MPC-325
Series

Multi Motorized-Micromanipulator Control System

Operation Manual

Rev. 3.21K (20201120)

SUTTER INSTRUMENT®

One Digital Drive
Novato, CA 94949

Voice: 415-883-0128 Web: www.sutter.com
Fax: 415-883-0572 Email: info@sutter.com

CE EU Declaration of Conformity

Application of Council Directives:
2014/30/EU (EMC), 2014/35/EU (LVD), and 2011/65/EU (RoHS 2)

Manufacturer's Name: Sutter Instrument Company

Manufacturer's Address: One Digital Drive
Novato, CA. 94949 USA
Tel: +1 415 883 0128

Equipment Tested: **MPC-325** Series-Motorized Multi Micromanipulator System

Model(s):

System:	**MPC-325 Series**
Controller:	**MPC-200**
User Control Device:	**ROE-200**
Motorized Micromanipulator:	**MP-225/M** (may also include **MP-285/M** and/or **MP-265/M** manipulator, and/or **MT-800** translator)

Conforms to Standards:

EMC Emissions: EN 61326-1:2013, including:
EN 55011: 2009 Class B;
EN 61000-3-2:2015, & EN 61000-3-3:2014

EMC Immunity: EN 61000-4-2:2009,
EN 61000-4-3:2011,
EN 61000-4-4:2012,
EN 61000-4-5:2014,
EN 61000-4-6:2014,
EN 61000-4-8:2010, &
EN 61000-4-11:2004

LVD (Safety): EN 61010-1:2010

Tested/Verified By: ITC Engineering Services, Inc.
9959 Calaveras Road, PO Box 543
Sunol, CA 94586-0543 USA
Tel. +1 925 862 2944 Fax: +1 925 862 9013
Email: itcemc@itcemc.com Web: www.itcemc.com

Sutter Instrument

Test Report(s): 20120511-01A-R1_MC CE RptA R4, SI_EMC_MPC-200_20160713

Sutter Instrument Company hereby declares that the equipment specified above was tested and conforms to the EU Directives and Standards listed above, and further certifies conformation to the requirements of the European Union's Restriction on Hazardous Substances in Electronic Equipment Directive 2011/65/EU (RoHS 2).

Project Engineer: Jack Belgum, Ph.D.
Senior Vice President

SUTTER INSTRUMENT®

One Digital Drive, Novato, CA 94949 USA Phone: +1 415 883 0128 Fax: +1 415 883 0572
Email: info@sutter.com Web: http://www.sutter.com

DISCLAIMER

The **MPC-325**-series system consists of one or two MPC-200 controllers, an ROE-200 user control device, and one or more electromechanical micromanipulator devices. The purpose of the system is for the manipulation at the micro level of micropipettes and probes used in conjunction with a microscope. No other use is recommended.

This instrument is designed for use in a laboratory environment. It is not intended, nor should it be used in human experimentation or applied to humans in any way. This is not a medical device.

Do not open or attempt to repair the instrument. High voltages are present and inadvertent movement of the micromanipulator electromechanical could cause injury.

Do not allow unauthorized and/or untrained operative to use this device.

Any misuse will be the sole responsibility of the user/owner and Sutter Instrument Company assumes no implied or inferred liability for direct or consequential damages from this instrument if it is operated or used in any way other than for which it is designed.

SAFETY WARNINGS AND PRECAUTIONS

Electrical

- **Operate the MPC-200 using 110-- 240 V AC., 50-60 Hz line voltage. This instrument is designed for use in a laboratory environment that has low electrical noise and mechanical vibration. Surge suppression is recommended at all times. If the system includes two MPC-200 controllers daisy-chained together, surge suppression is strongly recommended.**
- **Fuse Replacement: Replace only with the same type and rating:**

  T2A, 250V, 5 x 20mm, Time Delay fuse (IEC 60127-2, Sheet III)
  (Examples: Bussmann GDC-2A. GMC-2A or S506-2-R (RoHS); or Littelfuse 218 200 or 218 200P (RoHS))

  A spare fuse is located in the power input module. Please refer to the fuse-replacement appendix for more details on fuse ratings and for instructions on how to change the fuse.

Avoiding Electrical Shock and Fire-related Injury

- **Always use the grounded power supply cord set provided to connect the system to a grounded outlet (3-prong). This is required to protect you from injury in the event that an electrical hazard occurs.**
- **Do not disassemble the system. Refer servicing to qualified personnel.**
- **To prevent fire or shock hazard do not expose the unit to rain or moisture.**

Electromagnetic Interference

To comply with FDA and CE electromagnetic immunity and interference standards; and to reduce the electromagnetic coupling between this and other equipment in your lab always use the type and length of interconnect cables provided with the unit for the interconnection

of one or more MP-2x5/M electromechanical devices, host computer via USB interface, (see the Technical Specifications appendix for more details).

Operational

Failure to comply with any of the following precautions may damage this device.

- This instrument is designed for operation in a laboratory environment (Pollution Degree I) that is free from mechanical vibrations, electrical noise and transients.
- This unit is not designed for operation at altitudes above 2000 meters nor was it tested for safety above 2000 meters.

- ⚠ DO NOT CONNECT OR DISCONNECT THE CABLES BETWEEN THE CONTROLLER AND THE MECHANICAL UNITS WHILE POWER IS ON.

  Please allow at least 20 seconds after turning the unit off before disconnecting the mechanical units. Failure to do this may result in damage to the electronics.
- Operate this instrument only according to the instructions included in this manual.
- Do not operate if there is any obvious damage to any part of the instrument.
- ⚠ Operate only in a location where there is a free flow of fresh air on all sides. NEVER ALLOW THE FREE FLOW OF AIR TO BE RESTRICTED.
- ⚠ Do not operate this instrument near flammable materials. The use of any hazardous materials with this instrument is not recommended and if undertaken is done so at the users' own risk.
- ⚠ Do not attempt to operate the instrument with the manipulator shipping screws in place - severe motor damage may result.
- ⚠ Do not operate if there is any obvious damage to any part of the instrument. Do not attempt to operate the instrument with the manipulator shipping screws in place - severe motor damage may result. When transporting the mechanical manipulator, be sure to install the shipping screws supplied in their correct locations. Failure to do this may result in damage to the motors.
- ⚠ Never touch any part of the micromanipulator electromechanical device while it is in operation and moving. Doing so can result in physical injury (e.g., fingers can be caught and pinched between the moving parts of the micromanipulator).
- ⚠ As with all microinjection devices, sharp micropipettes can fly out of their holder unexpectedly. Always take precautions to prevent this from happening. Never loosen the micropipette holder chuck when the tubing is pressurized, and never point micropipette holders at yourself or others. Always wear safety glasses when using sharp glass micropipettes with pressure microinjectors.
- ⚠ Do not handle the manipulator mechanical while the power is on, and take care to ensure no cables pass close to the mechanical manipulator.

Other

- Use this instrument only for microinjection purposes in conjunction with the procedures and guidelines in this manual.
- Retain the original packaging for future transport of the instrument.
- Some applications, such as piezo-impact microinjection call for the use of mercury in the micropipette tip. The use of any hazardous materials with any Sutter Instrument's instrument is not recommended and if undertaken is done so at the users' own risk.
- When transporting the mechanical manipulator, be sure to install the shipping screws supplied in their correct locations. Failure to do this may result in damage to the motors.
- This instrument contains no user-serviceable components — do not open the instrument casing. This instrument should be serviced and repaired only by Sutter Instrument or an authorized Sutter Instrument servicing agent.
- Sutter Instrument reserves the right to change specifications without prior notice.
- This device is intended only for research purposes.

Handling Micropipettes

 Failure to comply with any of the following precautions may result in injury to the users of this device as well as those working in the general area near the device.

- The micropipettes used with this instrument are very sharp and relatively fragile. Contact with the pulled micropipette tips, therefore, should be avoided to prevent accidentally impaling yourself.
- Always dispose of micropipettes by placing them into a well-marked, spill-proof "sharps" container.

(This page intentionally left blank.)

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

1. INTRODUCTION

1.1 Structure of the MPC-325 Documentation Package

The MPC-325-Series is a manipulator system comprised of the MPC-200 controller, the ROE-200 input device and one or more MP-225/M stepper motor manipulators. The manual consists of two parts, “Operations” that describes the functions of the MPC-200 controller and ROE input device and “Setup” that describes how to install the MP-225/M mechanicals.

1.2 Components of the MPC-325 Series

Carefully remove all components from the shipping container. In addition to this manual, the following should be included in each MP-325 system

1. **MPC-325** (a single manipulator system):
   - MPC-200 controller
   - ROE-200 Rotary Optical Encoder input device
   - MP-225/M manipulator mechanical
   - Two MP-225 Adapter boxes (adapts 3 wire MP-225/M mechanical to DB-25 cable)
   - DB-25 cables (connect adapter boxes to the controller).
   - RJ-45 cable (8 conductor) connects the ROE to the controller
   - RJ-12 cable (6 conductor) daisy-chains two MPC-200 controllers
   - USB cable for computer control of the MPC-325-series system
   - Power cable appropriate for your location
   - X285210 mounting adapter plates and hardware to attach mechanicals to their mounting surfaces
   - X285204 four-inch dovetail extensions for mounting headstages
   - Dovetail rod clamp
   - 2.5mm hex wrench(s) for removing the shipping screws
   - 1.5mm hex wrench(s) for adjusting pipette angle

2. **MPC-325-2** (a two-manipulator system). Same as for the MPC-325, with the following exceptions:
   - Two MP-225/M manipulator mechanicals
   - Two MP-225 Adapter boxes (adapts 3 wire MP-225/M mechanical to DB-25 cable)
   - Two DB-25 cables (connect adapter boxes to the controller).
   - Two X285210 mounting adapter plates and hardware to attach mechanicals to their mounting surfaces
   - Two X285204 four-inch dovetail extensions for mounting headstages
   - Two dovetail rod clamps

3. **MPC-325-3** (a three-manipulator system). Same as for the MPC-325-2, with the following exceptions:
   - Two MPC-200 controllers
   - Three MP-225/M manipulator mechanicals
   - Three MP-225 Adapter boxes (adapts 3 wire MP-225/M mechanical to DB-25 cable)
   - Three DB-25 cables (connect adapter boxes to the controller).
   - Three X285210 mounting adapter plates and hardware to attach mechanicals to their mounting surfaces
   - Three X285204 four-inch dovetail extensions for mounting headstages
   - Three dovetail rod clamps

4. **MPC-325-4** (a four-manipulator system). Same as for the MPC-325-3, with the following exceptions:
   - Four MP-225/M manipulator mechanicals
   - Four MP-225 Adapter boxes (adapts 3 wire MP-225/M mechanical to DB-25 cable)
   - Four DB-25 cables (connect adapter boxes to the controller).
   - Four X285210 mounting adapter plates and hardware to attach mechanicals to their mounting surfaces
   - Four X285204 four-inch dovetail extensions for mounting headstages
   - Four dovetail rod clamps

IMPORTANT

Once the MPC-325-series system has been unpacked, remove the two shipping screws, indicated by the red warning tags, from each MP-225/M micromanipulator mechanical. These screws must be removed before operating the manipulator. Save the screws, warning tags, and hex wrench if future transport of the manipulator is needed. Once these screws have been removed, handle the micromanipulator with care. The mechanism can be damaged if the axes are moved with the screws in place.

2. MPC-200 MULTI-MANIPULATOR CONTROLLER AND ROE-200 INPUT DEVICE OPERATIONS

2.1 Electrical Connections and Initial Operating Instructions

Initially, you may want to simply connect the two manipulators, the controller, and the ROE together and try some gross movements in order to get a feel for the controls and how to make simple movements. It is perfectly acceptable to set the manipulators in the middle of a bench top, make all electrical connections and then observe each unit's movement by eye. Even if you wish to directly install the manipulators in your rig, it is useful to follow the initial setup procedure to learn how to move the units to allow easy access to the mounting screws.

1. Connect the power cord to the power entry module on the back of the MPC-200 controller.

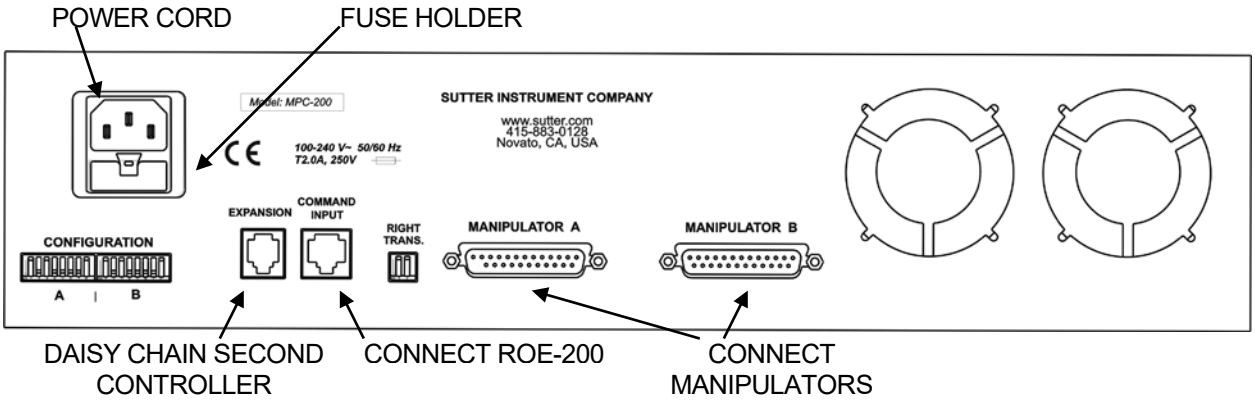


Figure 2-1. Rear of MPC-200 controller cabinet.

2. With the power OFF (front panel switch in the "0" position), connect the ROE-200 input box to the MPC-200 controller using the RJ-45 8-conductor cable.* Use the CONTROLLER output on the back of the ROE and the COMMAND INPUT on the back of the controller.
3. With the power OFF, run a DB-25 cable from each of the two MP-225/M mechanicals to the DB-25 connectors marked "MANIPULATOR A" and "MANIPULATOR B" on the back of the controller. *

*⚠ ***CAUTION: Never connect or disconnect the ROE or the MP-225/M while the power is on!***

2.2 Initial Operating Instructions

After all connections are made, power up the MPC-325 using the 0/I switch on the front of the controller. As it initializes, you will see a start up screen on the ROE-200 that briefly displays the name of the device and the version of the installed firmware. As the power switch is the only control you will need to access on the MPC-200, the controller can ultimately be placed in an out of the way location (e.g., under your bench).

Once the start-up sequence has finished, you will see a display that gives the coordinates of the manipulator. The LED marked 1 will light and the left-hand corner of the display shows “Drive A” to indicate that the ROE is ready to operate the MP-225/M connected at the MANIPULATOR A output. Confirm that you get a coordinate display and that you have removed the shipping screws from both manipulators. If you do not get a coordinate display, go to the trouble shooting section at the back of the manual. If you have not yet removed the shipping screws, turn the power off again and remove all shipping screws from both manipulators.

All functions necessary during normal operation are provided by 4 push buttons and two rocker switches on the top of the ROE-200. Other setup functions are done via buttons and DIP switches located on the back of the ROE-200 and DIP switches on the back of the MPC-200 controller.

The three ROE knobs control the three axes of either manipulator (right knob X, left knob Y, and top knob Z (see Page 15)). Turn any one of the three knobs and notice that the corresponding axis moves and the coordinate for that axis changes on the MP225/M connected to the MANIPULATOR A output.

The MPC-200 controller and ROE-200 have a built in Centering function. This is activated by pressing the white “CENTER” button on the back of the ROE. If both MP-225/M manipulators are sitting in a wide-open area, and the shipping screws are removed, press the CENTER button. The ROE-200 display will display the message “PLEASE WAIT MOVE IN PROGRESS” and the first manipulator will center. After the CENTER operation is complete, the manipulator axes will each be at the center of travel and the display will read 12500 for X, Y and Z.

From this location, you can move 12500 microns in each direction on each axis. The unit will stop automatically at each end of travel (00000 or 25000 microns). These ends are determined by firmware. Each axis also has magnetic end of travel switches that are not activated in normal operation. If the magnetic switches are activated, you will see the message EOT (for End Of Travel) on one of the displayed axes.

If you wish, you can easily switch to the second manipulator (connected to the MANIPULATOR B output on the back of the controller). This is done by pressing the Manipulator toggle once. The LED marked 2 will light and the left-hand corner of the display will change to “Drive B”. While you are controlling the second manipulator, press CENTER to make sure that this manipulator’s coordinate system is initialized. After centering, you can demonstrate that the manual knobs are now moving this manipulator.

When the MPC-200 controller is first turned on, the speed of movement is at its fastest, coarsest Mode. Movement mode can be finer and slower by changing the black “Mode” toggle switch. As MODE increases from 0, smaller movements are commanded by the same turn of the ROE knob. MODE 5 or 6 is probably what you will use for the final approach to a cell. MODE 0 or “Accelerated Mode” is used for fast movements to move the pipette large distances. In MODE 0, when you turn the ROE knobs slowly, you get relatively slow movement that is useful for final moves to place a pipette near a cell. Conversely, when you make prolonged, rapid turns of the ROE knob, the controller/ROE automatically accelerates to maximum speed to allow for prolonged, long distance movements. This would be most useful for manual pipette exchange.

If you toggle from Drive A to Drive B and back again you will see that the display coordinates and Mode settings are maintained for Drive A while you are using Drive B and vice versa.

The remaining functions of the ROE are explained in the next section.

2.3 Main Controls on the ROE-200

2.3.1 White Buttons:

DIAG/NORM: Pressing the DIAG/NORM button will cause the green LED near the button to light, indicating the MPC-200/ROE-200 is in Diagonal mode. In this mode, rotation of the Z-axis knob produces diagonal movement. A second press will put the manipulator back into Normal mode. When in diagonal mode, the X and Y knobs remain active, allowing you to readjust the X and Y positioning of the pipette as you approach a cell in diagonal mode. Angle of diagonal mode movement is set via DIP switches on the back of the MPC-200 controller (see controller DIP switch setting instructions on Pages 19-21). When using MODE 9 (MODE toggle set to 9), Diagonal mode produces short, quick, impulse-like movement that may be useful in sharp pipette impalements.

When you switch to Diagonal mode, the ROE-200 display is changed from absolute to relative coordinates and the current location is set to 0,0,0. This allows users to invoke relative measurements using the display as a measuring device. A fourth coordinate that gives movement along the diagonal is also added for users who wish to measure the movement of along the axis coaxial with a pipette. When you return to Normal mode, the absolute coordinate system is recovered. The relative coordinate feature can be disabled via DIP switch 2 on back of the ROE-200.

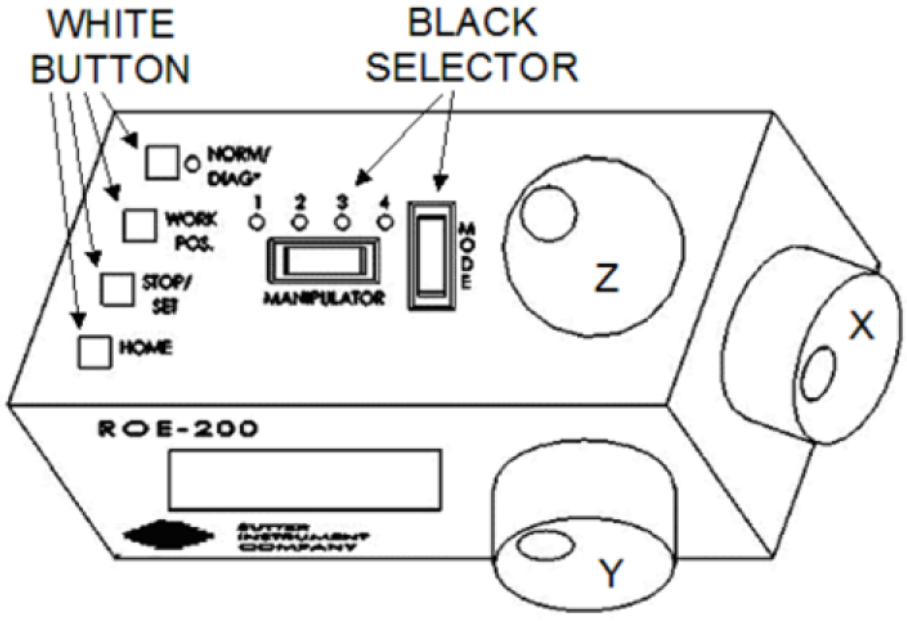


Figure 2-2. Top view of ROE-200.

HOME: When pressed, the manipulator will make a move along a stereotypic path to the location 0,0,0 or "home". Home is the location where you would most likely exchange your pipette and is maximal up on the Z-axis, maximal right on the X-axis (maximal left on a left-handed manipulator) and maximal front on the Y-axis. The stereotypic path of the movement

is first along the currently set diagonal until either the X-axis or Z-axis reaches its origin (0). Which one of these occurs first is a function of the diagonal angle and the location at the time HOME is pressed. Once the first limit is reached, the unit will move the two remaining axes simultaneously to their origins (0). The only allowed change in this stereotyped move is that the Y-axis move can be eliminated. This is done via DIP switch 8 on the back of the MPC-200 controller. (See Controller DIP switch setting instructions on Page 21).

WORK POS.: This button has three functions:

1. With the STOP/SET button is held down, a momentary press of WORK POS. makes the current location the "Work Position". A beeper will sound to indicate that the operation is complete, and the location has been saved. Typically, this is a location where the pipette tip is under the microscope objective and near the cells or tissue of interest.
2. Once you have defined a Work Position, a momentary press of WORK POS. will cause the manipulator to move to the defined Work Position, providing the manipulator last move was to Home. The move will occur along the predefined path that the manipulator moved to get to Home (described above) but in the opposite direction. This is the reason why Work Position moves **must** follow Home moves; the move to Home defines the return trip. In either case, the movement along the diagonal as you come in and out of the preparation/dish/bath should assure that the pipette tip will not hit anything on the way in or out.
3. When WORK POS. is held down for longer than 2 seconds, the current manipulator is locked so that none of the buttons or the ROE knobs will cause it to move. The lock is released by holding WORK POS. down again. A beep will indicate that the lock is enabled or disabled, and the display will indicate the locked state.

Note that like HOME, the Y axis movement can be disabled when WORK POS is pressed, as set via DIP switch 8 on the back of the MPC-200 controller (see Controller DIP switch setting instructions on Page 21).

STOP/SET: This button has two functions:

1. When held down, STOP/SET" performs a "Set" function in combination with the "WORK POS." key. Think of it as a shift key when held down.
2. A momentary press of STOP/SET during a robotic move (see HOME, WORK POS. and CENTER) will immediately "Stop" the movement. **Think of this as your panic button when you see your pipette headed somewhere that you don't want it to go!**

2.3.2 Black Selector Switches:

MODE: The MODE Selector controls the speed and the relative fineness of movement of the manipulator produced by rotating the ROE knobs. As MODE increases from 0 to 9, movement gets finer and slower. As explained in "INITIAL OPERATING INSTRUCTIONS", MODE 0 is Accelerated Mode. In MODE 0, slow turns of the ROE knob produce medium course moves for moving a pipette under a microscope in the vicinity of a cell. Prolonged, fast turns of the ROE knobs cause the controller to accelerate to top speed for long, imprecise movements for rapid manual positioning of the pipette. The remaining MODES (1-9) produce moves of increasing sensitivity and decreasing speed. In practice, most users will find that MODE 5 or 6 will provide the necessary dexterity of movement for the final approach to a cell. The current MODE setting is displayed in the upper right of the ROE-200 display.

MANIPULATOR: The MANIPULATOR Selector toggles the active manipulator. Both an LED and the named manipulator on the ROE display change to signify which manipulator is active. When the MANIPULATOR A output is selected, LED 1 will light, and the display will say “Drive A” in the upper right-hand corner. When the MANIPULATOR B output is selected, LED 2 will light, and the display will say “Drive B”.

The status of a manipulator is preserved when you toggle to the other manipulator. Status includes the current position, current MODE (speed) setting, whether or not you are in diagonal or orthogonal movement, and whether the manipulator is currently locked. In addition, a separate WORK POS is maintained for each manipulator in use.

A separate set of DIP switches is present on the back of the controller for controlling setup of the two different manipulator outputs (see Controls on the MPC-200).

You can also configure how the MANIPULATOR Selector operates. The selector can function as a two position toggle, where pressing the left side of the toggle selects Manipulator 1 and pressing the right side of the toggle selects Manipulator 2, or the selector can function in a cyclical fashion, pressing once on either side selects the other manipulator or pressing twice reselects the manipulator you are already on. Selection method is determined by DIP switch 3 on the back of the ROE.

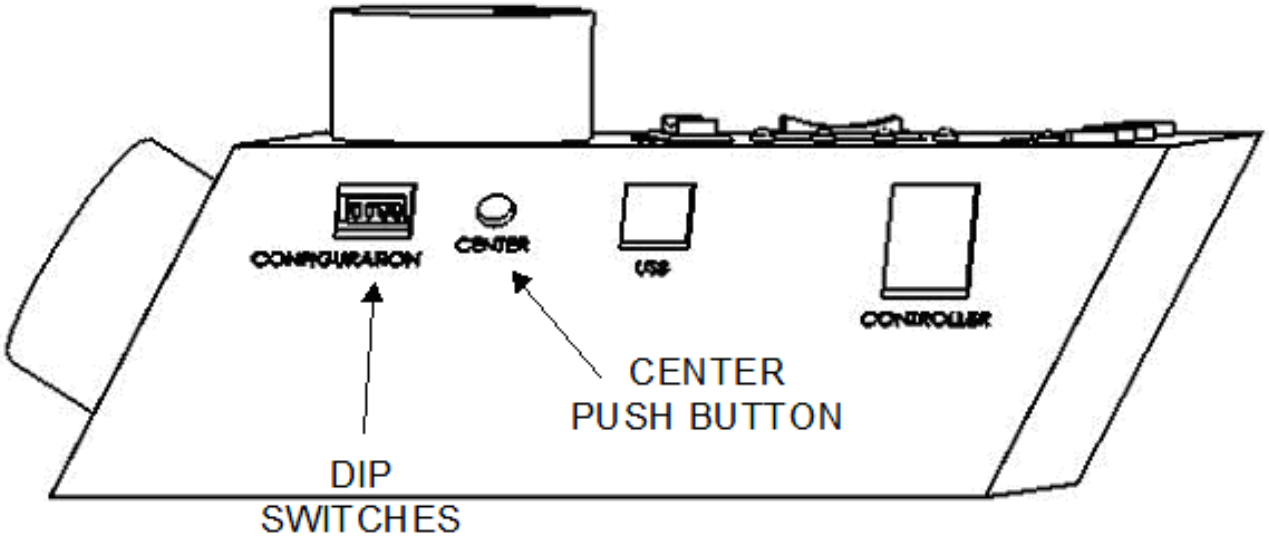


Figure 2-3. Side view of ROE-200.

2.3.3 Other Controls on the ROE-200

CENTER (round push button on the back of ROE-200): CENTER is an initialization function that is used when the unit is first set up and occasionally during normal operation. **CENTER should only be done in the absence of a pipette as the manipulator makes large robotic movements to its extreme ranges of motion.** To CENTER, press and release the white button on the back of the ROE-200. This will cause a prolonged movement in each axis to the end of travel (EOT) sensors beyond the origin (0,0,0). Once the sensors are found, a short move in the opposite direction is made and this location is defined as (0,0,0). Finally, the unit moves to the location (12500 (X), 12500 (Y), 12500 (Z)), the center of travel of each axis. If the unit is turned off or STOP/SET is pressed during the running of CENTER, the unit will

not be correctly initialized. In this case, it is necessary to cycle the power off and on and run CENTER again to its completion.

DIP Switches (on back of ROE-200): There are four DIP switches on the back of the ROE-200 which govern global and/or ROE settings.

> **Switch 1:** When ON, all modes on the MODE selector, except Mode 0 and 5, are disabled. Some users may find that they only need Accelerated Mode and a single fine mode. This will allow them to more easily switch between the two. Factory default is OFF, enabling all modes.
>
> **Switch 2:** When OFF, relative coordinates during Diagonal Mode are disabled. The factory default is ON (display of relative coordinates enabled during Diagonal Mode).
>
> **Switch 3:** When OFF, the MANIPULATOR Selector functions in a cyclical fashion. After reaching the highest-numerated manipulator, a further push of MANIPULATOR cycles the user back to the lowest-numerated manipulator. When DIP switch 3 is set to ON, the selector does not cycle back to the first manipulator. Factory default is OFF, allowing cycling back.
>
> **Switch 4:** Reserved for future use. **NOTE: Must be kept ON for proper functioning!**

2.4 Controls on the MPC-200

Power Switch: The power switch for the MPC-200 is located on the front panel of the controller. At power up, the microprocessor in the ROE-200 scans the attached equipment and configures the system accordingly. Among the checks/configurations that are made:

1. Determines the number and type of manipulators that are attached. The MPC-200/ROE-200 system is able determine how many and what type of manipulators (MP-285/M, MP-225/M, or MP-265/M) are connected and to what outputs they are connected. It then sets the current for each output to the correct value for the mechanicals found. If no manipulators are found, the controller will return the message "NO MANIPULATOR DETECTED, PLEASE TURN OFF CONTROLLER AND ATTACH MANIPULATOR"
2. The ROE-200 can connect to more than one MPC-200 controller. On power up, the ROE determines of how many controllers are attached and configures itself properly. If the power is off on the second controller, the ROE-200 displays a message "PLEASE TURN ON ALL CONTROLLERS, THEN PRESS SET TO START".

DIP Switches: Two banks of 8 DIP switches are located on the back of the MPC-200 controller. Each bank is assigned to, and configures, one of the two manipulator outputs on the back of the controller (MANIPULATOR A or B). Users familiar with the Sutter Instrument MP-225 controller will find that they have the same function as the configuration DIP switches on the MP-225 ROE. The switches are numbered 1 through 8. In all cases, the 0 or OFF position is opposite the direction of the switch number and the 1 or ON position is in the direction of the switch number and is also indicated by an arrow and the word “ON” next to Switch 1. For any new switch settings to take effect, the controller must be powered off and on.

The figure below shows the two banks of switches on the back of the MPC-200 controller.

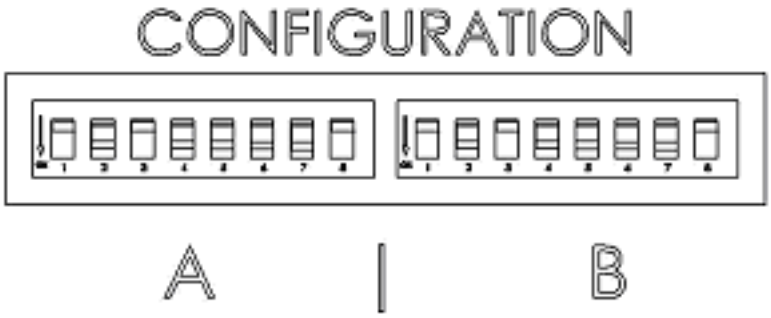
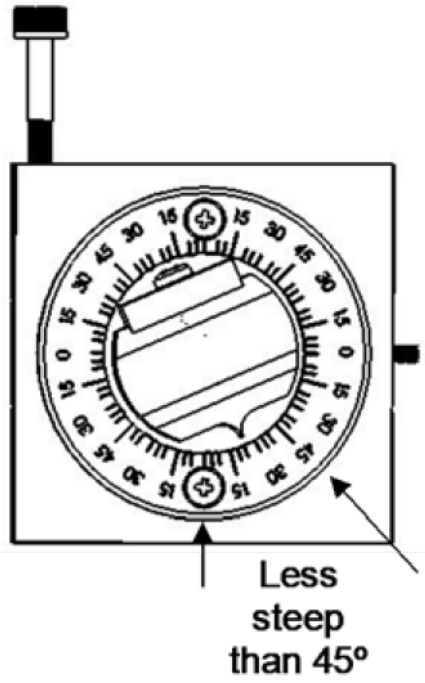


Figure 2-4. Configuration switches on rear of MPC-200 controller cabinet.

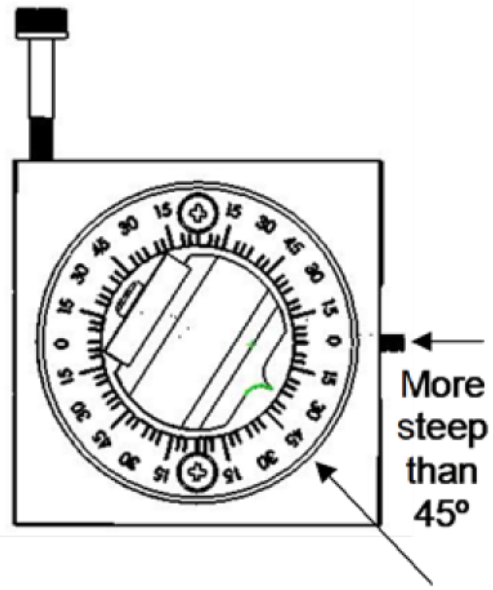
Switches 1, 2, 3 and 4 set the angle of the Diagonal mode movement.

The following table provides the angles that can be used and the DIP switch settings of switches 1, 2, 3 and 4. As indicated in the inset to the left of the table, the angles fall into two different quadrants according to whether the angles are more or less steep than 45 degrees.

Table 2-1. Configuration switch settings for different angles of steepness.



Angle	DIP switch number 1	2	3	4
7	1	1	1	1
11	0	1	1	1
14	1	0	1	1
21	0	0	1	1
27	1	1	0	1
29 *	0	1	0	1
35	1	0	0	1
39	0	0	0	1
45	1	1	1	0



Angle	DIP switch number 1	2	3	4
39	0	1	1	0
35	1	0	1	0
29	0	0	1	0
27	1	1	0	0
21	0	1	0	0
14	1	0	0	0
11	0	0	0	0

*Factory default near 30 degrees

Switches 5, 6 and 7 set the direction of the movement produced by a clockwise turn (advancing right hand screw) of the ROE knob for each axis.

With the switch set to 0, a clockwise turn of the knob produces a decrement in the display; when the switch is set to 1, a clockwise turn of the knob produces an increment in the display. An increment in the display coincides with movement downward in the Z axis,

movement toward the rear of your setup in the Y axis and movement producing pipette advancement in the X axis.

The factory default is 1,1,1 for switches 5,6 and 7.

Switch number	5	6	7
Corresponding axis	X	Y	Z

Switch 8 determines whether the Y-axis is included in HOME and WORK POS. robotic moves. If switch 8 is set to 0, the Y axis is moved to a location where the pipette is towards the user in HOME move and is moved back to whatever Y coordinate was recorded during SET-WORK POS. in the WORK POS. move. If switch 8 is set to 1, the Y axis is not moved (Y position ignored) during the HOME or WORK POS. moves. The factory default for switch 8 is 0; the Y axis will move during HOME and WORK POS. moves.

Remember that the settings on the A switches apply to the MANIPULATOR A output and the settings on the B switches apply to the MANIPULATOR B output. Thus, you can have, for example, different angles of approach on your two manipulators or a different direction of turning to advance the pipette on a left versus a right-handed manipulator.

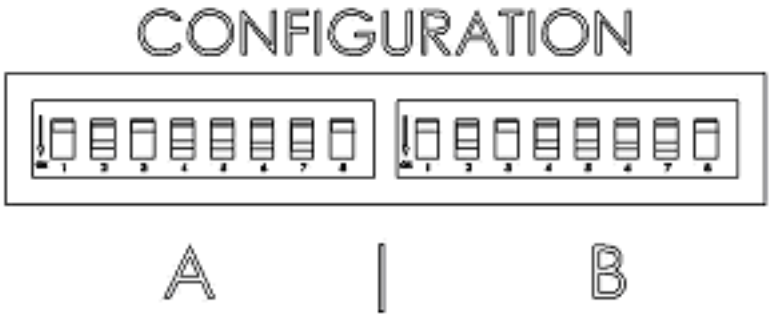


Figure 2-5. Configuration switches (rear of MPC-200 controller).

(This page intentionally blank.)

3. MP-225/M MANIPULATOR MECHANICAL MOUNTING INSTRUCTIONS

The following sections describe how to mount your MP-225/M to a stand using the mounting adapter plate, how to adjust pipette angle and change pipettes, how to mount different headstages and finally, the modular nature of the mechanical. It is assumed that if you are setting up an MPC-325-2, that you will repeat the setup instructions for two mechanicals. The figure below shows a right-handed MP-225/M. You may have a left-handed unit, especially if you have a two-manipulator system, but the setup is identical.

3.1 Mounting MP-225/M to a Stand or Platform

The MP-225/M mounts to the mounting adapter plate (X285210) using four tapered pegs with locking screws. The figure below shows the location of the front locking setscrews:

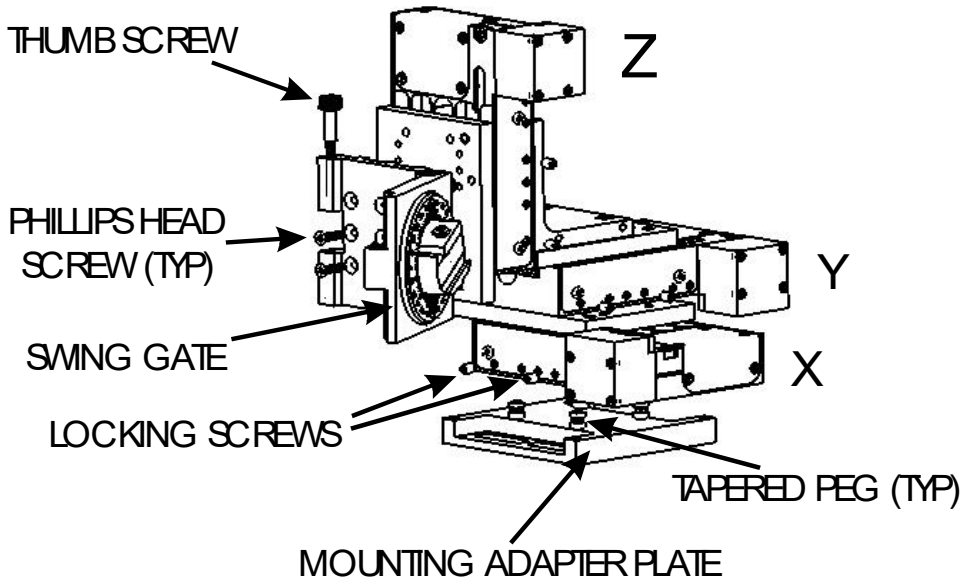


Figure 3-1. Angled side view of MP-225/M showing swing gate and mounting adapter plate.

The rear pair of screws is in a similar location in the back of the manipulator. **The manipulator is shipped with the adapter plate in place; it must be removed to mount the manipulator!**

To remove the plate, first loosen the locking screws and then pull the mounting plate straight down. The figure above shows the plate removed from the X axis. Once removed, the mounting adapter plate can be secured to any flat surface carrying ¼-20 or 10-32 holes on one-inch centers (such as a Sutter MT-stand or MD series platform) using the supplied hardware. After mounting the adapter plate on your stand or platform, align the pegs on top

of the plate with the holes in the manipulator, push the X-axis firmly onto the plate, and retighten the locking hex set screws.

3.2 Setting Headstage/Pipette Angle and Pipette Exchange

Mounted on the front of the Z-axis of the manipulator is the “swing-out gate”. The swing-out gate is the mounting surface for the rotary dovetail that holds various electrophysiological headstages and/or micro tools at defined angles. The swing-out gate also provides for easy exchange of pipettes during an experiment.

The angle of the rotary dovetail is adjusted by loosening the hex set screw located on the hinge side of the swing-out gate (see figure below). You can set an angle using the knife-edge on the dovetail and the scale on the faceplate. After choosing an angle, press the rotary dovetail firmly into the pocket in the swing gate and retighten the screw to fix the angle.

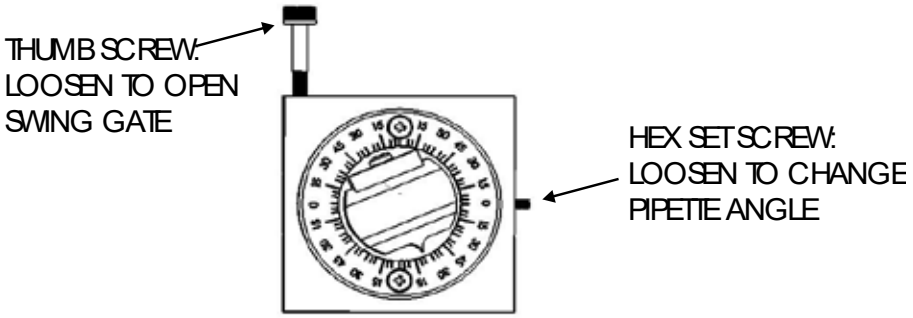


Figure 3-2. Locations of screws used to open swing gate and changing pipette angle.

To change pipettes, loosen the thumbscrew on the swing-out gate (above left). The gate will open allowing the headstage and holder to rotate almost 90 degrees. After replacing your pipette, make sure to close the gate tightly and tighten the thumbscrew securely while holding the gate closed. The thumbscrew is designed to pull the gate closed with tightening. Tightening with thumb and finger is enough.

The height of the swing-out gate on the front of the Z-axis is adjustable. To change the position, open the gate and loosen the 4 Phillips-head screws that mount the swing gate. As shipped from the factory, the gate is positioned to allow access to the Z-axis shipping screw holes. You may find it beneficial to move the gate up before you start using your MP-225/M.

3.3 Headstage Mounting

Sutter Instrument IPA® and dPatch® headstages, Axon headstages 203B or CV-7, and the Heka EPC-10 headstage have an integral dovetail that slides directly into the rotary dovetail on the MP-225/M. The figure below shows an example of this type of headstage mounted in a left-handed manipulator and in profile (on the right), the location of the Phillips-head screw that secures the headstage dovetail in its mate on the manipulator.

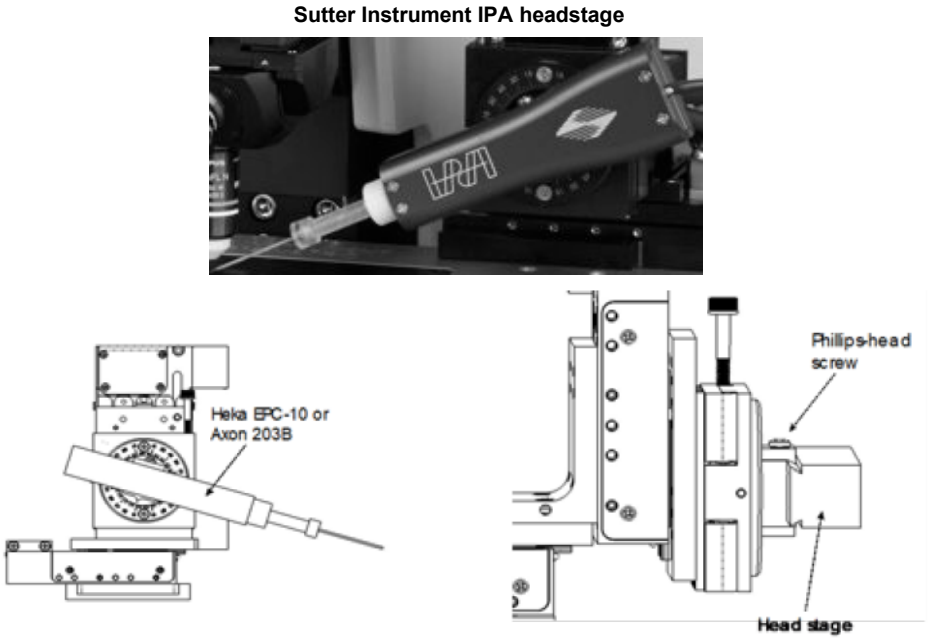


Figure 3-3. Headstage mounting.

Older Axon and Heka headstages mount using the 4-inch dovetail (X285204) and a plastic plate. A typical headstage of this type is shown mounted in a right-handed MP-225/M (right panel). The plastic plate used with the 4-inch dovetail is shown in the left panel of the figure and the holes are indicated to mount common headstages. Additional holes could be easily added to accommodate less common headstage footprints.

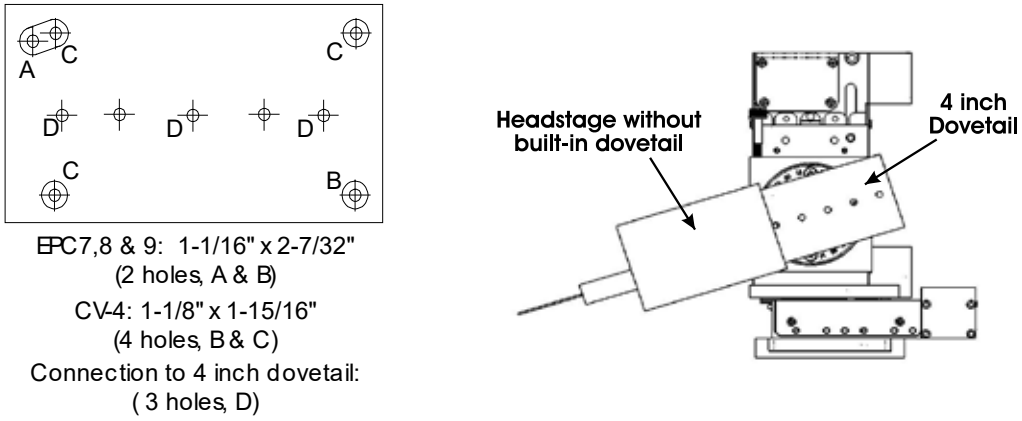


Figure 3-4. Using a dovetail for headstage mounting.

Rod mounted headstages and micro tools are accommodated by use of a rod clamp that fits into the dovetail (not shown). All the headstage adapters and mounting hardware are included with the manipulator and are shipped in a zip lock plastic bag.

3.4 Modular Construction

The three axes of the MP-225/M are identical. They are connected to each other with tapered pegs and hex locking screws, four between each axis. The peg and locking screw attachment is identical to the attachment of the mounting adapter plate to the bottom of the

manipulator. The manipulators are shipped pre-assembled in either the right or the left-handed configuration. Because of the modular construction, handedness can be easily switched. If you wish to change one of your mechanicals from right-handed to left-handed or vice versa see "Instructions for Changing Handedness" included later in this section.

The modular construction of the MP-225/M mechanical allows for some flexibility in the connections between axes. In the standard configuration, the axes are stacked X, Y, and Z from bottom to top (see figure on Page 23). However, the 3 axes can be used separately or assembled in non-standard configurations. **If you assemble the manipulator in other configurations, make sure that the axes do not interfere with one another!** For example, if the Z-axis is mounted in a lower location on the right angle that connects it to the Y-axis, it may interfere with the full travel of the Y-axis.

You may have also received one or more accessories for mounting your MP-225/M and/or modifying the headstage mount to the manipulator (i.e. rotating base, microscope stage mount, gantry stand, dovetail extension). Setup of these accessories is normally covered in documentation accompanying the accessory.

If you intend to use the right-angle adapter (285300) with your MP-225/M in order to rotate the manipulator 90 degrees, please see "Instructions used in Special Installations Only" near the end of this manual.

3.5 Minimizing Electrical Noise

We are aware of one potential noise source that users coupling their MP-325 with high-gain, high-input impedance, electrophysiological recording amplifiers may experience. Under certain circumstances, the manipulator and/or the drive cable coming from the controller may act as an antenna picking up electric field noise from nearby electrical equipment and bringing it into your setup. Grounding the manipulator will largely eliminate this noise source. Try to attach to one of the silver Phillips-head screws on the side of one or more of the axes. It should be noted that the manipulator produces negligible electrical noise when it is not moving because it is powered by a linear power supply with no AC current present.

3.6 Instructions for Changing Handedness

To switch from Right-handed to Left-handed:

1. Loosen four hex set screws, two on each side of the Y axis, that lock four tapered pins on the top of the X-axis in the holes in the bottom of the Y-axis (the four screws are indicated in the left and right panels below).
2. Pull the X-axis straight down (left panel).
3. Rotate the X-axis 180 degrees in the X-Y plane so that the motor and wire of the X axis are on the left side of the manipulator (middle panel).
4. Reinsert the pins into the holes in the bottom of the Y axis (right panel).
5. Retighten the hex set screws in the sides of the Y-axis.
6. Open the swing gate and remove the four screws that attach it to the Z-axis (swing gate shown open on Page 23).
7. Rotate the swing gate 180 degrees in the Z-X plane and reattach to the front of the Z-axis.
8. Remove the thumbscrew from the bottom of the swing gate and install in the similar hole in the top of the swing gate (rotated swing gate shown in right panel).

To switch from left to right, simply reverse these directions.

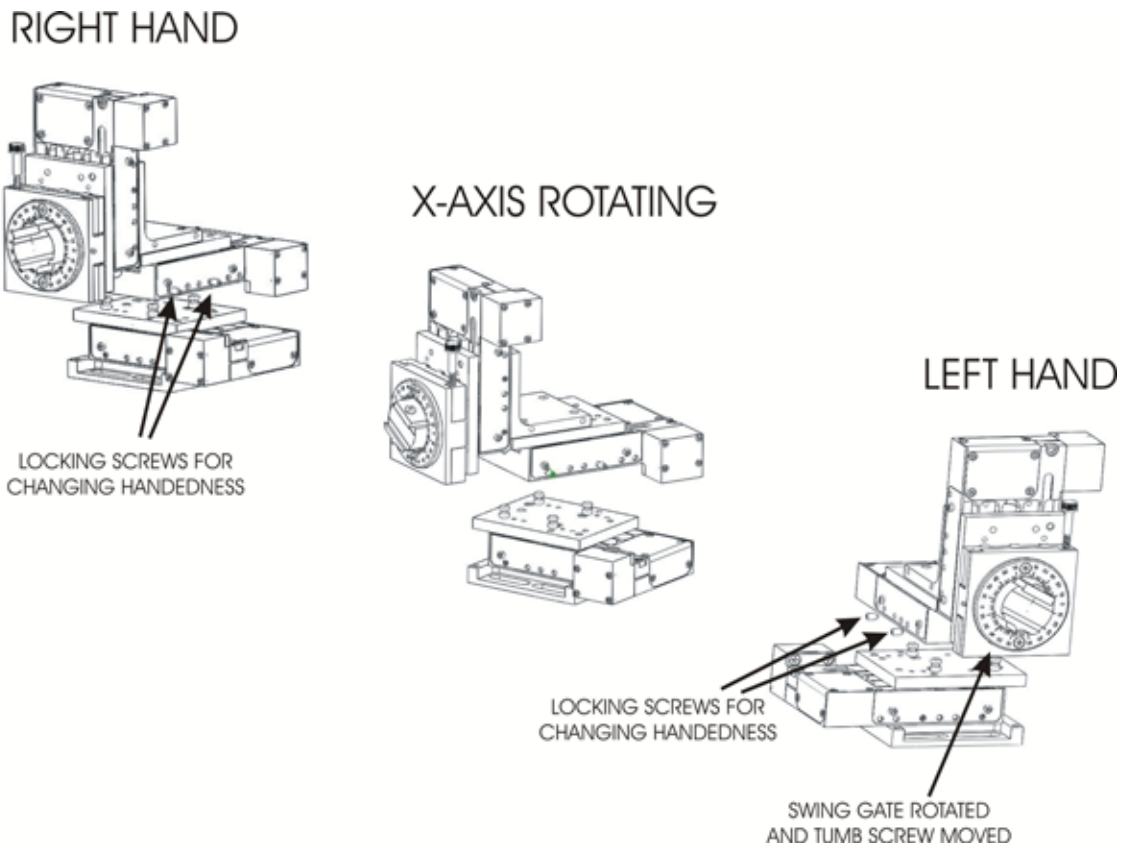


Figure 3-5. Rotating the MP-225/M for right and left handedness.

3.7 Instructions Used in Special Installations Only

TO INSTALL AND USE THE RIGHT-ANGLE ADAPTER (285300)

Open the swing-out gate and remove it from the front of the MP-225/M by removing the four Phillips-head screws. Next, install the right-angle adapter on the front of the MP-225/M using the supplied M3-0.5 hex head screws. Finally, install the swing out gate on the right-angle adapter using the four Phillips-head screws. With the right angle in place, the manipulator (right-handed) can be turned 90 degrees clockwise so that its bulk faces to the right instead of the back of your microscope.

Having made the 90-degree rotation, if you wish to use the automated features and diagonal movement mode of the MP-225/M, two other minor changes must be made:

First, as the X and Y-axes have interchanged with the 90-degree rotation, the wires controlling these two axes must be switched at the connector box.

Second, in order to have the Y axis movement be in the forward direction during the home move, you need to rotate the Y axis 180 degrees with respect to the rest of the manipulator. To reverse the Y (bottom axis in the rotated configuration), locate and loosen the 4 set screws that attach the Y to the X. These are on the sides of the X or middle axis in the stack. When the set screws are loose, you can move the Y off the X, rotate it 180 degrees and reinsert its mounting posts into the correct holes in the X (the holes with set screws). Finally, press the

two axes firmly together and retighten the setscrews. Now the Y-axis will move forward during the home move and back during the work position move.

4. OPERATIONS

4.1 First Time Use

4.1.1 Line Power (Mains)

The power cord provided with the Lambda 10-3 connects to the Power Entry Module located on the back of the unit (see diagram below). This module also includes the line fuse .

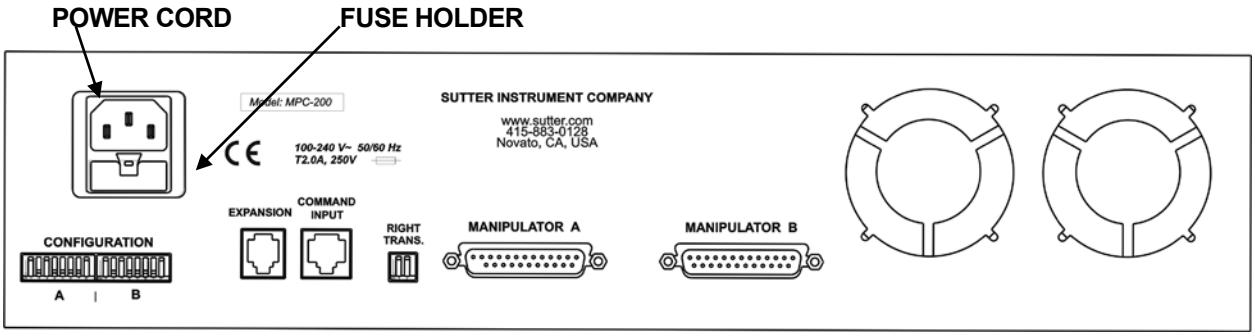


Figure 4-1. MPC-200 Controller cabinet (rear view) showing power connection and fuse.

The power switch is located on the front panel as shown in Figure 4-2.

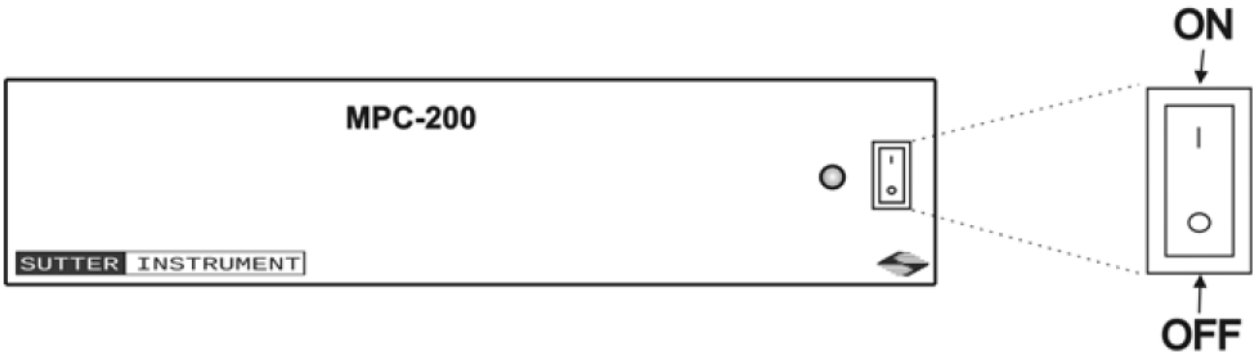


Figure 4-2. Power switch (front panel).

The MPC-200 has a “universal” power supply that runs on 110 volts or 220 volts AC, 50 or 60 Hz. You do not need to change settings or fuses to suit local conditions. Make certain that the ON/OFF Switch located on the front panel of the MPC-200 cabinet is turned OFF. Plug the power cord provided with the MPC-200 into the Line Input socket on the Power Entry Module and then to a power source of the appropriate voltage and frequency.

You must replace the fuse with the appropriate value (see the Technical Specifications), otherwise your protection from fire and electric shock will be compromised.

4.2 Make It Go

Turn on the power using the ON/OFF switch on the front panel of the MPC-200 controller cabinet.

4.3 Basic Operation

4.3.1 Initialization

At power on the first display is the copyright message:

```
 SUTTER INSTRUMENT
 MPC-200 CONTROLLER
    VERSION 3.15
 INITIALIZING.....
```

4.3.2 Standard Screen

```
ROE-200        MODE:0
       Z=00000
 X=00000    Y=00000
```

4.3.3 No Manipulator Connected Screen

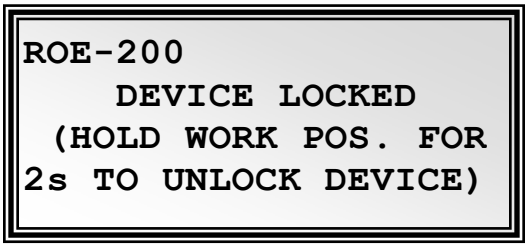
```
   NO MANIPULATOR
  DETECTED, PLEASE
TURN OFF CONTROLLER
 ATTACH MANIPULATOR
```

4.3.4 Diagonal-Mode Screen (DIAGONAL)

```
ROE-200        MODE:0
       Z=00000
 X=00000    Y=00000
       D=00000
```

4.3.5 Move in progress screen:

4.3.6 Setting Work Position (WORK POS)



(This page intentionally blank.)

5. EXTERNAL CONTROL

5.1 General

Controlling the MPC-325 externally via computer is accomplished by sending commands to the MPC-200 controller over the USB interface between the computer and the USB connector on the rear of the ROE-200 that's connected to the MPC-200 controller. The USB device driver for Windows is downloadable from Sutter Instrument's web site (www.sutter.com). The MPC-325 (MPC-200) requires USB CDM (Combined Driver Model) Version 2.10.00 or higher. The CDM device driver for the MPC-325 (MPC-200) 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 MPC-325 (MPC-200). 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.

5.1 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 5-1. USB-VCP interface serial port settings.

Property	Setting
Data ("Baud") Rate (bits per second (bps))	128000
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.

5.2 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.

5.3 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 transmitted to the controller must contain an unsigned binary value. Attempting to code command sequences as “strings” is not advisable. Any command data returned by the controller should be initially treated as a sequence of unsigned byte values upon reception. 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 MPC-200, all axis position values (number of microsteps) are stored as “unsigned long” 32-bit positive-only values, and each is transmitted and received to and from the controller as four contiguous bytes.

5.4 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 (see Microns/microsteps conversion table for conversion factors).

Microsteps are stored as positive 32-bit values (“unsigned long” for C/C++, “uint32” for MATLAB, “U32” for LabVIEW, etc.). “Unsigned” means the value is always positive; negative values are not allowed. The positive-only values can also be stored in signed data type variables if necessary, in which case care must be taken to ensure that only positive values are exchanged with the controller (do not allow values that are less than 0).

The 32-bit value 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/AMD 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 and Python (script programming language) are examples of environments that adapt to the system on which each is running, so Little-Endian enforcement may be needed if running on a Big-Endian system. Some processors (e.g., ARM) can be configured for specific endianess.

5.5 Microsteps and Microns (Micrometers)

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

Table 5-2. Microns/microsteps conversion factors (multipliers).

System/Device	From/To Units	Conv. Factor
MP-225/M micromanipulator	μsteps → μm	0.0625
	μm → μsteps	16

** Same applies to MP-285/M & MP-265/M micromanipulator, 3DMS & MPC-78 stages, and MOM & SOM microscope objective movers. Other devices:*

- *MP-x45[S]/M & MP-865/M μmanipulators and MPC-x8 series stages: 1 μstep = 0.046875 μm; 1 μm = 21.333333333 μsteps.*
- *MT-800-based translators: 1 μstep = 0.078125 μm; 1 μm = 12.8 μsteps.*

For accuracy in your application, type these conversion factors as "double" (avoid using the "float" type as it lacks precision with large values). When converting to microsteps, type the result as a 32-bit "unsigned long" (C/C++), "uint32" (MATLAB), or "U32" (LabVIEW) integer (positive only) value. When converting to microns, type the result as a "double" (C/C++, MATLAB) or "DBL" (LabVIEW) 64-bit double-precision floating-point value.

5.1 Travel Ranges & Bounds

The following table shows the travel lengths, ranges, and bounds for each axis of each supported device.

Table 5-3. Travel distances and bounds.

Device	Axis	Len. (mm)	Microns (Micrometers (μm))	Microsteps (μsteps)
MP-225/M *	X	25mm	0 – 25,000	0 – 266,667
	Y	25mm	0 – 25,000	0 – 266,667
	Z	25mm	0 – 25,000	0 – 266,667

** Same applies to MP-285/M μmanipulators, 3DMS & MPC-78-series stages, & SOM objective mover.*
Other devices:

- *TRIO MP-x45/M series micromanipulator and MPC-x8 series stages: 25mm (533,333 microsteps) for X, Y, & Z. (Requires firmware v3.19+.)*
- *TRIO MP-865/M micromanipulator: 50mm for X, 12.5mm for Y, and 25mm for Z. (Requires firmware v3.21+.)*
- *MP-265/M micromanipulator: 25mm for X & Z, 12.5mm for Y. (Discontinued product – replaced by TRIO MP-865/M.)*
- *MT-8x0 (MT-22xx) series translator: 22mm in all three axes. Only X & Y are connected (Z can be optionally connected to another device (e.g., a focus drive)).*
- *MOM objective mover (firmware v3.13 or 3.16, and device Port A only): 21.5mm in all three axes.*

5.1 Travel Speed

The following table shows the travel speeds for single-, double-, and triple-axis movements for supported devices using orthogonal move commands.

Table 5-4. Travel speeds.

Device	mm/sec or μm/ms		
	Single Axis	Dual Axis (x 1.4)	Triple Axis (x 1.7)
MP-225/M*	3	4.2	5.1

** Applies also to the MP-245[S]/M, MP-845[S]/M, MP-865/M. & MP-265/M μmanipulators; and MPC-x8 series stages.*

NOTE: The MP-285/M has a single-axis speed of 5 mm/sec., 7 for double axis, and 8.5 for triple-axis movement. The same applies also to the 3DMS & MPC-78 stages, MT-800-based translators, and SOM & MOM objective movers.

5.2 Commands

5.2.1 Get Connected-Devices Status ('A') Command (FW <v3)

This command is used to obtain information on how many devices are connected. The command sequence consists of one byte as shown in the following table, and 0 or 2 bytes for the returned data: 1 byte for the number of devices connected followed by the completion indicator (1 byte).

Table 5-5. Get Connected Devices Status ('A' Command (FW <3).

Tx/-Delay/-Rx	Ver.	Total Bytes	Byte Offset (Len.)	Value			Alt-key-pad #	Ctrl-char	ASCII def./-char.	Description
				Dec.	Hex.	Binary				
Tx	<3	1	0	65	41	0100 0001	0065		A	Command. Returns the number of devices connected (0-4)
	3+	1	0	85	55	1000 0101	0085		U	
Rx	All	0								Zero bytes = no devices connected. ROE displays "NO MANIPULATOR CONNECTED".
		6	0	1 - 4	01 – 04	0000 0000 – 0000 0100		^A – ^D	<SOH> – <EOT>	Number of devices connected

5.2.2 Get Connected-Devices Status ('U') Command (FW v3+)

This command is used to obtain information on how many devices are connected and the connected status of each one on port basis. The command sequence consists of one byte as shown in the following table, and six bytes for the returned data: 1 byte for the number of devices connected followed four bytes containing connected status for Port 1 through 4, and completion indicator (1 byte).

Table 5-6. Get Connected Devices Status ('A' (FW <3) or 'U' (FW 3+)) Command.

Tx/-Delay/-Rx	Ver.	Total Bytes	Byte Offset (Len.)	Value			Alt-key-pad #	Ctrl-char	ASCII def./-char.	Description
				Dec.	Hex.	Binary				
Tx	**<3**	1	0	65	41	0100 0001	0065		**A**	Command. Returns the number of devices connected (0-4), and the connected status of Ports 1 - 2 (1st controller) and 3 - 4 (2nd controller daisy chained to the first)
	3+	1	0	85	55	1000 0101	0085		**U**	
Rx	All	0								Zero bytes = no devices connected. ROE displays "NO MANIPULATOR CONNECTED".
		6	0	1 - 4	01 – 04	0000 0000 – 0000 0100		^A – ^D	<SOH> – <EOT>	Number of devices connected
										Device # Connected Status
			1	0	0	0000 0000		^@	<NUL>	1: No
				1	1	0000 0001		^A	<SOH>	1: Yes
			2	0	0	0000 0000		^@	<NUL>	2: No
				1	1	0000 0001		^A	<SOH>	2: Yes
			3	0	0	0000 0000		^@	<NUL>	3: No
				1	1	0000 0001		^A	<SOH>	3: Yes
			4	0	0	0000 0000		^@	<NUL>	4: No
				1	1	0000 0001		^A	<SOH>	4: Yes
			13	13	0D	0000 1101		^M	<CR>	Completion indicator

5.2.3 Get Active Device & Firmware Version ('K') Command

This command is used to obtain information on which of the connected devices is currently active. The command also returns the controller's firmware version (in Firmware Ver. 3 and above). The command sequence consists of one byte as shown in the following table, and two or four bytes for the returned data: 1 byte for the active device number (1 – 4), two bytes for Firmware version number (Ver. 3 and above only), and completion indicator (1 byte).

Table 5-7. Get Active Device & Firmware Version ('K') Command.

Tx/-Delay/-Rx	Ver.	Total Bytes	Byte Offset (Len.)	Value			Alt-key-pad #	Ctrl-char	ASCII def./-char.	Description
				Dec.	Hex.	Binary				
Tx	All	1	0	75	4B	0100 1011	0075		**K**	Command
Rx	**<3**	2	0	1 - 4	01 – 04	0000 0000 – 0000 0100		^A – ^D	<SOH> – <EOT>	Currently-active device (1 – 4).
			1	13	0D	0000 1101		^M	<CR>	Task-completion indicator
Rx	**3+**	4	0	1 - 4	01 – 04	0000 0000 – 0000 0100		^A – ^D	<SOH> – <EOT>	Currently-active device (1 – 4).
			1	Minor version number coded in BCD (e.g., if ver. = 3.15, then byte = 0x15 (upper nibble = 1; lower nibble = 5))						
			2	Major version number coded in BCD (e.g., if ver. = 3.15, then byte = 0x03 (upper nibble = 0; lower nibble = 3))						
			3	13	0D	0000 1101		^M	<CR>	Completion indicator

Extracting the MPC-200 Firmware Version Number: The firmware version number returned by the 'K' command is encoded in BCD (Binary Coded Decimal) in two bytes, with minor version byte first, followed by major version byte, each of which contains two digits with first in the upper nibble and the next in the lower nibble. For example, if the complete version is 3.15, then the bytes at offsets 1 and 2 will show (in hexadecimal) as 0x15 0x03 (ret[1] and ret[2] as shown in the following code snippets). The following code shows how to extract and convert the 4 BCD digits into usable forms for later comparison without altering the original command return data (written in C/C++ and easily portable to Python, Java, C#, MATLAB script, etc.).

```
/* "ret" is the array of bytes containing
the 'K' command's return data */
/* define variables */
unsigned char verbyte;  /* temp work byte */
int minver, majver, majminver;
float version;
```

Minor version number as an integer (e.g., 15):

```
verbyte = ret[1]; /* get minor ver. digits */
/* get 1's digit & then get & add 10's digit */
minver = (verbyte & 0x0F) +
   ((verbyte >>4 & 0x0F) * 10);
```

Major version number as an integer (e.g., 3):

```
verbyte = ret[2];  /* get major ver. digits */
majver = (verbyte & 0x0F) +
   ((verbyte >>4 & 0x0F) * 10);
```

Complete (thousands) version as an integer (e.g., 315):

```
majminver = majver * 100 + minver;
```

Complete version as a floating-point number (e.g., 3.15):

```
version = majminver * .01;
```

5.2.4 Get Current Position ('C') Command

This command is used to obtain the current position (X, Y, & Z coordinates) of the manipulator or stage. The command sequence consists of one byte as shown in the following table, followed by 12 bytes containing currently-active device number 1 – 4 (1 byte), X, Y, & Z position values in microsteps (4 bytes each), and completion indicator (1 byte).

Table 5-8. Get Current Position (‘C’) command.

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 (‘C’)	Tx	All	1	0	67	43	0100 0011	0067		C	Command
	Rx		14	0	1-4	01 – 04	0000 0000 – 0000 0100		^A – ^D	<SOH> – <EOT>	Drive number (1 – 4) to which the current position applies
					Three 4-byte (32-bit) values (current positions in μsteps of X, Y, & Z) + 1 byte for completion indicator. See *Ranges and bounds* table for minimum and maximum values.						
				1-4							X microsteps
				5-8							Y microsteps
				9-12							Z microsteps
				13	13	0D	0000 1101		^M	<CR>	Completion indicator

5.2.5 Change Active Device (‘I’) Command

This command is used to change which device (1- 4) is currently active. The command sequence consists of one byte as shown in the following table, and returns 1 (Ver. below 1.06) or 2 bytes: 1st byte containing specified device number if it exists or ASCII ‘E’ (for “error”) if not, and the completion indicator (1 byte).

Table 5-9. Change Active Device (‘I’) command.

Command	Tx/-Delay/-Rx	Ver.	Total Bytes	Byte Offset (Len.)	Value			Alt-key-pad #	Ctrl-char	ASCII def./-char.	Description
					Dec.	Hex.	Binary				
Change Active Device (‘I’)	Tx	All	3	0	73	49	0100 1001	0073		I	Command
				1	1-4	01 – 04	0000 0001 – 0000 0100	0001 – 0004	^A – ^D	<SOH> – <EOT>	Device number (by value) to change (1 through 4)
	Rx	1-1.05	1	0	13	0D	0000 1101		^M	<CR>	Task-completion indicator
		1.06+	2	0	1-4 or 69	01 – 04 or 45	0000 0001 – 0000 0100 or 0100 0101		^A – ^D	<1> – <4> or ‘E’	If device specified exists, then device number (1-4) is returned. Else, ‘E’ (error) is returned.
				1	13	0D	0000 1101		^M	<CR>	Completion indicator

5.2.6 Move to Controller-Defined HOME Position (‘H’) Command

This command moves to the position last defined by the HOME key. Movement is equivalent to pressing the HOME key.

Table 5-10. Move to controller-defined HOME position (‘H’) command.

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 Home Position[1] (‘H’)	Tx	All	1	0	72	48	0100 1000	0072		H	Command
	Rx	All	1	13	13	0D	0000 1101		^M	<CR>	Completion indicator

5.2.7 Move to Controller-Defined WORK Position (‘Y’) Command

This command moves to the position defined last defined for the WORK key. Movement is equivalent to pressing the WORK key.

Table 5-11. Move to controller -defined WORK position (‘Y’) command.

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 Work Position [2] (‘Y’)	Tx	All	1	0	89	59	0101 1001	0089		Y	Command
	Rx	All	1	13	13	0D	0000 1101		^M	<CR>	Completion indicator

5.2.8 Move to Center Position (‘N’) Command (FW =< 1.03)

This command moves to the center position (midpoint between beginning of travel and end of travel positions). Movement is equivalent to pressing the CENTER key.

NOTE: This command (‘N’) along with the momentary “CENTER” switch on the back of the ROE-200 have the “Move to Center Position” functionality only in a system programmed with Firmware version 1.03 or lower.

Table 5-12. Move to Center Position (‘N’) command (FW <= 1.03).

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 Center Position (‘N’)	Tx	All	1	0	78	4E	0100 1110	0078		N	Command
	Rx	All	1	0	13	0D	0000 1101		^M	<CR>	Completion indicator

5.2.9 Calibrate (‘N’) Command (FW > 1.03)

This command moves to the Beginning Of Travel (BOT) position. It has the same functionality as pressing the momentary “CALIBRATE” switch on back of the ROE-200.

[1] The “Home Position” is defined manually on the ROE-200.
[2] The “Work Position” is defined manually on the ROE-200.

NOTE: This command (‘N’) along with the momentary “CALIBRATE” switch on the back of the ROE-200 have the “calibrate” functionality only in a system programmed with a firmware version above 1.03.

Table 5-13. Calibrate (‘N’) command (FW > 1.03).

Command	Tx/-Delay/-Rx	Ver.	Total Bytes	Byte Offset (Len.)	Value			Alt-key-pad #	Ctrl-char	ASCII def./-char.	Description
					Dec.	Hex.	Binary				
Calibrate (‘N’)	Tx	All	1	0	78	4E	0100 1110	0078		N	Command
	Rx	All	1	0	13	0D	0000 1101		^M	<CR>	Completion indicator

5.2.10 Move to Specified Position Orthogonally at Full Speed (‘M’) Command

This command instructs the controller to move all three axes to the position specified, in a stereotypic or orthogonal manner, at the highest speed. The command sequence consists of thirteen bytes: Command byte followed by three sets of four bytes containing position information in microsteps for X, Y, and Z.

Table 5-14. Move to specified position (‘M’) command.

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 Specified Position Orthogonally at Full Speed (‘M’)	Tx	All	14	0	77	4D	0100 1101	0077		M	Moves X, Y, and Z to specified position (stereotypic at fastest speed)
					X, Y, & Z target absolute positions, in microsteps, each consisting of 4 contiguous bytes representing a single 32-bit positive integer value (see *Ranges and bounds* table).						
				1-4							X μsteps
				5-8							Y μsteps
				9-12							Z μsteps
				13	13	0D	0000 1101	0013	^M	<CR>	Command seq. term.
	=> 30ms										Time of travel (see Notes)
	Rx	All	1	0	13	0D	0000 1101		^M	<CR>	Completion indicator

5.2.11 Move to Specified Position in a Straight Line at Specified Speed (‘S’) Command

Move to Specified Position in a Straight Line at Specified Speed (‘S’)CommandThis command instructs the controller to move all three axes to the position specified, in straight line at the specified speed (Speed Level 0 – 15). The command sequence consists of fourteen bytes.

Table 5-15. Move to Specified Position in a Straight Line at Specified Speed ('S') command.

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 Specified Position in a Straight Line at Specified Speed ('S')	**Tx**	**3+**	14	0	83	53	0101 0011	0083		**S**	Command
				1	0 – 15	00 – 0F	0000 0000 – 0000 FFFF	0000 – 0015	^@ – ^O	<NUL> – <SI>	Velocity (0 = slowest, 15 = fastest) (See next table)
	30ms										⚠ **Required delay between Velocity byte and remaining bytes**
	Tx				X, Y, & Z target absolute positions, in microsteps, each consisting of 4 contiguous bytes representing a single 32-bit positive integer value (see *Ranges and bounds* table).						
				2-5							X μsteps
				6-9							Y μsteps
				10-13							Z μsteps
	=> 30ms										Time of travel (see *Travel Lengths and Durations* note)
	Rx		1	0	13	0D	0000 1101		^M	<CR>	Completion indicator (streaming positional data is OFF – see 'F' command) once movement completes; or, streaming current position data (see 'O' command) and then completion indicator once movement completes. (See *'S' Command's Streaming Return Data for Current Position* note.)

Table 5-16. Straight-Line Move 'S' Command Speeds.

Speed Setting	mm/sec or μm/ms	μm/sec or nm/ms	nm/sec	% of Max.
15	1.30000	1300.00	1300000	100.00%
14	1.21875	1218.75	1218750	93.75%
13	1.13750	1137.50	1137500	87.50%
12	1.05625	1056.25	1056250	81.25%
11	0.97500	975.00	975000	75.00%
10	0.89375	893.75	893750	68.75%
9	0.81250	812.50	812500	62.50%
8	0.73125	731.25	731250	56.25%
7	0.65000	650.00	650000	50.00%
6	0.56875	568.75	568750	43.75%
5	0.48750	487.50	487500	37.50%
4	0.40625	406.25	406250	31.25%
3	0.32500	325.00	325000	25.00%
2	0.24375	243.75	243750	18.75%
1	0.16250	162.50	162500	12.50%
0	0.08125	81.25	81250	6.25%

5.2.12 Interrupt Move ('^C') Command

This command interrupts a move in progress that was originated by an external command. The command sequence consists of one byte and returns one byte (completion indicator).

Table 5-17. Interrupt move in progress ('^C') command.

Command	Tx/-Delay/-Rx	Ver.	Total Bytes	Byte Offset (Len.)	Value			Alt-key-pad #	Ctrl-char	ASCII def./-char.	Description
					Dec.	Hex.	Binary				
Interrupt Move (^C)	Tx	All	2	0	3	03	0000 0111	0003	^C	<ETX>	Command
	Rx		1	0	13	0D	0000 1101		^M	<CR>	Completion indicator

NOTE: The "Interrupt Move in Progress" (^C) command is the only command to which the controller will respond while a command-initiated move is in progress.

5.2.13 Turn OFF S-Move command streaming data ('F') Command

This command is sent before the 'S' (straight-line move) command to disable the return of current position streaming while an 'S' command initiated move is in progress. Command sequence consists of one byte and returns one byte (completion indicator) and must be sent before the 'S' command to disable the return of streaming data to be effective.

Table 5-18 Command to disable current-position streaming data during an 'S'-command-initiated move.

Command	Tx/-Delay/-Rx	Ver.	Total Bytes	Byte Offset (Len.)	Value			Alt-key-pad #	Ctrl-char	ASCII def./-char.	Description
					Dec.	Hex.	Binary				
Turn OFF S-Move command streaming data ('F')	Tx	3+	1	0	70	46	0100 0110			F	Command (see *'S' Command's Streaming Return Data for Current Position* note)
	Rx		1	0	13	0D	0000 1101		^M	<CR>	Completion indicator

5.2.14 Turn ON S-Move command streaming data ('O') Command

This command is sent before the 'S' (straight-line move) command to enable the return of current position streaming while an 'S' command initiated move is in progress. Command sequence consists of one byte and returns one byte (completion indicator) and must be sent before the 'S' command to enable the return of streaming data to be effective.

Table 5-19 Command to disable current-position streaming data during an 'S'-command-initiated move.

Command	Tx/-Delay/-Rx	Ver.	Total Bytes	Byte Offset (Len.)	Value			Alt-key-pad #	Ctrl-char	ASCII def./-char.	Description
					Dec.	Hex.	Binary				
Turn ON S-Move command streaming data ('O')	Tx	3+	1	0	79	4F	0100 1111			O	Command (see *'S' Command's Streaming Return Data for Current Position* note)
	Rx		1	0	13	0D	0000 1101		^M	<CR>	Completion indicator

5.2.15 Set ROE MODE ('L') Command

Set ROE MODE ('L') Sets the ROE MODE (0 – 9). Command sequence consists of two bytes (command byte and MODE (0 – 9)) and returns one byte (completion indicator).

Table 5-20 Set ROE MODE ('L') command.

Command	Tx/-Delay/-Rx	Ver.	Total Bytes	Byte Offset (Len.)	Value			Alt-key-pad #	Ctrl-char	ASCII def./-char.	Description
					Dec.	Hex.	Binary				
Set ROE MODE ('L')	Tx	All	2	0	76	4C	0100 1100	0076		L	Command
				1	0-9	0-9	0000 0000 – 0000 1001	0000 – 0009			Mode 0 - 9 (coarsest/fastest to finest/slowest)
	Rx	All	1	0	13	0D	0000 1101		^M	<CR>	Task-completion indicator

5.2.16 Command Notes

The following list of notes apply to all the external control commands described in this chapter.

1. **Task-Complete Indicator:** All commands will send back to the computer the "Task-Complete Indicator" to signal the command and its associated function in controller is complete. The indicator consists of one (1) byte containing a value of 13 decimal (0D hexadecimal), and which represents an ASCII CR (Carriage Return).
2. **Intercommand Delay:** A short delay (usually around 2 ms) is recommended between commands (after sending a command sequence and before sending the next command).
3. **Clearing Send/Receive Buffers:** Clearing (purging) the transmit and receive buffers of the I/O port immediately before sending any command is recommended.
4. **Positions in Microsteps and Microns:** All positions sent to and received from the controller are in microsteps (μsteps). See *Microns/microsteps conversion* table) for conversion between μsteps and microns (micrometers (μm)).

   Declaring position variables in C/C++:

```
/* current position for X, Y, & Z */
unsigned long  cp_x_us, cp_y_us, cp_z_us;  /* microsteps */
double         cp_x_um, cp_y_um, cp_z_um;  /* microns */
/* specified (move-to) position for X, Y, & Z */
unsigned long  sp_x_us, sp_y_us, sp_z_us;  /* microsteps */
```

   Use the same convention for other position variables the application might need.

   Declaring the microsteps/microns conversion factors in C/C++:

```
/* conversion factors for MP-225/M, MP-285/M, or MP-265/M based config.
*/
double us2umCF = 0.0625; /* microsteps to microns */
double um2usCF = 16;     /* microns to microsteps */
/* conversion factors for MP-245[S]/M, MP-845[S]/M, or MP-865/M based
config. */
double us2umCF = 0.046875;     /* microsteps to microns */
double um2usCF = 21.333333333; /* microns to microsteps */
/* conversion factors for MT-800 config. */
```

```
double us2umCF = 0.078125; /* microsteps to microns */
double um2usCF = 12.8;     /* microns to microsteps */
```

Converting between microsteps and microns in C/C++:

```
/* converting X axis current position */
cp_x_um = cp_x_us * us2umCF;  /* microsteps to microns */
cp_x_us = cp_x_um * um2usCF;  /* microns to microsteps */
```

Do the same for Y and Z, and for any other position sets used in the application.

5. **Ranges and Bounds:** See *Ranges and Bounds* table for exact minimum and maximum values for each axis of each compatible device that can be connected. All move commands must include positive values only for positions – negative positions must never be specified. All positions are absolute as measured from the physical beginning of travel of a device's axis. In application programming, it is important that positional values be checked (>= 0 and <= max.) to ensure that a negative absolute position is never sent to the controller and that end of travel is not exceeded. All computational relative positioning must always resolve to accurate absolute positions.

   Declaring minimum and maximum absolute position variables in C/C++:

```
/* minimum and maximum positions for X, Y, & Z */
double min_x_um, min_y_um, min_z_um;  /* minimum microns */
double max_x_um, max_y_um, max_z_um;  /* maximum microns */
```

   Set minimum and maximum absolute positions for each axis – see Ranges & Bounds table.

```
/* initialize all minimum positions in microns*/
min_x_um = 0;
min_y_um = 0;
min_z_um = 0;
/* initialize all maximum positions in microns*/
/* MP-225/M, MP-285/M, MP-845[S]/M, MP-245[S]/M, etc. */
max_x_um = 25000;
max_y_um = 25000;
max_z_um = 25000;
/* MP-865/M */
max_x_um = 50000;
max_y_um = 12500;
max_z_um = 25000;
/* MP-265/M */
max_x_um = 25000;
max_y_um = 12500;
max_z_um = 25000;
```

6. **Absolute Positioning System Origin:** The Origin is set to a physical position of travel to define absolute position 0. The physical Origin position is fixed at beginning of travel (BOT). This means that all higher positions (towards end of travel (EOT)) are positive values; there are no lower positions and therefore no negative values are allowed.
7. **Absolute vs. Relative Positioning:** Current position ('c') and move commands always use absolute positions. All positions can be considered "relative" to the Origin (Position 0), but all are in fact absolute positions. Any position that is considered to be "relative" to the current position, whatever that might be, can be handled synthetically by external programming. However, care should be taken to ensure that all relative position

calculations always result in correct positive absolute positions before initiating a move command.

Declaring relative position variables in C/C++:

```
/* relative positions for X, Y, & Z */
double rp_x_um, rp_y_um, rp_z_um;  /* microns */
/* initialize all relative positions to 0 after declaring them */
rp_x_um = rp_y_um = rp_z_um = 0;
```

Enter any positive or negative value for each relative position (e.g., rp_x_um = 1000; rp_y_um = 500; rp_z_um = -200 … etc.

For each axis, check to make sure that the new resultant absolute position (to which to move) is within bounds. Reset the relative position to 0 if not. If relative value is negative, its positivized value must not be greater than the current position. Otherwise, if positive, adding current position with relative position must not exceed the maximum position allowed. If out of bounds, resetting relative position to 0 allow the remaining conversions and movement to resolve without error.

```
/* check to make sure that relative X is within bounds */
if ( ( rp_x_um < 0 && abs(rp_x_um) > cp_x_um ) ||
     (cp_x_um + rp_x_um > max_x_um) )  /* out of bounds? */
    rp_x_um = 0; /* yes, so reset relative pos. to 0 */
```

Repeat the above bounds check for each of the remaining axes.

For each axis, calculate new absolute position in microns and then convert to microsteps before issuing a move command.

```
/* convert X relative position to absolute position */
sp_x_um = cp_x_um + rp_x_um; /* add relative pos. to current pos. */
/* convert new absolute X position in microns to microsteps */
sp_x_us = sp_x_um * um2usCF;
```

Repeat for each of the remaining axes as required before issuing a move command.

8. **Position Value Typing:** All positions sent and received to and from the controller are in microsteps and consist of 32-bit integer values (four contiguous bytes). Position values in microsteps are always positive, so data type must be an “unsigned” integer that can hold 32 bits of data. Although each positional value is transmitted to, or received from, the controller as a sequence of four (4) contiguous bytes, for computer application computational and storage purposes each should be typed as an unsigned 32-bit integer (“unsigned long” in C/C++, “uint32” in MATLAB, “U32” in LabVIEW, etc.).

   Position values in microns (micrometers or μm) should be data typed as double-precision floating point variables (“double” in C/C++ and MATLAB, “DBL” in LabVIEW, etc.).

   Note that in Python, incorporating the optional NumPy package brings robust data typing like that used in C/C++ to your program, simplifying coding and adding positioning accuracy to the application.

9. **Position Value Bit Ordering:** 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 most 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. 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).

10. **Travel Lengths and Durations:** "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.
11. **Orthogonal Move Speed:** Full speed for the "Orthogonal Move 'M'" command is 5000 microns/sec. (5 mm/sec. or microns/millisecond) for single-axis movements (3000 μm/sec. (3 mm/sec. or μm/ms) for MP-225/M).
12. **Straight-Line Move Speeds:** Actual speed for the "Straight-Line Move 'S'" command can be determined with the following formula: (1300 / 16) * (sp +1), where 1300 is the maximum speed in microns/second and "sp" is the speed level 0 (slowest) through 15 (fastest). For mm/second or microns/millisecond, multiply result by 0.001.

Table 5-21.Straight-line move 'S' command speeds.

Speed Setting	mm/sec or μm/ms	μm/sec or nm/ms	nm/sec	in/sec or mil/ms	% of Max.
15	1.30000	1300.00	1300000	0.051181102	100.00%
14	1.21875	1218.75	1218750	0.047982283	93.75%
13	1.13750	1137.50	1137500	0.044783465	87.50%
12	1.05625	1056.25	1056250	0.041584646	81.25%
11	0.97500	975.00	975000	0.038385827	75.00%
10	0.89375	893.75	893750	0.035187008	68.75%
9	0.81250	812.50	812500	0.031988189	62.50%
8	0.73125	731.25	731250	0.028789370	56.25%
7	0.65000	650.00	650000	0.025590551	50.00%
6	0.56875	568.75	568750	0.022391732	43.75%
5	0.48750	487.50	487500	0.019192913	37.50%
4	0.40625	406.25	406250	0.015994094	31.25%
3	0.32500	325.00	325000	0.012795276	25.00%
2	0.24375	243.75	243750	0.009596457	18.75%
1	0.16250	162.50	162500	0.006397638	12.50%
0	0.08125	81.25	81250	0.003198819	6.25%

13. **Multi Axis Movement Speed Increase:** Specified travel speeds are for single-axis movements. When travel traverses a 45° diagonal within a dual-axis square, speed is increased by 40% (x 1.4), and by 70% (x 1.7) within a triple-axis cube.

14. **Move Interruption:** A command should be sent to the controller only after the task of any previous command is complete (i.e., the task-completion terminator (CR) is returned). One exception is the "Interrupt Move" (^C) command, which can be issued while a command-initiated move is still in progress.

15. **Extracting the MPC-200 Firmware Version Number:** The firmware version number returned by the 'K' command is encoded in BCD (Binary Coded Decimal) in two bytes, with minor version byte first, followed by major version byte, each of which contains two-digit pairs , the first of which is in the upper nibble and the next in the lower nibble. For example, if the complete version is 3.15, then the bytes at offsets 1 and 2 will show (in hexadecimal) as 0x15 0x03 (ret[1] and ret[2] as shown in the following code snippets). The following code shows how to extract and convert the 4 BCD digits into usable forms for later comparison without altering the original command return data (written in C/C++ and is easily portable to Python, Java, C#, MATLAB script, etc.).

```
/* "ret" is the array of bytes containing
the 'K' command's return data */
/* define variables */
unsigned char verbyte;  /* temp work byte */
int minver, majver, majminver;
float version;
```

Minor version number as an integer (e.g., 15):

```
verbyte = ret[1]; /* get minor ver. digits */
/* get 1's digit & then get & add 10's digit */
minver = (verbyte & 0x0F) +
   ((verbyte >>4 & 0x0F) * 10);
```

Major version number as an integer (e.g., 3):

```
verbyte = ret[2];  /* get major ver. digits */
majver = (verbyte & 0x0F) +
   ((verbyte >>4 & 0x0F) * 10);
```

Complete (thousands) version as an integer (e.g., 315):

```
majminver = majver * 100 + minver;
```

Complete version as a floating-point number (e.g., 3.15):

```
version = majminver * .01;
```

16. **'S' Command's Streaming Return Data for Current Position:** The Straight-Line Move ('S') command has two modes of operation:

    a. a CR is returned when the target position has been reached (F' (Off) command before the 'S' command sequence), or

    b. streaming positional data is returned while movement is occurring, and then a CR once movement is complete ('O' (On) command before the 'S' command sequence).

    Positional data is streamed at every 1 micron of movement, and the rate (data per second) depends on the 'S' command speed level used. Each positional data block streamed consists of 12 bytes:

    1 The 1st three bytes each contains FF hexadecimal (255 decimal) as a data block signature,

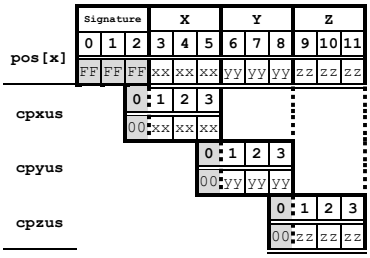
2 the next 3 contains positional data for the X axis,

3 the penultimate is for Y, and

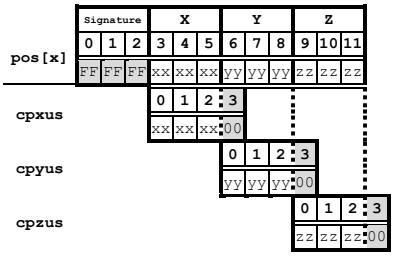
4 last for Z.

All positional data are in microsteps. Each 3-byte position needs to be converted into 4-byte blocks by prepending a byte containing 0, so that the resulting data (now 4 bytes) can be treated programmatically as an unsigned 32-bit "long" (C/C++) or "U32" (LabVIEW) data type. All positional data streamed is in Little-Endian bit/byte order (Wintel), so conversion to 32-bit longs will require bit-order reversal (byte swapping) for Big-Endian platforms (e.g., LabVIEW). The appropriate microstep-to-microns conversion factor is needed according to the device type being moved (see *Microns/microsteps conversion factors (multipliers)* table).

Little-Endian bit/byte order environment:



Big-Endian bit/byte order environment ("pos" is in Little-Endian format):



The following C/C++ code snippets can be used to process the streaming data.

Array for a streamed 12-byte block of data containing current position

```
unsigned char pos[12];
```

32-bit variables for current position in microsteps, all initialized to 0 to ensure MSB allows only positive values

```
unsigned long cpxus, cpyus, cpzus;
cpxus = cpyus = cpzus = 0;
```

Copy 24-bit (3-byte) position for each axis to 32-bit (4-byte) equivalents. Use the byte position offsets shown in the diagram above. ("le" means Little Endian; "be" means Big Endian bit/byte order.)

If in Little-Endian environment (e.g., Windows, Intel-MacOSX), copy all 3 U24 bytes for each axis to the respective U32 variables.

```
memcpy(&cpxus[1], &pos[3], 3); /* X */
memcpy(&cpyus[1], &pos[6], 3); /* Y */
memcpy(&cpzus[1], &pos[9], 3); /* Z */
```

If in Big-Endian environment (e.g., legacy MacOS, LabVIEW), copy U24 to U32 byte at a time (1st to 3rd, 2nd to 2nd, & 3rd to 1st). Note that "pos" is always in Little-Endian bit/byte order.

```
memcpy(&cpxus[2], &pos[3], 1); /* X */
memcpy(&cpxus[1], &pos[4], 1);
memcpy(&cpxus[0], &pos[5], 1);
memcpy(&cpyus[2], &pos[6], 1); /* Y */
memcpy(&cpyus[1], &pos[7], 1);
memcpy(&cpyus[0], &pos[8], 1);
memcpy(&cpzus[2], &pos[9], 1); /* Z */
memcpy(&cpzus[1], &pos[10], 1);
memcpy(&cpzus[0], &pos[11], 1);
```

Ready to update UI with current position in microsteps using 32-bit integer values.

Convert microsteps to microns. Use double-precision variables for current position in microns; initialize each to 0.

```
double cpxum, cpyum, cpzum;
cpxum = cpyum = cpzum = 0;
```

Microsteps-to-microns conversion factor (see "Microns / microsteps conversion" table for appropriate factor)

```
double us2umCF = 0.0625;
```

Get microns from microsteps for each axis

```
cpxum = cpxus * us2umCF;
cpyum = cpyus * us2umCF;
cpzum = cpzus * us2umCF;
```

Ready to update UI with current position in microns using double-precision values. Loop for next data block as desired until streaming ends.

For LabVIEW, a 3-byte positional value for an axis can be transferred into a byte array, and then into a U32 data type via a byte-swap function to ensure 24-bit to 32-bit conversion while making sure that no high-order value is misinterpreted as a sign bit (there should never be a negative positional value in the MPC-200). LabVIEW data types (e.g., U16, U32, I32) are always in Big-Endian bit/byte order, while MPC-200 multibyte values are always transcieved in Little-Endian bit/byte order.

A single completion indicator byte (ASCII CR) is returned when streaming ends and target position has been reached.

6. MAINTENANCE

Routine cleaning of the MPC-325-series system is required to prevent excessive dust accumulations. This is done by wiping all exterior surfaces with a dry, soft, cotton cloth.

All cables and connections should be periodically inspected to make sure that all connections are made well and that all connectors are well and evenly seated.

CAUTION: The MP-225/M electromechanical is a precision-machined part, mounted on three stepper motor shafts. As such, it DOES NOT REQUIRE LUBRICATION. Attempting to lubricate any part of the electromechanical assembly will void the warranty and may harm one or more of the motors.

(This page intentionally blank.)

APPENDIX A. LIMITED WARRANTY

- Sutter Instrument Company, a division of Sutter Instrument Corporation, limits the warranty on this instrument to repair and replacement of defective components for two years from date of shipment, provided the instrument has been operated in accordance with the instructions outlined in this manual.
- Abuse, misuse, or unauthorized repairs will void this warranty.
- Warranty work will be performed only at the factory.
- The cost of shipment both ways is paid for by Sutter Instrument during the first three months this warranty is in effect, after which the cost is the responsibility of the customer.
- The limited warranty is as stated above and no implied or inferred liability for direct or consequential damages is intended.
- Consumables, PMTs, galvanometers, and Uniblitz®[1] shutters are exempt from this warranty.
- An extended warranty for up to three additional years can be purchased at the time of ordering, or until the original warranty expires. For pricing and other information, please contact Sutter Instrument.

[1] Uniblitz® is a registered trademark of Vincent Associates.

(This page intentionally blank.)

APPENDIX B. ACCESSORIES

285204	4-inch dovetail extension
285210[1]	Mounting adapter plate
285RBI	Rotating base for MP-285
225RBI	Rotating base for MP-225
X285300	Right angle adapter
X285305	Z-axis vertical extension
X285310	Z-axis horizontal extension
BR-AW	Rod holding clamp for XenoWorks injectors
MP-ROD	Rod holder
285HEA	Hinged headstage mount

[1] For use with MT- or MD-series stands/platforms, or any surface with 1-inch centered holes.

(This page intentionally blank.)

APPENDIX C. FUSE REPLACEMENT

In the event that the controller fails to power up when the power switch is turned on, check the line power fuses to see if either or both have blown. The fuses are located in the fuse holder on the power entry module on the back of the controller. To remove the fuse holder first unplug the power cord from the power entry module. Press down on the lever that is located just above the fuse holder and below the power receptacle and pry the holder straight out of the power entry module.

The fuse holder holds two fuses. Both fuses are of the same type and rating. If either fuse is blown, it is recommended that both fuses be replaced.

The type and rating of both fuses are as follows:

5 x 20 mm glass tube, Time Delay (IEC 60127-2, Sheet III)
T2A, 250V
(Examples: Bussmann GDC-2A. GMC-2A or S506-2-R (RoHS); or
Littelfuse 218 200 or 218 200P (RoHS))

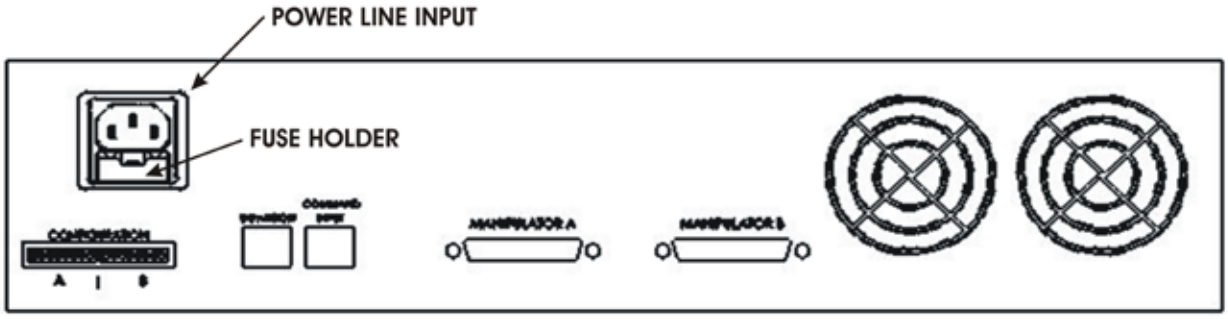


Figure C-1. Rear view of the MPC-200 controller cabinet showing the power entry module and fuse location.

(This page intentionally blank.)

APPENDIX D. TECHNICAL SPECIFICATIONS

D.1. MP-225/M

Resolution	Minimal microstep size is 62.5 nanometers per microstep. Display has single micron resolution.
Speed	3 mm/sec. maximum
Drift	< 1 micron/hr drive mechanism

D.2. MPC-200 Controller

Dimensions (H x W x D):	4 x 15.94 x 11 in (10.16 x 40.49 x 27.94 cm)
Weight:	6.65 lb (6 lb., 10.4 oz.; 3.02 kg)
Electrical:	
Input voltage (Mains)	100 - 240 V, 50/60 Hz
Power consumption	170 Watts maximum per MPC-200 controller (340W max. for an MPC-325-3 or MPC-325-4 system)
Mains fuses (rear of cabinet)	refer to the FUSE REPLACEMENT appendix for details
Cables	(Refer to the following tables for a description of all possible cables.)

Table 6-1. MPC-325 Controller cables and receptacles/connectors.

Controller Rear Panel Port Connector/Receptacle	Cable Connector Types	Connects to ...	Cable Type	Cable Max. Length
MPC-200: (Power entry) 3-pin male connector	← 3-pin power standard (female) | 3-pin male → (Geographical region dependent)	Mains power source.	10A, 250V, with safety ground plug	3 meters (approx. 10 feet)

Controller Rear Panel Port Connector/Receptacle	Cable Connector Types	Connects to ...	Cable Type	Cable Max. Length
MPC-200 Controller Cabinet: MANIPULATOR A or MANIPULATOR B (each a 25-Pin DSUB female receptacle	← DB-25 male | DB-25 female → (Straight-through)	MP-225M (adaptor) (MP-225M Adapter has three RJ45 connectors for three cables that connect to the MP-225M electromechanical)	Minimum of 26 awg stranded wire with 500 Volt.	3 meters (approx. 10 feet)
MPC-200: COMMAND INPUT and ROE-200 CONTROLLER	← RJ45 male | RJ45 male →	ROE-200 User Input Device receptacle named "CONTROLLER"		
MPC-200: EXPANSION	← RJ11 male | RJ11 male →	a second MPC-200 controller (and its EXPANSION receptacle)		
ROE-200: USB USB "A" (Device) female receptacle (full-sized)	← A connector | B connector →	Host computer's USB "B" receptacle (full-sized)	Dielectric separation of circuits. Foil shielding.	3 meters (approx. 10 feet)

D.3. ROE-200

Dimensions: 10 x 6 x 4 in (25.4 x 15.24 x 10.16 cm)

Weight: 3.5 lbs. (1.6 kg)

APPENDIX E. QUICK REFERENCE

E.1. Manual Operation

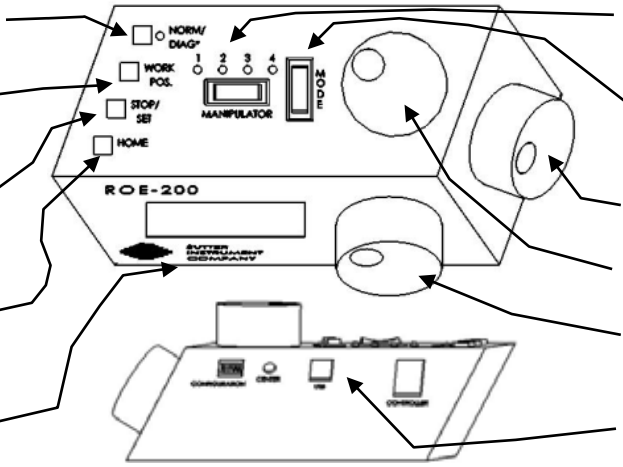
NORM/DIAG: Normal or Diagonal (LED on) mode.

WORK POS: Moves to defined Work Position. Hold >2 sec., lock/unlocks ROE.

STOP/SET: Stops movement. Hold down and press WORK sets Work pos. to current pos.

HOME: Moves to position 0,0,0.

Display: Shows current position along with other state information.



MANIPULATOR: Selects Manipulator 1 through 4 (as connected).

MODE: Selects Mode 0 through 9, or 0 and 5 (ROE Switch 1 OFF).

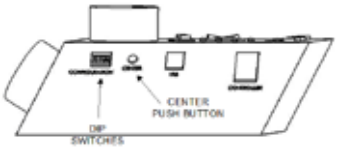
X-Axis Movement Knob:

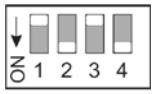
Z-Axis Movement Knob:

Y-Axis Movement Knob:

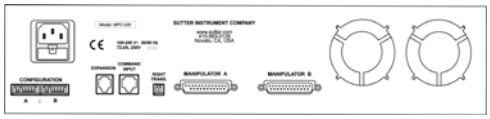
CENTER (CALIBRATE):

E.2. Configuration





ROE DIP Sw



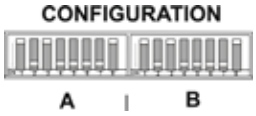


Table E-1. ROE-200 configuration switches (rear).

Switch #	Definition	State	Setting	Position
1	Modes	**All modes**	**ON**	**DOWN**
		Mode 0 and 5	OFF	UP
2	Relative coordinates during Diagonal Mode	**Enabled**	**ON**	**DOWN**
		Disabled	OFF	UP
3	Manipulator selection	Not cyclical	ON	DOWN
		Cyclical	**OFF**	**UP**
4	Reserved		**ON**	**DOWN**

Table E-2. MPC-200 rear-panel config. Switches 1 – 4 (angle setting) for Device, A or B.

Steepness Relative to 45°	Angle	Switch 1	2	3	4	(Value)
More	**11**	0	0	0	0	0
	14	1	0	0	0	1
	21	0	1	0	0	2
	27	1	1	0	0	3
	29	0	0	1	0	4
	35	1	0	1	0	5
	39	0	1	1	0	6
None	**45**	1	1	1	0	7
Less	**39**	0	0	0	1	8
	35	1	0	0	1	9
	29	0	1	0	1	10
	27	1	1	0	1	11
	21	0	0	1	1	12
	14	1	0	1	1	13
	11	0	1	1	1	14
	7	1	1	1	1	15

0 = OFF (Up); 1 = ON (Down)

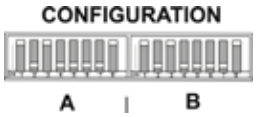


Table E-3. MPC-200 rear-panel config. Switches 5 – 8 for Device A or B.

Switch #	Definition	State	Setting	Position
5	**X-axis knob rotation for forward movement**	**Clockwise ***	**ON**	**DOWN**
		Counterclockwise	OFF	UP
6	**Y-axis knob rotation for forward movement**	**Clockwise ***	**ON**	**DOWN**
		Counterclockwise	OFF	UP
7	**Z-axis knob rotation for forward movement**	**Clockwise ***	**ON**	**DOWN**
		Counterclockwise	OFF	UP
8	**Y-Axis lockout during HOME & WORK position moves**	Y is moved	ON	DOWN
		Y not moved **	**OFF**	**UP**

* Normal operation (factory default).

** Normal operation (factory default) in MPC-365 series systems.

RIGHT TRANS

Table E-4. MPC-200 rear-panel RIGHT TRAN config. switches for Device A or B.

Switch #	Definition	State	Setting	Position
1	**Manipulator A right angle installed state**	Installed	ON	DOWN
		Not installed *	**OFF**	**UP**
2	**Manipulator B right angle installed state**	Installed	ON	DOWN
		Not installed *	**OFF**	**UP**

* Normal operation (factory default).

E.3. External Control

Controlling the MPC-325 externally via computer is accomplished by sending commands to the MPC-200 controller over the USB interface between the computer and the USB connector on the rear of the ROE-200 that's connected to the MPC-200 controller. The USB device driver for Windows is downloadable from Sutter Instrument's web site (www.sutter.com). The MPC-325 (MPC-200) requires USB CDM (Combined Driver Model) Version 2.10.00 or higher. The CDM device driver for the MPC-325 (MPC-200) 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 MPC-325 (MPC-200). 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 E-5. USB-VCP interface serial port settings.

Property	Setting
Data ("Baud") Rate (bits per second (bps))	128000
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 transmitted to the controller must contain an unsigned binary value. Attempting to code command sequences as "strings" is not advisable. Any command data returned by the controller should be initially treated as a sequence of unsigned byte values upon reception. 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 MPC-200, all axis position values (number of microsteps) are stored as "unsigned long" 32-bit positive-only 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 (see *Microns/microsteps conversion* table for conversion factors).

Microsteps are stored as positive 32-bit values ("unsigned long" for C/C++, "uint32" for MATLAB, "U32" for LabVIEW, etc.). "Unsigned" means the value is always positive; negative values are not allowed. The positive-only values can also be stored in signed data type variables if necessary, in which case care must be taken to ensure that only positive values are exchanged with the controller (do not allow values that are less than 0).

The 32-bit value 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/AMD 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 and Python (script programming language) are examples of environments that adapt to the system on which each is running, so Little-Endian enforcement may be needed if running on a Big-Endian system. Some processors (e.g., ARM) can be configured for specific endianess.

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 E-6. Microns/microsteps conversion factors (multipliers).

Device	From/To Units	Conversion Factor (multiplier)
MP-225/M micromanipulator	μsteps → μm	0.0625
	μm → μsteps	16

** Same applies to MP-285/M & MP-265/M micromanipulator, 3DMS & MPC-78 stages, and MOM & SOM microscope objective movers. Other devices:*

- *MP-x45[S]/M & MP-865/M μmanipulators and MPC-x8 series stages: 1 μstep = 0.046875 μm; 1 μm = 21.333333333 μsteps.*
- *MT-800-based translators: 1 μstep = 0.078125 μm; 1 μm = 12.8 μsteps.*

For accuracy in your application, type these conversion factors as "double" (avoid using the "float" type as it lacks precision with large values). When converting to microsteps, type the result as a 32-bit "unsigned long" (C/C++), "uint32" (MATLAB), or "U32" (LabVIEW) integer (positive only) value. When converting to microns, type the result as a "double" (C/C++, MATLAB) or "DBL" (LabVIEW) 64-bit double-precision floating-point value.

Table E-7. Travel distances and bounds.

Device	Axis	Len. (mm)	Microns (Micrometers (μm))	Micro-steps (μsteps)
MP-225/M *	X	25mm	0 – 25,000	0 – 266,667
	Y	25mm	0 – 25,000	0 – 266,667
	Z	25mm	0 – 25,000	0 – 266,667

** Same applies to MP-285/M μmanipulators, 3DMS & MPC-78-series stages, & SOM objective mover. Other devices:*

- *TRIO MP-x45/M series micromanipulator and MPC-x8 series stages: 25mm (533,333 microsteps) for X, Y, & Z. (Requires firmware v3.19+.)*
- *TRIO MP-865/M micromanipulator: 50mm for X, 12.5mm for Y, and 25mm for Z. (Requires firmware v3.21+.)*
- *MP-265/M micromanipulator: 25mm for X & Z, 12.5mm for Y. (Discontinued product – replaced by TRIO MP-865/M.)*
- *MT-8x0 (MT-22xx) series translator: 22mm in all three axes. Only X & Y are connected (Z can be optionally connected to another device (e.g., a focus drive)).*

- *MOM objective mover (firmware v3.13 or 3.16, and device Port A only): 21.5mm in all three axes.*

Travel Speed: The following table shows the travel speeds for single-, double-, and triple-axis movements for supported devices using orthogonal move commands.

Table E-8. Travel speeds.

Device	mm/sec or μm/ms		
	Single Axis	Dual Axis (x 1.4)	Triple Axis (x 1.7)
MP-225/M *	3	4.2	5.1

** Applies also to the MP-245[S]/M, MP-845[S]/M, MP-865/M. & MP-265/M μmanipulators; and MPC-x8 series stages.*

NOTE: The MP-285/M has a single-axis speed of 5 mm/sec., 7 for double axis, and 8.5 for triple-axis movement. The same applies also to the 3DMS & MPC-78 stages, MT-800-based translators. and SOM & MOM objective movers.

Command Reference: The following table lists all the external-control commands for the MPC-325 (MPC-200).

Table E-9. MPC-325 (MPC-200) 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 Connected-Devices Status ('A') (FW <3)	**Tx**	**<3**	1	0	65	41	0100 0001	0065		**A**	Returns the number of devices connected (0-4). *NOTE: This command is replaced by ('U') command in Firmware Ver. 3 and above (described next).*
	Rx	**<3**	0	Zero bytes = no devices connected. ROE displays "NO MANIPULATOR CONNECTED".							
	Rx	**<3**	2	0	1 - 4	01 - 04	0000 0000 - 0000 0100		^A - ^D	<SOH> - <EOT>	Number of devices connected
				1	13	0D	0000 1101		^M	<CR>	Completion indicator
Get Connected-Devices Status ('U') (FW 3+)	**Tx**	**3+**	1	0	85	55	1000 0101	0085		**U**	Returns the number of devices connected (0-4), and the connected status of Ports 1 - 2 (1st controller) and 3 - 4 (2nd controller daisy chained to the first). *NOTE: This command replaces the previous command ('A') in Firmware Ver. 3 and above.*
	Rx	**3+**	0	Zero bytes = no devices connected. ROE displays "NO MANIPULATOR CONNECTED".							
	Rx	**3+**	6	0	1 - 4	01 - 04	0000 0000 - 0000 0100		^A - ^D	<SOH> - <EOT>	Number of devices connected
											Device # Connected Status
				1	0	0	0000 0000		^@	<NUL>	1: No
					1	1	0000 0001		^A	<SOH>	1: Yes
				2	0	0	0000 0000		^@	<NUL>	2: No
					1	1	0000 0001		^A	<SOH>	2: Yes
				3	0	0	0000 0000		^@	<NUL>	3: No
					1	1	0000 0001		^A	<SOH>	3: Yes
				4	0	0	0000 0000		^@	<NUL>	4: No
					1	1	0000 0001		^A	<SOH>	4: Yes
				5	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				
Get Active Device & Firmware Version ('K')	**Tx**	All	1	0	75	4B	0100 1011	0075		**K**	Command
	Rx	**<3**	2	0	1 – 4	01 – 04	0000 0000 – 0000 0100		^A – ^D	<SOH> – <EOT>	Currently-active device (1 – 4).
				1	13	0D	0000 1101		^M	<CR>	Task-completion indicator
	Rx	**3+**	4	0	1 – 4	01 – 04	0000 0000 – 0000 0100		^A – ^D	<SOH> – <EOT>	Currently-active device (1 – 4).
				1	Minor version number coded in BCD (e.g., if ver. = 3.15, then byte = 0x15 (upper nibble = 1; lower nibble = 5))						
				2	Major version number coded in BCD (e.g., if ver. = 3.15, then byte = 0x03 (upper nibble = 0; lower nibble = 3))						
				3	13	0D	0000 1101		^M	<CR>	Completion indicator
Get Current Position ('C')	**Tx**	All	1	0	67	43	0100 0011	0067		**C**	Command
	Rx		14	0	1-4	01 – 04	0000 0000 – 0000 0100		^A – ^D	<SOH> – <EOT>	Drive number (1 – 4) to which the current position applies
					Three 4-byte (32-bit) values (current positions in μsteps of X, Y, & Z) + 1 byte for completion indicator. See *Ranges* table for minimum and maximum values.						
				1-4 (4)							X pos. in μsteps
				5-8 (4)							Y pos. in μsteps
				9-12 (4)							Z pos. in μsteps
				13	13	0D	0000 1101		^M	<CR>	Completion indicator
Change Active Device ('I')	**Tx**	**All**	2	0	73	49	0100 1001	0073		**I**	Command
				1	1-4	01 – 04	0000 0001 – 0000 0100	0001 – 0004	^A – ^D	<SOH> – <EOT>	Device number (by value) to change (1 through 4)
	Rx	**1 - 1.05**	1	0	13	0D	0000 1101		^M	<CR>	Task-completion indicator
	Rx	**1.06 +**	2	0	1-4 or 69	01 – 04 or 45	0000 0001 – 0000 0100 or 0100 0101		^A – ^D	<1> – <4> or 'E'	If device specified exists, then device number (1-4) is returned. Otherwise, 'E' (error) is returned.
				1	13	0D	0000 1101		^M	<CR>	Completion indicator
Move to Home Position[5] ('H')	**Tx**	All	1	0	72	48	0100 1000	0072		**H**	Command
	Rx	All	1	0	13	0D	0000 1101		^M	<CR>	Completion indicator
Move to Work Position [6] ('Y')	**Tx**	All	1	0	89	59	0101 1001	0089		**Y**	Command
	Rx	All	1	0	13	0D	0000 1101		^M	<CR>	Completion indicator
Move to Center Position ('N') (FW <=1.03)	**Tx**	<= 1.03	1	0	78	4E	0100 1110	0078		**N**	Command: Moves active device to the center of travel position.
	Rx	All	1	0	13	0D	0000 1101		^M	<CR>	Completion indicator
Calibrate ('N') (FW >1.03)	**Tx**	>1.03	1	0	78	4E	0100 1110	0078		**N**	Command: Calibrates the active device.
	Rx	All	1	0	13	0D	0000 1101		^M	<CR>	Completion indicator

[5] The "Home Position" is defined manually on the ROE-200.
[6] The "Work Position" is defined manually on the ROE-200.

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
Move to Specified Position Orthogonally at Full Speed ('M')	**Tx**	All	13	0	77	4D	0100 1101	0077		**M**	Moves X, Y, and Z to specified position (stereotypic at fastest speed)
					X, Y, & Z target absolute positions, in microsteps, each consisting of 4 contiguous bytes representing a single 32-bit positive integer value (see *Ranges* table).						
				1-4 (4)							X μsteps
				5-8 (4)							Y μsteps
				9-12 (4)							Z μsteps
	=> 30ms										Time of travel (see Notes)
	Rx	All	1	0	13	0D	0000 1101		^M	<CR>	Completion indicator
Move to Specified Position in a Straight Line at Specified Speed ('S')	**Tx**	**3+**	14	0	83	53	0101 0011	0083		**S**	Command
				1	0 - 15	00 - 0F	0000 0000 - 0000 FFFF	0000 - 0015	^@ - ^O	<NUL> - <SI>	Velocity (0 = slowest, 15 = fastest) (See Notes)
	30ms										⚠ **Required delay between Velocity byte and remaining bytes**
	Tx				X, Y, & Z target absolute positions, in microsteps, each consisting of 4 contiguous bytes representing a single 32-bit positive integer value (see *Ranges* table).						
				2-5 (4)							X μsteps
				6-9 (4)							Y μsteps
				10-13 (4)							Z μsteps
	=> 30ms										Time of travel (see Notes)
	Rx		1	0	13	0D	0000 1101		^M	<CR>	Completion indicator (streaming positional data is OFF – see 'F' command) once movement completes; or, streaming current position data (see 'O' command) and then completion indicator once movement completes. (See *'S' Command's Streaming Return Data for Current Position* note.)
Interrupt Move (^C)	**Tx**	All	1	0	3	03	0000 0111	0003	^C	<ETX>	Interrupts a move in progress that was previously initiated by any move command.
	Rx		1	0	13	0D	0000 1101		^M	<CR>	Completion indicator
Turn OFF S-Move command streaming data ('F')	**Tx**	3+	1	0	70	46	0100 0110			**F**	Command (see *'S' Command's Streaming Return Data for Current Position* note)
	Rx		1	0	13	0D	0000 1101		^M	<CR>	Completion indicator
Turn ON S-Move command streaming data ('O')	**Tx**	3+	1	0	79	4F	0100 1111			**O**	Command (see *'S' Command's Streaming Return Data for Current Position* note)
	Rx		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				
Set ROE MODE ('L')	**Tx**	All	2	0	76	4C	0100 1100	0076		L	Command
				1	0-9	0-9	0000 0000 – 0000 1001	0000 – 0009			Mode 0 - 9 (coarsest/fastest to finest/slowest)
	Rx	All	1	0	13	0D	0000 1101		^M	<CR>	Task-completion indicator

NOTES:

1. **Task-Complete Indicator:** All commands will send back to the computer the "Task-Complete Indicator" to signal the command and its associated function in controller is complete. The indicator consists of one (1) byte containing a value of 13 decimal (0D hexadecimal), and which represents an ASCII CR (Carriage Return).

2. **Intercommand Delay:** A short delay (usually around 2 ms) is recommended between commands (after sending a command sequence and before sending the next command).

3. **Clearing the I/O Send & Receive Buffers:** Clearing (purging) the transmit and receive buffers of the I/O port immediately before sending any command is recommended. Note that this clearing of the buffers affects only the computer-side I/O; it does not (necessarily) clear the buffers on the controller side, requiring, when necessary, to reset/power-cycle the controller. Following the rules described will generally avoid problems with getting garbage data in the I/O buffers of both the computer and controller (i.e., using exact number of bytes for both command sequences and return data (as per the *Commands* table), never sending a command before the previous command is finished with its task, etc.).

4. **Positions in Microsteps:** All positions sent to and received from the controller are in microsteps (μsteps). See *Microns/microsteps conversion* table) for conversion between μsteps and microns (micrometers (μm)).

   Declaring position variables in C/C++:

```
/* current position for X, Y, & Z */
unsigned long  cp_x_us, cp_y_us, cp_z_us;  /*
microsteps */
double         cp_x_um, cp_y_um, cp_z_um;  /*
microns */
/* specified (move-to) position for X, Y, & Z */
unsigned long  sp_x_us, sp_y_us, sp_z_us;  /*
microsteps */
double         sp_x_um, sp_y_um, sp_z_um;  /*
microns */
```

   Use the same convention for other position variables the application might need.

   Declaring the microsteps/microns conversion factors in C/C++:

```
/* conversion factors for MP-225/M, MP-285/M, or
MP-265/M based config. */
double us2umCF = 0.0625; /* microsteps to microns
*/
double um2usCF = 16;     /* microns to microsteps
*/
/* conversion factors for MP-245[S]/M, MP-
845[S]/M, or MP-865/M based config. */
double us2umCF = 0.046875;     /* microsteps to
microns */
double um2usCF = 21.333333333; /* microns to
microsteps */
/* conversion factors for MT-800 config. */
double us2umCF = 0.078125; /* microsteps to
microns */
double um2usCF = 12.8; /* microns to microsteps */
```

   Converting between microsteps and microns in C/C++:

```
/* converting X axis current position */
cp_x_um = cp_x_us * us2umCF; /* microsteps to
microns */
cp_x_us = cp_x_um * um2usCF; /* microns to
microsteps */
```

   Do the same for Y and Z, and for any other position sets used in the application.

5. **Ranges and Bounds:** See Ranges and Bounds table for exact minimum and maximum values for each axis of each compatible device that can be connected. All move commands must include positive values only for positions – negative positions must never be specified. All positions are absolute as measured from the physical beginning of travel of a device's axis. In application programming, it is important that positional values be checked (>= 0 and <= max.) to ensure that a negative absolute position is never sent to the controller and that end of travel is not exceeded. All computational relative positioning must always resolve to accurate absolute positions.

   Declaring minimum and maximum absolute position variables in C/C++:

```
/* minimum and maximum positions for X, Y, & Z */
double min_x_um, min_y_um, min_z_um;  /* minimum
microns */
double max_x_um, max_y_um, max_z_um;  /* maximum
microns */
```

   Set minimum and maximum absolute positions for each axis – see Ranges & Bounds table.

```
/* initialize all minimum positions in microns*/
min_x_um = 0;
min_y_um = 0;
min_z_um = 0;
/* initialize all maximum positions in microns*/
/* MP-225/M, MP-285/M, MP-845[S]/M, MP-245[S]/M,
etc. */
max_x_um = 25000;
max_y_um = 25000;
max_z_um = 25000;
/* MP-865/M */
max_x_um = 50000;
max_y_um = 12500;
max_z_um = 25000;
/* MP-265/M */
max_x_um = 25000;
max_y_um = 12500;
max_z_um = 25000;
```

6. **Absolute Positioning System Origin:** The Origin is set to a physical position of travel to define absolute position 0. The physical Origin position is fixed at beginning of travel (BOT). This means that all higher positions (towards end of travel (EOT)) are positive values; there are no lower positions and therefore no negative values are allowed.

7. **Absolute vs. Relative Positioning:** Current position ('c') and move commands always use absolute positions. All positions can be considered "relative" to the Origin (Position 0), but all are in fact absolute positions. Any position that is considered to be "relative" to the current position, whatever that might be, can be handled synthetically by external programming. However, care should be taken to ensure that all relative position calculations always result in correct positive absolute positions before initiating a move command.

   Declaring relative position variables in C/C++:

```
/* relative positions for X, Y, & Z */
double rp_x_um, rp_y_um, rp_z_um;  /* microns */
/* initialize all relative positions to 0 after
declaring them */
rp_x_um = rp_y_um = rp_z_um = 0;
```

Enter any positive or negative value for each relative position (e.g., rp_x_um = 1000; rp_y_um = 500; rp_z_um = -200 … etc.

For each axis, check to make sure that the new resultant absolute position (to which to move) is within bounds. Reset the relative position to 0 if not. If relative value is negative, its positivized value must not be greater than the current position. Otherwise, if positive, adding current position with relative position must not exceed the maximum position allowed. If out of bounds, resetting relative position to 0 allow the remaining conversions and movement to resolve without error.

```
/* check to make sure that relative X is within
bounds */
if ( ( rp_x_um < 0 && abs(rp_x_um) > cp_x_um ) ||
     (cp_x_um + rp_x_um > max_x_um) )  /* out of
bounds? */
    rp_x_um = 0; /* yes, so reset relative pos. to
0 */
```

Repeat the above bounds check for each of the remaining axes.

For each axis, calculate new absolute position in microns and then convert to microsteps before issuing a move command.

```
/* convert X relative position to absolute
position */
sp_x_um = cp_x_um + rp_x_um; /* add relative pos.
to current pos. */
/* convert new absolute X position in microns to
microsteps */
sp_x_us = sp_x_um * um2usCF;
```

Repeat for each of the remaining axes as required before issuing a move command.

8. **Position Value Typing:** All positions sent and received to and from the controller are in microsteps and consist of 32-bit integer values (four contiguous bytes). Position values in microsteps are always positive, so data type must be an "unsigned" integer that can hold 32 bits of data. Although each positional value is transmitted to, or received from, the controller as a sequence of four (4) contiguous bytes, for computer application computational and storage purposes each should be typed as an unsigned 32-bit integer ("unsigned long" in C/C++; "uint32" in MATLAB, "U32" in LabVIEW, etc.).

   Position values in microns (micrometers or μm) should be data typed as double-precision floating point variables ("double" in C/C++ and MATLAB, "DBL" in LabVIEW, etc.).

   Note that in Python, incorporating the optional NumPy package brings robust data typing like that used in C/C++ to your program, simplifying coding and adding positioning accuracy to the application.

9. **Position Value Bit Ordering:** 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 most 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. 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).

10. **Travel Lengths and Durations:** "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.

11. **Orthogonal Move Speed:** Full speed for the "Orthogonal Move 'M'" command is 5000 microns/sec. (5 mm/sec. or microns/millisecond) for single-axis movements (3000 μm/sec. (3 mm/sec. or μm/ms) for MP-225/M).

12. **Straight-Line Move Speeds:** Actual speed for the "Straight-Line Move 'S'" command can be determined with the following formula: (1300 / 16) * (sp +1), where 1300 is the maximum speed in microns/second and "sp" is the speed level 0 (slowest) through 15 (fastest). For mm/second or microns/millisecond, multiply result by 0.001.

Table E-10. Straight-line move 'S' command speeds.

Speed Setting	mm/sec or μm/ms	μm/sec or nm/ms	nm/sec	in/sec or mil/ms	% of Max.
15	1.30000	1300.00	1300000	0.051181102	100.00%
14	1.21875	1218.75	1218750	0.047982283	93.75%
13	1.13750	1137.50	1137500	0.044783465	87.50%
12	1.05625	1056.25	1056250	0.041584646	81.25%
11	0.97500	975.00	975000	0.038385827	75.00%
10	0.89375	893.75	893750	0.035187008	68.75%
9	0.81250	812.50	812500	0.031988189	62.50%
8	0.73125	731.25	731250	0.028789370	56.25%
7	0.65000	650.00	650000	0.025590551	50.00%
6	0.56875	568.75	568750	0.022391732	43.75%
5	0.48750	487.50	487500	0.019192913	37.50%
4	0.40625	406.25	406250	0.015994094	31.25%
3	0.32500	325.00	325000	0.012795276	25.00%
2	0.24375	243.75	243750	0.009596457	18.75%
1	0.16250	162.50	162500	0.006397638	12.50%
0	0.08125	81.25	81250	0.003198819	6.25%

13. **Multi Axis Movement Speed Increase:** Specified travel speeds are for single-axis movements. When travel traverses a 45° diagonal within a dual-axis square, speed is increased by 40% (x 1.4), and by 70% (x 1.7) within a triple-axis cube.

14. **Move Interruption:** A command should be sent to the controller only after the task of any previous command is complete (i.e., the task-completion terminator (CR) is returned). One exception is the "Interrupt Move" (^C) command, which can be issued while a command-initiated move is still in progress.

15. **Extracting the MPC-200 Firmware Version Number:** The firmware version number returned by the 'K' command is encoded in BCD (Binary Coded Decimal) in two bytes, with minor version byte first, followed by major version byte, each

of which contains two digits , the first of which is in the upper nibble and the next in the lower nibble. For example, if the complete version is 3.15, then the bytes at offsets 1 and 2 will show (in hexadecimal) as 0x15 0x03 (ret[1] and ret[2] as shown in the following code snippets). The following code shows how to extract and convert the 4 BCD digits into usable forms for later comparison without altering the original command return data (written in C/C++ and is easily portable to Python, Java, C#, MATLAB script, etc.).

```
/* "ret" is the array of bytes containing
the 'K' command's return data */
/* define variables */
unsigned char verbyte;  /* temp work byte */
int minver, majver, majminver;
float version;
```

Minor version number as an integer (e.g., 15):

```
verbyte = ret[1]; /* get minor ver. digits */
/* get 1's digit & then get & add 10's digit */
minver = (verbyte & 0x0F) +
   ((verbyte >>4 & 0x0F) * 10);
```

Major version number as an integer (e.g., 3):

```
verbyte = ret[2];  /* get major ver. digits */
majver = (verbyte & 0x0F) +
   ((verbyte >>4 & 0x0F) * 10);
```

Complete (thousands) version as an integer (e.g., 315):

```
majminver = majver * 100 + minver;
```

Complete version as a floating-point number (e.g., 3.15):

```
version = majminver * .01;
```

16. **'S' Command's Streaming Return Data for Current Position:** The Straight-Line Move ('S') command has two modes of operation:

    a. a CR is returned when the target position has been reached (F' (Off) command before the 'S' command sequence), or

    b. streaming positional data is returned while movement is occurring, and then a CR once movement is complete ('O' (On) command before the 'S' command sequence).

    Positional data is streamed at every 1 micron of movement, and the rate (data per second) depends on the 'S' command speed level used. Each positional data block streamed consists of 12 bytes:

    1 The 1st three bytes each contains FF hexadecimal (255 decimal) as a data block signature,

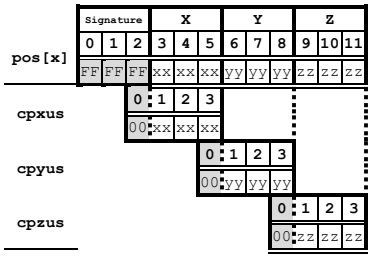
    2 the next 3 contains positional data for the X axis,

    3 the penultimate is for Y, and

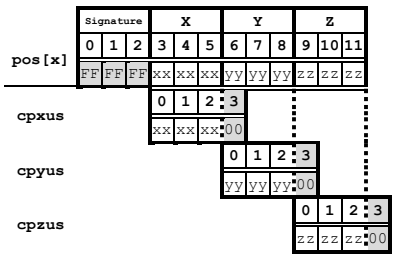
    4 last for Z.

    All positional data are in microsteps. Each 3-byte position needs to be converted into 4-byte blocks by prepending a byte containing 0, so that the resulting data (now 4 bytes) can be treated programmatically as an unsigned 32-bit "long" (C/C++) or "U32" (LabVIEW) data type. All positional data streamed is in Little-Endian bit/byte order (Wintel), so conversion to 32-bit longs will require bit-order reversal (byte swapping) for Big-Endian platforms (e.g., LabVIEW). The appropriate microstep-to-microns conversion factor is needed according to the device type being moved (see *Microns/microsteps conversion factors (multipliers)* table).

    Little-Endian bit/byte order environment:



Big-Endian bit/byte order environment ("pos" is in Little-Endian format):



The following C/C++ code snippets can be used to process the streaming data.

Array for a streamed 12-byte block of data containing current position

```
unsigned char pos[12];
```

32-bit variables for current position in microsteps, all initialized to 0 to ensure MSB allows only positive values

```
unsigned long cpxus, cpyus, cpzus;
cpxus = cpyus = cpzus = 0;
```

Copy 24-bit (3-byte) position for each axis to 32-bit (4-byte) equivalents. Use the byte position offsets shown in the diagram above. ("le" means Little Endian; "be" means Big Endian bit/byte order.)

If in Little-Endian environment (e.g., Windows, Intel-MacOSX), copy all 3 U24 bytes for each axis to the respective U32 variables.

```
memcpy(&cpxus[1], &pos[3], 3); /* X */
memcpy(&cpyus[1], &pos[6], 3); /* Y */
memcpy(&cpzus[1], &pos[9], 3); /* Z */
```

If in Big-Endian environment (e.g., legacy MacOS, LabVIEW), copy U24 to U32 byte at a time (1st to 3rd, 2nd to 2nd, & 3rd to 1st). Note that "pos" is always in Little-Endian bit/byte order.

```
memcpy(&cpxus[2], &pos[3], 1); /* X */
memcpy(&cpxus[1], &pos[4], 1);
memcpy(&cpxus[0], &pos[5], 1);
memcpy(&cpyus[2], &pos[6], 1); /* Y */
memcpy(&cpyus[1], &pos[7], 1);
memcpy(&cpyus[0], &pos[8], 1);
memcpy(&cpzus[2], &pos[9], 1); /* Z */
memcpy(&cpzus[1], &pos[10], 1);
memcpy(&cpzus[0], &pos[11], 1);
```

Ready to update UI with current position in microsteps using 32-bit integer values.

Convert microsteps to microns. Use double-precision variables for current position in microns; initialize each to 0.

```
double cpxum, cpyum, cpzum;
cpxum = cpyum = cpzum = 0;
```

Microsteps-to-microns conversion factor (see "Microns / microsteps conversion" table for appropriate factor)

```
double us2umCF = 0.0625;
```

Get microns from microsteps for each axis

```
cpxum = cpxus * us2umCF;
cpyum = cpyus * us2umCF;
cpzum = cpzus * us2umCF;
```

Ready to update UI with current position in microns using double-precision values. Loop for next data block as desired until streaming ends.

For LabVIEW, a 3-byte positional value for an axis can be transferred into a byte array, and then into a U32 data type via a byte-swap function to ensure 24-bit to 32-bit conversion while making sure that no high-order value is misinterpreted

as a sign bit (there should never be a negative positional value in the MPC-200). LabVIEW data types (e.g., U16, U32, I32) are always in Big-Endian bit/byte order, while MPC-200 multibyte values are always transcieved in Little-Endian bit/byte order.

A single completion indicator byte (ASCII CR) is returned when streaming ends and target position has been reached.

NOTES:

INDEX

NOTES

SUTTER INSTRUMENT®

Addendum

To

All Operation Manuals of MPC-200/ROE-200-Based MPC-Series Systems

Rev. 1.00 – November 28, 2007

As of Version 3.11 (November 12, 2007) of the firmware for the MPC-200 micromanipulator controller and ROE-200 input device, the CENTER routine associated with the white button on the rear of the ROE-200 has been replaced with a CALIBRATE routine that is less likely to break a pipette. Thus, it can be used in the middle of an experiment when you see the message EOT (end of travel) displayed on the ROE-200. Please note that all references to CENTER in the current manual should be replaced with CALIBRATE. Furthermore, the detailed instructions regarding the centering routine in section 2.3.3 should be replaced with the CALIBRATE instructions below.

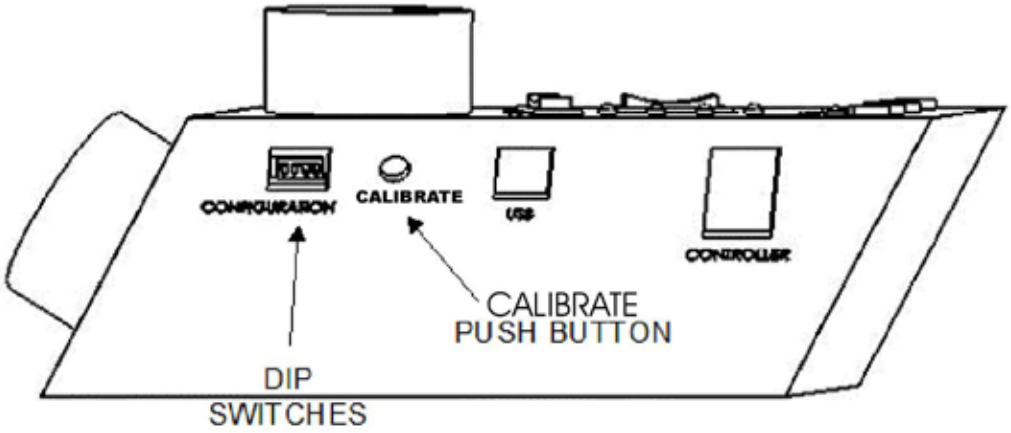


Figure 1. Location of the CALIBRATE button on the ROE-200.

CALIBRATE is used in two ways. When the unit is first set up, CALIBRATE is used to establish the zero location. Then, occasionally, during normal operation, CALIBRATE is used to reestablish the zero location. CALIBRATE follows a more conservative path than CENTER and can generally be used in the presence of a pipette.

To CALIBRATE, press and release the white button on the back of the ROE-200. The manipulator will back away from the current location along the established diagonal (like a HOME move), and ultimately move to the end of travel (EOT) sensors, beyond the origin (0,0,0). Once the sensors are found, a short move in the opposite direction is made and this location is defined as (0,0,0). The purpose of CALIBRATE is to allow 0,0,0 or HOME to be safely reestablished during an experiment without risking damage to the pipette.

If the unit is turned off or STOP/SET is pressed during the running of CALIBRATE, the unit will not be correctly initialized. In this case, it is necessary to cycle the power off and on, and then run CALIBRATE again to its completion.