving X & Y together first, and then D (see Ranges table)						
											1 (4)	X $\mu\text{steps}$					
											5 (4)	Y $\mu\text{steps}$					
											9 (4)	D $\mu\text{steps}$					
	Rx	All	1	0	13	0D	0000 1101		^M	<CR>	Completion indicator						

Command	Tx/ Delay/ Rx	Ver	Total Bytes	Byte Offset (Len.)	Value			Alt- key- pad #	Ctrl- char	ASCII def./- char.	Description
					Dec.	Hex.	Binary				
Move to speci- fied X axis Position ('x' or 'X')	Tx	All	5	0	120	78	0111 1000	0120		'x'	Move X axis to specified position (see Ranges table)
					or 90	or 5A	or 0101 1010				
	1 (4)										X $\mu$ steps
	Rx	All	1	0	13	0D	0000 1101			<CR>	Completion indicator
Move to speci- fied Y axis Position ('y' or 'Y')	Tx	All	5	0	121	79	0111 1001	0121		'y'	Move Y axis to specified position (see Ranges table)
					or 91	or 5B	or 0101 1011				
	1 (4)										Y $\mu$ steps
	Rx	All	1	0	13	0D	0000 1101			<CR>	Completion indicator
Move to speci- fied D axis Position ('d' or 'D')	Tx	All	5	0	100	64	0110 0100	0100		'd'	Move D-axis to specified position (see Ranges table)
					or 68	or 44	or 0100 0100				
	1 (4)										D $\mu$ steps
	Rx	All	1	0	13	0D	0000 1101			<CR>	Completion indicator

**NOTES:**

- All positions sent to and received from the controller are in microsteps. See **Microns/microsteps conversion** table for conversion between microns (micrometers ( $\mu\text{m}$ )) and microsteps ( $\mu\text{steps}$ ).
- See **Ranges** table for exact minimum and maximum values for each axis of each compatible device that can be connected.
- A short delay (usually around 2 ms) is recommended between commands (after the reception of one command and the sending of the next command).
- All positions sent and received to and from the controller are in microsteps and consist of 32-bit positive integer values (four contiguous bytes). For C/C++, these are typed as “unsigned long”; “U32” for LabVIEW. Although a signed type might be desirable for computation and compatibility purposes, the application must ensure that only unsigned longs or U32 values are sent to, and received from, the controller (i.e., no negative values).
- All 32-bit position values transmitted to, and received from, the controller must be bit/byte-ordered in “Little Endian” format. This means that the least significant bit/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 endianness 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. If your development platform does not have built-in Little/Big Endian conversion functions, bit reordering can be accomplished by first swapping positions of the two bytes in each 16-bit half of the 32-bit value, and then swap positions of the two halves. This method efficiently and quickly changes the bit ordering of any multibyte value between the two Endian formats (if Big Endian, it becomes Little Endian, and if Little Endian, it becomes then Big Endian).

- “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:**