MPC-200 USB commands, version 1.10

General Information

The controller is ready to accept remote input about one second after power-on. Command requests are single bytes. The data stream consists of full bytes (all 8 bits — not ASCII). The lowest order byte (for example, of the four bytes encoding the X coordinate) is the first into the controller and is the first out. There are no delimiters within command strings. The controller will reply with carriage return (CR, 0Dh) at the completion of normal command processing. Wait for the CR before sending a new stream. Full travel on each axis is 25000 µm, so 0 to 25000 µm are valid values for the move command (note: position info is sent in microsteps, see Data Format).

The MPC200 firmware is field updatable. If your firmware is newer than the version listed above, check with Sutter to see if new USB commands have been added.

Remote (USB) Commands

Change drive (manipulator)

command	'I'D (D is 01-04h)
returns	value of D if manipulator connected or E if not. Followed by CR

Get current position

command	'C'
returns	DxxxxyyyyzzzzCR (D designating the currently active drive + three signed long integers + 0Dh)

Get number N of connected manipulators and drive status n for all 4 drives

command	'U'
returns	NnnnnCR n=1 if connected, n=0 if not connected

Get currently active drive

command	'K'
returns	D Vl Vh then CR (D designates the currently active drive, Vl and Vh designate firmware version)

There are two generic move commands. Before using either, the manipulator to be moved must be selected either manually or via the USB port using the I command. Only the active manipulator can make USB moves.

The first move command, S, is for straight-line moves and should be used when you need to move point-to-point at a fixed, user-defined, velocity. The second move command, M, is for rapid moves that accelerate to the fastest velocity following a stereotypic, multiple-axis path. The path taken in M moves is similar to that in the Home and Work Pos moves evoked from the ROE.

Straight-line movement with user defined velocity

command	'S'X xxxxyyyyzzzz 053h + three signed long integers
returns	streams position info until the move is finished

Fast, stereotypic movement with firmware controlled velocity

command	‘M’xxxxyyyyzzzz	04Dh + three signed long integers
returns	CR	0Dh

Moves related to manual functions of the ROE200

Home

command	‘H’	048h
returns	CR	0Dh

Work Position

command	‘Y’	059h
returns	CR	0Dh

Center

command	‘N’	048h
returns	CR	0Dh

The final command is an interrupt to halt any USB move while in progress. This and the Stop button are the only means to interrupt a move before it is finished.

Interrupt Move

command	control-C	03h
returns	CR	0Dh

Specifics on the various commands

1) Straight-line movement: ‘S’+X+coordinates.

Structure of the command is S followed immediately by X followed immediately by “coordinates” (the desired final location for the manipulator making the move). X passes the velocity of the move as follows:

The value of X may range from 0x00 to 0x0F. 0x0F is the fastest speed for S moves and is about 1.3 mm/s. Assuming more than one axis is involved in the move, X is the speed for the axis making with the longest travel in the three axis move. We require all axes to finish the move at the same time, thus the speed of movement of the other axes will be slower in order to force the moving axes to finish simultaneously.

A value for X of 0x0F encodes the fastest speed along the longest axis and is about 1.3mm/sec. A value of 0x00 for the low nibble is the slowest speed and is 1/16th of the fastest speed. This means if X is 0x08, the speed of S movement will only be half of the speed when X is set at 0x09.

“Coordinates” is the final position of the manipulator. For each axis, you must specify a long integer (4 characters). Therefore “coordinates” will consist of 3 long integers. It is up to the programmer to assure that the “coordinates” is the correct length, if it is not, the controller and/or the USB interface will hang.

Other constraints on coordinates: Values cannot exceed the travel of each axis (0-25000 microns. “Coordinates” is actually passed in microsteps. See Data Format, below).

After receiving the S command, the manipulator starts from its current location and moves to the specified final location along a straight line with the speed set in X. The actual speed is determined by the axis with the longest move. The difference in time to start moving between any two axes is less than 0.06 ms

While the move is in progress, the current coordinates of the moving manipulator are sent back to the PC using the following format: Three 0xFF followed by the coordinates of the manipulator making the move. The formatted string of coordinate information repeats itself until movement finishes.

Note also, to make the fastest moves, you should use the M command, however, this move is not straight line. See details under M.

2) Detail on speed of communication:

The baud rate of communication over the USB port 128,000. This has to be set in the PC program.

3) Detail on the ‘U’ command

The U command, when issued, returns the number of currently connected manipulators. If no manipulators are connected, the ROE will display “No manipulator connected” and no response will be received over the USB. If manipulators are connected, the controller returns an initial character that gives the number of connected manipulators and then four characters that give the status of each drive. The value of the four characters is 1, if a manipulator is connected and 0 if a manipulator is not connected. After number and status drive info is sent, the controller ends with CR (0x0D).

4) Detail on the ‘K’ command

The K command returns three characters. The first character, D tells which drive is active (the drive currently connected to the knobs on the ROE). The drives are numbered in order, as DriveA, DriveB, Drive2A, Drive2B or Manipulators 1,2,3,4 on the top of the ROE. The second and third returned characters, Vl and Vh, give the version of the controller firmware. For example, if the firmware is version 1.10, Vh will be 1 and Vl will be 10. After D and V are passed the controller terminates with CR (0x0D).

5) Detail on the ‘C’ command (Note, version 1.03 had a simpler C command)

When position is read, the first character returned represents the currently active drive (the one the ROE knobs are connected to). For example, if the user manually switched the drive from drive A to drive2A, the next ‘C’ command would return ‘3’ followed by the coordinates of drive2A. After the coordinate information is passed, the controller sends a CR (0x0D). It is not necessary to either keep track of the current manually active manipulator or to switch to the active manipulator to poll position via USB.

Thus, one can keep track of position info of all manual movement by constantly polling with C. Even if the user is making a “continuous” manual move, position information to the USB C command will be “instantaneously” updated. If the user manually changes to a different manipulator (drive), the first character of the C command will follow that change and C will begin giving coordinates of the new drive. As only one drive can move at a time, the info passed over the USB port and read with the C command in a tight loop will completely document the positions of all manipulators.

Finally, note that when issuing M or S type move commands, manual movement and further communication over the USB port is locked out (with the exception of “Interrupt Move” or manual press of Stop on the ROE). Once the M or S move is complete, the C interrogation can continue. Thus, complete documentation of all manipulator positions is maintained.

6) Detail on the 'I' command (Note, version 1.03 had a different I command)

The I command switches USB AND manual control to the specified drive and echoes back information about the drives’ status. I returns the manipulator drive number if a manipulator is connected and returns 'E' if nothing is connected to this drive. If E is returned, the controller does not switch drives. Either return is then terminated by a CR (0x0D).

7) Detail on the move command “M” (Note, version 1.03 had no other move command)

While the S, straight-line move, is the most generic move that can be made over the USB port, the original move command, M, is available for fast, long distance moves. We assume this might be used for fast pipette changes or other moves where speed is paramount and path is not important. The move accelerates, so speed of the movement is not constant and is a function of the length of the move. M may be also very useful for fast single axes moves.

8) Detail stopped moves. It is possible for the user to interrupt moves from the ROE200 by pressing the Stop button. When this is done during a S, M, H, Y or C move, the controller will send an I + CR to the USB port to indicate to the host that a manual interrupt has occurred.

Data format

Position information is displayed on the ROE in microns relative to the absolute zero set using the Center routine. However, postion is not communicated in microns either in the microprocessor or over the USB interface. Rather, it is represented in stepper motor microsteps. The resolution of movement used over the USB port is the minimal microstep size of 0.0625 micrometers (2 micron whole steps divided by 32 microsteps per whole step).   As most PC GUIs will take information from the user in microns, the USB interface will need to incorporate a conversion factor of 16 to convert microns into microsteps for the MPC200 controller.

For example, if you want to move one of the axes of the MPC200 to the location 100 microns. The actual position you will send is 1600 microsteps (100 times 16). You also need to convert this number into the characters (hexdecimal) which represent this number (not the characters of "1", "6", "0", "0").

One way to understand the data format is to manually move the controller to some integer number e.g. x=100, y=200, z=300, and then use the C command to get the current position and make sense out of the numbers you get back from the controller. In this example, C should return 0x40, 0x06, 0x00,0x00 for X. The other axes follow in a similar manner. You would then construct coordinates in the same number format when you want to send a move to the controller using M or S.