COMPUTER INTERFACE/SUPPLIED SOFTWARE

General Information

The controller is ready to accept remote serial input about one second after power-on. Command requests are single bytes followed by optional parameters and terminated by a carriage return (CR, 0Dh). 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. The default Baud rate is 9600. Commands are processed bytewise by interrupt and executed only after the terminating CR is received. 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. The serial port is a minimal 3-wire interface and the CR reply is the only form of handshaking. Errors are reported as ASCII numerals to the host and on the controller display at the upper right-hand corner. These are:

0	SP Over-run	The previous character was not unloaded before the latest was received
1	Frame Error	A valid stop bit was not received during the appropriate time period
2	Buffer Over-run	The input buffer is filled and CR has not been received
4	Bad Command	Input cannot be interpreted – Command byte not valid
8	Move interrupted	A requested move was interrupted by input on the serial port. This code is: ORed with any other error code. The value normally returned is “=”, i.e., 8 ORed with 4. "4” is reported on the vacuum fluorescent display

Remote Commands

Get Current Position

command	‘c’CR	063h + 0Dh
returns	xxxxyyyyzzzzCR	three signed long integers + 0Dh

Go To Position

command	‘m’xxxxyyyyzzzzCR	06Dh + three signed long integers + 0Dh
returns	CR	0Dh

Change Velocity

command	'V'xxCR	056h + one 16-bit unsigned integer + 0Dh (Note: Capital ‘V’) Note: High-order bit is resolution; 0=10, 1=50 uSteps/step
returns	CR	0Dh

Set Origin

command	'o'CR	06Fh + 0Dh

returns	CR	0Dh

Download Program

command	‘d’nncc … CR	064h + integer + integer + vectors +0Dh
		Note: integers are program number and the number of vectors
returns	CR	0Dh

Execute Stored Program

command	‘k’nnCR	06Bh + integer (program no.) + 0Dh
returns	CR	0Dh

Upload Program

command	‘u’nnCR	075h + integer (program no.) + 0Dh
returns	CR	0Dh

Absolute mode

command	'a'CR	061h + 0Dh
returns	CR	0Dh (Note: No VFD update)

Relative mode

command	'b'CR	062h + 0Dh
returns	CR	0Dh (Note: No VFD update)

Continue After Pause

command	'e'CR	065h + 0Dh
returns	CR	0Dh

Refresh VFD Display

command	'n'CR	06Eh + 0Dh
returns	CR	0Dh (Note: refreshes XYZ display only)

Interrupt Move

command	control-C	03h
returns	CR	0Dh

Reset Controller

command	'r'CR	072h + 0Dh
returns	nothing	---

Status

command	's'CR	073h + 0Dh
returns	(32 bytes)CR	The following are contained in 32 bytes (8 long integers):

FLAGS	byte	Program flags
UDIRX	byte	User-defined values for
UDIRY	byte	motor axis directions
UDIRZ	byte	Valid values 0 .. 5
ROE_VARI	word	µsteps per ROE click
UOFFSET	word	User-defined period start value
URANGE	word	User-defined period range
PULSE	word	Number of u-steps per pulse
USPEED	word	Adjusted pulse speed µsteps/s
INDEVICE	byte	Input device type
FLAGS_2	byte	Program flags
JUMPSPD	word	“Jump to max at” speed

HIGHSPD	word	“Jumped to” speed
DEAD	word	Dead zone, not saved
WATCH_DOG	word	programmer’s function
STEP_DIV	word	Divisor yields μsteps/ μm
STEP_MUL	word	Multiplier yields μsteps/nm
XSPEED	word	Remote speed (μm/s) & res. (bit 15)
VERSION	word	Firmware version

FLAGS

0..3	SETUP #	Currently loaded setup number. Values 0..9 correspond to displayed 1..0		
4	ROE_DIR	ROE direction	Set = last dir=neg	Clear = last dir=pos
5	REL_ABS_F	Display origin	Set = absolute origin	Clear = relative origin
6	MODE_F	Manual mode flag	Set = continuous mode	Clear = pulse mode
7	STORE_F	Setup condition	Set = setup is stored	Clear = setup is erased

FLAGS_2

0	LOOP_MODE	Program loops	Set = do Loops	Clear = execute program once
1	LEARN_MODE	Learn mode status	Set = learning now	Clear = not Learning
2	STEP_MODE	Movement resolution	Set = 50 usteps/step	Clear = 10 usteps/step
3	SW2_MODE	Enable joy side button	Set = joy side button enable	Clear = disable side button/SW2
4	SW1_MODE	Enable FSR/Joystick	Set = FSR/joystick sw enabled	Clear = keypad changes Cont/Pulse
5	SW3_MODE	Enable ROE switch	Set = ROE switch enabled	Clear = disabled
6	SW4_MODE	Enable SW 4&5	Set = switches 4&5 enabled	Clear = disabled
7	REVERSE_IT	Reverse program	Set = reverse program	Clear = normal program sequence

Setting up for Serial Communication

First, use the 9-pin serial port cable provided with the MP-285 to connect the “serial port” of your computer to that of the MP-285 controller. Next configure your terminal emulator (e.g., Hyperterminal in Windows 95) to the following settings (or their equivalent):

- TTY mode
- echo typed characters locally only (do not echo input to the computer serial port back to the controller)
- baud rate to 9600
- 8 data bits, no parity, 1 stop bit

A simple test can now be made to confirm that the RS232 cable is properly connected and the computer terminal emulator is properly configured. The “o” command can serve this purpose but it is important to know that THIS COMMAND RESETS THE ABSOLUTE ORIGIN OF THE MP-285 CONTROLLER! It is possible to minimize the relative effect of this command by moving the manipulator to a very short distance from the ABSOLUTE ORIGIN (e.g., X=0.04 μm, Y=0.04 μm, Z=0.04 μm) before issuing the “o” command from the remote computer. Now, type “o”

(followed by a RETURN). The MP-285 CONTROLLER display should reset to X=0.00 μm, Y=0.00 μm, Z=0.00 μm. There may also be a change in the overall appearance of the display. The original display configuration can be restored by entering an "n" (followed by a RETURN) from the computer keyboard or by pressing the reset button on the front panel of the MP-285 CONTROLLER.

You may also want to try the serial interface test "SIO test" that is built in to the MP-285 controller. For directions on how to use this function see the subheading "SIO test [PRGM\Setup\Utilities\SIO test]" in the Controller Configuration section of the manual.

If these tests of the serial port connection fail, try another COM port assignment and/or recheck your computer's configuration to make certain that they conform to the above specifications.

Now that you have the cable connected and port configured, you should next try one of the following:

- Run one of the simple program fragments on the following pages.
- Download the self-extracting sample programs (“MP285.exe”) from the Sutter Instrument FTP site “FTP.Sutter.com/pub”. The “readme” document found within MP285.exe briefly describes those QuickBASIC and PowerBASIC programs and their usage.
- Try the Visual Basic MP-285 interface program supplied with your manipulator.

These programs and code fragments are meant to be used only as examples of how the data stream can be handled to and from the controller and are not mature software packages. The MP285COM.bas program found within MP285.exe may be useful, however, as a starting point for developing a more advanced PowerBASIC interface or software in MODULA-2 and C programming languages, as well.

Further notes:

- The approach used for handling the data stream in the PowerBasic programs uses UNIONs. The UNION allows the bytes in the data stream to be simultaneously “TYPED” as LONG INTEGERs (for display of and operations on the data from the MP-285 controller) or as STRINGs of BYTES (for sending coordinates back to the MP-285 controller).
- An executable version of the MP285COM.bas program is also included in the package and is named MP285COM.exe.
- Sutter Instrument Company welcomes your feedback about any successes or problems that you may experience in developing serial communication software for the MP-285. With your permission, we will put your comments and suggestions on our FTP site for the benefit of other scientists who might gain from your efforts.

```
'QuickBASIC "getit" ROUTINE

'This program is written in QuickBasic 4.5.  It will get the position
'coordinates from the MP-285, (in usteps), convert them to um's and
'then print them to the screen.

CLS

'First, open the appropriate COM port (without a handshaking protocol)
OPEN "COM2:9600,N,8,1,CD0,CS0,DS0" FOR RANDOM AS #1

'Next send the "get position" command and a CR to the MP-285
PRINT #1, "c";
PRINT #1, CHR$(13);

'Finally, for each axis convert 4 bytes from the INPUT BUFFER into
'a LONG INTEGER using the CVL function, multiply that value by .04 to
'convert it from usteps to um's and then PRINT it to the screen.  All
'three of these steps are contained in one line of code below.
PRINT "MP-285 COORDINATES"
PRINT USING "X = +#####.## um "; .04 * (CVL(INPUT$(4, 1)))
PRINT USING "Y = +#####.## um "; .04 * (CVL(INPUT$(4, 1)))
PRINT USING "Z = +#####.## um "; .04 * (CVL(INPUT$(4, 1)))
'This line dumps the last byte (a CR) from the buffer
endchar$ = INPUT$(1, 1)

CLOSE #1
END
```

```
            'QuickBASIC "goto" ROUTINE

'This program is written in QuickBASIC 4.5.  It will prompt you to
'enter the coordinates (in um's) to which you want the manipulator to
'move, convert the values to usteps and then send them to the MP-285

CLS

'First, open the appropriate COM port (without a handshaking protocol)
OPEN "COM2:9600,N,8,1,CD0,CS0,DS0" FOR RANDOM AS #1

'The next step will prompt you to input the coordinates in um's
INPUT "X = ", X
INPUT "Y = ", Y
INPUT "Z = ", Z

'This step will send the MOVE command to the MP-285 followed by a 13 byte
'data stream.  The first 12 bytes in the data stream are three 4-byte strings
'representing the LONG INTEGERS entered above.  Those values are reduced here
'to bytes by the MKL$ function after they have been converted from um's to
'usteps (* 25).  The last byte is a CR.
PRINT #1, "m";
PRINT #1, MKL$(X * 25);
PRINT #1, MKL$(Y * 25);
PRINT #1, MKL$(Z * 25);
PRINT #1, CHR$(13);
CLOSE #1
END
```

```
'PowerBASIC "getit" ROUTINE (Pbgetit.bas)
'This program is written in PowerBASIC 3.0.  It will get the position
'coordinates from the MP-285 (in usteps), convert them to um's and
'then print them to the screen.

CLS
'Open the appropriate COM port (without a handshaking protocol).
OPEN "COM2:9600,N,8,1,CD0,CS0,DS0" FOR RANDOM AS #1

UNION fourbytes                 'creates a data "TYPE" to allow the
  longfield AS LONG             '4 bytes describing an axis' position
  fourstring AS STRING * 4      'to be arrayed as a "LONG" integer and
END UNION                       'as a "Fixed-length" string called
                                'longfield and fourstring, respectively

DIM XVAL AS fourbytes           'dimensions the arrays to
DIM YVAL AS fourbytes           'which the byte values of the
DIM ZVAL AS fourbytes           'coordinates will be written and
                                'declares their "TYPE" (a UNION
                                'called fourbytes in this instance)

PRINT #1, "c"                   'requests the coordinates
PRINT #1, CHR$(13)

XVAL.fourstring = INPUT$(4,#1)  'reads the coordinates (4 bytes
YVAL.fourstring = INPUT$(4,#1)  'for each) into the appropriate
ZVAL.fourstring = INPUT$(4,#1)  'array

PRINT "MP-285 COORDINATES"
PRINT USING "X = +#####.## um ";XVAL.longfield * .04 'converts the ustep
PRINT USING "Y = +#####.## um ";YVAL.longfield * .04 'values to um's and
PRINT USING "Z = +#####.## um ";ZVAL.longfield * .04 'prints to screen

endchar$ =  INPUT$(1,#1)         'dumps the last byte (a CR) from the buffer
CLOSE #1
END
```

```
'PowerBASIC "goto" ROUTINE (Pbgoto.bas)
'This program is written in PowerBASIC 3.0.  It will prompt you to enter
'the coordinates (in um's) to which you want the manipulator to move,
'convert those values to usteps and then send them to the MP-285.

CLS
'Open the appropriate COM port (without a handshaking protocol).
OPEN "COM2:9600,N,8,1,CD0,CS0,DS0" FOR RANDOM AS #1

UNION fourbytes                 'a data "TYPE" created to allow the
  longfield AS LONG             '4 bytes describing an axis' position
  fourstring AS STRING * 4      'to be arrayed as a "LONG" integer and
END UNION                       'as a "Fixed-length" string called
                                'longfield and fourstring, respectively

DIM XVAL AS fourbytes           'dimensions the arrays to
DIM YVAL AS fourbytes           'which the coordinates in the
DIM ZVAL AS fourbytes           'buffer will be written and
                                'declares their "TYPE" (a UNION
                                'called fourbytes in this instance)

INPUT "X = ",X                  'prompts you to input the coordinates
INPUT "Y = ",Y                  '(in um's) of the position to which
INPUT "Z = ",Z                  'you want the manipulator to move

XVAL.longfield = X * 25         'converts the values entered from
YVAL.longfield = Y * 25         'um's to usteps and enters them into
ZVAL.longfield = Z * 25         'LONG INTEGER array named longfield

PRINT #1, "m";                  'sends the MOVE command to the MP-285
PRINT #1, XVAL.fourstring;      'followed by the 3 strings of bytes
PRINT #1, YVAL.fourstring;      '(4 each) that represent the LONG INTEGERS
PRINT #1, ZVAL.fourstring;      'put into the longfield array in the last
PRINT #1, CHR$(13);             'step (a CR ends the data stream)

CLOSE #1
END
```

The Visual Basic MP-285 Interface: Version 1.00

Preface

Sutter Instrument Company has created a visual basic program as an object-oriented interface to control the MP-285 micromanipulator via a host computer. Use this software as a template for creating your own programs. This brief introduction is intended for users who are already familiar with using the MP-285. If you are not already familiar with the MP-285 you should do this first.

The software is designed for executing micromanipulator movements and controlling various functions of MP-285 via a serial interface.

Installing the Visual Basic PC Controller

1.

1 **Windows 95 should be installed and running on a PC, DX486 or faster.**

2 **Insert “Disk 1" into A: drive.**

3 **When prompted, insert “Disk 2" into A: drive.**

1. **Follow setup directions as they appear.**

2. **Once setup has competed, make sure your MP-285 is connected properly to the serial port of your computer.**

4 **Turn on your MP-285 *first*, then go to the MP-295 folder inside the Program Files folder and double click on the MP-285 icon.**

5 **The MP-285 work space should appear:**

PICTURE REMOVED TO MAKE FILE SMALLER

A Brief Tour Of The User Interface

On the right hand side, the Absolute **box with three panels shows you where your manipulator is, each panel corresponding to one of the three axes. Each one indicates the position, in μm, relative to the absolute position read from the controller when the software began communication. In each box you can manually specify the next position you want to move to along each axes. The move is then performed when you click on the “move” button below.**

The Step **box below allows you to move the manipulator in specified increments along each axis. The panel at the base of the box allows you to specify how many μm the manipulator moves each time you click on one of the buttons in the box.**

The five buttons on the left end of the top bar send the manipulator to the Home **position,** Position 1**,** Position 2**,** Position 3**, and** Position 4 **respectively. Directly underneath are corresponding buttons that let you** Define **where 'Home', 'Position 1', 'Position 2', etc. are in the field of possible positions. (**Position 4 **is always the "start up" position.)**

The Stop **button to the right becomes red only when it is active, i.e. when manipulator is in the process of moving to another position. This button allows you to stop the movement of the manipulator.**

The button to the far right allows you to Update Display Coordinates **after the manipulator has received input commands from another device such as the Rotary Optical Encoder (ROE). Though input from other devices is processed and executed, the display on the MP-285 controller is updated with the new position while the PC interface is not. Pressing this button updates the coordinates on the PC interface.**

The two boxes at the bottom left of the window display the resolution (Res**) and** Velocity **at which the pipette tip is set to move.**

Within the blue box **in the main workspace, the mouse cursor becomes a black cross. With a click, you can move your manipulator tip relative to the current position, represented by the smaller cross. While the large block represents the X and Y dimensions the** narrow blue strip **along the right represents the Z-axis.**

Along the bottom, three boxes show the Cursor Position **along the axes so that the user can accurately and quickly move the pipette tip to a new position.**

The Robotics **menu at the top of the screen lets you create, edit, and execute preset programs stored in the controller or on the PC (**Robotics/Host**), and transfer programs back and forth from the PC to the controller for editing and execution. (**Robotics/Controller**).**

The Setup **menu lets you fine tune your manipulator so that you can select the specifications that are appropriate for your experiments. Here you can choose a specific** Velocity**, and High or Low** Resolution**. Under** Communications**, you can specify Serial Port and Baud Rate.** Hardware **lets you choose a new origin, and reset the Controller, but please refer to the main MP-285 manual before doing so.**

For tech support and information about the Visual Basic Interface, choose About **from the menu bar.**