

# **Software Manual**

## **Web Version 1.0**

Ho Group  
STM Software  
User's Manual

Ho Group  
Dept. of Physics  
Clark Hall  
Cornell University

## **Purpose**

This document is a reference manual for users of the Ho group's STM software. This is a large, complex program. This document is intended to serve as a guide for the various dialog boxes, buttons, and other features of the software. It should answer the question: "What does this button do?" This document is not intended to detail how the program may be used to effectively perform experiments; such knowledge comes with experience.

This software runs under Windows 3.11. The reader is assumed to have a basic knowledge of how that operating system works (buttons, windows, dialogs, etc.).

## **Revision History**

This software is constantly being revised. It is constantly evolving with the needs of the research group. Thus, it is important to note which version of the software is reflected in this document.

This document reflects the software as of: 06/18/98

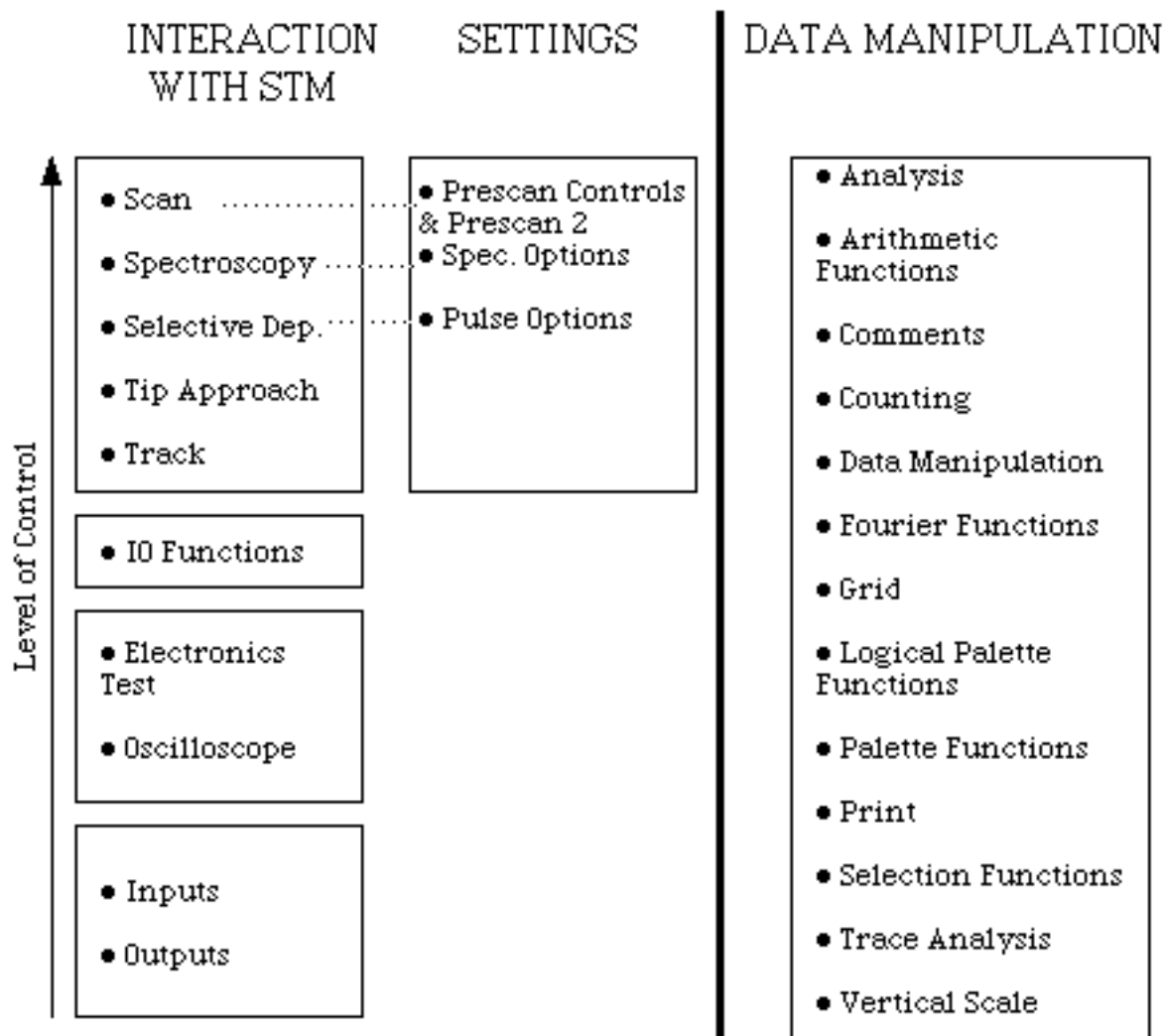
## **Overview and Organization**

The Ho group's STM software is complex. The user interface is immense: there are nearly thirty dialogs with somewhere in the neighborhood of 500 different buttons, radio buttons, scroll bars and text fields. In principle, each dialog has a distinct function. This manual describes each dialog separately.

The dialogs are covered in alphabetical order. A flow chart in the appendix shows the relationship between the dialogs and the page numbers where they are described. Consult this chart to find the location of the dialog which interests you. An effective way to explore a particular dialog would be to run the program with this manual open to the page corresponding to that dialog.

If you are already familiar with the program, you might refer to this manual occasionally to clarify a certain aspect of the program or to refresh your memory about how the program works.

The discussions of the individual dialogs cover many details of the software, but do not offer much insight into the overall organization. One way to view the overall structure of this software is in terms of two kinds of general tasks the software is meant to accomplish: interacting with the STM and Data Manipulation. The figure on the following page categorizes the major features of the program based on which of these two tasks they accomplish.



Dialogs whose primary function is interaction with the STM are shown on the left side of this figure. These dialogs offer various levels of control over the STM. At the highest level of control, you can instruct the STM to complete sophisticated tasks, such as scanning and tip approach. A number of the high-level-control dialogs are so complex that they require separate dialogs for setting various parameters. At the lowest level of control, you can send and receive patterns of bits from the computer with the INPUTS and OUTPUTS dialogs. The lowest levels of control puts the burden on you, the user, to know exactly what you are doing with the STM. That is not to say that you don't need to know what you're doing while using the high-level-control dialogs, but there are safeguards built into the high-level dialogs. There are no safeguards built into the program on the lower levels of control. In the past, the lowest levels of control – ELECTRONICS TEST, OSCILLOSCOPE, INPUTS, and OUTPUTS dialogs – have been used to test the STM electronics rather than to perform experiments.

Dialogs whose primary function is to manipulate data for analysis or presentation are shown on the right. All of these dialogs are accessible via the DATA MANIPULATION dialog. Since they don't involve interaction with the STM, there are fewer dangers in using these features of the program.

The SEQUENCE SUMMARY and DATA INFO dialogs are not shown here. Their primary function is to present you with summary information.

## General Information

There are several recommended practices that apply to the entire program. In general, if a field can be changed by either typing in a text field or by adjusting a scroll bar, it is best to adjust the value with the scroll bar. For instance, say you wish to change the sample bias from 0.100 to 0.150. If you change the value by typing, the program changes the sample bias to 0 as soon as you type the "0" and proceeds to change the bias each time a you type a character. This is clearly undesirable and can be avoided by using a scrollbar.

Also, the conversion of distances to angstroms is calibrated to the piezotubes' response at room temperature and is not valid at other temperatures.

The software expects the following initialization files to be present in the directory c:\stm\new :

dep0.ini  
dep1.ini  
dep2.ini  
dep3.ini  
dosed.ini  
dtransl.ini  
dtranss.ini  
initdir.ini  
masspec.ini  
samples.ini  
scan0.ini  
scan1.ini  
scan2.ini  
scan3.ini  
sp0.ini  
sp1.ini  
sp2.ini  
sp3.ini  
stm.ini  
stmpr.ps

Also, the program reads and saves several different kinds of files, corresponding to the various kinds of data it can read and manipulate:

3D, topographic data (\*.stm)  
scan settings (\*.stp)  
2D, spectroscopic data (\*.spc)  
2D, deposition data (\*.dep)  
3D, grid data (\*.grd)  
3D, counting data (\*.cnt)  
2D, data cut from 3D data (\*.cut)  
2D, data binned by plateau duration (\*.b1,\*.b2,\*.bs1,\*.bs2)  
2D, data plateau voltages (\*.t1, \*.t2)  
print layouts (\*.fig)

## **Suggestions to Avoid Crashing the Tip**

One of the most troublesome things that can happen while using the software is inadvertently crashing the tip. Here are some suggestions that will help you avoid crashing the tip:

- During TIP APPROACH, avoid having a low sample bias and a high minimum tunneling current.
- If you wish to adjust the gains for the feedback electronics, use the radio buttons labeled Z GAIN and FREQUENCY in the PRE-SCAN CONTROLS dialog rather than the radio buttons labeled Z and Z2 in the IO FUNCTIONS dialog.
- Avoid executing an AUTO- or AUTO+ (or making any change to z offset) with the feedback off.
- To flip the sample bias, use the FLIP button in the PRE-SCAN CONTROLS dialog. Do *not* scroll through zero, especially on semiconductor surfaces.

| Analysis               |                      |                      |                      |   |
|------------------------|----------------------|----------------------|----------------------|---|
| Delta (2-1) First Sel. |                      |                      |                      |   |
| X                      | Y                    | Z                    | Avg. Z               |   |
| <input type="text"/>   | <input type="text"/> | <input type="text"/> | <input type="text"/> | V |
| <input type="text"/>   | <input type="text"/> | <input type="text"/> | <input type="text"/> | Å |
| <input type="text"/>   | <input type="text"/> | <input type="text"/> | <input type="text"/> |   |

| Delta (2-1) Sec. Sel. |                      |                      |                      |                                     |
|-----------------------|----------------------|----------------------|----------------------|-------------------------------------|
| X                     | Y                    | Z                    | Avg. Z               |                                     |
| <input type="text"/>  | <input type="text"/> | <input type="text"/> | <input type="text"/> | V                                   |
| <input type="text"/>  | <input type="text"/> | <input type="text"/> | <input type="text"/> | Å                                   |
| <input type="text"/>  | <input type="text"/> | <input type="text"/> | <input type="text"/> | <input type="button" value="Exit"/> |

Use the ANALYSIS dialog to perform two simple calculations on data that has been selected with the SELECTION FUNCTIONS dialog. These two simple calculations are: 1) subtracting one set of selected coordinates from another 2) calculating the average value of z within a selected region.

This dialog consists of two segments. The upper segment refers to Selection 1 made with the SELECTION FUNCTIONS dialog. The lower segment refers to Selection 2 made with the SELECTION FUNCTIONS dialog.

TEXT FIELDS:

The text fields form a table with four columns and three rows. The first three columns are labeled x, y, and z. The fourth column is “labeled” by the AVG Z button.

The program displays the difference between two sets of selected coordinates in the SELECTION FUNCTIONS dialog in the first row in units of pixels. For example if Selection 1 in the SELECTION FUNCTIONS dialog is a line with endpoints (x1, y1, z1) and (x2, y2, z2) the first row of the upper segment will show (x1-x2), (y1-y2) and (z1-z2). In the second row, the program displays the same information as the first row, but in units of volts. The final row, labeled with the angstrom symbol, does not have a function at the present time.

One way to better understand the ANALYSIS dialog is to alter selections with the SELECTION FUNCTIONS dialog while the ANALYSIS dialog is open.

AVG Z BUTTON:

Use this button to calculate the average value of z within a square region of 3D data. The program will place the result in the fourth column of editable text fields. The program displays the value in pixels and in volts. If the selection is not a square region, the result of the AVG Z calculation is not meaningful.

Use the ARITHMETIC OPERATIONS dialog to perform simple arithmetic and smoothing operations on scanned data. This dialog works in conjunction with the DATA MANIPULATION dialog. Note that all of the arithmetic operations can be constrained to a selected region by using the CONSTRAIN TRANSFORMS check box in the SELECTION FUNCTIONS dialog.

#### EXECUTING THE OPERATION:

1,2,3,4 BUTTONS: Use the 1,2,3 buttons to execute the current operation. The 4 button does nothing.

#### SPECIFYING THE OPERATION:

Use the radio buttons on the left side of the dialog to specify the current operation.

**SUBTRACT:** Use this radio button to select the subtraction operation. The subtraction operation is  $(A - B)$  where A is the currently selected data in the DATA MANIPULATION dialog and B is the data corresponding to the 1,2,3 button you press. The program displays the result in the lower right quadrant of the DATA MANIPULATION dialog (quadrant 4). This operation does nothing if  $A = B$ . This operation does nothing if A and B are different kinds of data (one is 3D, one is 2D).

**AVERAGE:** Use this radio button to select the average operation. The average operation is  $(A + B)/2$  where A is the currently selected data in the DATA MANIPULATION dialog and B is the data corresponding to the 1,2,3 button that is pressed. The program displays the result in the lower right quadrant of the DATA MANIPULATION dialog (quadrant 4). This operation does nothing if A and B are different kinds of data (one is 3D, one is 2D). If  $A=B$  and the data is 2D, the program smooths the data by averaging sets of neighboring points into one point (the overall number of data points will be reduced). If  $A=B$  and the data is 3D, this operation does nothing.

**DIVIDE:** The divide operation has not been implemented.

**SMOOTH:** Use this radio button to select the smooth operation. The smooth operation is for 2D data only. The program smooths the data specified by the 1,2,3 button by performing a running average (the overall number of data points remains the same). Note that the 1,2,3 button must be the same as the currently selected data. The program displays the result in the lower right quadrant of the DATA MANIPULATION dialog (quadrant 4).

**$(dI/dV)/(I/V)$ :** Use this radio button to select the  $(dI/dV)/(I/V)$  operation. The  $(dI/dV)/(I/V)$  operation is for 2D data only.  $dI/dV$  is the currently selected data in the DATA MANIPULATION dialog;  $I/V$  is the data corresponding to the 1,2,3 button that you press.



**SCALE:** Use this radio button to select the scale operation. The scale operation is for 2D data only. The program multiplies each data point in the quadrant specified by the 1,2,3 button by a constant multiplier. Note that the 1,2,3 button must be the same as the currently selected data. The program displays the result in the lower right quadrant of the DATA MANIPULATION dialog (quadrant 4).

**INTEGRATE:** Use this radio button to select the integration operation. The integration operation is for 2D spectroscopic data only. The integration operation does not alter a data set, but integrates the data specified by the 1,2,3 button to produce a numerical result. Note that the 1,2,3 button must be the same as the currently selected data. The program displays the result in the operation parameter RESULT.

**OPERATION PARAMETERS:**

Use the text fields to specify parameters for each operation. Note each operation has a different set of parameters. For example, when the operation is SUBTRACT, the parameters are X OFFSET and Y OFFSET.

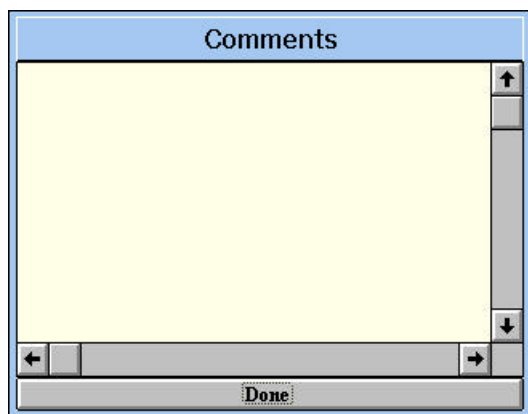
**SUBTRACT:** Use X OFFSET and Y OFFSET to offset one set of data from the other when subtracting. Use the GET button to import these coordinates from a point selected in the SELECTION FUNCTIONS dialog.

**AVERAGE:** If A=B, use BIN to specify the number of points to average over, i.e. the program will average every BIN data points into one data point. If A does not equal B, use X OFFSET and Y OFFSET to offset one set of data from the other when averaging. Use the GET button to import these coordinates from a point selected in the SELECTION FUNCTIONS dialog.

**SMOOTH:** Use BIN to specify the number of points to average over, i.e. the program will replace each data point by the average of itself and (BIN-1) neighboring data points.

**INTEGRATE:** The program displays the result of the integration in RESULT.

**SCALE:** Use MULTIPLIER to specify the constant factor that is multiplied by each data point.



Use the COMMENTS DIALOG to add comments to and view comments previously added to a set of data.

This dialog is straightforward. There is space to enter and edit comments and a button to press when you're DONE.

You may add comments to scanned data (\*.stm files), deposition data (\*.dep files), counting data (\*.cnt files) and spectroscopy data (\*.spc files).

| Counting  |     |     |     |
|---|-----|-----|-----|
| LMB:  | 255 | 0   | 0   |
| RMB:  | 0   | 255 | 0   |
| Shift LMB:  | 0   | 255 | 255 |
| Shift RMB:  | 255 | 0   | 255 |
| <input type="button" value="Clear All"/> <input checked="" type="radio"/> Add <input type="radio"/> Erase <input type="radio"/> Convert <input checked="" type="checkbox"/> Sloppy Erase <input type="button" value="Save"/> <input type="button" value="Load"/> <input type="button" value="Exit"/> <input type="button" value="Comment"/> |     |     |     |
| Symbol Size:  | 4   |     |     |

Use the COUNTING dialog to count features in an image in the DATA MANIPULATION dialog. The technique is fairly simple: with the COUNTING dialog open, click on features in the DATA MANIPULATION dialog. The program labels each point where you click with a colored dot and automatically keeps track of the number of clicks you have made. You can erase the colored dots in an analogous way.

#### DISPLAYED COUNTS:

LMB, RMB, shift LMB, shift RMB: The program displays the number of clicks that have been recorded. The four different labels refer to four different kinds of clicks: left mouse button, right mouse button, shift key + left mouse button and shift key + right mouse button.

#### DOT COLOR, SIZE:

R, G, B: In the upper right hand corner of the dialog, there are twelve editable text fields (three for each type of click). Use each set of three fields to set the color of the corresponding dots in the image. The color is specified in RGB (Red-Green-Blue) format, i.e. a set of three numbers from 0 to 255. (Examples: 255-0-0 is red, 0-0-0 is black).

SYMBOL SIZE: Use this text field to set the diameter of the colored dots in pixels.

#### SETTING THE COUNTING MODE:

ADD, ERASE, CONVERT: By clicking the mouse on the image, you can do three things: add a new dot, erase a current dot or convert a current dot to another kind. Use these radio buttons to switch between these modes of operation.

SLOPPY ERASE: Check this box to activate the sloppy erase. With sloppy erase, you can erase all kinds of dots by clicking the left mouse button.

MISCELLANY:

**LOAD and SAVE:** Use these buttons to save and load counting data (\*.cnt files). Note that the counting data is distinct and separate from scanned data. When you save or copy scanned data via the DATA MANIPULATION dialog, you do NOT save or copy the corresponding counting data.

**CLEAR ALL:** Use this button to erase all dots and reset the displayed counts to 0.

**HIDE:** Check this box to hide all of the dots. Note that this will not erase the dots.

**SCROLL BARS:** Use the scroll bars to shift the entire set of counting dots in the image.

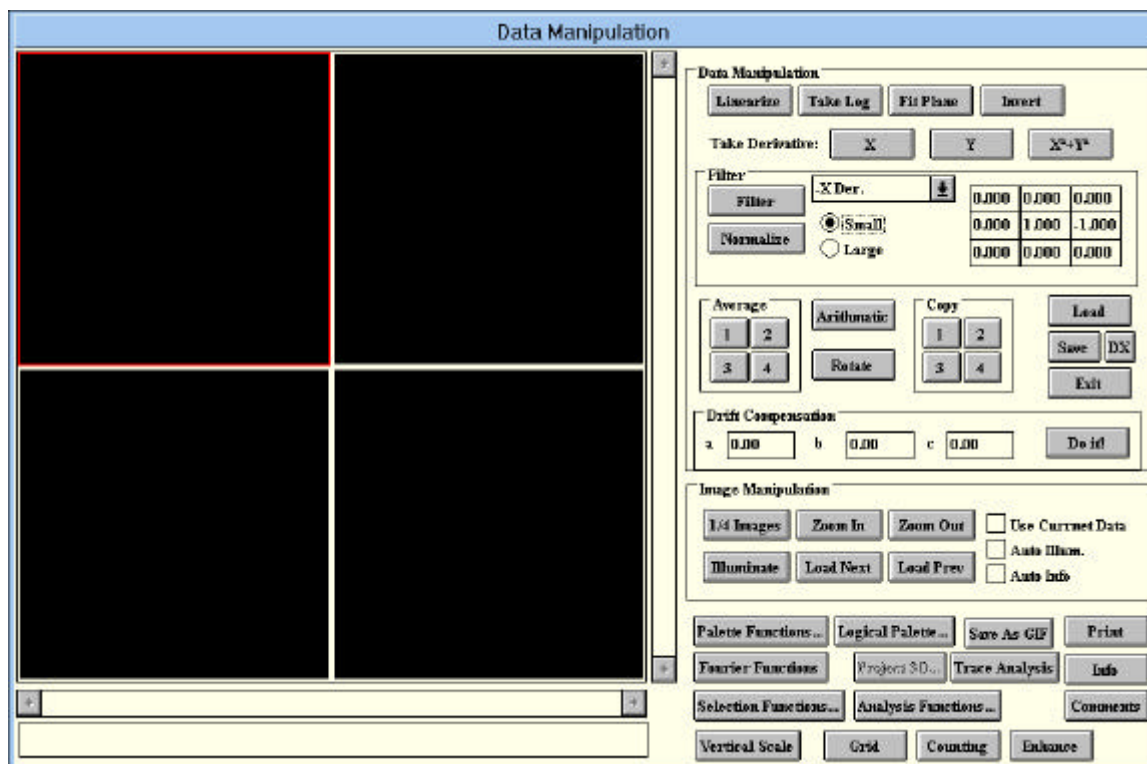
**COMMENT:** Use this button to open the COMMENT dialog. Comments added here will be saved along with \*.cnt files.

| Data Info  |                 |                       |                         |
|--|-----------------|-----------------------|-------------------------|
| Filename:  | File Version: 7 |                       |                         |
| Width:   | 1 Pixels        | 9.77 Å                | 0.05 V                  |
| Step Size:                                       | 1               | Z Freq: 10.0          |                         |
| Temperature:                                     | 0.00 K          | Input Ch.             | 1                       |
| Dep. Filename:                                   | Sequence #: 0   |                       |                         |
| Gains:   |                 |                       |                         |
| X: 10.0  | Y: 10.0         | Z: 10.0               |                         |
| X offset:  | -100.00 V       | Y Offset:             | -100.00 V               |
| Z offset:  | -10.00 V        | Z-Offset Crash Prot.: | 0                       |
| Scan Order and Direction: Y First, Read Forward. |                 |                       |                         |
| Preamp Gain:                                     | 8               | Current:              | 10.00 nA Bias: -10.00 V |
| Delays (µs):                                     |                 |                       |                         |
| Inter-step:                                      | 0               | Step:                 | 0                       |
| Inter-line:                                      | 0               | Seq Summary           |                         |
| Digital Feedback: No With Tip Spacing:           |                 |                       |                         |
| Feedback: On. Dither 0: Off. Dither 1: Off.      |                 |                       |                         |
| Overshot by: None                                |                 |                       |                         |
| Overshot waits:                                  |                 |                       |                         |
| Sample Type:                                     |                 |                       |                         |
| <input type="text"/>                             |                 |                       |                         |
| Dosed with:                                      |                 |                       |                         |
| <input type="text"/>                             |                 | 0.0000                | L                       |
| Comment:   |                 |                       |                         |
| <input type="text"/>                             |                 |                       |                         |
| Exit   |                 |                       |                         |

Use the DATA INFO dialog to display the physical parameters for a scanned image (3D data).

**BUTTON:**

**SEQ SUMMARY:** Use this button to open the SEQUENCE SUMMARY dialog.



Use the DATA MANIPULATION dialog to manipulate data for interpretation and presentation. This dialog works in conjunction with several other dialogs. For instance, making selections with the SELECTION FUNCTIONS dialog and counting features with the COUNTING dialog require you to click on data in the DATA MANIPULATION dialog.

#### DISPLAYED IMAGES:

**1/4 BUTTON:** Use this button to toggle between viewing one set of data and four sets of data at one time.

If you are viewing four sets of data, you can select a quadrant by clicking on it. The program displays a red outline around the selected quadrant. This manual refers to the selected quadrant as the "currently selected quadrant" and the data in that quadrant as the "currently selected data." If you are viewing only one set of data, that data is the "currently selected data."

#### LOADING AND SAVING DATA:

**LOAD and SAVE:** Use these buttons to load or save data (\*.stm, \*.spc, \*.cut or \*.dep files). The program loads data into and saves data from the currently selected quadrant. Note that manipulating data (with ILLUMINATE, ROTATE, etc) does not alter previously saved data.

**LOAD NEXT and LOAD PREV:** Use these buttons to load data in sequence from a directory. If the **USE CURRENT DATA** box is checked, the program will load the data that follows/precedes the currently selected data. Otherwise, the program will load the data that follows/precedes the data that was loaded by the program most recently.

**AUTO INFO:** Check this box to automatically open the DATA INFO dialog or DEPOSITION DATA INFO dialog when the program loads data.

**DX:** Use this button to save data in a format for export to another program (\*.exp files). 3D data saved this way readable by the Data Explorer and Mathcad applications. 2D data saved in this way is saved as an ASCII text file.

**SAVE AS GIF:** Use this button to save data as a gif image (\*.gif files).

#### DERIVATIVES AND OTHER MATHEMATIC OPERATIONS:

**(Take Derivative) X:** Use this button to calculate  $d/dx$  at all points in the currently selected data. The program displays the result in quadrant 4. When you execute this operation on an image, steep positive slopes in the x direction will appear bright and steep negative slopes will appear dark. The surface will appear to be illuminated from the left as you view it on the screen. This operation is for 3D data only.

**(Take Derivative) Y:** Use this button to calculate  $d/dy$  at all points in the currently selected data. The program displays the result in quadrant 4. When you execute this operation on an image, steep positive slopes in the y direction will appear bright and steep negative slopes will appear dark. The surface will appear to be illuminated from the bottom as you view it on the screen. This operation is for 3D data only.

**(Take Derivative)  $X^2 + Y^2$ :** Use this button to calculate the magnitude of the gradient at all points in the currently selected data. The program displays the result in quadrant 4. This operation is for 3D data only.

**FIT PLANE:** Use this button to fit a plane to the currently selected data of the form:  $z = ax + by - c$ . The program replaces the value of z at each point in the data by its value relative to the plane. The program automatically adds the values of a, b, and c to the comments for the data. This operation is for 3D data only.

**INVERT:** Use this button to invert the currently selected data. The program replaces the value of z at each point in the currently selected data by -z. This operation is for 3D data only.

**TAKE LOG:** Use this button to take the logarithm of the currently selected data. The program replaces the value of z at each point in the currently selected data by the base 10  $\log(z)$ . The program automatically appends a description of this operation to the comments. This operation is for 3D data only.

**LINEARIZE:** Use this button to rescale the range of z values for the currently selected data. The program rescales the range to 256 distinct values. The program automatically appends a description of this operation to the comments. This operation is for 3D data only.

#### ZOOMING:

**ZOOM IN/ZOOM OUT:** Use these buttons to zoom in or out of the currently selected data by a factor of two.

**SCROLL BARS:** The program activates the scroll bars when you have zoomed in on the currently selected data. Use them to scroll across the image.

#### FILTERING:

**EDITABLE TEXT FIELDS:** Nine editable text fields to the right of the Filter button form a 3x3 convolution matrix. The matrix represents a point and its eight nearest neighbors. When the program filters an image, each point is replaced by a weighted sum of the datum at that point and the data from its nearest neighbors. The convolution matrix specifies the weight for each data point.

**FILTER:** Use this button to filter the currently selected data. The program displays the result in the lower right quadrant (quadrant 4).

Use the filter listbox to choose a filter. The program displays the filter's convolution matrix in the editable text fields.

**SMALL, LARGE:** These radio buttons refer to a feature that has not been implemented.

**NORMALIZE:** Use this button to multiply the convolution matrix by  $1/(\text{sum of all matrix elements})$ , i.e. to normalize the matrix. It is highly recommended that you normalize the convolution matrix before filtering data.

**DRIFT COMPENSATION:**

Use these text fields and button to map a square set of data to a parallelogram in order to compensate for piezo drift. The algorithm and definition of the parameters can be found in Sang-il Park's thesis.

**MISCELLANEOUS MANIPULATIONS:**

**ROTATE:** Use this button to rotate the currently selected data 90 degrees counter-clockwise. The program displays the result in the lower right hand quadrant (quadrant 4).

**COPY 1,2,3,4:** Use these buttons to copy the currently selected data into the quadrant corresponding to the button number.

**ILLUMINATE:** Use this button to alter the currently selected image so that raised features appear to cast shadows (This involves taking a derivative). The program displays the result in the lower right quadrant (quadrant 4). If the AUTO ILLUMINATE button is checked, the program automatically illuminates images when it loads them. This operation is for 3D data only.

**ENHANCE:** Use this button to increase the data size to 512x512 pixels. The program interpolates the values of z in the new 512 x 512 data from the old data. The program displays the result in the lower right quadrant. This operation is for 3D data only.

**BUTTONS THAT OPEN OTHER DIALOGS:**

**ANALYSIS FUNCTIONS:** Use this button to open the ANALYSIS dialog.

**ARITHMATIC:** Use this button to open the ARITHMATIC OPERATIONS dialog.

**COMMENTS:** Use this button to open the COMMENTS dialog. The program saves comments added here with \*.stm, \*.spc, and \*.dep files.

**COUNT:** Use this button to open the COUNTING dialog.

**FOURIER FUNCTIONS:** Use this button to open the FOURIER FUNCTIONS dialog.

**GRID:** Use this button to open the GRID dialog.

**INFO:** Use this button to open the DATA INFO dialog if the currently selected data is 3D. Use this button to open the DEPOSITION DATA INFO dialog if the currently selected data is 2D.

**LOGICAL PALETTE:** Use this button to open the LOGICAL PALETTE FUNCTIONS dialog.



PALETTE FUNCTIONS: Use this button to open the PALETTE FUNCTIONS dialog.

PRINT: Use this button to open the PRINT dialog.

SELECTION FUNCTIONS: Use this button to open the SELECTION FUNCTIONS dialog.

TRACE: Use this button to open the TRACE ANALYSIS dialog.

VERTICAL SCALE: Use this button to open the VERTICAL SCALE dialog.

| Deposition Data Info  |  |
|---|--|
| Filename:   | 01269804 File Version: 5                               |
| Temperature:  | 8.00 K ZFreq: 9  |
| Pulse Duration:   | 10000000 $\mu$ s Feedback: Off                         |
| Measured:   | Current  |
| <div> Ramp Speeds<br/> Bias: Step      Current: 1.000<br/> Z-Offset: 1.000 </div> |  |
| Z offset:   | 0.000 V Z-Offset Crash Prot.: No                       |
| Z-Offset used movement bias:  | Yes  |
| Preamp Gain:  | 9 Current 1.00 nA Bias: 0.100 V                        |
| <div> Delays (<math>\mu</math>s):<br/> Move: 5      Write: 5 </div>               |  |
| <div> Bias (V):<br/> Move: 0.100      Write: 0.400 </div>                         |  |
| <div> Currents:<br/> Move: 1.000      Write: 1.000 </div>                         |  |
| Averaged Samples:   | 10   |
| Sample Type:  | Cu(100)  |
| Dosed with:   | C2H2      0.0000 L                                     |
| Comment:  | Wed Jan 28 02:00:21 1998<br>y offset = 60 bits<br>C2h2 |
| Exit  |  |

Use the DEPOSITION DATA INFO dialog to display information about 2D data. Note that the information displayed for spectroscopic data (\*.spc files) is slightly different from that displayed for deposition data (\*.dep files).

**Electronics Test**

**Analog Outputs**

Channel:  Range: ☐ +/- 5V ☒ +/- 10V

Output:  Binary Bits:  Voltage:

**Inputs**

|                                    | Input                          | Bits   | Voltage                              |
|------------------------------------|--------------------------------|--|--------------------------------------|
| <input type="checkbox"/> Channel 0 | <input type="text" value="0"/> | <input type="text" value="0000 0000 0000 0000"/> | <input type="text" value="-9.9997"/> |
| <input type="checkbox"/> Channel 1 | <input type="text" value="0"/> | <input type="text" value="0000 0000 0000 0000"/> | <input type="text" value="-9.9997"/> |
| <input type="checkbox"/> Channel 2 | <input type="text" value="0"/> | <input type="text" value="0000 0000 0000 0000"/> | <input type="text" value="-9.9997"/> |
| <input type="checkbox"/> Channel 3 | <input type="text" value="0"/> | <input type="text" value="0000 0000 0000 0000"/> | <input type="text" value="-9.9997"/> |

**Digital Outputs**

Channel:

12 11 10 9 8 7 6 5 4 3 2 1

☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐

**Automatic Tests**

Out Ch.:  In Ch.:

**Ramp**

From:  Bits

To:  Bits

**Bit Test**

Use the ELECTRONICS TEST dialog to check the STM electronics. Note that “Outputs” refers to outputs from the computer to the STM and “Inputs” refers to inputs from the STM to the computer.

#### ANALOG OUTPUTS:

**CHANNEL:** Use this scroll bar or text field to choose one of twelve channels, indexed from 0 to 11. The output channel is displayed, in binary form, on the four LEDs labeled OUTPUT 0,1,2 and 3 on the Opto-Isolator unit.

**RANGE:** Use these radio buttons to set the range of output voltages for each channel.

**SET RANGE:** This button does nothing.

**OUTPUT:** Use the text fields or scroll bar to set the output that will be sent to the STM. The program displays the output in decimal, binary and as a voltage. Use the button to send the data. One way to check if the data was sent successfully is to connect a multimeter to the appropriate connector on the Computer Interface unit. Additionally, the appropriate LEDs in the D/A row on the Opto-Isolator unit should light up.

#### ANALOG INPUTS:

**UPDATE:** Use this button to read input from the STM electronics. The program displays the input in decimal, binary, and as a voltage. To properly perform a test, connect a known voltage to the appropriate connector on the Computer Interface unit. Additionally, the appropriate LEDs in the A/D row on the Opto-Isolator unit should light up.

**CHECK BOXES:** Check these boxes if you want the corresponding channels to be read during the next UPDATE. The input channel is displayed, in binary form, by the LEDs labeled input 0, 1 on the Opto-Isolator unit.

**DIGITAL OUTPUT:**

**CHANNEL:** Use the scroll bar to select the output channel. Note that in this portion of the dialog, the twelve output channels and the sequence of bits that are sent are improperly indexed from 1 to 12 instead of 0 to 11.

Use the check boxes at the bottom of the screen to specify the sequence of bits to send to the STM.

**OUTPUT:** Use this button to send the sequence of bits to the specified channel. For example, say you select channel 7 and check the 7 and 9 boxes. When the OUTPUT button is hit, the number 0, 1, 2 output LED's and the 6 and 8 LED's in the D/A row on the Opto-Isolator unit should light.

**AUTOMATIC TESTS:**

**OUT CH and IN CH:** In the automatic testing mode, connect one output channel to one input channel on the Computer Interface unit. Use these text fields to indicate which two channels are connected.

**RAMP TEST:** This test sends a linearly increasing signal to the STM and reads it back in. Use the FROM and TO text fields to specify the range of the test signal. Use the DO IT! button to execute the test. The program displays the results of the test on the graph. If the electronics are working properly, the graph will be a straight line.

**CALIBRATION TEST:** This test sends twelve bits to the output channel and reads those twelve bits from the input channel. The program compares the input and output bit by bit and the comparisons are shown in a message box.

**TEST INPUT:** This test is analogous to the CALIBRATION TEST, but should be used if the output electronics are known to be ok.

**TEST OUTPUT:** This test is analogous to the CALIBRATION TEST, but should be used if the input electronics are known to be ok.

**Fourier Functions**

FFT   Inv. FFT   FHT   Inv. FHT   Filter

**Filter**

Low Cut-Off: 0.20   ☐ Rectangle   ☐ Triangle

High Cut-Off: 1.00   ☒ Hanning   ☐ Hamming

Order: 10   ☐ Bartlett

☒ Low Pass   ☐ High Pass   ☐ Band Pass   ☐ Band Stop

Exit

Use the FOURIER FUNCTIONS dialog to perform Fourier transforms on 3D data.

The transforms are for 3D data only. In every case, the currently selected data is transformed by the operation. The program displays the transformed data in quadrant 4 of the DATA MANIPULATION dialog.

FFT and INV FFT: Use these buttons to perform a fast Fourier transform and inverse fast Fourier transform on the currently selected data.

FHT and INV FHT: Use these buttons to perform a fast Hartley transform and inverse fast Hartley transform on the currently selected data.

FILTER: Use this button to filter the currently selected data according to the settings in the Filter segment of the dialog and then perform a fast Hartley transform on the filtered data.

#### FILTER SETTINGS:

ORDER: Use this text field to set the diameter of the filter.

HIGH and LOW CUT-OFF: Use these text boxes to set the cut-off frequencies for the filter. Use the HIGH CUT-OFF to specify the cut-off for a HIGH PASS filter. Use the LOW CUT-OFF to specify the cut-off for a LOW PASS filter. Use both cut-offs to specify the band of frequencies for BANDPASS and BANDSTOP filters.

RECTANGLE, TRIANGLE, HANNING, HAMMING, BARTLETT: Use these radio buttons to select the type of window for the filter.

LOW PASS, HIGH PASS, BAND PASS, BAND STOP: Use these radio buttons to select the type of filter.

Use the GRID dialog to superimpose a grid over a set of data.

A grid is a mesh composed of up to three sets of parallel lines. There are three columns of parameters in the upper part of the dialog. Each column corresponds to one of the three sets of parallel lines.

Note that grids are distinct from data sets. The program loads and saves grids separately from data sets. Saving data does not save a grid. Each of the four quadrants in the DATA MANIPULATION dialog has its own grid.

#### PARAMETER TEXT FIELDS:

**ANGLE:** Use these text fields or scroll bars to set the angle between a set of parallel lines and the horizontal. For example, set the angle to 90 degrees for a vertical line. The range of acceptable angles is from -90 to 90 degrees.

**DISTANCE:** Use these text fields or scroll bars to set the separation between consecutive lines. The units (angstroms) are calculated based on the piezotube response at room temperature and are not valid at other temperatures.

**R:** Use these text fields or scroll bars to set the distance of the bottom-most line from the bottom edge. The magnitude of this number can't be greater than what is specified in DISTANCE. The units (angstroms) are calculated based on the piezotube response at room temperature and are not valid at other temperatures.

CLIPPING:

CLIP TO LINE: Check this box to clip the grid in the PostScript file produced by the PRINT dialog. Note that the grid will not appear to be clipped in the PRINT dialog. This feature clips the grid only, not the scanned image.

GET: Use this button to import the border of the clipping region from the SELECTION FUNCTIONS dialog. The line is displayed in four editable text fields to the left of the GET button.

CLIP < Y: Check this box to clip the portion of the grid that is less than the line specified in the editable text fields.

OTHER CONTROLS:

FROM LINE/NUMBER OF LINES: With two lines selected with the SELECTION FUNCTIONS dialog, use the GET button to display the number of grid lines between the two selected lines.

HIDE: Check this box to hide the corresponding set of parallel lines. Grids are initially hidden by default.

LOCK: These boxes do nothing.

BUTTONS:

COPY 1,2,3,4: Use these buttons to copy the grid (but *not* the data) from the currently selected quadrant in the DATA MANIPULATION dialog to the quadrant corresponding to the button that is pressed.

HIDE GRIDS: This button does nothing.

MATCH: Use this button to match the angle and separation of the right column set of lines with the middle and left column sets of lines. The program will alter the ANGLE and DISTANCE text fields accordingly. Note that you may have to adjust the R value of the right column to see the match.

LOAD and SAVE: Use these buttons to load and save grid data (\*.grd files).

SCROLL BARS: Use the scroll bars to shift the entire grid. The program will change the values of R in each column in response.

| Inputs  |       |                     |         |
|---|-------|---------------------|---------|
|   | Input | Bits                | Voltage |
| <input checked="" type="checkbox"/> Channel 0 | 0     | 0000 0000 0000 0000 | -9.9997 |
| <input type="checkbox"/> Channel 1            | 0     | 0000 0000 0000 0000 | -9.9997 |
| <input type="checkbox"/> Channel 2            | 0     | 0000 0000 0000 0000 | -9.9997 |
| <input type="checkbox"/> Channel 3            | 0     | 0000 0000 0000 0000 | -9.9997 |

Use the INPUTS DIALOG to read data from the channels of the a-to-d converter. Note that “Inputs” refers to inputs from the STM to the computer. This dialog is intended for use during testing of the STM and the software.

**CHANNEL:** Check these boxes if you want the corresponding channel to be read during an UPDATE.

**UPDATE:** Use this button to read from each channel with a checked CHANNEL box. The program displays the data that is read in the editable text fields. In the input column, the program displays the data in decimal format. In the bits column, the program displays the data as a binary string. This binary string should match the A/D LED display on the opto-isolator box. In the voltage column, the program displays the data as a voltage.



IO Functions

**Positioning**  
☒ X ☐ Y ☐ Z Channel: 2  
Range: ☐  $\pm 5$  V ☒  $\pm 10$  V Send  

| Bits | Voltage | Angs. |  |
|------|---------|-------|--|
| 2048 | 0.00    | 0.00  | <div style="display: flex; justify-content: space-around;"> <span>←</span> <span>→</span> </div> |

**Offsets**  
☒ X ☐ Y ☐ Z Channel: 5  
Range: ☐  $\pm 5$  V ☒  $\pm 10$  V Send  

| Bits | Voltage | Angs. |  |
|------|---------|-------|--|
| 2048 | 0.00    | 0.00  | <div style="display: flex; justify-content: space-around;"> <span>←</span> <span>→</span> </div> |

**Gains**  
X ☐ 0.1 ☒ 1.0 ☐ 10 Y ☐ 0.1 ☒ 1.0 ☐ 10  
Z ☒ 0.1 ☐ 1.0 ☐ 10 Z2 ☐ 0.1 ☐ 1.0 ☒ 10

**Sample Movement**  
Mode: ☒ Translate ☐ Rotate  
Movement: ☐ Coarse ☒ Fine

**Sample Bias**  

| Bits | Voltage |  |
|------|---------|--|
| 2253 | 1.001   | <div style="display: flex; justify-content: space-around;"> <span>←</span> <span>→</span> </div> |

**Tip Current**  
PreAmp Gain ☐ 8 ☒ 9  
(10th powers) ☐ 10  

| NanoAmps | 0.1 to 10 nA   | I SET Bits |
|----------|--|------------|
| 1.2521   | <div style="display: flex; justify-content: space-around;"> <span>←</span> <span>→</span> </div> | 1848       |

**Dither**  
Ch. 0: ☐ On ☒ Off Bias Square Wave  
Ch. 1: ☐ On ☒ Off Delay: 10000  $\mu$ s

**Tip Movement**  
Feedback: ☒ On ☐ Off Defaults  
Retract: ☐ On ☒ Off Reinit Exit

Emergency Exit

Use the I/O FUNCTIONS dialog to carry out basic STM actions: setting the sample bias, moving the offset and fine coordinates, retracting the tip, etc. This dialog contains a complete set of "low-level" controls of the STM. These controls are useful when testing the piezo and feedback control electronics, but only a few of them have been used during experiments to date. Note that this dialog does not provide any safeguards to prevent you from making drastic mistakes with the STM.

The Ho group's STM follows the design of Besocke. In this design, the sample sits on three piezotubes arranged in a triangle. At the center of the triangle is a fourth piezotube that holds the tip.

Signals sent to the three outer piezotubes move the sample in the lateral (x, y) and vertical (z) directions. The net x signal is the sum of two signals: fine x and x offset. Similarly, the net y signal is the sum of fine y and y offset. Only one signal controls the outer piezotubes' z motion. It is referred to as zo or "z outer." The program determines the values of these signals.

The center piezotube moves only in the z direction. The center piezotube z signal is the sum of two separate signals: z offset and feedback z. The feedback z signal is provided by the feedback electronics. The feedback z signal is what is read during a 3D, topographic scan (provided feedback is on). The program controls the value of the z offset signal.

#### POSITIONING:

X, Y, Z: These radio buttons refer to the fine x, fine y, and zo signals respectively. Use these radio buttons to select which signal to set. The program displays the output CHANNEL that corresponds to the selected signal.

+5V, +-10V: Use these radio buttons to set the range of the selected signal.

Use the text fields and scroll bar to set the value of the selected signal, i.e. the destination of the sample. The program displays this value in units of bits, volts, and angstroms.

SEND: Use this button to send the sample to the destination.

#### OFFSETS:

The offsets controls function in the same way as the positioning controls, except that here, X, Y, and Z refer to the x offset, y offset, and z offset signals, respectively.

#### GAINS:

X, Y, Z, Z2: X and Y refer to the gain settings for fine x and fine y. The fine x and fine y gains are only active when the piezotubes are in translation mode. When the piezotubes are in the rotation mode, the fine x and fine y gains are fixed at ten. Z and Z2 refer to gain settings for the feedback electronics. Z2 is the same as the gain labeled Z GAIN in the PRE-SCAN CONTROLS dialog. The FREQUENCY in the PRE-SCAN CONTROLS dialog is determined by the product of Z and Z2. Note that the program automatically executes an AUTO- before changing the gain in the PRE-SCAN CONTROLS dialog. No such precautions are taken when you change the gain from the IO FUNCTIONS dialog.

#### SAMPLE MOVEMENT:

MODE, MOVEMENT: Use these radio buttons to set the piezotube mode and type of motion. In translation mode, the outer piezotubes move together along mutually perpendicular directions. In rotation mode, the outer piezotubes move cooperatively to rotate the sample. In coarse mode, the zo signal is "off," while in fine mode, it is "on." For example, during scans, the mode is coarse-translation. During tip approach, the mode is fine-rotation.

#### SAMPLE BIAS:

+5V, +-10V: Use these radio buttons to set the range of the sample bias signal.

Use the text field or scroll bar to adjust the sample bias. It is highly recommended that you use the scroll bar to adjust the sample bias since the sample bias changes in real time as you change the dialog.

#### TIP CURRENT:

PRE-AMP GAIN: These radio buttons do NOT actually change the pre amp gain. That must be done manually. This setting is saved with data.

Use the text field or scroll bar to adjust the tip current. As with the sample bias, it is highly recommended that you use the scroll bar to make adjustments since the tip current changes in real time as you change the dialog.

**DITHERING:**

ON and OFF: Use these radio buttons to turn dithering on or off for channel 0 or channel 1. Dithering input channels is usually done in conjunction with the use of a lock-in amplifier.

**TIP MOVEMENT:**

FEEDBACK ON and OFF: Use these radio buttons to turn the feedback on or off.

RETRACT ON and OFF: Use these radio buttons to retract or unretract the tip. It is advisable to retract the tip whenever you are going to physically handle the apparatus (during dosing or while changing the pre-amp gain, for example).

**OTHER BUTTONS:**

BIAS SQUARE WAVE: Use this button to send the sample bias through a sequence of square wave pulses with a minimum voltage of 0.5 volts and a maximum voltage of 1.0 volts. Use the DELAY text field to set the delay between consecutive pulses. Use the ESC key to terminate the wave. In the past, this feature has been used to test the response of the feedback electronics.

DEFAULTS: This button does nothing.

EMERGENCY EXIT: Use this button to exit the entire STM program.

REINIT: Use this button to reset the computer's ports and registers and to reset the output channels.

| Lock-in Settings                    |                      |             |
|-------------------------------------|----------------------|-------------|
| Sensitivity 1:                      | <input type="text"/> | mV          |
| Sensitivity 2:                      | <input type="text"/> | mV          |
| Oscill. Freq:                       | <input type="text"/> | Hz          |
| Oscill Mag:                         | <input type="text"/> | mV          |
| Time Const:                         | <input type="text"/> | ms          |
| N(oise) 1:                          | <input type="text"/> | mV/sqrt(Hz) |
| Phase 1:                            | <input type="text"/> | Deg.        |
| Phase 2:                            | <input type="text"/> | Deg.        |
| <input type="button" value="Exit"/> |                      |             |

In the LOCKIN SETTINGS dialog, you can type in several parameters related to a lockin amplifier. However, these parameters are neither used nor saved by the program. This dialog essentially does nothing.

Use the LOGICAL PALETTE FUNCTIONS dialog to create a logical palette and apply it to the currently selected image.

A palette is an array of 256 specified colors that acts as a look up table, mapping data values to colors in the array (see PALETTE FUNCTIONS dialog). A logical palette differs from a palette in that only a small number of RGB colors are specified in a logical palette. The program interpolates the colors for data that does not map to one of the few explicitly specified colors.

With this dialog, you can specify the small number of points that constitute a logical palette. The program will automatically interpolate for data that does not map directly to the specified points, providing shades between the colors you have specified.

Note that although an applied logical palette will be copied along with data in the DATA MANIPULATION dialog, logical palettes do not actually alter data and are not saved along with data.

### WHY IS THIS USEFUL?

If you create a histogram of STM data, most data sets are bell-shaped and fill a large portion of the palette. However, some of the histograms have multiple peaks or are narrow. Images of such data will lack contrast if a palette is used. You can use a logical palette to insure that the colors vary slowly with the data over plateaus in the histogram and that the colors vary quickly with the data over peaks in the histogram. This will produce an image with greater contrast.

### CREATION / DESTRUCTION:

CREATE: Use this button to create one logical palette.

DESTROY: Use this button to destroy the logical palette and restore the old palette.

VISUALIZE: This button does nothing.

REPAINT: Use this button to apply a logical palette to the currently selected data.

### CURRENT POINT

INDEX: The program displays the index of a point in the logical palette in this text field. The index runs from 0 to 255. Do not try to add points by typing an index in this field. Instead, use the ADD POINT button.

R, G, B: In these text fields, the program displays a series of three numbers that specify the color for the current point. Each value must be within the range from 0 to 255. Examples: 0, 0, 255 is blue. 0, 0, 0 is black. 0, 255, 255 is blue-green.

**GAMMA:**

Gamma is a parameter that describes how the program interpolates the shades between the colors you specify. If gamma is 1, the program will interpolate linearly. If gamma is 2, the program interpolates quadratically. For example, say index 0 is RGB 0, 0, 255 and index 255 is 0, 255, 255. If gamma is 1, the shades will vary smoothly from blue to purple. If gamma is 3, the most of the shades will be blue and only near index 255 will the shades be closer to purple. Use this text field to adjust gamma for the logical palette.

**REMOVE:** Use this button to remove the specification of the current point from the logical palette.

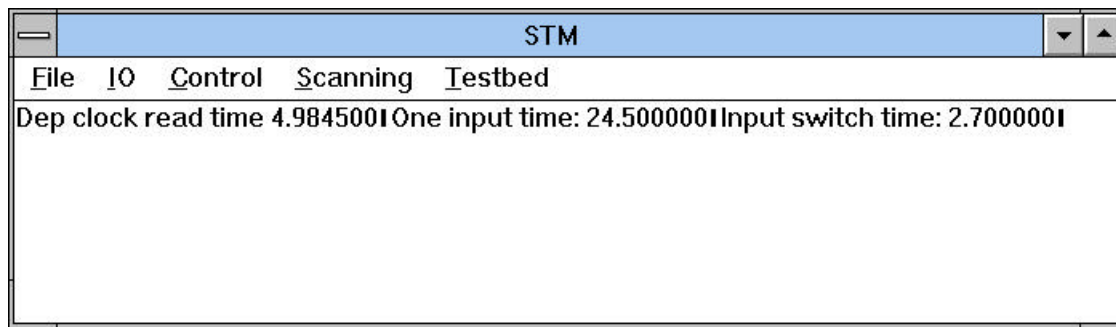
**EQUALIZE:** Check this box to EQUALIZE the currently selected image (see the PALETTE FUNCTIONS dialog for more information on EQUALIZE).

**SPECIFYING A NEW POINT**

**ADD POINT:** Use this button to specify another point in the logical palette. The index of the point you add will appear in the INDEX text field. The values of R, G and B that will initially appear correspond to the shade that the program interpolates at that index. By default, the initial value of the ADD POINT text field is halfway between the current point and the next point.

**MOVING THROUGH THE POINTS**

**PREV / NEXT:** Use these buttons to move through the series of points that you have specified.



The MAIN WINDOW consists of five menus that provide access to dialog boxes.

MENU ITEMS THAT OPEN DIALOGS:

SCAN

DATA MANIPULATION

TIP APPROACH

SELECTIVE DEPOSITION

INPUTS

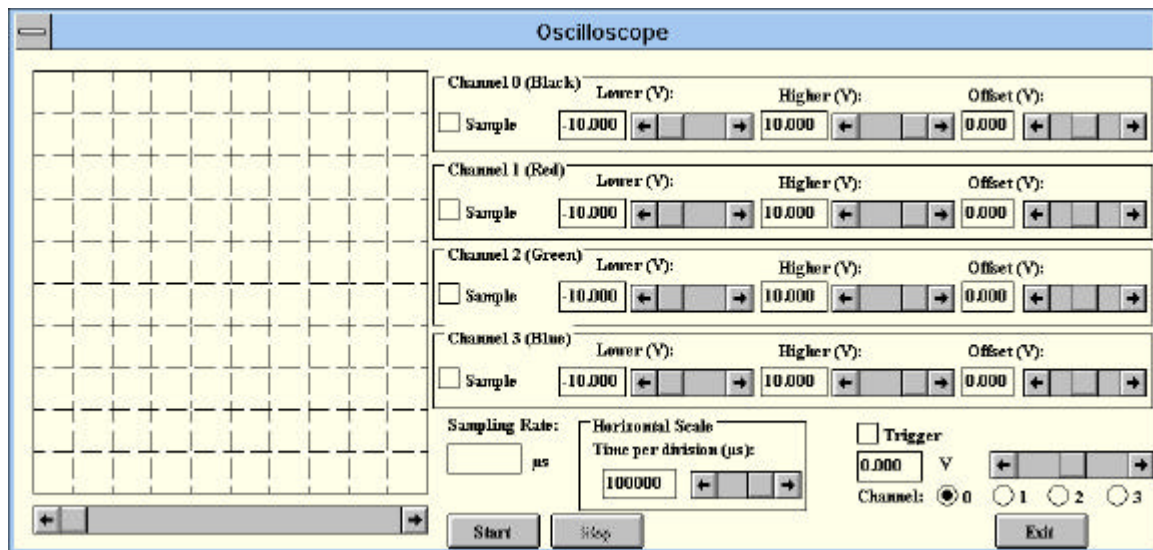
OUTPUTS

ELECTRONICS TEST

I/O CONTROL

OSCILLOSCOPE

Additionally, there are several menu items which do not open dialogs. These menu items perform specialized tests of the electronics. It is highly recommended that you do not run these specialized tests.



Use the OSCILLOSCOPE dialog to run the program as an oscilloscope. The oscilloscope can display up to four signals, one for each input channel.

#### VOLTAGE (VERTICAL AXIS):

**SAMPLE:** Check this box to display the input signal from the corresponding input channel on the oscilloscope.

**LOWER and HIGHER:** Use these text fields and scroll bars to set range of displayed voltages on the oscilloscope's vertical axis.

**OFFSET:** Use these text fields to set the voltage offset on the oscilloscope's vertical axis.

**START and STOP:** Use these buttons to start and stop the trace.

#### TIME (HORIZONTAL AXIS):

**SAMPLING RATE:** The program displays the time that elapses between each sampling of the inputs in this text field.

**HORIZONTAL SCALE:** Use this text field and scroll bar to set the time scale on the oscilloscope's horizontal axis.

#### TRIGGERING:

**TRIGGER:** Check this box to use one of the input channels as a trigger for the oscilloscope.

**V:** Use this text field and scroll bar to set the triggering voltage.

**CH 0,1,2,3:** Use these radio buttons to determine which channel to use as the trigger.



**Outputs**

**Channel**

← [ ] [ ] [ ] → 7

**Range**

☐ +/- 5V ☒ +/- 10V

**Output**

2070      100000010110      0.10742

← [ ] [ ] [ ] →

Set Range      OUTPUT      Exit

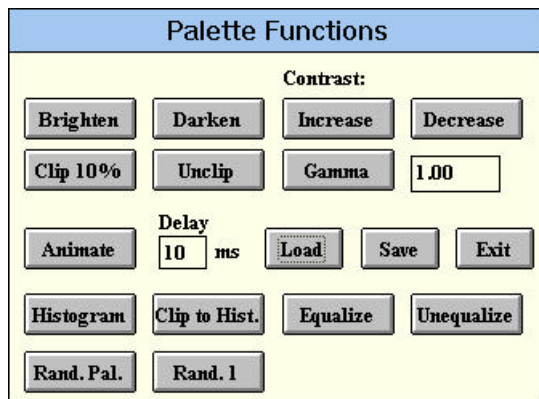
Use the **OUTPUTS** dialog to send data to a channel of the d-to-a converter. Note that “Outputs” refers to outputs from the computer to the STM. This dialog is intended for use during testing of the STM and the software.

Use the text fields or scroll bar to specify the data to be sent to the STM. The program displays the data in decimal format, as a string of binary data and as a voltage. If the electronics and the program are working properly, the channel will be displayed in binary form on the three output LEDs and the data will be displayed on the D/A LEDs.

CHANNEL: Use this scroll bar to select which channel (0-15) to send data to.

RANGE: Use these radio buttons to set the range of the output channel.

**SET RANGE BUTTON:** This button does nothing.



Use the PALETTE FUNCTIONS dialog to alter the palette of currently selected 3D data in the data manipulations dialog. The palette is an array of 256 colors used to color in the topograph. The palette acts as a look-up table that maps data values to colors. The contrast available with a palette is limited, since there are only 256 colors in the array.

**LOAD / SAVE:** Use these buttons to load and save palette data (\*.pal files).

**HISTOGRAM:** Use this button to display a histogram of the currently selected data. The histogram is displayed in quadrant 4 of the DATA MANIPULATION dialog. Palette colors are along the horizontal axis of the histogram and the frequencies of the corresponding data are along the vertical axis. Note that this histogram describes the data, but not necessarily the image. The currently selected quadrant must be 1,2 or 3.

**EQUALIZE / UNEQUALIZE:** Use this button to display an image with maximum contrast. When you hit the equalize button, the program adjusts the image so that each color in the palette appears with the same frequency. Note that equalizing alters the currently selected image, but not the currently selected data.

**GAMMA:** Gamma is a parameter that describes the mapping between data values and colors. If gamma is one, the colors change linearly with the data values. If gamma is 2, the colors change quadratically with the data values. Use this button and text field to adjust gamma for the palette of the currently selected data.

**RAND PAL:** Use this button to apply a randomly generated palette that includes shades of a random number of colors.

**RAND 1:** Use this button to apply a randomly generated palette that includes shades of a single color.

**CLIP TO HIST, CLIP 10%, UNCLIP:** These features are not useful and may not function as planned at the present time.

**BRIGHTEN and DARKEN:** These buttons do nothing.

**CONTRAST INCREASE and DECREASE:** These buttons do nothing.

**ANIMATE, DELAY:** This button and text field do nothing.

**Pre-Scan Controls**

**Resolution**  
 Z Gain ☐ 0.1 ☐ 1.0 ☒ 10  
 Frequency ☒ 1 kHz ☐ 10 kHz ☐ 100 kHz

**Tip Current**  
 PreAmp Gain ☐ 8 ☒ 9 ☐ 10  
 NanoAmps 0.1 to 10 nA I SET Bits  
 1.2521  1848

**Sample Bias**  
 Bits Voltage   
 2253 1.001

**Z Offset**  
 Bits Voltage Angstroms   
 Z 2048 0 0   
 ☒ Crash Protect

**Step Delay**  
☐ Freq/Feedback Dependent  
☒ Fixed: 500  μs

**Inter-Step Delay**  
 50  μs    
 Velocity 19531.2  Å/s  
☐ Wait for tip-spacing of:  
 0.50  Å    
☒ Abort after 1000  tries.  
 Require 1  consecutive, in range values.

**Crash Protection**  
 If measured quantity is within 5.00  % of its limits,  
☒ Do nothing.  
☐ Stop scanning.  
☐ Change Z-offset a little and continue.  
☐ Find min/max for Z-offset and continue.

**Read Sequence (waits in μs)**  
 Sequence #: 1 of 1  
☒ Feedback On  
 Wait 1: 0   
☐ Dither Ch. 0 On  
 Wait 2: 0   
☐ Dither Ch. 1 On  
 Wait 3: 0   
☒ Record Channel: 1 x 5

**Move Parameters**  
☒ Feedback On  
☐ Dither Ch. 0 On  
☐ Dither Ch. 1 On

**Inter-Line Delay**  
 20  ms

The PRE-SCAN CONTROLS dialog enables you to adjust several scan parameters. (It is advisable that you understand the SCAN dialog before trying to understand the PRE-SCAN CONTROLS).

#### RESOLUTION:

**Z GAIN and FREQUENCY:** Use these radio buttons to adjust the gain settings for the feedback electronics. Z GAIN is the same setting as Z2 in the IO FUNCTIONS dialog and effectively controls the range of motion of the inner piezotube. The FREQUENCY is the product of the Z and Z2 settings in the IO FUNCTIONS dialog. As a precautionary measure, the program executes an AUTO+ or AUTO- command (described below) before changing the gain.

#### SAMPLE BIAS:

Use the editable fields or scroll bar to adjust the sample bias. It is highly recommended that you use the scroll bar to adjust the sample bias since the sample bias changes in real time as you move the scroll bar.

**FLIP:** Use this button to change the sample bias from V to -V.

#### TIP CURRENT:

**PRE-AMP GAIN:** These radio buttons do NOT actually change the pre amp gain. That must be done manually. This setting is saved with the data.

Use the editable field or scroll bar to adjust the tip current. As with the sample bias, it is highly recommended that you use the scroll bar to make adjustments since the tip current changes in real time as you change the dialog.

#### Z OFFSET:

The z offset signal is added to the feedback z signal before it is sent to the inner piezotube. Use the editable text field and a scroll bar to adjust the z offset signal. The z offset signal does not change until you hit the SEND button. The program displays the offset in units of bits, volts, and angstroms. Note that the feedback electronics keep the tip-sample separation the same as you change the z offset.

**ZERO:** Use this button to set the z offset to zero volts. When you click this button, the program will display zero in the text field and scroll bar. However, you must still click the SEND button for the change to take effect.

**AUTO- or AUTO+:** Use these buttons to adjust the z offset so that the feedback z is zero volts. If the feedback is on, the net signal sent to the inner piezotube (z offset plus feedback z) will not change. AUTO- goes slightly below the target. AUTO+ goes slightly above the target.

**CRASH PROTECT:** Check this box to have crash protection on when you send the z-offset.

#### STEP DELAY:

Use this text field to set a fixed step delay. The step delay is the number of microseconds that the tip pauses over a position where it will take data before it actually takes the data.

#### INTER-STEP DELAY:

Use this text field to set the inter-step delay. The inter-steps are the small movements of the tip in the fast direction. It may take several inter-steps to get from one position where data is taken to another. The inter-step delay is the number of microseconds that the tip waits between each inter-step movement. The longer the STEP DELAY and INTER-STEP DELAY, the longer the scan will take.

**WAIT FOR TIP SPACING:** Check this box to instruct the program to wait at an interstep position until the variance of the tunneling current is within a certain tolerance. Use the text field or scroll bar to set the tolerance.

**ABORT:** Check this box to instruct the program to abort if the tunneling current fails to fall within the specified tolerance.

#### INTER-LINE DELAY:

Use this text field to set the inter-line delay. The inter-line delay is the number of milliseconds that the tip pauses before moving over one pixel in the slow direction.

#### PARAMETERS:

**DITHER CH1 ON and DITHER CH2 ON:** Check these boxes to have dithering on during a scan.

**FEEDBACK ON:** Check this box to have the feedback on during the scan. If the feedback is on, the STM is operating in a constant current mode. Otherwise, the STM is operating in a constant height mode.

### CRASH PROTECTION:

Use these Radio buttons to specify the STM's response when the measured signal (feedback z in constant current mode ) is with a certain percentage of the maximum (usually 10 V) or minimum. The options are:

DO NOTHING: The program ignores the situation (Crash protection is off).

STOP SCANNING: The program terminates the scan.

CHANGE Z OFFSET A LITTLE: The program changes z offset so that feedback z is three times the specified tolerance away from the limit.

FIND MIN/MAX...: The program adjusts z offset so that feedback z is centered (around 0) rather than near the limit. The program keeps track of the change and insures that the data is continuous.

The finally option is the safest.

### READ SEQUENCE:

The read sequence establishes what the STM will do each time it is at a position where data will be taken. The program will carry out up to 1000 sequences at each position where data will be taken. Use the checkboxes for turning the FEEDBACK and DITHERING on and off during a sequence, use the editable text fields to WAIT during a sequence, and use the check box to RECORD data during a sequence. Use the text fields to the right of the RECORD CHANNEL check box to set which input channel and number of times to record.

NEXT and PREV: Use these buttons to move through the sequences.

ADD: Use this button to append a new sequence.

INSERT: Use this button to add a new sequence right after the one that is being viewed.

DELETE: Use this button to remove the sequence that is currently being viewed.

START SCAN: (see the START SCAN button in the SCAN dialog)

### BUTTONS THAT OPEN OTHER DIALOGS:

CONTROLS: Use this button to open the IO FUNCTIONS dialog.

SUMMARY: Use this button to open the SEQUENCE SUMMARY dialog.

**Prescan 2**

---

**Offset Overshoot Drift Compensation**

☐ Fast Dir.    ☒ Slow Dir.

Overshoot by  %

Then wait  s

Come back and wait  s

---

Temperature:  K

**Sample Type**

None

---

**Dosed With**

L

---

**Ranges**

X    ☐ ±5 V    ☒ ±10 V  
 Y    ☐ ±5 V    ☒ ±10 V  
 Z    ☐ ±5 V    ☒ ±10 V

---

**Offset Ranges**

X    ☐ ±5 V    ☒ ±10 V  
 Y    ☐ ±5 V    ☒ ±10 V  
 Z    ☐ ±5 V    ☒ ±10 V

Use the PRE-SCAN2 dialog to adjust a few scan parameters that didn't fit in the PRE-SCAN CONTROLS dialog.

#### OFFSET OVERSHOOT:

One way to combat the creep in the piezotubes is with offset overshoot. At the start of a scan, the program instructs the tip to overshoot the starting location (lower left corner of SCAN dialog), wait and then move back to the starting location.

**FAST DIR and SLOW DIR:** Check these boxes to instruct the program to overshoot in the corresponding directions before scanning.

**OVERSHOOT BY:** Use this text field to set the amount by which to overshoot the starting location.

**WAIT:** Use this text field to set the time to wait before coming back to starting location.

**COMEBACK AND WAIT:** Use this text field to set the time to wait at the starting location before continuing on with the other stages of the scan.

#### PHYSICAL PARAMETERS:

**TEMPERATURE:** Use this text field to indicate the temperature of the STM. This value is saved along with the data.

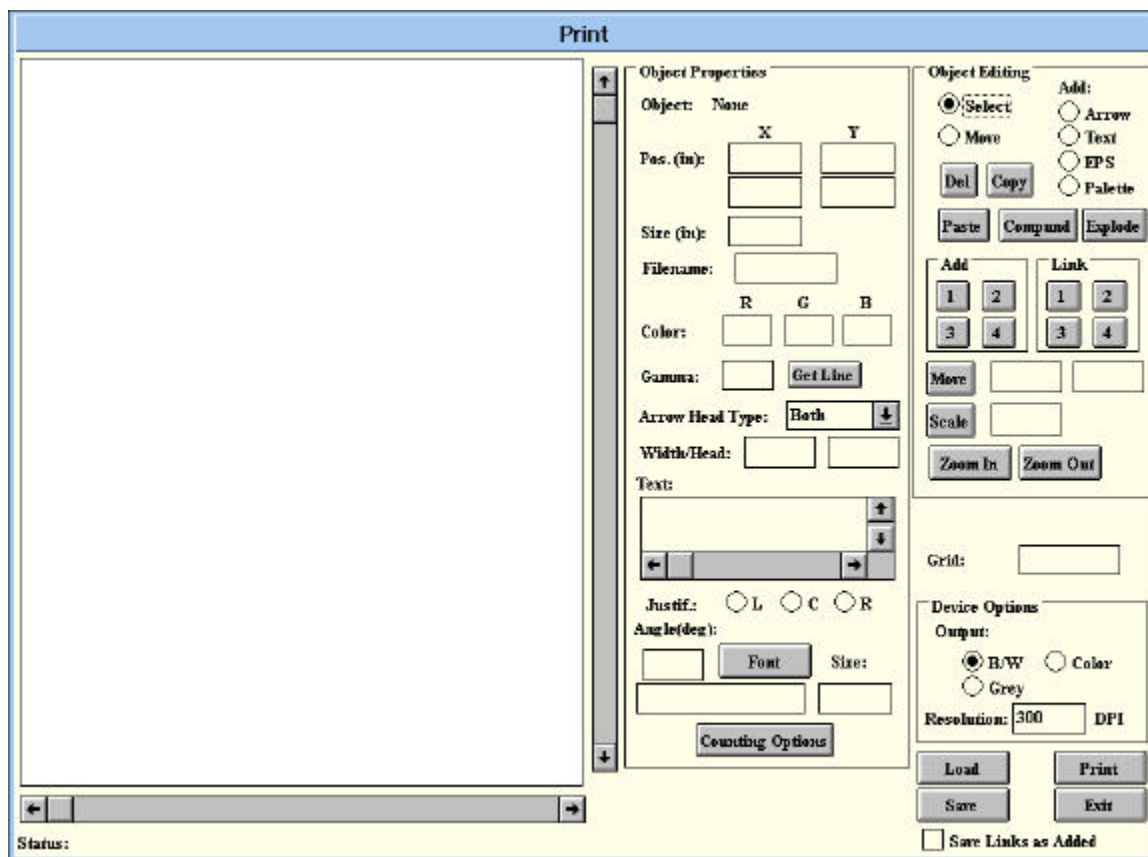
**SAMPLE TYPE and DOSED WITH:** Use these listboxes to choose the sample type and the type of dosing. If you are using a sample that is not listed, the SAMPLES.INI file must be edited in order for a new sample to appear on the list. If you are dosing with a molecule that is not listed, the DOSED.INI file must be edited for a new dosing molecule to appear on the list. One way to do this is via DOS: "edit samples.ini" and "edit dosed.ini".

RANGES:

X, Y, Z: X and Y refer to the fine x and fine y signals, respectively. Z refers to the zo signal. See the IO FUNCTIONS dialog for a more complete description of these signals. Use the +/- 5 V and +/- 10 V radio buttons to adjust the ranges.

OFFSET RANGES:

X, Y, Z: Here X, Y, and Z refer to the x offset, y offset, and z offset signals, respectively. Use these radio buttons in a manner analagous to the fine coordinate ranges.



Use the PRINT dialog to arrange, print and add text to scanned images.

The white area on the left half of the screen is for creating a layout of the page you wish to print. To create a layout, add objects to this area. Currently, you can add three types of objects: images from the data manipulation window, arrows and text.

#### OBJECT EDITING:

These features are used to add, edit and move objects in the layout.

**SELECT, MOVE:** Use this radio buttons to choose between two possible modes for using the mouse in the layout. In the SELECT mode, click the left mouse button on an object to select it. A border around an object indicates that it has been selected. Multiple objects may be selected at the same time by holding the shift key down while making selections. In the MOVE mode, drag the selected object(s) around with the left mouse button down.

**ADD TEXT:** When the TEXT radio button is pressed, click the left mouse button in the layout to add a text object to the layout. If you haven't yet specified a font, the program will automatically ask which font you wish to use. The location of the text depends on the JUSTIFICATION. The program will display left justified text so that the text will begin where you clicked. The program will display center justified text so that the text will be centered where you clicked. The program will display right justified text so that the text will end where you clicked.



**ADD ARROW:** When the ARROW radio button is pressed, click the left mouse button in the layout to add arrow objects. The beginning of the arrow is determined by the location where you push the left mouse button and the end of the arrow is determined by the location where you release the left mouse button.

**ADD EPS:** At this time, the program can't add Encapsulated Postscript file (EPS) objects.

**ADD PALETTE:** At this time, the program can't add palette objects.

**ADD 1,2,3,4:** Use these buttons to copy an image object from the DATA MANIPULATION dialog and place it in the lower left corner of the layout. Grids, counting dots, and the palette are all imported from the DATA MANIPULATION dialog.

**LINK 1,2,3,4:** Use these buttons to add an image object to the layout that is linked to an image in the DATA MANIPULATION dialog. When changes are made to the image in the DATA MANIPULATION dialog, the changes are reflected in the layout. For example, say you add a linked image object to the window by pressing LINK 3. If you then exit the PRINT dialog, change the palette of image 3 in the DATA MANIPULATION dialog, and then return to the PRINT dialog, the palette of the image object in the layout will also change.

**DEL, COPY, PASTE:** Use these buttons for deleting, copying and pasting objects. They can be used in the same way as the corresponding Edit menu items in a word processor program.

**ZOOM IN and ZOOM OUT:** Use these buttons to zoom in or out of the layout. Zooming in activates the scroll bars for the page layout.

**COMPOUND, SCALE, EXPLODE, MOVE:** These features are not currently implemented.

### OBJECT PROPERTIES:

The program displays the properties of the currently selected object in this portion of the dialog. At the very top of this portion of the dialog, the program displays the type of OBJECT. Some properties apply to only one type of object.

**POS:** For an image object, the program displays the x,y coordinates of the lower left corner of the object. For a text object, the program displays the x,y coordinates of the click where the text was added. For an arrow object, the program displays the two sets of x,y coordinates: one for the beginning of the arrow and one for the end. All x,y coordinates are given in inches, with the lower left corner of the layout as the origin.

**SIZE:** (Image objects) The program displays the length of one side of the image object in inches.

**FILENAME:** (Image objects) The program displays the file where data corresponding to an image object has been saved.

**R, G, B:** (Text and Arrow objects) Use these text fields to specify the color of text or an arrow in RGB format.

**ARROW HEAD TYPE:** (Arrow objects) Use this listbox to select the location of arrow heads on an arrow. START puts arrow head at the beginning point, END points an arrow head at the end point, BOTH puts an arrow head at both points, and NONE turns the arrow into a line segment (A camel with no humps is a horse).

**WIDTH and HEAD:** (Arrow objects) Use these text fields to establish the size of an arrow object. The **WIDTH** describes the thickness of the line segment portion of the arrow. The **HEAD** describes the size of the base of the triangle that forms the arrow head. Both quantities are in units of inches.

**TEXT:** (Text objects) Type the text in here.

**JUSTIFICATION:** (Text objects) Use these three radio buttons to choose between text that is left justified (L), center justified (C) and right justified (R).

**ANGLE:** (Text objects) The program displays the angle of the text in this field.

**FONT:** (Text objects) Use the button to select the font and angle. The program displays the font in the nearby field.

**COUNTING OPTIONS:** This button does nothing.

**DEVICE OPTIONS:**

**B/W, GREY, COLOR:** Use these radio buttons to indicate what kind of output you will print.

**RESOLUTION:** Use this text field to set the resolution with which you will print. Clearly, these choices will be affected by what kind of printers are available.

**FILES:**

**PRINT:** Use this button to create a Postscript file from the layout.

**LOAD and SAVE:** Use these buttons to load and save layouts (\*.fig files). If you have not completed a layout, but wish to save your work, use the **SAVE** button to save it. These files are also useful for creating templates for certain formats of figures.

**SAVE LINKS AS ADDED:** Check this box to instruct the program to save the files that correspond to linked objects along with \*.fig files.

**Pulse Options**

**Pulse Duration:**  
 Max. dur.: ☐  $\mu$ s ☒ ms  
  
☐ Fixed  
☐ Wait for change in Z  
☒ Wait for change in I  
 Ignore Initial   $\mu$ s  
 Change Time   $\mu$ s  
 Average Pts   
 After Time   $\mu$ s  
 % Change

**Feedback**  
☐ On ☒ Off  
☐ Toggle Feedback after  
 Wait   $\mu$ s

**Feedback Freq.**  
☒ 1 kHz ☐ 10 kHz  
☐ 100 kHz

**Delta Z-offset**  
 Bits Voltage Angstroms  
    
   
☐ Crash Protect ☐ After  
☒ Use movement ☐ None  
 bias when moving. ☐ Auto  
☐ Back  
☐ Measure Current Once:  
 Wait   $\mu$ s

**Pulse Speeds:**  
 Bias (V/ms): ☒ Step  
    
 Current (V/ms):     
 Z-Offset (V/ms):

**Pulse Measurements**  
☒ Measure Current  
☐ Measure Z  
 Measure before and after:  
  $\mu$ s    
 Avg. every  pts.

**Repeated Pulses**  
 Number of Pulses:  
    
 Delay Between Pulses:  
  $\mu$ s

**Exit**

Use the PULSE OPTIONS dialog enables to adjust the settings for a voltage pulse applied from the SELECTIVE DEPOSITION dialog.

#### PULSE DURATION:

**FIXED and WAIT...:** Use these radio buttons to choose a pulse that lasts for a fixed amount of time or a pulse that lasts until a measured quantity (Z or I) changes.

**MAX. DUR.:** If the pulse will last for a FIXED amount of time, use this text field to set the duration. If the pulse will *not* last for a FIXED amount of time, use this field to set the maximum time to wait for a change. Use the ms and  $\mu$ s radio buttons to choose the units.

If a pulse is not FIXED, you must set several parameters: IGNORE INITIAL, CHANGE TIME, AVERAGE PTS, AFTER TIME and % CHANGE.

**IGNORE INITIAL:** Use this text field to specify the minimum amount of time that the pulse will last. Even if Z or I changes during this initial period, the program will not terminate the pulse.

**CHANGE TIME, % CHANGE:** Use these text fields to specify how rapidly the measured quantity must change in order for the program to terminate the pulse. The program will terminate the pulse if the measured quantity is changed by % CHANGE percent within a time window of CHANGE TIME.

**AFTER TIME:** Use this button to establish how much time will pass between the time the change is observed and the time when the pulse is actually terminated.

**AVERAGE PTS:** Use this button to set the number of points that will be sampled and averaged to determine whether or not the measured quantity has changed. This will prevent a noisy signal from appearing to be a change in the measured quantity.

**PULSE SPEEDS:**

**STEP:** Check this box if the bias should step directly up to the pulse voltage. Specify the pulse voltage in the WRITING text field for bias in the SELECTIVE DEPOSITION dialog. If you don't check this box, the bias will ramp up to the pulse voltage.

**BIAS SPEED:** Use this text field or scroll bar to set the rate that the voltage will ramp up. This field is not relevant if the STEP box is checked.

**CURRENT SPEED, Z-OFFSET SPEED:** The program does not utilize these parameters.

**FEEDBACK:**

**ON and OFF:** Use these radio buttons to turn the feedback on or off at the start of the pulse. If the TOGGLE AFTER box is not checked, the feedback will remain in this state for the duration of the pulse.

**TOGGLE AFTER:** Use this button to toggle the feedback during the pulse. The feedback will be toggled after WAIT  $\mu$ s have passed since the start of the pulse.

**1, 10, 100 kHz:** Use these radio buttons to specify the feedback frequency. This is the same setting as the one labeled FREQUENCY in the PRE-SCAN CONTROLS dialog. It is a product of the Z and Z2 gains. Note that the program does *not* execute an AUTO- command before changing the gain setting.

**PULSE MEASUREMENTS:**

**MEASURE CURRENT:** Check this box to measure the current during the pulse.

**MEASURE Z:** Check this box to measure z during the pulse.

**AVG EVERY [n] PTS:** Use the text field to specify the number of points to average over, i.e. the program will sample the measured quantity n times and average the samples.

**MEASURE BEFORE & AFTER:** The program does not utilize this text field or scroll bar.

**DELTA Z OFFSET:**

**Z OFFSET:** Use the scroll bar to indicate the amount which the z offset should change after the feedback is toggled. The program displays this amount in units of bits, voltage and angstroms.

**CRASH PROTECT:** Check this box if you would like crash protection to be on as the z offset is changed.

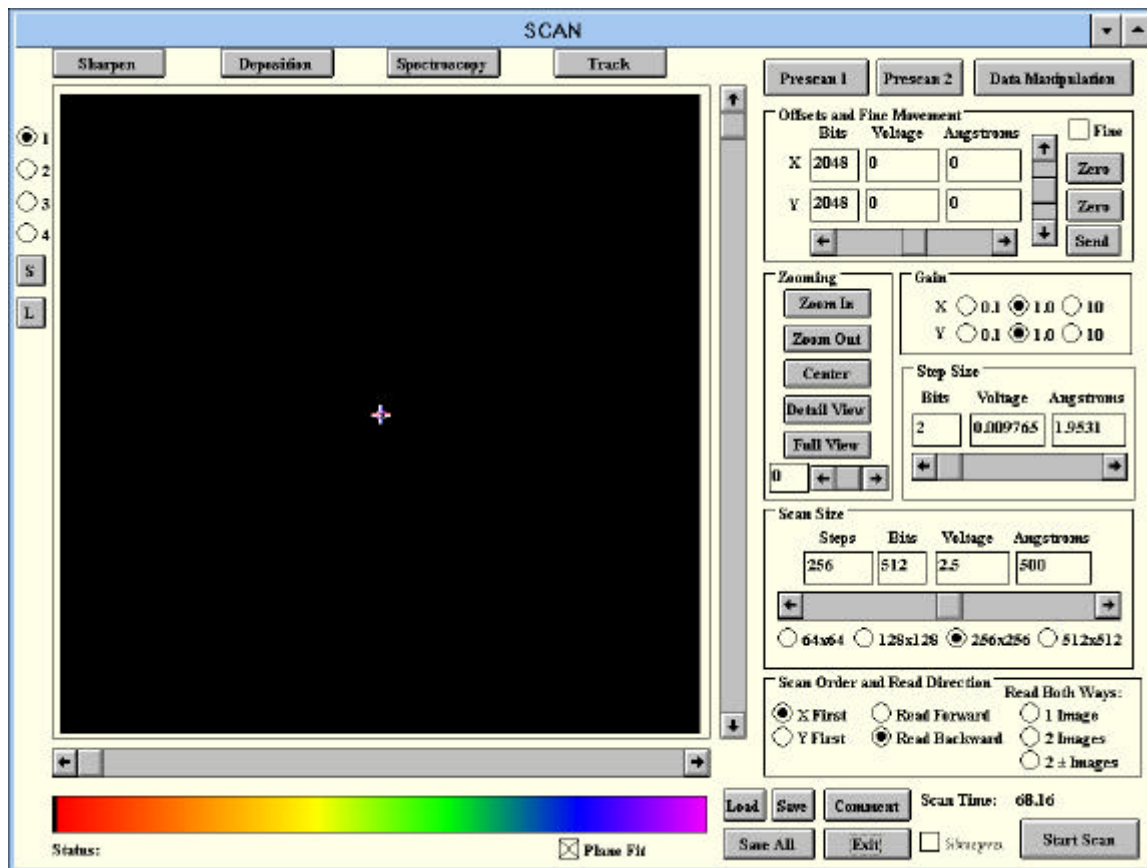
USE MOVEMENT...: Check this box to have the tip-sample bias set by the MOVEMENT text field for bias in the SELECTIVE DEPOSITION dialog. Note that for the CRASH default options, the movement bias is zero and the z offset is quite large.

NONE, AUTO, BACK: The program does not use these parameters.

MEASURE CURRENT ONCE: Check this box to have the current measured once during the voltage pulse. Use the text field to specify how long to wait before making the measurement. The wait is measured from the start of the pulse. The measured value (a voltage) is placed in the AVG CURRENT text field of the SELECTIVE DEPOSITION dialog.

REPEATED PULSES:

The repeated pulses feature is not implemented. The program does not use the NUMBER OF PULSES or DELAY BETWEEN PULSES.



Use the SCAN dialog to adjust scan settings, scan a surface or view the current scanned data.

#### OFFSETS:

**TEXT FIELDS:** Use these text fields or scroll bars to set the x and y offset destination coordinates. These coordinates refer to the center of the area that will be scanned and are given in bits, volts, and angstroms.

As the coordinates are changed, the outline of the scan area will shift. However, the tip does not actually move until the you press the SEND button. This is relevant during dosing, tip crashing and other activities where you move the tip away from the area that you are investigating.

**SEND:** Use this button to send the tip to the destination specified in the text fields. If you forget to send before scanning, the program will send the tip for you.

**ZERO:** Use these buttons to reset the x or y coordinate to zero.

**FINE:** Check this box to control fine x,y motion instead of x,y offsets

#### GAIN:

Use these radio buttons to set the gain on the fine x and fine y signals. In practice to date, the gains have been set to the same value: (.1,.1), (1,1) or (10,10).

#### STEP SIZE:

Use this text field and scroll bar to set the step size. The step size is the distance moved by the tip between measurements. The value is given in units of bits, volts, and angstroms.

#### SCAN SIZE:

64x64, etc.: Use these radio buttons to set the scan size. The scan size is determined by one parameter: the length of one side of a scan. Currently, all scans are square. The available sizes are: 64x64 steps, 128x128 steps, 256x256 steps, and 512x512 steps. The program displays the scan size in units of steps, bits, volts, and angstroms.

Note that the step size, scan size, and gain settings are interdependent. Changing one may introduce changes in the others. As you adjust these parameters, the SCAN TIME will also change. The SCAN TIME is given in seconds and represents an educated guess at how long the scan will take.

#### SCAN ORDER AND READ DIRECTION:

There are two possible settings for the scan order: X FIRST and Y FIRST. For X FIRST, the tip moves back and forth across the entire range of x coordinates, takes a small step in the y direction, then moves back and forth in the x direction, and so on, until the scan is completed. Y FIRST is analogous. The "first" direction is referred to as the "fast direction" and the other direction is referred to as the "slow direction."

As the tip moves back and forth in the fast direction, data may be read as the tip is moving forward, as it moving backward or during both stages of the motion. If data are being read during both stages of the motion, the scan can generate one composite set of data (1 IMAGE) or two separate sets of data (2 IMAGES). If two separate sets of data are obtained, it is possible to take the second one with a sample bias that is the opposite polarity from the first set (2 +- IMAGES).

The labels "1 IMAGE", "2 IMAGES", etc. are somewhat misleading, since in principle, up to one thousand sets of data can be taken on any one scan using the READ SUMMARY in the PRE-SCAN CONTROLS dialog.

### SCANNING:

**START SCAN:** Use this button to begin a scan. The algorithm for the scan is roughly:

- (1) Set the sample bias.
- (2) Set the tunneling current.
- (3) Set the piezo mode to coarse-translation.
- (4) Start dithering of channels 0 & 1, if necessary.
- (5) Zero the fine x and fine y, i.e. move them to the center of the area to be imaged.
- (6) "Synch Offsets" (In case the user forgot to hit "Send"). Set the offset x and offset y (coarse position controls) and the z offset.
- (7) Move fine x and fine y to the starting position (lower, left corner of area to be imaged, as it appears on screen). With the slow coordinate, overshoot the image edge and then move back. Do the same with the fast coordinate. The overshoot settings are set in the PRESCAN 2 dialog. This is intended to combat creep in the piezos.
- (8) The tip moves forward in the fast direction, taking measurements if appropriate.
- (9) If data is being taken in both directions to form one composite image, the tip is moved over one pixel in the slow direction. If two polar sets of data (2 +- IMAGES) are being obtained, flip the bias.
- (10) The tip moves backward in the fast direction, taking measurements if appropriate.
- (11) The tip is moved over one pixel in the slow direction.
- (12) If two polar sets of data (2 +- IMAGES) are being obtained, flip the bias.
- (13) Repeat until the entire image is scanned.

If you are taking measurements in both directions to produce one composite set of data, then the tip will raster back and forth across the sample. For any other read direction, the tip will trace out a "comb-like" pattern: up,down,over,up,down,over,etc.

The program will display the data as the scan is performed. Note that the palette for the display is sometimes updated during the scan. At the end of the scan, the program will adjust the palette a final time and, if the PLANE FIT checkbox is checked, a plane will be fit to the data (see FIT PLANE in the DATA MANIPULATION dialog). The program repaints the entire image when the scan is complete.

Use the ESC key to terminate a scan.

### LOADING AND SAVING DATA AND SETTINGS:

**LOAD and SAVE:** Use these buttons to load or save a set of 3D data (\*.stm files).

**SAVE ALL:** Use this button to save all sets of data that were read in during the most recent scan.

**1,2,3,4:** The program stores four sets of scan settings. Use these radio buttons to switch between these sets of scan settings.

**S and L:** Use these buttons to save and load scan settings (\*.stp files).

### ZOOMING:

**ZOOM IN and ZOOM OUT:** Use these buttons to zoom in or out of the currently selected image by a factor of 2.

**CENTER:** Use this button to repaint the screen so that the outline of the scan area lies in the center of the display.

**FULL VIEW:** Use this button to repaint the screen so that the display shows the entire area within the STM's range of motion.



**DETAIL VIEW:** Use this button to repaint the screen so that the scan area fills the entire display. In other words, the program will move the outline of the scan area the edge of the display.

**SCROLL BAR:** Use the unlabeled scroll bar beneath the FULL VIEW button to cycle through data sets that were taken during the most recent scan. The program will display the image for the current data set.

**BUTTONS THAT OPEN OTHER DIALOGS:**

**SHARPEN:** Use this button to open the SHARPEN TIP dialog.

**DEPOSITION:** Use this button to open the SELECTIVE DEPOSITION dialog.

**SPECTROSCOPY:** Use this button to open the SPECTROSCOPY dialog.

**TRACK:** Use this button to open the TRACK dialog.

**PRESCAN1:** Use this button to open the PRE-SCAN CONTROLS dialog.

**PRESCAN2:** Use this button to open the second PRESCAN 2 dialog.

**DATA MANIPULATION:** Use this button to open the DATA MANIPULATION dialog.

**COMMENT:** Use this button to open the COMMENTS dialog. The program saves comments added here along with 3D data (\*.stm files).

**Selection Functions**

**First Point**  
 X Y Z  
 [ ] [ ] [ ]  
 [ ] [ ] [ ] V  
 [ ] [ ] [ ] [ ]

**Selection Type**  
☒ Points  
☐ Line  
☐ Square  
☐ Circle

**Second Point**  
 X Y Z  
 [ ] [ ] [ ]  
 [ ] [ ] [ ] V  
 [ ] [ ] [ ] [ ]

☐ Constrain Transforms to Region

Sel 1/2 1 Lock  
 Min Max  
 Cut Copy  
 Invert Exit

Use the SELECTION FUNCTIONS dialog to select points, lines, square regions, and circular regions in an image that is displayed in the DATA MANIPULATION dialog.

#### SELECTION 1 AND SELECTION 2:

You can make two different selections in one image. Click the left mouse button in the DATA MANIPULATION dialog to make Selection 1. Click the right mouse button in the DATA MANIPULATION dialog to make Selection 2.

**SEL 1/2 BUTTON:** Use this button to toggle between the two different selections.

#### FIRST POINT AND SECOND POINT:

The program displays the coordinates of the selection in editable text fields in the upper left hand corner of the dialog. The program displays the coordinates in units of pixels and volts. The type of selection dictates the how the coordinates should be interpreted.

**POINT coordinates:** The program displays a Selection 1 point coordinate in the FIRST POINT text fields. The program displays a Selection 2 point coordinate in the SECOND POINT text fields. Thus, if both selections are POINTs, the program will display the coordinates of both selections.

If one of the Selections is a line, square, or circle, click on SEL 1/2 to toggle between the two sets of coordinates.

**LINE coordinates:** The program displays one endpoint of the line in FIRST POINT and the other endpoint in SECOND POINT.

**SQUARE coordinates:** The program displays the coordinates of the upper left hand corner in FIRST POINT and the coordinates of the lower right hand corner in SECOND POINT.

**CIRCLE:** The program displays the center of the circle in FIRST POINT and the radius of the circle in SECOND POINT.

The ANALYSIS FUNCTIONS dialog can be used in conjunction with the SELECTION FUNCTIONS dialog to compare sets of coordinates.

**SELECTION TYPE RADIO BUTTONS:**

POINT, LINE, SQUARE, CIRCLE: Use these radio buttons to set the types of selection 1 and selection 2.

For 2D data, line, square and circle selections are not available.

**EDITING SELECTIONS:**

COPY: Use this button to copy the selected data to quadrant 4. This operation only works for square selected regions of 3D data that are in quadrant 1,2 or 3.

MIN and MAX: Use these buttons to move the FIRST POINT to a point in the data where z is a maximum or minimum.

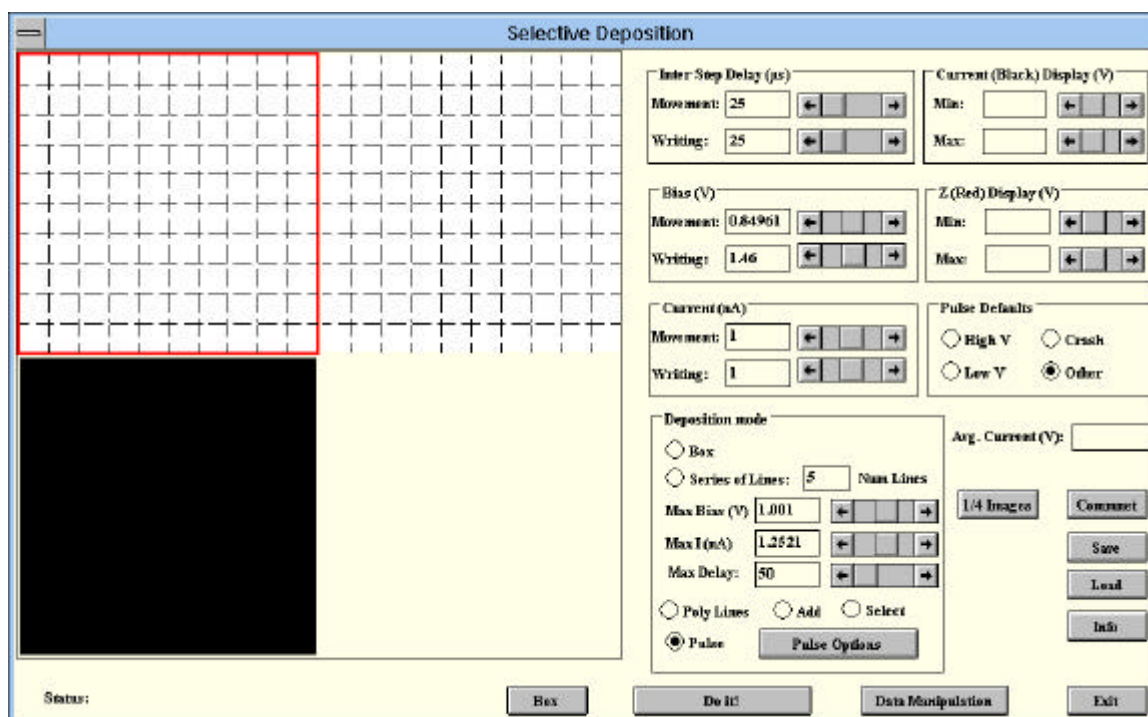
LOCK: This button does nothing.

CUT: Use this button to cut out selected data. If the selection is a line in 3D data in quadrant 1,2 or 3, the program will create a set of 2D data in quadrant 4 (a cross-section). 2D data created in this way can be saved (\*.cut files). If the selection is a square region of 3D data in quadrant 1,2 or 3 the selected area is cleared and the program displays the cut data in quadrant 4.

INVERT: Use this button to replace the value of z at each point outside the selected region by -z. Note that inverting twice will return the data to its original state. This operation is valid for square selected regions of 3D data that are in quadrant 1,2 or 3.

**CONSTRAIN TRANSFORMS TO REGION:**

If a square or circular selection has been made, check this box to limit the operations performed in the DATA MANIPULATION dialog to the selected region.



The SELECTIVE DEPOSITION dialog was originally intended to “draw” shapes along a scanned surface. Currently, we use this dialog to apply voltage pulses and to crash the tip.

#### DISPLAY:

**1/4 IMAGES:** The program can display one image or four image quadrants at one time. Use this button to toggle between the two modes.

In the four image quadrant mode, the program will display the image from the SCAN dialog in quadrant 3. The program will display data taken during a pulse in quadrant 1 and quadrant 2. You can load deposition data (\*.dep files) into quadrant 1 or quadrant 2.

#### DEPOSITION MODE:

**BOX:** Use this radio button to enter box mode. In box mode, the tip will trace out a square box on the scanned area shown in quadrant 3.

**SERIES OF LINES:** Use this radio button to enter the series-of-lines mode. In this mode, the tip will trace out a series of equally spaced lines on the scanned area shown in quadrant 3. Use the NUM LINES text field to specify the number of lines to trace. The program can ramp the bias, current, and delay from line to line. The program uses the WRITING values of these parameters as the starting value and ramps up to the values you specify in the MAX BIAS, MAX I and MAX delay text fields.

**PULSE:** Use this radio button to enter the pulse mode. In the pulse mode, the tip remains stationary while the tip-sample bias is pulsed. Use the PULSE OPTIONS dialog to adjust the settings for pulse mode.

**POLY LINES, ADD, SELECT:** These modes are not implemented.

#### DELAY, BIAS, CURRENT PARAMETERS:

MOVEMENT, WRITING: Use these text fields or scroll bars to specify the parameters.

MOVEMENT vs. WRITING: When the program is in box mode or series-of-lines modes, you must specify two sets of parameters. The first set of parameters, MOVEMENT, applies to the tip's motion between tracing out lines and as the tip moves to the starting position. The second set of parameters, WRITING, applies to the tip's motion while it is tracing out a box or series of lines.

In pulse mode, use the WRITING text field for the bias to specify the pulse voltage. If the feedback is on in pulse mode, use the WRITING text field for the current to specify the feedback current.

#### EXECUTION:

DO IT!: Use this button to draw a box in box mode, pulse the bias in pulse mode, etc.

BOX: Use this button to trace out a box with a tip. You could accomplish the same task if you are in box mode and use the DO IT! button.

#### LOADING AND SAVING DEPOSITION DATA:

LOAD and SAVE: Use these buttons to load or save deposition data and settings (\*.dep files).

INFO: Use this button to write the deposition data, but not the settings, to a file c:/dep.data.

#### BUTTONS THAT OPEN OTHER DIALOGS:

DATA MANIPULATION: Use this button to open the DATA MANIPULATION dialog.

COMMENT: Use this dialog to open the COMMENTS dialog. If you add comments in this dialog, they will be saved within the deposition data (\*.dep files).

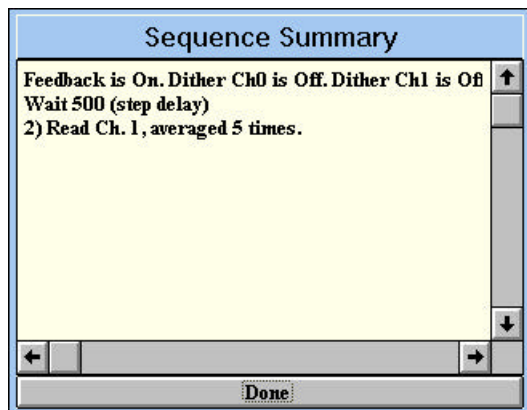
PULSE OPTIONS: Use this dialog to open the PULSE OPTIONS dialog.

#### OTHER:

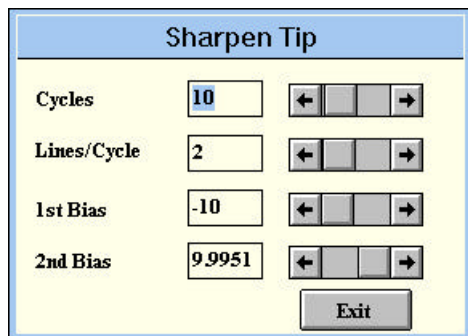
PULSE DEFAULTS: Use these radio buttons to select a set of default pulse options. There is no way to save or edit the default pulse options from within the STM program.

AVG CURRENT: If the MEASURE CURRENT ONCE box is checked in the PULSE OPTIONS dialog, the program will display the measured value of the current in this text field.

CURRENT (BLACK) DISPLAY, Z (RED) DISPLAY: The program does not utilize the MIN or MAX parameters.



Use the SEQUENCE SUMMARY dialog to view a text summary of the read sequence summary that is specified in the PRE-SCAN CONTROLS dialog. The program automatically updates this information when changes are made in the PRE-SCAN CONTROLS dialog.

The image shows a software dialog box titled "Sharpen Tip" with a blue header bar. The dialog has a yellow background and contains four rows of controls. Each row consists of a label, a text input field, and a three-button scroll bar. The labels are "Cycles", "Lines/Cycle", "1st Bias", and "2nd Bias". The input fields contain the values "10", "2", "-10", and "9.9951" respectively. The scroll bars have left, center, and right buttons. At the bottom center of the dialog is an "Exit" button.

| Sharpen Tip |        |         |
|-------------|--------|---------|
| Cycles      | 10     | ← [ ] → |
| Lines/Cycle | 2      | ← [ ] → |
| 1st Bias    | -10    | ← [ ] → |
| 2nd Bias    | 9.9951 | ← [ ] → |
| Exit        |        |         |

Use the SHARPEN TIP dialog to display the parameters for the tip sharpening procedure executed from the SCAN dialog. Currently, the sharpening feature is disabled and the SHARPEN check box in the SCAN dialog is greyed out.

If the SHARPEN box is checked in the SCAN dialog, the tip-sample bias will be switched as the tip moves back and forth across the surface in place of a usual SCAN. Use the text fields or scroll bars in this dialog to set the number of CYCLES of bias switching, the number of LINES/CYCLE to be scanned and the values of the 1ST BIAS and 2ND BIAS.

Spectroscopy. No Data.

**Pointer 1 (Blue)**

Voltage: -10.000

Current:

**Find Peak**

**Settings**

☒ 1 ☐ 2

☐ 3 ☐ 4

**Pointer 2 (Red)**

Voltage: 9.995

Current:

**Find Peak**

**Current Channel**

0

**Vertical Scale**

Max: 10.000

Min: -10.000

**Set to Max** **Center** **Set to Min**

**Pre Amp Scale:** 9

**Prescan**

**Options** **Lock-in Param.**

**Horizontal Scale**

**Zoom In** **Zoom Out**

**Scan/Stop**

**Clear** **Clear All**

**Save** **Load**

**Comment**

**Exit**

**Image Relative coordinates:**

X:  Y:

**Image:**

**Status:**

Bias: 1.0010 V      Scan Time: s

Use the SPECTROSCOPY dialog to measure the input channels as a function of output bias (V). Up to 8 sets of data can be taken at once (see the SPECTROSCOPY OPTIONS dialog for the types of data that can be taken and for more information about spectroscopic scans). The program graphs the sets of data versus V in the upper left corner of the dialog. If appropriate, the program displays the scanned image of the local surface in the lower left corner of the dialog.

**CURRENT CHANNEL:** Use this scroll bar to move through the eight sets of data.

#### POINTER 1 and POINTER 2:

The program can move two pointers along the data set corresponding to the CURRENT CHANNEL. They appear as vertical red and blue lines on the graph. Use the VOLTAGE text field or scroll bar to move along the data set. The program will display the corresponding CURRENT. The voltage can also be set by clicking on the graph with the left or right mouse button.

**FIND PEAK:** These buttons do nothing.



#### VERTICAL SCALE:

These functions are also available in the VERTICAL SCALE dialog.

MAX and MIN: Use these text fields or scroll bars to set the values of the vertical coordinate at the top and bottom edges of the graph.

SET TO MAX and SET TO MIN: Use these buttons to set MAX/MIN to correspond to the global maximum/minimum of the data.

CENTER: Use this button to center the data on the vertical scale. The program finds out whether the global maximum or the global minimum of the data has the largest magnitude. The program sets MAX and MIN to be +(this value) and -(this value).

#### HORIZONTAL SCALE:

ZOOM IN and ZOOM OUT: Use these buttons to zoom into or out of the data by a factor of 2. When you have zoomed into the data, use the scroll bar to scroll along the V axis of the graph.

#### IMAGE RELATIVE COORDINATES:

X,Y: The program displays the coordinates of the tip position. The program displays the coordinates in units of pixels, relative to the lower, left edge of the scanned image.

IMAGE: The program displays the name of the image or corresponding filename.

#### BUTTONS THAT OPEN OTHER DIALOGS:

PRESCAN: Use this button to open the PRESCAN CONTROLS dialog.

OPTIONS: Use this button to open the SPECTROSCOPY OPTIONS dialog.

LOCKIN PARAMS: Use this button to open the LOCKIN SETTINGS dialog.

COMMENT: Use this button to open the COMMENTS dialog. The program saves comments added here along with spectroscopy data (\*.spc files).

#### OTHER CONTROLS:

SETTINGS 1,2,3,4: Use these four radio buttons to load default scan options. Use the OPTIONS dialog to edit the sets of options. The program automatically saves the options when the SAVE or EXIT button is pressed.

PRE-AMP SCALE: The program does not use this setting. The pre-amp scale must be adjusted manually.

SAVE and LOAD: Use these buttons to save and load spectroscopy data (\*.spc files). If more than one set of data has been acquired, the program will prompt you to determine which of these sets of data should be saved.

CLEAR: Use this button to clear the data for the CURRENT CHANNEL.

CLEAR ALL: Use this button to clear the data for all eight channels.

SCAN and STOP: Use this button to start and stop taking data. Use the SPECTROSCOPY OPTIONS dialog to set the parameters for a spectroscopic scan.

Use the SPECTROSCOPY OPTIONS dialog for setting the parameters for a spectroscopic scan.

#### OUTPUT:

CHANNEL: Use this text field to set the channel for voltage output.

MODE: Use these radio buttons to indicate how the voltage range is specified. In the ABSOLUTE mode, the voltage range is specified explicitly. In the RELATIVE mode, the scan range is specified relative to the initial voltage of the output channel.

SCAN RANGE MIN/MAX: Use these text fields and scroll bars to set the scanning range.

#### MEASUREMENTS:

Each of the four input channels can be sampled during a spectroscopic scan. Measurements can be made while the voltage is increasing from minimum to maximum (the forward direction) or while the voltage is decreasing from maximum to minimum (the backward direction). Each trip forward and then back is one pass. Sampling all four channels in both directions would give eight sets of data.

MEASURE I, Z, CH 2, CH 3: Check these boxes if you want to sample the corresponding input channels.

SCAN DIR: Check these boxes to indicate when measurements should be made. If measurements are made in the FORWARD direction, the data can be seen in channels 0-3 in the SPECTROSCOPY dialog. If measurements are made in the BACKWARD direction, the data can be seen in channels 4-7 in the SPECTROSCOPY dialog. If measurements are made in the both directions and averaged (AVG BOTH), the data can be seen in channels 0-3 in the SPECTROSCOPY dialog.

MOVE DELAY: Use this text field to set the delay between each increment of V.

READ DELAY: The program doesn't use this setting.

BITS/STEP: Use this text field to set voltage increment between each measurement. The program will display the corresponding rate of voltage increase in V/sec.

NUMBER OF SAMPLES: Use this text field to indicate how many times the program should sample each time a data point is taken. The sampled values will be averaged to give the final data point.

NUMBER OF PASSES: Use this text field to specify the number of cycles the voltage will make forward and then back.

#### TRACKING:

Enable tracking if you want the program to track for extrema in the data periodically during the spectroscopy scan. Doing so will insure that the tip is always above an extremum, even if an adsorbed molecule moves. (See the TRACKING dialog for more information about tracking).

EVERY [n] PASSES: Use this text field to indicate how often the program should perform tracking during a spectroscopic scan. If the value of this field is zero, the program will never track.

ITERATIONS: Use this text field to specify the number of times the program will check the tunneling current at the current tip position and at the nearest eight pixels to find an extremum.

SAMPLE EVERY: Use this text field to set the delay between each set of 9 current measurements.

MAX/MIN: Use these radio buttons to toggle between searching for a max or a min in the tunneling current.

The program displays the tip position in absolute coordinates (offset x, offset y signals) and as an offset from the original tip position (fine x, fine y signals).

**MAX BITS:** Use this text field or scroll bar to limit the distance the tip can track during a spectroscopic scan.

**DELAY:** Use this text field to set the delay. This delay serves as both the STEP DELAY and INTER STEP DELAY as the tip is moved from pixel to pixel (see PRE SCAN CONTROLS dialog for explanations of these terms).

**AVG [n] DATA PTS:** Use this text field to set n. Each time the tunneling current is read, the program samples the current n times. The average of the n values is taken to be the value of the tunneling current at a given position.

**DELTA Z OFFSET:**

The program can ramp the sample bias and adjust the z offset signal before starting a spectroscopic scan. Use these text fields and scroll bars to set the corresponding values.

**DITHER DURING SCAN:**

**CH 0/ CH 1:** Use the radio buttons to have dithering ON or OFF during the spectroscopic scan.

**WAIT:** Use this text field to indicate how long the program should wait after dithering before continuing on with the spectroscopic scan.

**FEEDBACK DURING SCAN:**

**ON / OFF 1:** Use the radio buttons to have the feedback ON or OFF during the entire spectroscopic scan.

**ON EVERY [n] PASSES:** If you want the feedback to be turned on periodically during the scan, use this text field to indicate how often the program should turn the feedback on and off.

**WAIT:** Use this text field to indicate how long the program should wait after turning the feedback on or off before continuing on with the spectroscopic scan.

**CRASH PROTECTION:**

These radio buttons do nothing; they are an artifact of an older version of the program.

**Tip Approach**

**Waveform Control**

Delay: 10 ms

Acceleration: 0.2 % of g

**Sample Type**

**Dosed With**

Min Tunneling Current

Volts: 0.12238

Bits: 400

**Step Parameters**

# of Giant Steps: 5

Baby Step Size: 2

**Steps**

Number of steps: 10000

Bias: 1.001 V

Current: 1.2521 nA

Status: Idle.

Tunneled At:

Buttons: Tip Approach, Tip Retract, Tip Unretract, Oscilloscope, Controls..., Bias Square Wave, Reinit, Exit

Use the TIP APPROACH dialog to bring the tip close to the surface and establish an initial tunneling current.

Each time the TIP APPROACH dialog is opened, the following signals are set to zero: fine x, fine y, x offset, y offset, z offset, zo (see the IO FUNCTIONS dialog for a description of these signals). Additionally, feedback is turned off.

The SAMPLE TYPE and DOSED WITH listboxes currently do not work.

#### MINIMUM TUNNELING CURRENT:

Use these text fields or scroll bar to set the minimum value of the tunneling current that will end the tip approach.

#### STEP PARAMETERS:

During the tip approach, the piezotubes are in the fine-rotation mode. The sample is moved toward the tip with two different kinds of steps: giant steps and baby steps. At the outset of a giant step, the outer piezos are maximally extended in the z direction. The three outer piezos rotate the sample with a constant acceleration. Then, the three piezos stop and are lowered by zo. As the outer piezos are lowered, they rotate back to their original orientation. The sample, still rotating, falls down toward the tip. Thus, a giant step moves the piezo ends farther up or down the ramps on the bottom of the sample holder. Giant steps are followed by baby steps. The baby steps drop zo by a total of 4096 bits. During baby steps the tunneling current is measured. If the tunneling current is greater than the minimum tunneling current, tip approach is complete. Otherwise, the outer piezos are then reextended and the sample is moved through another cycle of giant steps and baby steps.

**GIANT STEPS, BABY STEP SIZE:** Use these text fields to set the number of giant steps and the size of baby steps taken. Note that baby steps always drop the sample by 4096 bits, but the number of baby steps required to achieve that depends on the baby step size.

### WAVEFORM CONTROL:

The waveform control is a set of parameters related to the giant steps.

DELAY: Use this text field to specify the amount of time to wait during a giant step after the piezotubes have stopped rotating before beginning to drop the piezotubes. Additionally, during a series of STEP UP or STEP DOWN movements, 1000 times this delay is the pause between steps.

ACCELERATION: Use this text field to specify the constant acceleration with which the sample is rotated. g refers to gravitational acceleration ( $9.8 \text{ m/s}^2$ ).

xo STEP SIZE: Use this text field to specify how far around the sample is rotated during each giant step.

zo STEP SIZE: Use this text field to specify how far the piezos drop.

UP DEFAULTS and DOWN DEFAULTS: Use these buttons to restore the default waveform.

### APPROACH AND RETRACTION:

TIP APPROACH: Use this button to start the Tip Approach.

STOP: Use this button to abort the Tip Approach.

TIP RETRACT and UNRETRACT: Use these buttons to retract and unretract the tip.

### "BLIND" STEPS

STEP UP and STEP DOWN: Use these buttons to move the sample up or down by the number of giant steps specified by the NUMBER OF STEPS text field.

### MISCELLANY:

TEMPERATURE: Use this text field to record the temperature. The program saves this information with data.

BIAS SQUARE WAVE: Use this button to send the sample bias through a sequence of square wave pulses with a minimum voltage of 0.5 volts and a maximum voltage of 1.0 volts. Use the DELAY text field to set the delay between consecutive pulses. Use the ESC key to terminate the wave.

REINIT: Use this button to reset the computer's ports and registers and to reset the output channels.

### BUTTONS THAT OPEN OTHER DIALOGS:

CONTROLS: Use this button to open the IO FUNCTIONS dialog.

OSCILLOSCOPE: Use this button to open the OSCILLOSCOPE dialog.

| Trace Analysis                                   |         |   |           |            |
|--|---------|---|-----------|------------|
| From:  | -10.000 | V   | To:       | 10.000 V   |
| This (ms):                                       | 0.000   | That (ms):  | 0.000     | Both (ms): |
|  | 0.000   |   | 0.000     | 0.000      |
| Num  | Avg ms: | Avg V:  | Bin (ms): |            |
| This: 1  | 0.000   | -10.000   | 1.000     |            |
| That: 1  | 0.000   | -10.000   | Save Bin  |            |
| This is a:                                       |         | Start as:   |           | Analyze    |
| <input checked="" type="radio"/> High            |         | <input type="radio"/> High <input checked="" type="radio"/> Undefined |           | Exit       |
| <input type="radio"/> Low                        |         | <input type="radio"/> Low   |           |            |
| <input type="checkbox"/> Reject "this" less than | 1.000   | ms  |           |            |
| <input type="checkbox"/> Reject "that" less than | 1.000   | ms  |           |            |
| Totals:  | Num     | Avg ms:   | Avg V:    |            |
| This:  | 0       | 0.000   | 0.000     |            |
| That:  | 0       | 0.000   | 0.000     |            |
|  |         |   | Add       |            |
|  |         |   | Clear     |            |

Use the TRACE ANALYSIS dialog to analyze 2D data with two or more distinct plateaus. For example, measurements of I vs. V with the tip directly over a rotating molecule will have several plateaus with the plateaus corresponding to different rotation states. With the TRACE ANALYSIS dialog, you can determine the number and average duration of each plateau.

#### TRACE PARAMETERS:

The trace analysis counts plateaus of two heights: "this" plateaus and "that" plateaus.

FROM and TO: Use these text fields to set the voltage range in which a plateau will be counted as a "this" plateau.

THIS IS A HIGH / LOW: Use this button to specify if "this" plateaus are higher or lower voltage than "that" plateaus. If "this" plateaus are HIGH, then plateaus below the specified "this" voltage range are counted as "that" plateaus. If "this" plateaus are LOW, then plateaus above the specified "this" voltage range are counted as "that" plateaus.

START AS HIGH / LOW / UNDEFINED: Use these radio buttons to specify what kind of plateau is at the left edge of the 2D data.

REJECT THIS / THAT LESS THAN... : Check these boxes to have the program disregard plateaus with short duration.

#### TRACE RESULTS:

To execute a trace analysis, use the ANALYZE button. The program will display the number, average duration and average voltage for THIS and THAT plateaus. The program will also display the total duration of each kind of plateau.

#### BINNING:

The trace analysis data can be binned and saved.

SAVE BIN: Saves four files: two sets of binned data and two sets of plateau width data. The program saves the data that corresponds to "this" plateaus with extensions .b1 and .t1. The program saves the data that corresponds to "that" plateaus with suffixes .b2 and .t2.

BIN(ms): Use this text field to set the bin width.

TOTALS:

In order to accumulate analyses, the program allows you to keep a running total of the number, average duration and average voltage of each kind of plateau.

ADD: Use this button to add the data from the most recent analysis to the running total. The program also saves the new running totals in files \*.bs1 and \*.bs2.

CLEAR: Use this button to clear the running totals.



| Track   |             |   |                                    |              |
|---|-------------|---|------------------------------------|--------------|
|   | Cur.Limits: |   | Cur.Pos.:                          | Offset       |
| X   | 1848        | 2248  | 2048                               | 0            |
| Y   | 1848        | 2248  | 2048                               | 0            |
| Max. Bits   | 200         | <input checked="" type="radio"/> Max<br><input type="radio"/> Min |                                    |              |
| Avg.  | 200         | data pts  | <input type="checkbox"/> Auto Auto |              |
| Sample Every  | 100         | $\mu$ s   | Delay                              | 1000 $\mu$ s |
| <input type="button" value="Start Here"/> <input type="button" value="Start Again"/> <input type="button" value="Stop/Offset"/> <input type="button" value="Exit"/> |             |   |                                    |              |

Use the TRACK dialog to look for a max (min) in the tunneling current. The program measures the current at a particular x,y pixel and at each of the eight neighboring pixels. The program moves the tip to the pixel with the highest (lowest) tunneling current. The program repeats the process until you instruct the program to stop or until the track hits the boundaries you have set.

MAX and MIN: Use these radio buttons toggle between searching for a max or a min in the tunneling current.

#### TRACKING LIMITS:

Tracking is restricted to a limited area, shown as a dotted outline in the SCAN dialog.

CUR. POS: The program displays the current tip position in these text fields (x offset and y offset signals). Tracking begins at the center of the outlined region.

CUR. LIMITS: The program displays the maximum and minimum values of x and y within the tracking limits.

MAX BITS: Use this text field or scroll bar to specify how far the x and y coordinates may wander relative to their value at the center of the tracking area.

OFFSET: The program displays the distance of the tip from the original starting position (change in fine x and fine y).

#### TRACKING PARAMETERS

SAMPLE EVERY: Use this text field to set the delay between each set of 9 current measurements.

DELAY: Use this text field to set the delay. This delay serves as both the STEP DELAY and INTER STEP DELAY as the tip is moved from pixel to pixel (see PRE SCAN CONTROLS dialog for explanations of these terms).

AVG [n] DATA PTS: Use this text field to set n. Each time the program reads the tunneling current, it samples the current n times. The average of the n values is the value of the tunneling current at a given position.

AUTO AUTO: Check this box to instruct the program to automatically perform AUTO Z - (see PRE SCAN CONTROLS ) during measurements.

TRACKING:

START HERE: Hitting this button begins the tracking at a point currently selected in the scan window.

START AGAIN: Starts tracking from the current position.

STOP/OFFSET: Terminates tracking.

As the program tracks, the program updates the CUR. POS. text fields to indicate the current tip location. The program also updates OFFSET text fields to reflect the position of the tip relative to the starting position.

| Vertical Scale |                      |   |   |
|----------------|----------------------|---|---|
| <b>Max:</b>    | <input type="text"/> | <input type="button" value="←"/> <input type="button" value="→"/> | <input type="button" value="Set to Max"/>   |
| <b>Min:</b>    | <input type="text"/> | <input type="button" value="←"/> <input type="button" value="→"/> | <input type="button" value="Max + Center"/> |
|                |                      |   | <input type="button" value="Set to Min"/>   |
|                |                      |   | <input type="button" value="Exit"/>         |

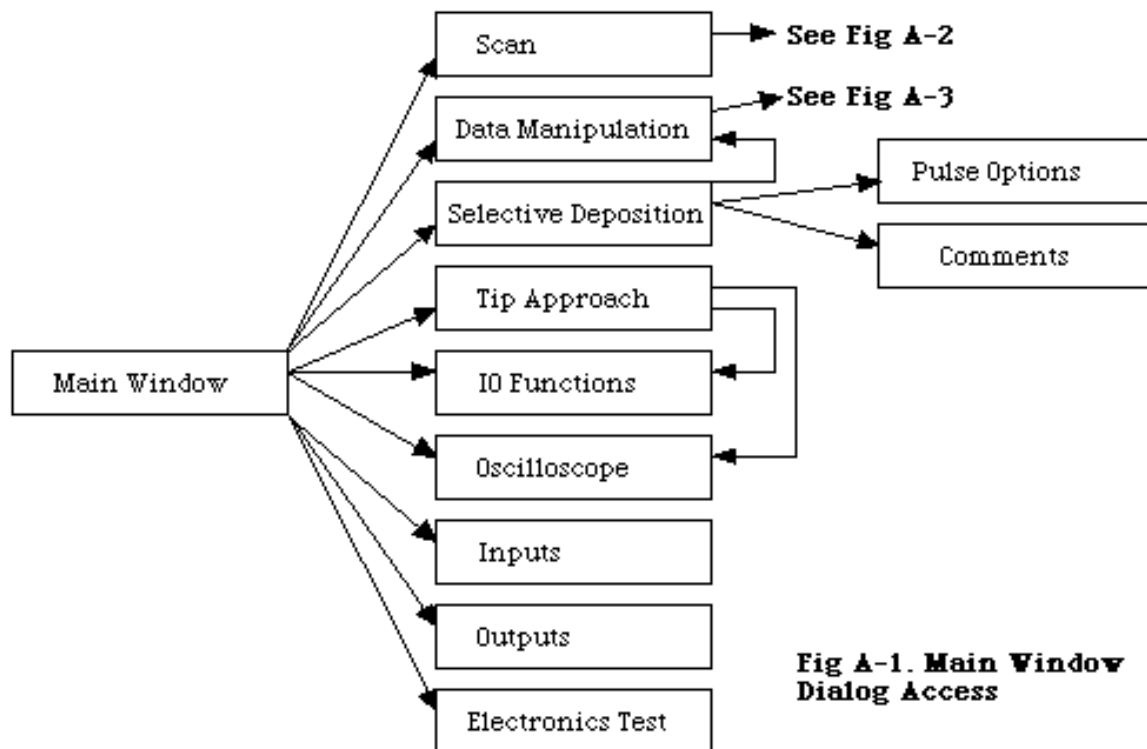
Use the VERTICAL SCALE dialog to rescale the vertical axis of 2D data.

**MAX and MIN:** Use these text fields or scroll bars to set the values of the vertical coordinate at the top and bottom edges of the graph.

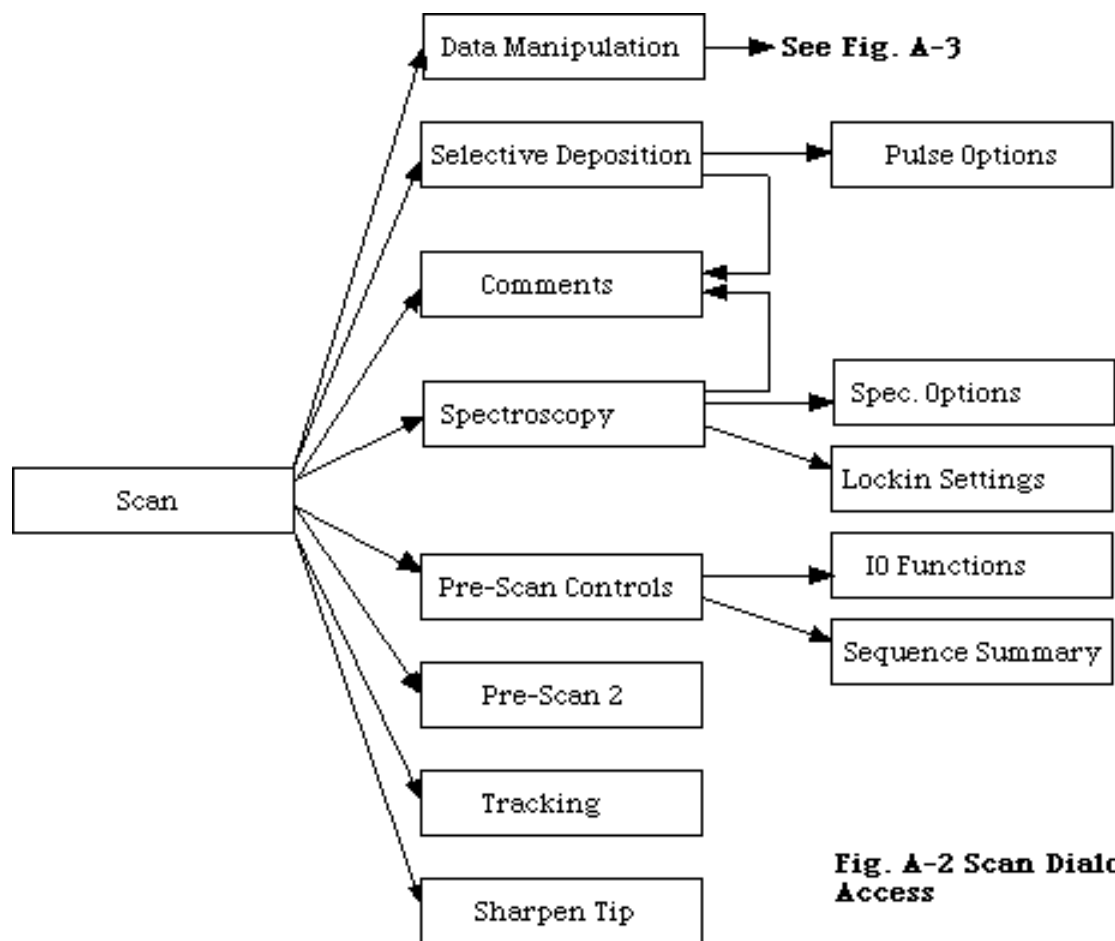
**SET TO MAX and SET TO MIN:** Use these buttons to set MAX/MIN to correspond to the global maximum/minimum of the data.

**MAX + CENTER:** Use this button to center the data on the vertical scale. The program finds out whether the global maximum or the global minimum of the data has the largest magnitude. The program sets MAX and MIN to be +(this value) and -(this value).

## Appendix

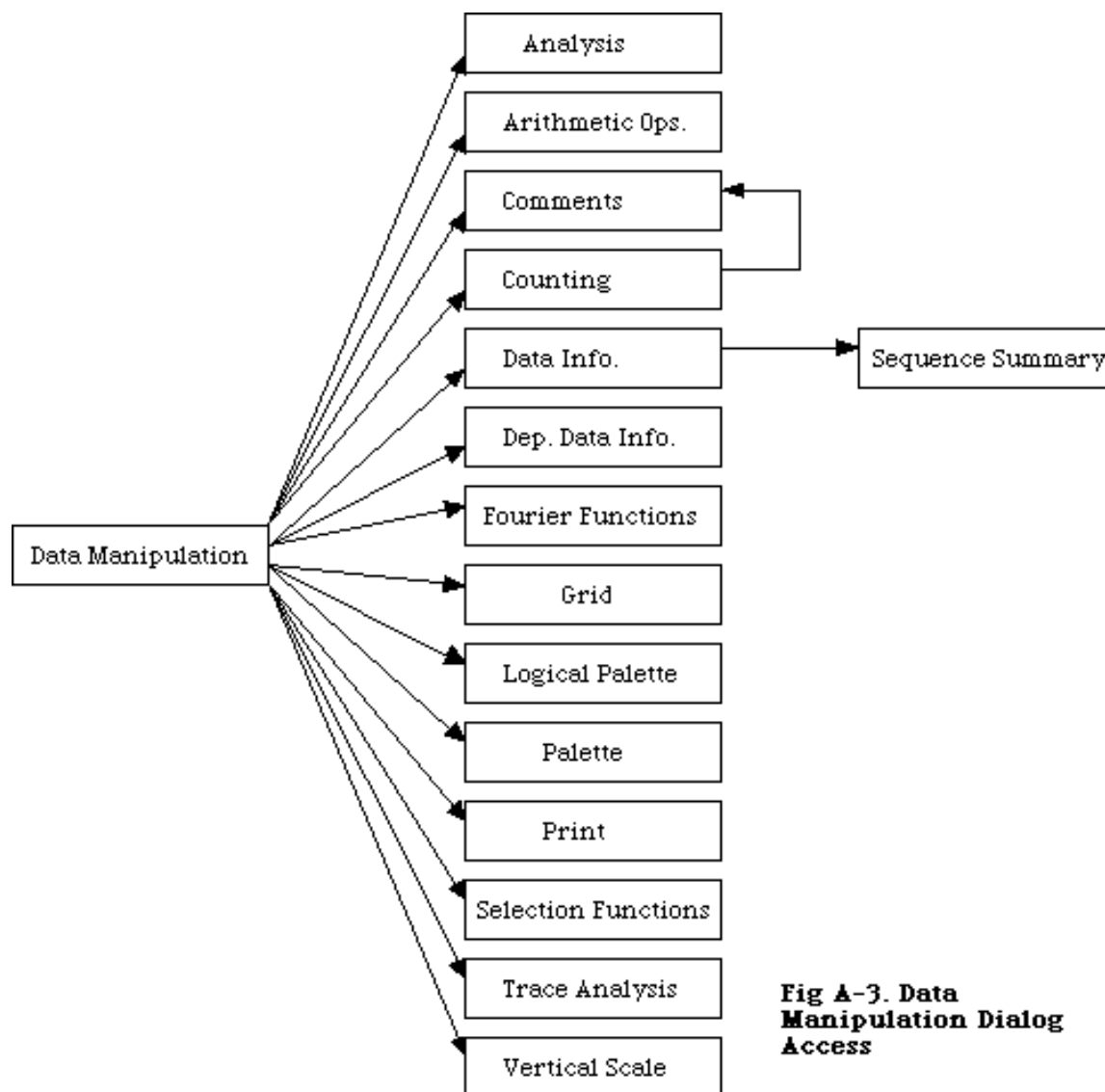


|                      |      |
|----------------------|------|
| Comments             | p 9  |
| Data Manipulation    | p 13 |
| Electronics Test     | p 18 |
| Inputs               | p 23 |
| IO Functions         | p 24 |
| Main Window          | p 30 |
| Oscilloscope         | p 31 |
| Outputs              | p 32 |
| Pulse Options        | p 42 |
| Scan                 | p 45 |
| Selective Deposition | p 51 |
| Tip Approach         | p 60 |



**Fig. A-2 Scan Dialog Access**

|                      |      |
|----------------------|------|
| Comments             | p 9  |
| Data Manipulation    | p 13 |
| IO Functions         | p 24 |
| Lockin Settings      | p 27 |
| Pre-Scan Controls    | p 34 |
| PreScan 2            | p 37 |
| Pulse Options        | p 42 |
| Scan                 | p 45 |
| Selective Deposition | p 51 |
| Sequence Summary     | p 53 |
| Sharpen Tip          | p 54 |
| Spectroscopy         | p 55 |
| Spectroscopy Options | p 57 |
| Tracking             | p 64 |



**Fig A-3. Data Manipulation Dialog Access**

|                           |      |
|---------------------------|------|
| Analysis                  | p 6  |
| Arithmetic Operations     | p 7  |
| Comments                  | p 9  |
| Counting                  | p 10 |
| Data Info                 | p 12 |
| Data Manipulation         | p 13 |
| Deposition Data Info      | p 17 |
| Fourier Functions         | p 20 |
| Grid                      | p 21 |
| Logical Palette Functions | p 28 |
| Palette Functions         | p 33 |
| Print                     | p 39 |
| Selection Functions       | p 49 |
| Trace Analysis            | p 62 |
| Vertical Scale            | p 66 |

Ho Group  
STM Software  
Code Manual

Ho Group  
Clark Hall  
Cornell University

## **Purpose**

This document outlines the Ho group's STM software. It covers the overall software design and details how that design is implemented as code. This document is intended to serve two purposes. First, it should aid competent programmers in becoming familiar with the code in a reasonable amount of time. Second, it should serve as a reference guide for the details of the software implementation.

The reader should be familiar with the C programming language and the fundamentals of STM. Readers who are not familiar with the use of the STM software are advised to read the Ho Group STM Software User's Manual.

## **Revision History**

This software is constantly being revised. The program is constantly evolving with the needs of the research group. Thus, it is important to note which version of the software is reflected in this document.

This document reflects the software as of: 01/01/98

## **Getting Started: Some General Information**

The Ho group's STM software is written in the C programming language. It runs under Windows 3.2 (Win32). The software serves several important purposes. The software:

1. Provides a graphical user interface (GUI) for operating the STM.
2. Enables communication between a personal computer and the STM via digital i/o.
3. Handles data, storing it temporarily in memory and permanently in files.
4. Manipulates data for interpretation and presentation.

The next four segments of this document discuss how the program accomplishes each of these major functions of the software. This document concludes with a reference of the C source files that make up the program. These four segments do not cover every detail of the STM software. Rather, they are intended to help the reader get started looking at and understanding the program.



## Graphical User Interface (GUI)

The windows, dialog boxes, buttons, and other parts of the GUI are consistent with Microsoft Windows 3.2. A summary of relevant features of Win32 program design follows. For more detailed information, consult one of the many sources of information about programming for Win3x (One such resource is McCord's *Windows 3.1 Programmer's Reference*).

The entry point for all Windows 3x programs is the WinMain function. In the STM program, this function can be found in the source file stm.c. The WinMain function performs some initialization and sets up the main application window. The main application window has five menus. These menus provide access to several dialog boxes which are used to control the STM and manipulate data. Many more dialog boxes may be accessed from these dialog boxes. This "nest" of dialog boxes constitutes the user interface for this program.

Windows is a message-driven operating system. The operating system sends messages to the windows and dialog boxes, carrying pertinent information such as "destroy this window" or "the user pressed the 'Scan' button." Each window and dialog box must process and respond to these messages. Usually, this is accomplished with a large switch statement. Messages for the main window in the STM software are handled by the function WndProc in stm.c. Messages for the various dialogs to messages are handled by relevant DialogProc's (for example, ScanDlg in scan.c).

From a design standpoint, the Windows-specific GUI implementation is a framework which the application is built around. The WndProc and DialogProc's handle messages and direct the program to STM-specific routines. Many of the source files are associated with individual dialog boxes. Some of the more important dialogs and their associated source files are:

|                          |           |
|--------------------------|-----------|
| scan dialog              | scan.c    |
| i/o control dialog       | io_ctrl.c |
| data manipulation dialog | data.c    |

Each of these files contains a DialogProc which handles messages sent to the relevant dialog box.

It is important to distinguish between modal and modeless dialogs. Modal dialogs forbid the user to interact with any other windows or dialog boxes while a user is free to switch between modeless dialogs.

The dialog-boxes are designed with the application Dialog Editor.

Other routines that are closely tied to the GUI are the routines for updating the appearance of the dialog boxes, buttons, editable text, and other items on the screen. These routines are usually named repaint\_\*. There are also routines for enabling and disabling on screen controls with names of the form enable\_\*.

One important subtlety of the GUI implementation is the "out\_smart" bug fix. A quick perusal of the code reveals the presence of the line

```
if( !out_smart) {...}
```

in many places. When the user adjusts a parameter that is associated with a decimal number (a voltage, for instance), it must be converted to a base-2, digital format. When dialog box receives a message indicating that such a parameter has been altered, the program converts the decimal representation of the parameter to a digital representation. The program updates the

editable text on the screen to reflect that change. However, when the editable text on the screen is changed, Windows once again sends a message to the dialog box indicating that the parameter has been adjusted. This could lead to an infinite loop in which the editable text is changed, leading to an internal change in representation, leading to the editable text being changed, etc. The out\_smart lines alleviate this potential problem.

## Digital i/o

As the user interacts with the GUI, the program must communicate the user's instructions to the STM. This is accomplished via the digital i/o.

The port(s) of the machine running the software should be connected to the 16-channel digital to analog converter which is, in turn, connected to the STM. Several Intel-specific routines are used to read and write data to and from the ports: inp, inpw, outp, outpw. Including conio.h allows access to these routines. The functions inp and outp allow communication of 8 bits at a time. outpw and inpw allow for 16 bits to be communicated at a time.

The state of the 16-channel digital to analog converter is stored in the array dac\_data[] (declared in output.c):

```
unsigned int dac_data[16];
```

Each element of dac\_data corresponds to a different channel. The channels are defined in dio.h. Some of them:

```
#define x_ch      2 /* fine x adjustment */
#define x_offset_ch 5 /* x offset (coarse x adjustment)*/
#define y_ch      3 /* fine y adjustment */
#define y_offset_ch 6 /* y offset (coarse y adjustment)*/
#define zo_ch     4
#define z_offset_ch 1
#define i_setpoint_ch 8
#define sample_bias_ch 0 /* sample bias (V)*/
```

Data may be sent out to the d-to-a converter with a short routine called dio\_out. The code for that routine, from dio.c:

```
void dio_out(unsigned int ch,unsigned int n)
{
    outpw(cfg1,0x0000+out1);
    outpw(cfg1,CFG1_CONST+out1);
    outpw(porta,n+(ch << 12));
}
```

The first two lines reset the port. The third line actually sends the data. The first four bits are used to indicate which channel is being changed. The data is placed in the remaining 12 bits. The output dialog, whose DialogProc is in output.c, allows control at this level. dio\_out is called using data and a channel selected by the user. This capability is included in the program for testing purposes and is seldom used during experiments.

dio\_in\_ch and dio\_in\_data combine to serve a similar role as dio\_out, except for reading from rather than writing to the d-to-a converter. The input dialog, whose DialogProc is in input.c, serves a role analogous to the output dialog for reading data in from the d-to-a converter.

At a more sophisticated level, there are routines for adjusting x and y coordinates, changing the piezotube mode, adjusting the tip-sample bias, and so forth. These actions are available to the user via the i/o control dialog, whose DialogProc is defined in io\_ctrl.c. The routines that

carry out these higher-level actions are in `dio.c`. Most of these routines have self-explanatory functions. For example, the routine `bias()` adjusts the tip-sample bias.

At the most sophisticated level, various actions that are accessible from the i/o control dialog are combined to perform complicated actions with the STM, the most important of which is scanning a sample surface. The routines relevant to scanning are in `scan.c` and `scan2.c`.

The scanning process is arbitrarily divided into two stages: pre-scanning and scanning. In broad terms, the algorithm for pre-scanning, from `pre_scan()` in `scan.c`, is:

- (1) Set the tip-sample bias. (V)
- (2) Set the tunneling current. (I)
- (3) Set the piezo mode, in case it has not been set.
- (4) Start dithering of channels 0 & 1, if necessary.
- (5) Zero the fine x and fine y, i.e. move them to the center of the area to be imaged.
- (6) "Synch Offsets" (In case the user forgot to hit "Send"). Set the offset x and offset y (coarse position controls) and the z offset.
- (7) Move to the starting position (lower, left corner of area to be imaged, as it appears on screen). With the slow coordinate, overshoot the image edge and then move back. Do the same with the fast coordinate. This is intended to combat creep in the piezos.

Once the surface has been pre-scanned, a scan may begin. In broad terms, the algorithm for scanning, from `scan` in `scan2.c`, is:

- (1) The tip moves up in the fast direction, taking measurements if appropriate.
- (2) The data just read (in `this_data`) is stuffed into `all_data[]`
- (3) For `scan_dir == BOTH_DIR1`, the tip is moved over one pixel. Or for `scan_dir == BOTH_DIR2_POLAR`, flip the bias. Otherwise, do nothing.
- (4) The tip moves down in the fast direction, taking measurements if appropriate.
- (5) Any data just read (in `this_data`) is stuffed into `all_data`.
- (6) The tip is moved over one pixel in the slow direction.
- (7) For `scan_dir == BOTH_DIR2_POLAR`, flip the bias. Otherwise, do nothing.
- (8) Repeat until the the entire image is scanned.

If the `scan_dir` is `BOTH_DIR1` (taking measurements in both directions to produce one composite image), then the tip will raster back and forth across the sample. For any other `scan_dir`, the tip will trace out a "comb-like" pattern: up,down,over,up,down,over,etc. During a scan, the program regularly checks to see if the ESC key has been pressed. If so, scanning is terminated.

## Data Handling

Scanned STM data and scan settings are stored in structs of the type `datadef`. The form of the struct declaration from `data.h` is shown below.

```
typedef struct tagdatadef {  
    // many, many fields  
} datadef;
```

This struct has many, many data fields. Only a few are discussed here.

There are several data fields that relate to the geometry of the scan:

```
unsigned int size; /* length of one side in pixels; all 3D scans are
                    square */
unsigned int x; /* x coordinate of the center of scan region */
unsigned int y; /* y coordinate of the center of scan region */
unsigned int z;
float x_gain;
float y_gain;
float z_gain;
int x_range;
int y_range;
```

The scanned data is pointed to by:

```
float *ptr; /* where the data is stored */
```

Note that the data stored in a datadef may be from a three-dimensional (topographic) scan or a two-dimensional scan.

There are several fields that describe physical parameters:

```
struct commentdef sample_type;
struct commentdef dosed_type;
unsigned int bias; /* tip-sample bias */
int i_set_range;
int bias_range;
unsigned int amp_gain; /* The tunnelling current gain. NOTE: Setting the
                        tunnelling current gain in the program records,
                        but does not actually set, this value. The tunnelling
                        current gain must be set manually */

float tip_spacing;
float temperature;
```

These fields detail how the scan is performed:

```
READ_SEQ *read_seq; /* array of reading sequences to carry out */
int read_seq_num; /* number of reading sequences */
unsigned int step; /* number of bits moved between measurements */
int scan_dir; /* scanning direction: FORWARD_DIR, BACKWARD_DIR,
              BOTH_DIR1, BOTH_DIR2, etc */
int step_delay; /* time to wait at a step before measurement */
int inter_step_delay; /* time to wait between inter_step movements
                      (movements without measurements) */
int inter_line_delay;
int digital_feedback; /* Boolean: whether or not the program's feedback
                      routine is used (in addition to electronics feedback) */
int crash_protection;
int overshoot; /* bitwise & with 0x1 to overshoot in fast dir,
               with 0x2 to overshoot in slow dir */
```

The following fields relate to reading data from and storing data in files.

```
int version; /* specifies version of datadef struct */
char filename[FILE_NAME_SIZE];
char dep_filename[FILE_NAME_SIZE];
char full_filename[FULL_FILE_NAME_SIZE];
```

Additionally, there are fields that are specifically related to pulse data and spectroscopy data.

Here are several global, important instances of `datadef`'s:

`datadef *all_data[1000];`

(declared in `data.c`) An array of up to 1000 sets of data taken on the most recent scan.

`datadef *data;`

(declared in `data.c`) The current data that the user has chosen to look at. Usually points to one of the `datadef`'s in `all_data[]`

`datadef **glob_data;`

(declared in `file.c`) When data is loaded or saved, it is loaded into or saved from `glob_data`.

`datadef *gendata[4]= {NULL, NULL, NULL, NULL};`

(declared in `image.c`) Contains the four current images in the data manipulation 4x view.

`datadef *scan_defaults_data[SCAN_NUM_DEFAULTS]` (declared in `scan.c`) Four sets of scan settings are stored. The user can select between these using four radio buttons in the scan window. NOTE: In `scan.h`, `SD` is defined to be the current selected element of `scan_defaults_data`.

`datadef`'s should be initialized with a call to `alloc_data`. Other routines for allocating and freeing memory associated with `datadef`'s.

All variables that are related to time are given in microseconds.

Saving to and loading from files is accomplished with the various routines in `file.c`. More details about `file.c` are given in the file catalog at the end of this document.

## Data Manipulation

Once a set of data has been acquired, the program allows a user to manipulate the data so that it is suitable for analysis or presentation. Possible manipulations include: superimposing a grid over the image, taking the fourier transform of the data, subtracting one set of data from another and several other operations.

Code for the data manipulation dialog is in `image.c`. Several of the operations that can be carried out are coded in that file. The remainder of the operations are accessed via their own dialogs and are described separately (`grid.c` for superimposing grids is an example). Most of the data manipulations are straightforward mathematical operations on the data.

Code for operations that require more than one set of data obey the following convention: the index of the currently selected data is "a" and the index of the data indicated by pressing a button is "b". For example, the subtraction operation generates a set of data from `gendata[a] - gendata[b]`. To select data in quadrant 2 from data in quadrant 3, one would select quadrant 3 and then hit the "2" button in the arithmetic functions dialog.

For several of the operations, the program uses source files written by people outside the group. For example, `ppmtogif.c`.

## File Reference

There are few comments in the Ho group STM software code. As a result, reading the code can be challenging. This portion of the manual is meant to serve as a reference for the \*.c and \*.h source files that make up the program. It is written as if it were a file full of comments.

The files are listed in alphabetical order. The list excludes \*.h files which consist only of #define'd constants and function declarations. Additionally, functions of the form repaint\_\* and enable\_\* are not listed. The repaint\_\* functions are used to update editable texts, graphics, etc. on the screen. The enable\_\* functions are used to enable or disable buttons and other controls based on the value of a boolean status parameter. Finally, code written by people outside the group is not documented here (such as ppmtogif.c).

```
// *****  
// anl.c  
// *****
```

```
BOOL FAR PASCAL AnlDlg(HWND hDlg, unsigned Message, WPARAM wParam,  
                      LPARAM lParam)
```

```
// A big switch( Message) that handles messages sent to the analysis  
// functions dialog, including those to various buttons and text fields  
// (WM_COMMAND's).
```

```
float avg_z( SEL_REGION *region)  
// returns the average value of z for all points within the region that  
// the user has selected in the data manipulation window
```

```
float avg_z_2d( SEL_REGION *region1, SEL_REGION *region2)  
// same as avg_z, except for two dimensional data
```

```
// *****  
// coarse.c  
// *****
```

```
BOOL FAR PASCAL CoarseDlg( HWND hDlg, unsigned Message, WPARAM wParam,  
                          LPARAM lParam)
```

```
// A big switch( Message) that handles messages sent to the coarse movement  
// dialog, including various buttons and text fields ( WM_COMMAND's ).
```

```
// *****  
// comment.c  
// *****
```

```
BOOL FAR PASCAL CommentDlg( HWND hDlg, unsigned Message, WPARAM wParam,  
                           LPARAM lParam)
```

```
// Handles messages sent to the comment dialog  
// Note that the comment dialog displays and allows editing of the global  
// commentdef gcomment. So, gcomment should be appropriately defined before  
// entering this dialog.
```

```

// *****
// comment2.c
// *****

// comment2 vs. comment
// comment2 is specific to the image manipulation dialog (image.c).
// comment2 is able to update itself based on which of the four images is
// selected. Also, comment2 is modeless while comment is modal.

BOOL FAR PASCAL Comment2Dlg( HWND hDlg, unsigned Message, WPARAM wParam,
                             LPARAM lParam)
// Handles messages sent to the comment2 dialog

// *****
// common.c
// *****

void CheckMsgQ()
// Routine for checking for Windows messages when we're hogging time
// (during scans, printing, etc.)

void init_listbox( HWND hDlg, int listboxid, LISTBOX *listbox)
// sets up listbox with appropriate items from listboxid

char *scan_freq_str( char *buf, unsigned int scale, unsigned int freq)
// looks up appropriate string, based on scale and freq

void fit_plane_simple( datadef *this_data, double *a, double *b, double *c)
// Fits a plane to the data in this_data using a least squares fit:
// 
$$z = a*x + b*y + c$$


void copy_data( datadef **dest, datadef **source)

void clear_area( HWND hDlg, int x1, int y1, int x2, int y2, COLORREF color)
// clears the rectangular area described by the coordinates and fills it
// with the specified color

float distance( float x1, float y1, float x2, float y2)

float interp_z( datadef *this_data, float row, float col)
// returns interpolated value of z at the point (row,col). for 3d data.

int bin_find( float *ptr, int find_min, int find_max, float goal)
// returns the index for which ptr[index] is nearest to goal
// looks for index in the range between find_min and find_max

char *bgr( datadef *this_data, float this_z, int *color32)

float linear_data( datadef *this_data, float row, float col)

int linear_data3( datadef *this_data, float row, float col)

int vert_int( float x1, float y1, float theta, float x2, float *y2)

```

```

int horiz_int( float x1, float y1, float theta, float *x2, float y2)

PRINT_EL *new_print_el( PRINT_EL **this_list, int type)

LOGPAL_EL *new_logpal_el( LOGPAL_EL **this_list, int index)

int new_count_el( COUNT_EL **this_list, int x, int y)

int remove_count_el( COUNT_EL **this_list, int x, int y)

void destroy_logpal( LOGPAL_EL **this_list)

void copy_logpal( LOGPAL_EL **dest, LOGPAL_EL *src)

LOGPAL_EL *sort_logpal_els( LOGPAL_EL **this_list)

float float_bin_find( float *ptr, int find_min, int find_max, float goal)

float point_line_dist( float x1, float y1, float x2, float y2, float x3, float y3)

void calc_r_theta( GRID this_grid, float *r, float *theta)

void wait_cursor()
// Sets the cursor to the wait cursor

void arrow_cursor()
// Sets the cursor to the arrow cursor

void find_min_max( datadef *data, float *min_z, float *max_z)

int color_pal( GENPAL this_pal)

double calc_i_set( unsigned int i_setpoint, int i_set_range,
                  unsigned int tip_gain)

void destroy_count( COUNT_DATA *list)

int index2d( datadef *this_data, int index)

void init_count_data( COUNT_DATA *this_data)
// Initializes the count_datadef pointed to by this_data

void copy_count( COUNT_DATA **dest, COUNT_DATA source)

void free_count( COUNT_DATA **this_data)

int rect_intersect_gen( float r1x1, float r1y1, float r1x2, float r1y2,
                      float r2x1, float r2y1, float r2x2, float r2y2 )

void paint_circles_gen( HDC hDC, int x_offset, int y_offset,
                      float pixel_size,
                      COUNT_EL *this_list, int size, COLORREF color)

```



```

// *****
// count.c
// *****

// Note that counting data is distinct from scanned data. Counting data
// is stored in structs of the type count_datadef, defined in data.h.

// important global variables
BOOL CountOn = FALSE; /* If CountOn==TRUE, clicking in image will add dot
                        and increment the total count for the current color */
COUNT_DATA count_data[4]; /* counting data for the 4x images in gendata[] */

BOOL FAR PASCAL CountDlg(HWND hDlg, unsigned Message, WPARAM wParam,
                        LPARAM lParam)
// A big switch( Message) that handles messages sent to the counting
// dialog various buttons and text fields ( WM_COMMAND's ).

// This routine is responsible for handling the counting dialog. However,
// actual counting is done through the data manipulation dialog with calls
// to countit() in image.c.

void init_count()
// Initializes each of the four count_datadef's in count_data[]

// Arrows in the counting dialog are for shifting the entire set of counter
// dots vertically or horizontally. Internally, the program uses
// count_horiz_shift and count_vert_shift to do this
void count_horiz_shift( HWND hDlg, int shift)

void count_vert_shift( HWND hDlg, int shift)

// *****
// data.c
// *****

void alloc_data( datadef **dataptr, int type, int size, int x_type,
                int y_type, int seq_size)
// Allocates memory and initializes fields in **dataptr

void alloc_data_ptrs( datadef **dataptr, int type, int size, int x_type,
                    int y_type, int seq_size)
// Allocates memory for (*dataptr)->ptr based on the type of data
// (topography, tunneling spectroscopy, etc).

void free_data( datadef *(*dataptr))
// frees dataptr and all ptrs within the struct

void free_data_ptrs( datadef *(*dataptr))
// frees all fields of dataptr struct that are not sub-structs, but are ptrs

void alloc_data_seq( datadef **dataptr, int seq_size)
// Allocates memory for (*dataptr)->read_seq and some other initialization.

```

```

void free_data_seq( datadef **dataptr)
// frees the read_seq field of dataptr

// *****
// dio.h
// *****

// define's

// input channels
#define zi_ch      1 /* z input with feedback on */
#define i_in_ch    0 /* i input with feedback off */
#define tip_ch     0 /* tip channel during approach */

// output channels (also correspond to indices of dac_data[] )
#define x_ch       2 /* fine x coordinate */
#define x_offset_ch 5 /* coarse x coordinate */
#define y_ch       3 /* fine y coordingate */
#define y_offset_ch 6 /* coarse coordinate */
#define zo_ch      4
#define z_offset_ch 1
#define i_setpoint_ch 8
#define sample_bias_ch 0
#define test_ch     9 /* for testing purposes */

#define range_ch    12
#define AD_ch       13
#define mode_ch     14 /* piezo mode */
#define feedback_ch 14 /* feedback on/off */
#define hold_ch     14 /* having hold on is same as feedback off, v.v */
#define retract_ch  14
#define gain_ch     14 /* z offset gain: 0.1, 1, 10 */

// The following routines can be redefined, if the macro NO_DIO_CARD is defined,
// i.e. if you want to run w/o a dio card installed:

#define dio_out( CH, VAL)
#define dio_start_clock( TIME)
#define dio_wait_clock()

// *****
// dio.c
// *****

int dio_dither_status( int ch)
// Returns the dither status (1=on, 0=off) of the channel ch.

void dio_dither( int ch, int status)
// Sets the dither status of channel ch to status (1=on, 0=off).

void dio_set_registers()
// Initializes i/o registers.

```

```

void dio_init()
// Initializes 16 channels of electronics and i/o registers.

int get_range( int ch)
// returns 1 if range of channel ch is +-5 V, 2 if +-10 V

void set_range( int ch, int range)
// Resets the STM-range to +-10 V. If variable range is 1, then sets to +-5 V.

void dio_feedback( int on)
// Turns the feedback on/off based on boolean interpretation of parameter on.

unsigned int *dio_blk_setup( unsigned int ch, unsigned int wave,
                           unsigned int n, float wave_range, int dir)

void dio_out( unsigned int ch, unsigned int n)
// Sends n to channel ch via the output register.

// Functions dio_blk_free and dio_blk_out are only called from coarse.c
// and are of questionable usefulness.
void dio_blk_free( unsigned int *data)
void dio_blk_out( unsigned int n, unsigned int *data)

unsigned int *para_move_setup( unsigned int step, unsigned int size,
                             unsigned int offset)
// This function is not called. It appears to have been copied as
// stair_move_setup(...), which does get called.

unsigned int para_size( unsigned int n)
// This function is not called. It appears to be superceded by stair_size.

void para_free( unsigned int *ptr)
// This function is not called.

unsigned int *stair_move_setup( unsigned int step, unsigned int offset)
// Sets up wave form for steps taken by piezos

unsigned int stair_size( unsigned int n)

void stair_free( unsigned int *ptr)

unsigned int *tip_setup( unsigned int xn, unsigned int zon,
                      unsigned int xstep, unsigned int zstep, int dir)
// Sets up wave form for steps taken by piezos during giant steps of tip approach

unsigned int *tip_zo_setup( unsigned int n)
// Sets up wave form for steps taken by piezos during baby steps of tip approach

void tip_free( unsigned int *data)

void dio_in_ch( unsigned int ch)

void dio_in_data( unsigned int *data)

```

```

double dio_read( unsigned int n)

double dtov( unsigned int data, int range)

double dtov_len( int data, int range)

unsigned int vtod( double v, int range)

char *dtob( unsigned int data)

unsigned int btod( char *str)

double in_dtov( unsigned int data)

int in_vtod( double data)

char *in_dtb( unsigned int data)
// data is a base-10 integer
// converts data to a string in binary
// Example data = 2177 is changed to "0000 1000 1000 0001"

void mode( int m)

void tip_offset( unsigned int data)

void move_to( unsigned int ch, unsigned int datai, unsigned int dataf)

void move_to_speed( unsigned int ch, unsigned int datai, unsigned int dataf,
                    int time, int digital, int Imin, int Imax,
                    int digital_feedback_max, int digital_feedback_reread)

void move_to_timed( unsigned int ch, unsigned int datai, unsigned int dataf,
                   int time)

unsigned int move_to_protect( unsigned int ch, unsigned int datai,
                             unsigned int dataf, int time, int crash_protection,
                             float limit_percent)

unsigned int move_to_protect2( unsigned int ch, unsigned int datai,
                              unsigned int dataf, int time, int crash_protection,
                              int digital, int Imin, int Imax, int digital_abort,
                              int force_it, int digital_feedback_max,
                              int digital_feedback_reread)
// datai, dataf are initial and final positions
// time is the delay (in microsecs) between each step during movement

void move_to2( unsigned int ch, unsigned int datai, unsigned int dataf,
              unsigned int steps)

// *****
// routines for setting gain on coordinate signals

void set_gain( unsigned int x_gain, unsigned int y_gain, unsigned int z_gain,
              unsigned int z2_gain)

```

```

void set_x_gain( unsigned int x_gain)

void set_y_gain( unsigned int y_gain)

void set_z_gain( unsigned int z_gain)

void set_z2_gain( unsigned int z2_gain)

// *****

void hold( int hold_on)
// ...as in "sample and hold." Having the hold_on is the same thing as
// having the feedback off. This is only called in one instance, during the tip
// approach.

void retract( int retract_on)

void adc_delay()
// When reading data in from the STM, need to allow about 15 microsecs
// for completion of analog to digital conversion.

void tip_current( unsigned int data)

void bias( unsigned int data)
// set the tip-sample bias to be data

void ramp_bias( unsigned int data, int time, int skip, int bits)

void ramp_ch( unsigned int ch, unsigned int data, int time, int skip, int bits)

int dio_digital_feedback( int Imin, int Imax, int digital_feedback_max,
                          int digital_feedback_reread)
// this routine monitors the current I and continues to run as long as
// I < Imin or I > Imax. digital_feedback_max is the maximum number of times
// that the routine will check the current before aborting. If digital_feedback_reread
// is nonzero, the routine will double check that Imin < I < Imax before exiting

unsigned int flipped_bias( unsigned int bias)

// *****
// dep.c
// *****

BOOL FAR PASCAL DepDlg( HWND hDlg, unsigned Message, WPARAM wParam,
                        LPARAM lParam)
// A big switch(Message) that handles messages sent to the deposition
// dialog, including scrolling (WM_HSCROLL, WM_VSCROLL), various buttons
// and text fields ( WM_COMMAND's ).

```

```

BOOL FAR PASCAL DepPulseDlg( HWND hDlg, unsigned Message, WPARAM wParam,
                             LPARAM lParam)
// A big switch(Message) that handles messages sent to the pulse options
// dialog, including scrolling (WM_HSCROLL, WM_VSCROLL), various buttons
// and text fields ( WM_COMMAND's ).

void dep_pulse2( HWND hDlg)

void dep_pulse3( HWND hDlg)

void set_data( datadef *this_data)

void get_data( datadef *this_data)

void dep_box( HWND hDlg)

void dep_lines( HWND hDlg)

void set_write_cond( HWND hDlg, unsigned int write_bias,
                    unsigned int write_i_setpoint)

void set_move_cond( HWND hDlg)

void set_scan_cond( HWND hDlg)

static void dep_set_title( HWND hDlg)

// *****
// etest.c
// *****

BOOL FAR PASCAL EtestDlg( HWND hDlg, unsigned Message, WPARAM wParam,
                           LPARAM lParam)
// A big switch( Message) that handles messages sent to the electronics
// test dialog, including scrolling (WM_HSCROLL, WM_VSCROLL) and various
// buttons ( WM_COMMAND's ).

static void calc_digi_ch( HWND hDlg)

static void etest_ramp( HWND hDlg)

static void etest_test_input( HWND hDlg)

static void etest_test_output( HWND hDlg)

static void etest_calibration( HWND hDlg)

// *****
// file.c
// *****

```

```

// global variables related to files
char *current_dir_stm; /*...and others. Current paths */
char *initial_dir_stm; /*...and others. Initial paths */
char *current_file_stm; /*...and others. Current file name: MMDDYY#.ext */
int file_stm_count; /*..and others. Keeps track of # being used in filename */

void init_file( char *dir, char *filename, char *ext, int *count)
// Sets the filename by date: MMDDYY##.ext where the ## is determined by
// count. Count may be incremented in order to produce a unique filename.

BOOL file_open( HWND hWnd, HANDLE hInstance, int file_type, int allow_all,
                char* this_file)
// Allows user to select and open a file. Returns TRUE if successful.

BOOL file_save_as( HWND hWnd, HANDLE hInstance, int file_type)
// Allows user to save data. (if it's image data, glob_data in particular)
// Returns TRUE if successful.

int load_image_old( char *filename)
// Load an image that is "old format" (no magic & version). Returns 0 if
// unsuccessful. If successful, returns non-zero value and places image
// data in glob_data datadef.

int load_image( char *filename, int compressit)
// Load an image by filename (a path+filename, really). Returns 0 if
// unsuccessful. If successful, returns non-zero value and places image
// data in glob_data datadef.

int load_image_fp( FILE *fp, int compressit))
// Load an image by file pointer. Returns 0 if unsuccessful. If successful,
// returns non-zero value and places image data in glob_data datadef.
// Uses version number of file to determine what fields will be present.

int save_image( char *filename)
// Save glob_data as filename (a path+filename, really). Returns if
// successful.

int save_image_fp( FILE *fp)
// Save glob_data to fp file pointer. Returns 1 if successful.
// Uses version number of file to determine what fields will be present.

void save_init( char *fname)
// Save important global variables to fname

void load_init( char *fname)
// (Mostly) initialization of global variables based on what's in fname

void init_dirs()
// Reads in directory paths from initialization file. Sets the corresponding
// global path variables (current_dir_stm, current_dir_pal, etc.).

```

```

// *****
// fine.c
// *****

BOOL FAR PASCAL FineDlg( HWND hDlg, unsigned Message, WPARAM wParam,
                        LPARAM lParam)
// A big switch( Message) that handles messages sent to the fine movement
// dialog, including various buttons and text fields ( WM_COMMAND's ).

void pre_fine()

void post_fine()

// *****
// grid.c
// *****

// Two structs that are relevant to this dialog that are declared in data.h
// are GRID AND GRID_LINE.

// The four grids that correspond to the 4X view in the data manipulation
// window are stored in the global variable:
GRID grids[4];

BOOL FAR PASCAL GridDlg(HWND hDlg unsigned Message, WPARAM wParam,
                        LPARAM lParam)
// A big switch( Message) that handles messages sent to the grid
// dialog, including those to various buttons and text fields (WM_COMMAND's).

void copy_grid( GRID *dest, GRID *source)

void init_grids()
// Grid lines are initially hidden, have theta (from horizontal) = 0, have
// r (offset from bottom) = 1, and are separated by GRID_INIT_DIST

void grid_get_line( int line)

void grid_get_clip_line()

void grid_match3( int one, int two, int three)
// matches the column three grid's angle and separation to the
// column one and column two grids (presently, only called with parameters
// (0,1,2))

int grid_hidden( int this_image)

```



```

// *****
// heater.c
// *****

BOOL FAR PASCAL HeaterDlg( HWND hDlg, unsigned Message, WPARAM wParam,
                           LPARAM lParam)
// A big switch( Message) that handles messages sent to the heater
// dialog, including scrolling (WM_HSCROLL, WM_VSCROLL) and various buttons
// and text fields ( WM_COMMAND's ).

// *****
// image.c
// *****

BOOL FAR PASCAL ImageDlg( HWND hDlg, unsigned Message, WPARAM wParam,
                           LPARAM lParam)
// A big switch( Message) that handles messages sent to the Data Manipulation
// dialog, including scrolling (WM_HSCROLL, WM_VSCROLL) and various buttons
// and text fields ( WM_COMMAND's ).

// Note that the Data Manipulation dialog receives messages from other
// dialogs to handle the updating of the images. In a sense, the Data
// Manipulation dialog facilitates the functions of the Grid dialog,
// the Counting dialog, and the Selection Functions dialog.

void read_smooth_values( HWND hDlg, int reverse)

void image_sel_cut( SEL_REGION *region, datadef *dest)

void image_sel_copy( SEL_REGION *region, datadef **dest, datadef, **source)

void linearize( int num)

int fit_plane_region( int num, SEL_REGION *region, double *a, double *b,
                     double *c)

void fit_plane( int num, SEL_REGION *region)
// fits a plane to the region of gendata[num] of the form
//  $z = ax + by - c$ 
// and replaced the values of z in gendata[num] by the values
// of z relative to this plane

void take_log( int num)
// replaces the values of z in gendata[num] by base 10 log(z)

void project3D( int num)

void take_deriv( HWND hDlg, int num, int dir)

void invert( int num)
// replaces the values of z in gendata[num] by -z

```

```

void drift_comp( int number)
// maps square data onto a parallelogram to compensate for drift
// This algorithm comes from Sung-il Park's thesis

int rect_intersect( int image, RECT area)

void normalize( HWND hDlg)

void smooth( HWND hDlg, int num)

void calc_bitmap( int num)

void calc_bitmap_gen( datadef *this_image, unsigned char **this_bitmap,
                      int *size);

int time_index( int number, float t)

void im_do_scrolls( HWND hDlg, int number)

int floatcmp( float *f1, float *f2)

void free_dtrans( DTRANS *dtrans)

void read_dtrans( DTRANS *dtrans, char *filename)

void copy_filter( float dest[], float *src)

void paint_line( HWND hDlg, SEL_REGION *region)

void paint_rect( HWND hDlg, SEL_REGION *region)

void im_invert( HWND hDlg, SEL_REGION *region)

void paint_ellipse( HWND hDlg, SEL_REGION *region)

void paint_cross( HWND hDlg, SEL_REGION *region, HBITMAP cross)

void find_clip_rect( HWND hDlg, RECT *r)

void save_gif(char *fname);

void square_coords( SEL_POINT pt1, SEL_POINT pt2, int image)

void circle_coords( SEL_POINT pt1, SEL_POINT pt2, int image)

void inc_pt_dist( int x1, int *x2, int dist)

int pt_in_region( int x, int y, SEL_REGION *region)

int rev_size( int size)

int image2pixel( float image_x, float image_y, int *pixel_x, int *pixel_y,
                 int image)

```

```

static int draw_grid_line( HWND hDlg, float r, float theta, int this_image)

void notify_all()

void image_set_title( HWND hDlg)

int color_index( datadef *this_data, int row, int column)

void new_image( HWND hDlg)

void reset_image( HWND hDlg, int num)

static void image_cut_profile( SEL_REGION *region)

static void countit( HWND hDlg, int this_image, int num, SEL_POINT temp_pt)

// *****
// info.c
// *****

BOOL FAR PASCAL InfoDlg( HWND hDlg, unsigned Message, WPARAM wParam,
                        LPARAM lParam)
// Handles messages sent to the information dialog

// *****
// infodep.c
// *****

BOOL FAR PASCAL InfodepDlg( HWND hDlg, unsigned Message, WPARAM wParam,
                           LPARAM lParam)
// Handles messages sent to the deposition information dialog

// *****
// infospc.c
// *****

BOOL FAR PASCAL InfospcDlg( HWND hDlg, unsigned Message, WPARAM wParam,
                           LPARAM lParam)
// Handles messages sent to the this dialog

// *****
// input.c
// *****

BOOL FAR PASCAL InputDlg( HWND hDlg, unsigned Message, WPARAM wParam,
                         LPARAM lParam)
// Handles messages sent to the input dialog.

```

```
// *****
// io_ctrl.c
// *****
```

```
BOOL FAR PASCAL IOCtrlDlg( HWND hDlg, unsigned Message, WPARAM wParam,
                          LPARAM lParam)
```

```
// A big switch( Message) that handles messages sent to the i/o control
// dialog, including scrolling (WM_HSCROLL, WM_VSCROLL) and various buttons
// and text fields ( WM_COMMAND's ).
```

```
// The i/o control dialog gives the user direct control over just about every
// facet of the digital i/o, including fine and coarse position, feedback,
// tip retraction and so on. An important note, radio buttons exist for
// choosing the tunnelling current gain. Selecting these radio buttons records
// but does not actually change the tunnelling current gain. This must be done
// manually.
```

```
// *****
// lockin.c
// *****
```

```
BOOL FAR PASCAL LockinDlg( HWND hDlg, unsigned Message, WPARAM wParam,
                          LPARAM lParam)
```

```
// A big switch( Message) that handles messages sent to the lockin
// dialog, including scrolling (WM_HSCROLL, WM_VSCROLL) and various buttons
// and text fields ( WM_COMMAND's ).
```

```
// *****
// logpal.c
// *****
```

```
BOOL FAR PASCAL LogpalDlg( HWND hDlg, unsigned Message, WPARAM wParam,
                          LPARAM lParam)
```

```
// A big switch( Message) that handles messages sent to the logical palette
// dialog, including scrolling (WM_HSCROLL, WM_VSCROLL) and various buttons
// and text fields ( WM_COMMAND's ).
```

```
static void remove_logpal_el( HWND hDlg, LOGPAL_EL *remove_el)
```

```
// *****
// masspec.h & masspec.c
// *****
```

```
// Mass spectroscopy was superceded by tunneling spectroscopy. Little or none
// of the code in these files is ever compiled or executed.
```

```

// *****
// oscill.c
// *****

BOOL FAR PASCAL OscillDlg(HWND hDlg, unsigned Message, WPARAM wParam,
                          LPARAM lParam)
// A big switch( Message) that handles messages sent to the oscilloscope
// dialog, including scrolling (WM_VSCROLL) and various buttons
// and text fields ( WM_COMMAND's ).

// *****
// output.c
// *****

// important global variables
unsigned int dac_data[16]; /* Reflects the current state of the electronics.
                           The channels that correspond to the indices
                           of dac_data are defined in dio.h */

BOOL FAR PASCAL OutputDlg( HWND hDlg, unsigned Message, WPARAM wParam,
                          LPARAM lParam)
// A big switch( Message) that handles messages sent to the output
// dialog, including scrolling (WM_HSCROLL, WM_VSCROLL) and various buttons
// and text fields ( WM_COMMAND's ).

// *****
// pal.c
// *****

BOOL FAR PASCAL PalDlg( HWND hDlg, unsigned Message, WPARAM wParam,
                       LPARAM lParam)
// A big switch( Message) that handles messages sent to the palettes
// dialog, including various buttons ( WM_COMMAND's ).

void animate_pal( HWND hDlg, int del);

void apply_gamma( unsigned char *dacbox);

void MakeColorLUT(int *pLut, /* array to hold LUT */
                 double gamma); /* exponent of stretch */

int spindac( int direction, int step, unsigned char *dacbox);
// Originally, this shifted the palette by step. Repeated calls to this would
// "spin" the colors of the image in a manner that some might call "trippy."
// It is currently broken, but a call of spindac(0,1, ...) will stuff the
// dacbox into the Windows logical palette.

```

```

void ramp( int colindex, int mincol, int maxcol, int start, int end
          unsigned char* dacbox)
// for RGB, colindex = 0 is R, 1 is G, 2 is B
// fills in the colindex (R,G, or B) segment from index start to end with
// shades between mincol and maxcol

void set_Plasma_palette(unsigned char *dacbox);
// fills in dacbox with shades of colors red->orange->yellow->green->blue

void get_rand_color( unsigned char *color)

void get_rand_high_color( unsigned char *color)

void set_random_pal( unsigned char *dacbox);

void set_random_one_pal( unsigned char *dacbox);

void histogram()

void equalize( datadef *this_data, float min_z, float max_z, float *fhist)

// *****
// pre_scan.c
// *****

BOOL FAR PASCAL PrescanDlg( HWND hDlg, unsigned Message, WPARAM wParam,
                          LPARAM lParam)
// A big switch( Message) that handles messages sent to the Prescan 1
// dialog, including scrolling (WM_HSCROLL, WM_VSCROLL) and various buttons
// and text fields ( WM_COMMAND's ).

BOOL FAR PASCAL Prescan2Dlg( HWND hDlg, unsigned Message, WPARAM wParam,
                          LPARAM lParam)
// A big switch( Message) that handles messages sent to the Prescan 2
// dialog, including scrolling (WM_HSCROLL, WM_VSCROLL) and various buttons
// and text fields ( WM_COMMAND's ).

int auto_z_below( int target)
// Adjusts the z-offset signal so that it is just below target. Note that
// the feedback electronics will maintain a constant tunneling current,
// so that actual distance bewteen the tip and sample will be unchanged.

// Important calls to auto_z_below and auto_z_above are made just before
// switching the gain (see WM_COMMAND, SCAN_SCALE_001, etc.). This prevents
// the tip from crashing.

int auto_z_above( int target)
// Adjusts the z-offset signal so that it is just above target. Note that
// the feedback electronics will maintain a constant tunneling current,
// so that actual distance bewteen the tip and sample will be unchanged.

void read_sample_list( SAMPLELIST *list, char *fnam, char *first_name,
                     char *first_dir)

```

```

void free_sample_list( SAMPLELIST *list)

int isdir( char *name)

unsigned int calc_i_setpoint( double i_set, int i_set_range,
                             unsigned int tip_gain)

static int z_read()

static int check_crash_protection( HWND hDlg)

static void check_digital_feedback( HWND hDlg)

static void copy_seq( READ_SEQ *dest, READ_SEQ *src)

static void set_dumb_seq( READ_SEQ *this_seq)

static void add_sequence()

static void insert_sequence()

static void del_sequence()

int num_data()
// Returns the number of read sequences that record a channel of data

static void summary( HWND hDlg)

// *****
// print.c
// *****

BOOL FAR PASCAL IPrintDlg(HWND hDlg unsigned Message, WPARAM wParam,
                          LPARAM lParam)
// A big switch( Message) that handles messages sent to the printing
// dialog, including scrolling (WM_VSCROLL) and various buttons
// and text fields ( WM_COMMAND's ).

void print( HWND hDlg)

int place_image( PRINT_EL *this_el, int image_number, FILE *fp)

void postscript_preamble( FILE *fp)

void postscript_postamble( FILE *fp)

void write_image( HWND hDlg, PRINT_EL *this_image, int image_number,
                 FILE *fp, int this_output_color)

void print_bit( FILE *fp, int this_bit)

void flush_row( FILE *fp)

```

```

void finish_print_bit( FILE *fp)

void start_print_bit( FILE *fp, int num)

void clear_maybe()

void select_next( int multi_select)

void clear_selection()

int maybe_select( POINT pt)

int pt_in_image( PRINT_EL *this_el, POINT pt)

int pt_in_rect( int x1, int y1, int x2, int y2, POINT pt)

int coord_in_rect( float x, float y, float x1, float y1, float x2, float y2)

void selection_squares( HDC hDC, int x1, int y1, int x2, int y2)

void selection_square( HDC hDC, int x, int y)

void count_selected()

void print_status( HWND hDlg, char *mes)

static float pof_bw( float z, float gamma)

static float pof_grey( float z, float gamma)

static float pof_color( int color32, float gamma)

int line_side( GRID this_grid, int x, int y)

static void write_circles( PRINT_EL *this_el, FILE *fp)

static void write_circles_list( COUNT_DATA *count_data, int num,
                                float pixel_size, FILE *fp)

static void write_grid( PRINT_EL *this_el, FILE *fp)

int intersection_pts( float r, float theta, float *x1, float *y1,
                     float *x2, float *y2, int image_size)

static int write_grid_line( float r, float theta, int image_size,
                           float print_size, FILE *fp)

void re_link()

static void remove_print_el( PRINT_EL **print_list, PRINT_EL *remove_el)

void delete_selected()

static void copy_print_el( PRINT_EL **this_list, PRINT_EL *this_el)

```



```

static void copy_selected()

static void paste_seleceted()

void delete_all()

void print_char( FILE *fp, int c)

void get_char_bitmap( HWND hDlg, LOGFONT *font, char chr, char **bitmap,
                     FONTBITMAPINFO *fbinfo)

static void write_text( HWND hDlg, HDC hDC, PRINT_EL *this_el)

static void place_char( char *bitmap, float x, float y, int bitmap_x,
                      int bitmap_y, STM_COLOR color, FILE *fp)

static void write_arrow( HWND hDlg, PRINT_EL *this_el, FILE *fp)

static void prune()

void init_arrow_heads()

static PRINT_EL *add_image( int num)

static void reset_current()

static int cur_obj_type( int type)

static float max_print_x_origin()

static float max_print_y_origin()

static void print_do_scrolls( HWND hDlg)

static void print_do_x_scroll( HWND hDlg)

static void print_do_y_scroll( HWND hDlg)

static int draw_grid_line( HDC hDC, float r, float theta, int left,
                          int bottom, int right, int top, int x_offset,
                          int y_offset, float pixel_size)

static void repaint_grids( HDC hDC, PRINT_EL *this_el)

// *****
// scan.h
// *****

// preprocessor define (for convenience)
#define SD scan_defaults_data[scan_current_default]

```

```

// *****
// scan.c
// *****

// important global variables
float scan_x_gain;    /* for fine x signal; can be 0.1, 1, or 10 */
float scan_y_gain;    /* for fine y signal; can be 0.1, 1, or 10 */

datadef *scan_defaults_data[SCAN_NUM_DEFAULTS]

BOOL FAR PASCAL ScanDlg( HWND hDlg, unsigned Message, WPARAM wParam,
                        LPARAM lParam)
// A big switch( Message) that handles messages sent to the scan dialog,
// including scrolling (WM_HSCROLL, WM_VSCROLL) and various buttons
// and text fields (WM_COMMAND's).

void calc_gains( unsigned int *x_gain, unsigned int *y_gain,
                unsigned int *z_gain, unsigned int *z2_gain)

void calc_step_delay()
// sets step delay for the current scan default data (SD), based on scanning
// scale and scanning frequency

float calc_initial_time()

float calc_overshoot_fast_time()

float calc_overshoot_slow_time()

float calc_total_time()

float calc_seq_time()

float calc_time( int lines_left)

void sharpen( HWND hDlg)

void init_bitmap_data()
// if the current data ( in global datadef *data) is valid, fills in scan bitmap
// accordingly

int find_size( int size)

void scan_status( HWND hDlg, char *text, float num)

float calc_z_gain( unsigned int scan_scale)

static void scan_set_title( HWND hDlg)
// Sets the title of the scan dialog according to the filename of the current data

```

```

void set_scan_defaults( datadef *this_data)
// Sets fields in datadef pointed to by this_data to default values

int pre_scan( HWND hDlg)
// pre_scan performs the following tasks, in this order:
// (1) Set the tip-sample bias. (V)
// (2) Set the tunneling current. (I)
// (3) Set the piezo mode (just to be sure).
// (4) Start dithering of channels 0 & 1, if necessary.
// (5) Zero the fine x and fine y, i.e. move them to the center of the
// area to be imaged.
// (6) "Synching Offsets". In case the user forgot to hit "Send."
// Set the offset x and offset y (coarse position controls) and
// the z offset.
// (7) Move to the starting position (lower, left corner of area to be
// imaged, as it appears on screen). With the slow coordinate,
// overshoot the image edge and then move back. Do the same with the
// fast coordinate.

// *****
// scan2.c
// *****

void scan( HWND hDlg)
// Basic scanning routine. Directs digital i/o and updates on-screen image.
// This routine assumes that pre_scan has already been called to set V & I,
// to set the x and y offset to the appropriate position, and to set the fine
// x and y to the lower left corner of the area to be imaged.
// The scan follows these steps:
// (1) The tip moves up in the fast direction, taking measurements
// if appropriate.
// (2) The data just read (in this_data) is stuffed into all_data
// (3) For scan_dir == BOTH_DIR1, the tip is moved over one pixel.
// Or for scan_dir == BOTH_DIR2_POLAR, flip the bias. Otherwise, do
// nothing.
// (4) The tip moves down in the fast direction, taking measurements
// if appropriate.
// (5) Any data just read (in this_data) is stuffed into all_data.
// (6) The tip is moved over one pixel in the slow direction.
// (7) For scan_dir == BOTH_DIR2_POLAR, flip the bias. Otherwise, do
// nothing.
// (8) Repeat until the the entire image is scanned.
// The program regularly checks to see if the ESC key has been pressed. If so,
// scanning is terminated.
// Note that for all possible values of scan_dir except BOTH_DIR1, the scan
// does not actually raster across the image. Rather, it follows a "comb-like"
// pattern: up, back, over, up, back, over, etc.

```

```

static void update_line( int num, int back)
// Updates one row or column of bitmap_data in memory. SetDIBitsToDevice(...)
// must be called to actually update the image on-screen.

static void set_scan_parameters( int num)
// Updates all fields in all_data[num] to reflect the line just scanned.


// *****
// scan2.h
// *****

// Several routines are defined here. Preprocessor #define's are used instead
// of procedures in order to maximize performance (These routines get invoked
// a lot!).

#define SET_DATA(DATA)
// OLD. Currently unused.

#define READ_Z()
// For each of the read sequences for this scan, turn feedback on/off, wait,
// dither, and record measurements as appropriate.

#define READ_Z_OLD()
// OLD. Currently unused.

#define Z_CALC_MINMAX()

#define Z_CRASH_PROTECT()

#define Z_CRASH_PROTECT_BACK()

#define DO_DIGITAL_FEEDBACK()
// Waits for signal to be between Imin and Imax. Calls dio_digital_feedback.

// *****
// sel.c
// *****

BOOL FAR PASCAL SelDlg( HWND hDlg, unsigned Message, WPARAM wParam,
                        LPARAM lParam)
// A big switch( Message) that handles messages sent to the Selection Functions
// dialog, including scrolling (WM_HSCROLL, WM_VSCROLL) and various buttons
// ( WM_COMMAND's ).

int StartSelection( HWND hWnd, POINT ptCurrent, LPRECT lpSelectRect,
                  int fFlags)

int UpdateSelection( HWND hWnd, POINT ptCurrent, LPRECT lpSelectRect,
                  int fFlags)

int EndSelection( POINT ptCurrent, LPRECT lpSelectionRect)

```

```

int ClearSelection( HWND hWnd, LPRECT lpSelectRect, int fFlags)

int find_coords( SEL_POINT *pt, LPARAM lParam)

// *****
// sharp.c
// *****

BOOL FAR PASCAL SharpDlg( HWND hDlg, unsigned Message, WPARAM wParam,
                          LPARAM lParam)
// A big switch( Message) that handles messages sent to the Sharpen Tip dialog,
// including scrolling (WM_HSCROLL, WM_VSCROLL) and various buttons
// ( WM_COMMAND's ).

// *****
// stm.c
// *****

// One routine is defined here. Preprocessor #define is used instead
// of a procedure in order to maximize performance (This routine gets invoked
// a lot!).

#define READ_DEP_CLOCK()

int PASCAL WinMain( HANDLE hInstance, HANDLE hPrevInstance,
                   LPSTR lpszCmdParam, int nCmdShow)
// WinMain is the entry point for the application ( Like main() in standard
// C programs ). Creates the application's window, allocates memory,
// initializes variables, and sets up the message loop.

long FAR PASCAL WndProc( HWND hWnd, unsigned Message,
                        WPARAM wParam, LPARAM lParam)
// A big switch( Message) that handles messages sent to the application's
// window, including standard Windows stuff ( like WM_CREATE and WM_DESTROY )
// and menu commands that create our dialogs ( WM_COMMAND'S ).

void set_smooth_vals()
// Initializes smooth_s and smooth_l.

void calibrate( HWND hWnd)

// *****
// spec.c
// *****

BOOL FAR PASCAL SpecDlg( HWND hDlg, unsigned Message, WPARAM wParam,
                        LPARAM lParam)
// A big switch( Message) that handles messages sent to the spectroscopy
// dialog, including scrolling (WM_HSCROLL, WM_VSCROLL) and various buttons
// and text fields ( WM_COMMAND's ).

static void calc_integral()

static void calc_part_integral()

```

```

static void find_background()

void calc_didv()

static void repaint_bias( HWND hDlg, unsigned int bias)

static void set_data( datadef *this_data)

static void get_data( datadef *this_data)

int sp_data_valid( int ch)

static int init_spec( HWND hDlg, int ch, int size, int type)

static void sp_status( HWND hDlg, int pass, char *status)

static void spec_set_title( HWND hDlg)

static void sp_update( HWND hDlg, int value)

// *****
// specopt.c
// *****

BOOL FAR PASCAL SpecOptDlg( HWND hDlg, unsigned Message, WPARAM wParam,
                           LPARAM lParam)
// A big switch( Message) that handles messages sent to the spec. opt.
// dialog, including scrolling (WM_HSCROLL, WM_VSCROLL) and various buttons
// and text fields ( WM_COMMAND's ).

static void sp_enable_abs_rel( HWND hDlg)

static void recalc_track_limits()

void sp_calc_scan()

// *****
// subtract.c
// *****

BOOL FAR PASCAL SubDlg( HWND hDlg, unsigned Message, WPARAM wParam,
                       LPARAM lParam)
// A big switch( Message) that handles messages sent to the arithmetic
// dialog, including scrolling (WM_HSCROLL, WM_VSCROLL) and various buttons
// and text fields ( WM_COMMAND's ).

void subtract_data( int a, int b, float sub_x, float sub_y, float sub_z)
// Calculates gendata[a]-gendata[b] and puts the result in gendata[3]
// Does nothing if gendata[a] and gendata[b] are not of the same type

void average_data( int a, int b, float sub_x, float sub_y, float sub_z)
// Calculates gendata[a]+gendata[b]/2 and puts the result in gendata[3]
// Does nothing if gendata[a] and gendata[b] are not of the same type

```

```

void didv( int a, int b, float sub_x, float sub_y, float sub_z)
// Calculates (dI/dV)/(I/V) for two 2D graphs. The data in gendata[a] should
// be dI/dV (selected data). The data in gendata[b] should be I/V (button pressed)
// Result is put in gendata[3]. Does nothing if current image is 3D.

static datadef *create_single_step( datadef *this_data)
// Returns a copy of this_data that has been altered to correspond to a step size
// of one bit.

void average_1_data( int num, float x)
// Averages every x data points for a 2D graph gendata[num]. Does nothing if
// current image is 3D.

void smooth_1_data( int num, float x)
// Smooths data by averaging each data point with x neighboring data points for a
// 2D graph gendata[num]. Does nothing if current image is 3D.

// *****
// summary.c
// *****

BOOL FAR PASCAL SummaryDlg( HWND hDlg, unsigned Message, WPARAM wParam,
                           LPARAM lParam)
// Displays information from the datadef glob_data
// Handles messages sent to the summary dialog

// *****
// tip.c
// *****

BOOL FAR PASCAL TipDlg( HWND hDlg, unsigned Message, WPARAM wParam,
                       LPARAM lParam)
// A big switch( Message) that handles messages sent to the tip approach
// dialog, including scrolling (WM_HSCROLL, WM_VSCROLL) and various buttons
// and text fields ( WM_COMMAND's ).

void tip_approach( HWND hDlg, unsigned int accel)
// Calculates the parabolic waveform for giant steps and then cycles through
// giant steps/baby steps until tunnelling

void giant_steps( unsigned int xn, unsigned int zon, unsigned int *data)
// During a giant step, the three outer piezos rotate the sample
// with a constant acceleration. Then, the three piezos stop and are lowered.
// The sample, still rotating, falls down toward the tip.

unsigned int baby_steps( HWND hDlg, unsigned int n, unsigned int *data)
// During a baby step, the z voltage of the outer piezos, z0, is adjusted
// by small amounts. During baby steps the tunneling current is measured.
// If the tunneling current is greater than the minimum tunneling current,
// tip approach is complete.

void tip_step( HWND hDlg, int dir)

```

```

void pre_tip()

void post_tip()

// *****
// trace.c
// *****

BOOL FAR PASCAL TraceDlg( HWND hDlg, unsigned Message, WPARAM wParam,
                          LPARAM lParam)
// A big switch( Message) that handles messages sent to the trace
// dialog, including scrolling (WM_HSCROLL, WM_VSCROLL) and various buttons
// ( WM_COMMAND's ).

static int trace_in_range( datadef* this_data, int pos)

static void new_trace_el( TRACE_EL **this_list, float width)

static void remove_last_trace_el( TRACE_EL **this_list)

static void destroy_trace_list( TRACE_EL **this_list)

static void trace_save_width( TRACE_EL *this_list, char *ext)

static void trace_bin( TRACE_EL *bin, TRACE_EL *list, float width)

static void trace_save_binned( TRACE_BIN bin, char *ext)

static void trace_save_bin()

// *****
// track.c
// *****

// important global variable
int tracking_mode; /* indicates track for max current or min current */

BOOL FAR PASCAL TrackDlg(HWND hDlg unsigned Message, WPARAM wParam,
                          LPARAM lParam)
// A big switch( Message) that handles messages sent to the tracking
// dialog, including scrolling (WM_VSCROLL) and various buttons
// ( WM_COMMAND's ).

static void recalc_track_limits()
// Calculates the tracking limits in terms of x,y coordinates from limits
// in terms of bits

```



```

void track( int track_auto_auto, int high_limit, int low_limit,
           int track_avg_data_pts, int tracking_mode, int step_delay,
           int inter_step_delay, int track_limit_x_min,
           int track_limit_x_max, int track_limit_y_min,
           int track_limit_y_max);
// track looks for a max (min) in the tunneling current. The current is
// measured at a particular x,y pixel and at each of the eight neighboring
// pixels. The tip is then moved to the pixel with the highest (lowest)
// tunneling current. The process is repeated.

// NOTE that the local variable tracking_mode is different than the global
// one. The global variable is set to either TRACK_MAX or TRACK_MIN. The local
// variable is essentially a boolean, set to non-zero if in TRACK_MAX mode

// *****
// vscale.c
// *****

BOOL FAR PASCAL VscaleDlg(HWND hDlg unsigned Message, WPARAM wParam,
                          LPARAM lParam)
// A big switch( Message) that handles messages sent to the tracking
// dialog, including scrolling (WM_VSCROLL) and various buttons
// ( WM_COMMAND's ).

// Vertical scale allows the vertical scale of a 2D graph to be rescaled. All
// controls are disabled if the current image is 3D.

```