ZyVector™

# *STM Lithography System*

# User Manual

## for Scanz 3.0

A Scanning Tunneling Microscopy Control System for Atomically Precise Lithography

August 2018

zyvex™ LABS

Zyvex Labs, LLC.
1301 North Plano Road
Richardson, Texas 75081, USA
www.zyvexlabs.com
972.792.1671

**Information**: info@zyvexlabs.com

This User's Guide originally written in Fall 2008 by Jim Von Ehr.
2009 revisions by Yu Ding and Jim Von Ehr.
2015 revisions by J. N. Randall, J. Ballard, E. Fuchs, S. W. Schmucker and J.H.G. Owen
2017 revisions by J. Ballard, E. Fuchs, and J.H.G. Owen
2018 revisions by J. Lake, H. Alemansour, and J.H.G. Owen

## Acknowledgements and references

We thank Professor Joe Lyding, at the University of Illinois, Urbana-Champaign, and his many graduate students, for their extraordinarily valuable STM design and great help in our project.

Neither APM nor ZyVector would have become what they are without the generous funding of DARPA, the State of Texas, the Army Research Office, and Zyvex Founder Jim Von Ehr.

Zyvex Asia played an integral role in the development of APM, building the original scanners, UHV systems, and software required for APM. Dr. Lerwen Liu was highly instrumental in getting this entity set up and underway, and we thank all the Zyvex Asia staff for their involvement.

Singapore's Institute for Material Research and Engineering was among the first government agencies to recognize the potential of APM, and we thank them, and particularly IMRE Director Dr. Khiang Wee Lim for their vision, and the help of our many IMRE collaborators. Singapore's Economic Development Board recruited us to Singapore, and was instrumental in fostering this collaboration.

We also wish to thank Zyvex Instruments for assistance with various vacuum issues, SEM access, and machine shop time.

We are indebted to Scienta Omicron for excellent support in adapting ZyVector to their VT STM tool.

And finally, we want to thank our spouses and significant others for their patience during this lengthy development process.

# Safety Instructions

## CAUTION

The following safety precautions must be observed while configuring, operating, testing, servicing, or repairing the system. Failure to comply with the safety precautions or warnings in this document is a violation of the intended use of this document and may impair the built-in protections herein. The safety of any system incorporating this equipment is the responsibility of the assembler of the system. If the equipment is used in a manner not specified by the manufacturer, the protection provided by the equipment may be impaired.

## GROUNDING

The ZyVector Lithography System is a Safety Class 1 instrument. During operation, to minimize shock hazard, the instrument chassis of the ZyVector analog electronics box and power supply must be connected to an electrical ground through the provided protective earth terminal. The instrument must be connected to the AC power supply mains through a three conductor power cable, with the ground wire firmly connected to an electrical ground at the power outlet. Any interruption of the protective ground conductor, or disconnection of the protective earth terminal, will cause a potential shock hazard that might cause personal injury.

## INPUT RATINGS

The input voltage and frequency rating of this equipment is 115/230V (configured on the ZyVector Power Supply by Zyvex Labs qualified personnel as part of installation), 50-60Hz, single-phase AC power, 250VA maximum. For safety reasons, the mains supply voltage fluctuations should not exceed ±10% of nominal voltage.

The detachable mains supply cord should be replaced only by adequately rated cords. Other electrical cables should be replaced only by the manufacturer or his agent.

## LIVE CIRCUITS

Personnel operating this instrument should not remove the instrument cover. No adjustment or component replacement of internal systems is permitted except by Zyvex Labs qualified personnel. Never replace components with the power cable connected. To avoid injuries, always disconnect mains power, discharge circuits, and remove external voltages sources from the equipment before touching any live components.

## PARTS SUBSTITUTIONS AND MODIFICATIONS

Substitution or parts and modifications to the equipment are not permitted, except by authorized Zyvex Labs service personnel. For any required repairs or modifications, the equipment should be returned to Zyvex Labs or to a service facility authorized by Zyvex Labs.

## ENVIRONMENTAL CONDITIONS

The ZyVector Lithography System safety approval applies to the following operating conditions:

Indoor use
Altitude up to 2000m
Temperatures ranging from 5°C to 40°C with maximum relative humidity 80% for temperatures up to 31°C decreasing linearly to 50% relative humidity at 40°C

## INSTALLATION REQUIREMENTS

This equipment is designed for installation on a surface or within a rack-mounted configuration, and requires adequate ventilation for safe operation. Equipment is supplied with rubberized footings, which must be installed during operation for adequate ventilation. Vent openings within the included Data Translation DSP module, and any other equipment, must remain accessible and unrestricted for adequate ventilation. The equipment should be installed so as not to restrict access to the main power cord.

## CLEANING OF THE EQUIPMENT

The equipment is designed to require minimal cleaning and maintenance. If necessary, the outer surfaces of the equipment can be cleaned with a damp cloth. No solvents or strong cleaning agents should be used. Before cleaning, equipment should be powered down and all electrical wiring disconnected.

## AC CABLES

If an AC cable is required, it should be ordered in accordance with the following specifications:

USA 10A, 125V, unshielded, 6ft typical length. IEC 320 connector on one end, NEMA-5-15P connector on the other
UK 10A, 250V, unshielded, 2m typical length, IEC 320 connector on one end, BS 1363 (Type G) connector on the other

## FCC COMPLIANCE NOTICE:

This equipment generates, uses, and can radiate radio frequency energy and if not installed and used in accordance with the instructions in this manual, may cause interference to radio communications. It has been tested and found to comply with the limits for a Class A computing device pursuant to Subpart B of Part 15 of the FCC Rules, which are designed to provide reasonable protection against such interference in which case the user at his own expense will be required to take whatever measures may be required to correct interference.

Changes or modifications to this equipment not expressly approved by Zyvex Labs could void your authority to operate the equipment.

This product was verified to meet electromagnetic compatibility and FCC requirements under test conditions that included use of certain shielded cables and connectors. It is important to use shielded cables and connectors where applicable to reduce the possibility of causing interference to radio, television, and other electronic devices. Unless otherwise stated herein, cables should be examined or supplied only by the manufacturer or his agent.

## CANADIAN DEPARTMENT OF COMMUNICATIONS STATEMENT:

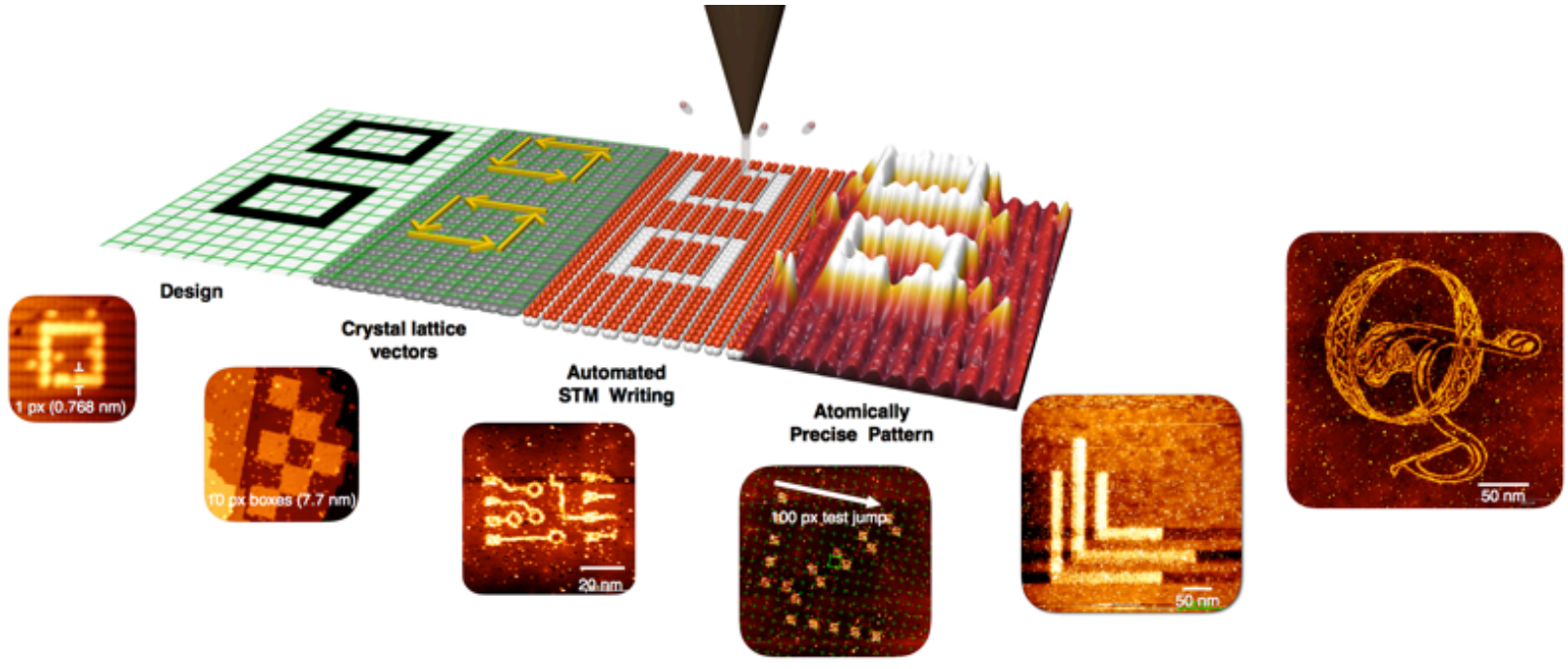
This digital apparatus has been tested and does not exceed the Class A limits for radio noise for digital apparatus set out in the Radio Interference Regulations of the Canadian Department of Communications. The regulations are designed to provide reasonable protection against radio noise interference in which case the user at his own expense will be required to take whatever measures may be required to correct interference.

## EUROPEAN COMMUNITIES ELECTROMAGNETIC COMPATIBILITY:

This equipment has been tested and found to comply with the limits of CISPR 22 and EN 55022 Class A for information technology equipment. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications.

This is a Class A product. In a domestic environment, this product may cause interference in which case the user may be required to take adequate measures.

# Table of Contents




## Overview

The ZyVector Lithography System (ZyVector for short) was created to support the development of Atomically Precise Manufacturing (APM). In particular, ZyVector enables atomically precise (AP) nanolithography using an Ultra High Vacuum (UHV) Scanning Tunneling Microscope (STM) to build AP structures on silicon. This User's Guide describes how to use ZyVector to image and pattern at the atomic scale.

This system gives the user an easy path towards AP lithography capability, which can be coupled with gas-phase chemistry for:

- Patterned Dopant Deposition to make phosphorus atom qubits and other nanodevices.
- Area-Selective Atomic Layer Deposition (ALD) to make hardmasks and other structures.
- Area-Selective Atomic Layer Epitaxy (ALE+) to make atomically precise 3D structures.
- Area-Selective Atomic Layer Etching (ALE-) to make atomically precise 3D structures.

## Hydrogen Depassivation Lithography

Hydrogen-passivated silicon provides an intrinsically digital, atomically-precise platform for AP Lithography. The H layer may be thought of as an atomically-thin, self-developing resist layer. An STM may be used as a lithography tool for extremely high resolution patterning by removal of H atoms from the Si(001) surface.  This is known as Hydrogen Depassivation Lithography (HDL). HDL has much in common with e-beam lithography:

- It has an electron source: at positive sample bias, the STM tip is essentially a cold field emitter.
- It delivers a spatially confined electron beam to a surface where it exposes a resist.
- It uses multiple spot size modes to expose fine features and larger features.
- In e-beam lithography terms it is a Gaussian-beam, variable-spot, vector-scan exposure tool.

However, some characteristics of HDL are significantly different from conventional e-beam lithography. For instance:

- The monolayer of H atoms self-develop by electron-stimulated desorption.
- HDL functions in two different lithography regimes. While these are superficially similar to the small spot and large spot modes in conventional e-beam lithography, they differ mainly in the voltage of the electrons rather than just the current.
  - In the small spot mode, the bias is only about 4 volts and the tip is in tunneling mode. This is known in ZyVector as Atomically Precise (AP) mode. The efficiency of H atom extraction is very low, so that the electron doses required are relatively high, and the exposure process is very slow, but the results have sub-nm resolution with very clean edge definition.
  - The large spot mode uses higher biases (8-10V), for which electrons are field-emitted from the tip. This is known in ZyVector as Field-Emission (FE) mode. The spot size is significantly larger with a significantly rougher line edge. However, the efficiency of H removal is several orders of magnitude greater allowing much faster exposure of larger area features.
  - Similar to conventional e-beam lithography pattern compiler tools, ZyVector fractures the patterns into different regions to be exposed by the small spot and large spot modes.
- This monolayer of H atoms is different from most resists in that it does not directly make an effective mask for etching and there is no "lift off" process. Also, there is no such thing as partial exposure; either the H-Si bond is broken, or it is not. However, it can be used for:
  - precise placement and study of dangling bond dots and wires.
  - selective deposition for patterned Si and Ge Atomic Layer Epitaxy (ALE).
  - selective deposition of dopant atoms (at least P and possibly others).
  - selective deposition for area-selective Atomic Layer Deposition (ALD) ($TiO_2$).
  - precise placement of a large variety of atomic and molecular species.
- There is a low-voltage imaging mode that does not expose the resist, using the same probe as used for writing. This results in several fortuitous capabilities not available in most other lithography tools.
  - The Si(001) surface organizes itself into dimer rows of surface Si atoms where the dimers are two Si atoms each with a single unsatisfied covalent bond to which a H atom can be attached. This regular lattice can then be imaged nondestructively and used as an address grid for patterning. In effect, the surface lattice becomes a global fiducial grid, and the STM tip can position itself on this grid before writing.
  - ZyVector can write its own alignment marks and return to register to them.
  - Alternatively, features which form part of a written pattern can be used for alignment.
  - ZyVector can examine a pattern after writing and, in some cases, correct errors in the pattern.

## ZyVector as a Digital Lithography Tool

These differences from conventional e-beam lithography allow us to implement something that is very different from virtually all other lithographic techniques, which we refer to as a digital patterning process. Virtually all other high resolution lithographic techniques treat matter as if it is infinitely divisible, providing fundamentally analog fabrication techniques. In contrast, in ZyVector, HDL has been realised as a digital patterning technique because:

- The resist response is digital. Either the H-Si bond is broken or it is not. There is no accumulation of partial exposure that leads to proximity effects.
- The resist is laterally (and vertically – being a monolayer) quantized.
- The lateral quantization is used in ZyVector to form sub-nm pixels. Each pixel is formed of two dimers on a single dimer row, as shown in Figure 1.1. The pixel size is 0.768 nm. Sub-px lithography is also possible, down to single-atom removal.
- The state of each pixel may be read and written by the same tool.
- A hallmark of a digital system is that there is some tolerance of the parameters of the system. If this tolerance is not exceeded, the output of the digital system is essentially perfect. We have determined that the tolerance in tip position is ~±0.15 nm. In other words, perfect tip positioning is not required to do perfect patterning. The tolerance is not large, but it is within the capabilities of current technologies.

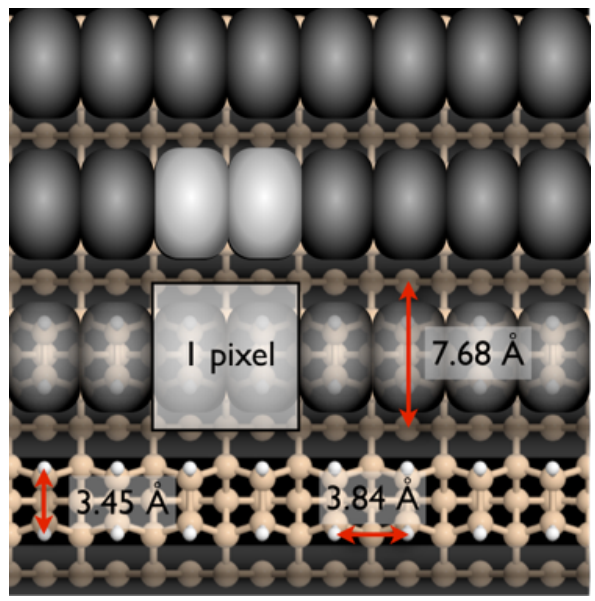



**Figure 1.1: Schematic of the Si(001):H surface, showing the 2-dimer pixel used as a basis for digital lithography in ZyVector.**

## The ZyVector Lithography Approach

Through a combination of control hardware and software, ZyVector connects the user to the UHV STM hardware, which images and patterns the surface atoms. The goal is to deliver a powerful, automated nanolithography system with excellent ease-of-use, and to support easy extensibility as new capabilities are needed.

The ZyVector SCANZ software supports user interactivity in three ways:

- Via the standard user interface of clicking buttons, entering text in fields, etc.
- Via an interactive window allowing line-by-line Python command execution
- Via supplied and user-written scripts which can execute more complex code

### Automation

While most operations can be carried out by an operator in a manual fashion through our sophisticated user interface, many basic functions are already automated, including tip approach, creep and hysteresis correction (after calibration), and lattice identification.

We have also provided a Python-based scripting language which allows users to create scripts so that the system can run unattended while carrying out complicated lithography tasks or experiments designed by the user. Unlike typical microscopy applications, our aim is to minimize operator time, and to maximize productivity, instead of locking users into performing tasks envisioned by their designers. ZyVector offers a modern programming environment with the flexibility for users to write their own routines to automate tasks.

### Pattern Entry

Patterns for lithography can be entered in three different ways:

- Via the graphical user interface by selecting among numerous options and combinations.
- Via the script window to produce primitive geometries entered at the command line or more complex geometries detailed in a script.
- Via automated vector generation of a desired pattern from a binary bitmap image, with a bitmap pixel corresponding to the 0.768 nm pixel defined on the silicon (2×1) surface lattice.

In the case of bitmap entry, our software will automatically fracture the pattern into parts to be written by the AP and FE modes, generate the tip vectors in an optimized fashion, and order the vectors (a “route inspection problem”) to minimize tip travel and mode transitions. Additionally, the scripting capabilities allow the user to read in their own list of lithography operations, and therefore leverage vector generation tools.

### Patterning Accuracy

While we regularly image the surface to identify the lattice and/or written alignment marks, doing so continually would make a relatively slow patterning process even slower. There are several positional errors that affect tip position: Thermal drift, lattice miscalibration, piezo creep and hysteresis, and tip changes to name just a few. Under the best of conditions with an STM, it is extremely difficult to do perfect patterning. While Zyvex Labs is working on a better understanding of the tip/surface interactions during HDL, we do have the capability to position the tip to enable near perfect patterning in a practical time frame, typically with sub-nm (Atomic precision or ± one atom spacing) precision in pattern placement over a write field of ~100×100nm.

To achieve this level of precision and accuracy we have developed an automated real time creep and hysteresis correction algorithm and automated lattice identification algorithms. Write fields can be stitched together to make larger atomically precise patterns.

For patterns (or subsections of patterns) where atomic precision may not be required, larger spot sizes can be used exclusively over larger write fields to achieve much more time efficient exposures.

### Definition of Terms

- User interface fields and action buttons will be indicated in the manual as underlined text.
- Coordinates and units will be indicated by italics whenever appropriate.
- Scripts and command line functions will be indicated by Courier font.

Other formatting such as bold will be used for emphasis.

# Chapter 2: ZyVector Hardware and Software Overview

## ZyVector Controller Hardware

The ZyVector Controller consists of several components, including a computer controlled digital signal processor (DSP), a high voltage power supply, an analog high voltage box, and associated cables. Figure 2.1 shows the assembled components and the PC software, SCANZ. ZyVector can be adapted to work with a variety of STMs, including one of the most popular Scanning Tunneling Microscopes (STM) the ScientaOmicron VT, shown in Figure 2.2. The system replaces a Matrix or other STM controller completely.

**Figure 2.1: ZyVector System. Top: Analog High Voltage Box. Middle: Power Supply. Bottom: DSP.**

**Figure 2.2 One configuration of the Omicron VT STM.**
**(reprinted here with the permission of ScientaOmicron)**

## System Architecture

As shown schematically in Figure 2.3, the ZyVector system comprises several parts.

ZyVector SCANZ, the primary control software described in detail in upcoming chapters, operates on a PC under Microsoft Windows® 7 or higher for user interfacing and generation of control statements, with specialized drivers loaded on a DSP; communication between the PC and the DSP is performed over a standard Universal Serial Bus (USB2) or ethernet connection. The DSP interfaces the control computer and analog box, converting instructions into STM control waveforms with an update frequency of either 50 kHz or 100 kHz, depending upon configuration.

The DSP performs real-time positioning control including creep and hysteresis correction, tip height control using a P/I feedback loop, and voltage control for lithography and imaging applications. The DSP uses one analog to digital input for tunneling current detection, eight analog outputs, and several digital outputs as summarized in Table 2.1 below. These signals are delivered to/from the analog box via DC-37 cables, as described below.

The analog box converts the outputs from the DSP (±10V) into control voltages to drive STM piezo motors. The X, Y, Z Coarse channels drive the coarse piezo motors at ±200 V. The X, Y, Z Fine channels drive the piezo tube scanner at ±135 V, with two configurations possible, shown in Figure 2.4. This configuration is jumper selectable, and should only be modified by a trained technician.

Providing power to the system: For optimal system noise performance, all components of the system (including any control computer, oscilloscope, spectrum analyzer, etc.) should share the same isolated power source with no unnecessary components attached. Isolation units could be an isolation transformer or an Uninterruptible Power Supply.

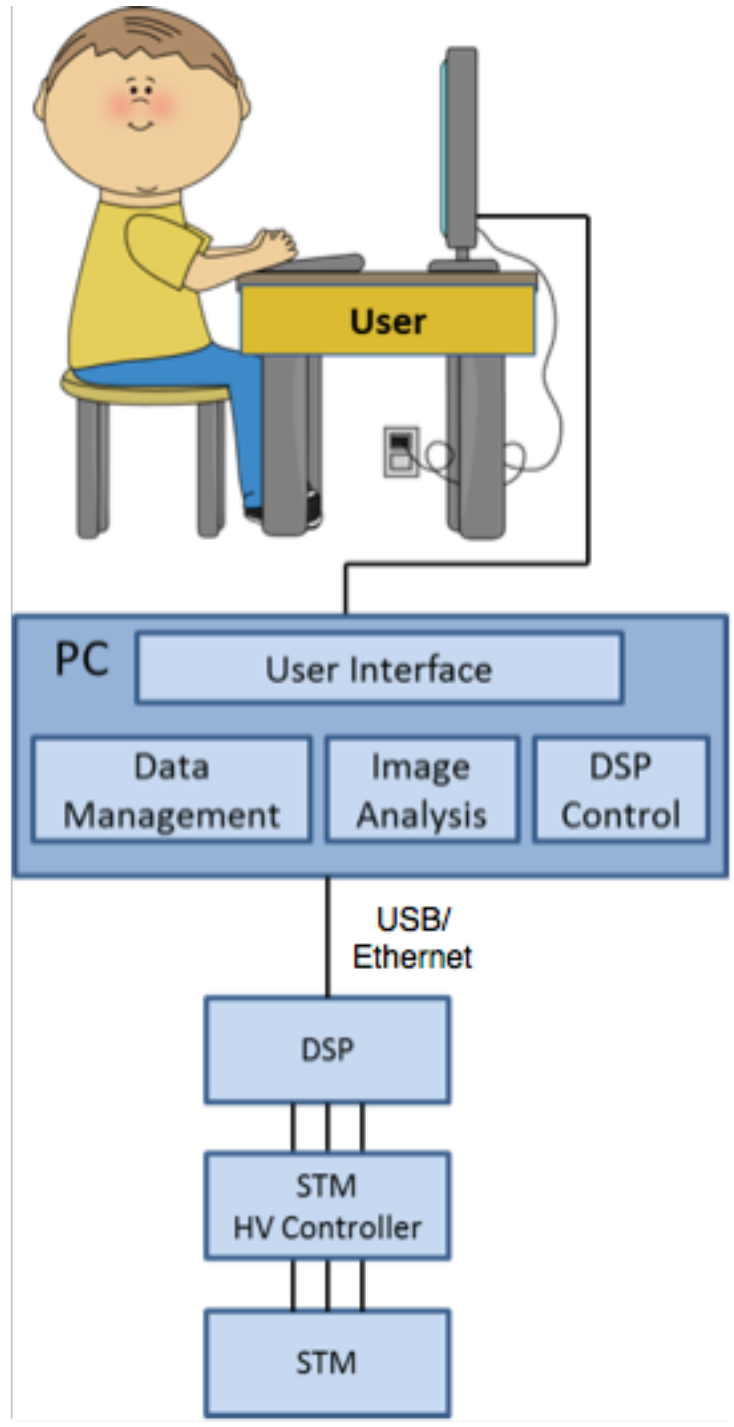



**Figure 2.3: ZyVector system architecture**

| |
|---|
| **Analog In:** |
| amplified STM current |
| **Analog Out:** |
| X, Y, Z Coarse |
| X, Y, Z Fine |
| Bias |
| Spare |
| **Digital Out** |
| X, Y, Z Coarse filter switches |
| X, Y, Z Fine filter switches |
| X/Y, Z Fine range selections |
| Pre-amp current range control |
| Pre-amp bias range control |
| Signal monitoring selection |

**Table 2.1: List of inputs and outputs from the DSP**

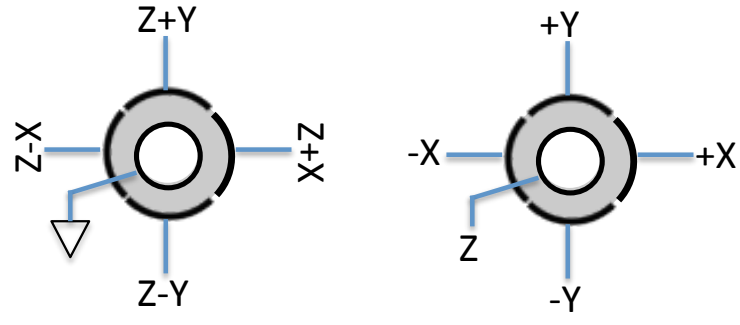



**Figure 2.4: Fine motor configurations.**

**Left: X, Y, and Z signals are mixed in the analog box, with the inner part of the piezo tube grounded (default setup). Right: Optional configuration with Z controlled by a voltage on the inner part of the tube.**

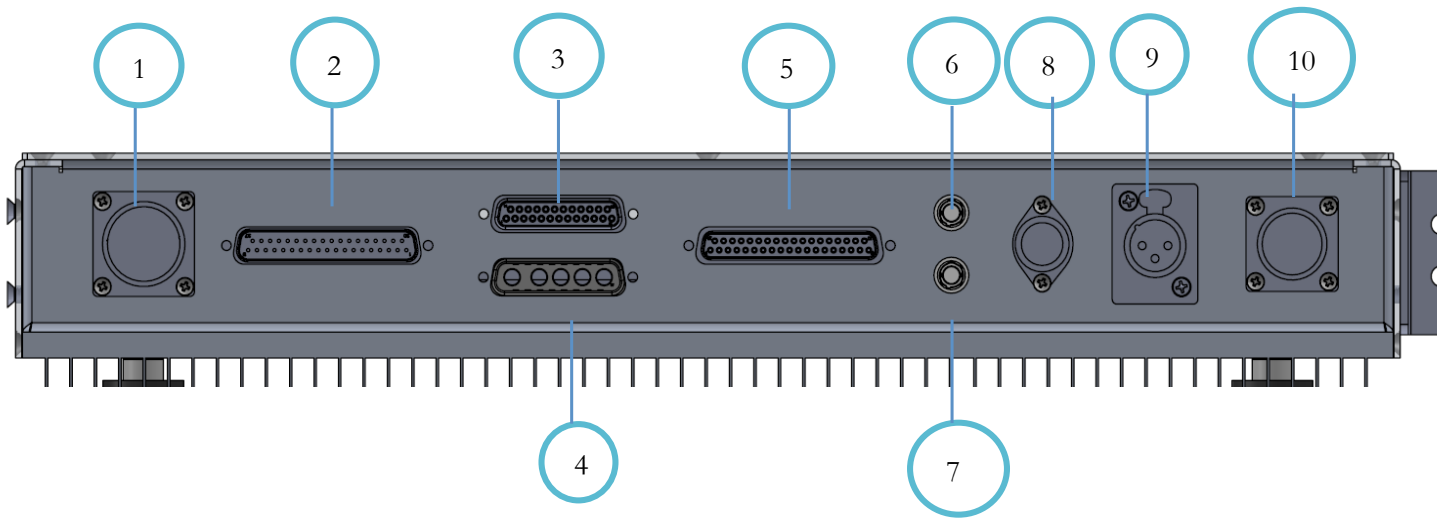



**Figure 2.5: Analog control box connections.**

## Analog control box socket list.

1. Power-1. Power supply lines from the Power Box.
2. Digital I/O.
3. HV Coarse Amplifier outputs.
4. HV Amplifier Outputs.
5. Analog I/O.
6. Bias. Output to the Pre-amplifier.
7. Current. Input from the Pre-amplifier.
8. Pre-Amplifier Power and Control signals.
9. 5VDC. Output to the Pre-amplifier.
10. Power-2. Pre-Amplifier related power supply connections from the Power Box.

## 20-bit Digital Control box LED Front panel

The LEDs on the front panel of the 20-bit DSP provides important status information about ZyVector. A list of LED states with their indications are listed in the table below. The LCD front panel display also provides more information, such as the tunnel current and voltage, even when the PC software Scanz is not working.

| State | Status | Alert |
|---|---|---|
| Initial boot | Red | Off |
| Booting | Yellow/red | Off |
| Waiting for PC | Yellow | Off |
| PC disconnecting (momentary) | Yellow | Red |
| Connected – disengaged | Green | Off |
| Connected – engaged | Green | Green |
| Communication lost | Green | Yellow |

**Table 2.2: LED lights on the DSP front panel**

## Connecting ZyVector

To connect ZyVector to an Omicron VT system, please refer to Figures 2.5 and 2.6:

- Place digital control box, analog box and power supply box in a rack so that there is access to the STM control cables without straining them.
- Connect power supply to analog box sockets 1 and 10 using round ended cables.
- Connect digital control box to analog box sockets 2 and 5 using provided DC-37 cables. Note that one cable is male-male and one is female-female, so attachment between the digital and analog boxes is unambiguous.
- Connect control computer to digital control box using a USB to microUSB cable.
- Attach external filter box to “Current” BNC socket 7 (optional).
- Check STM tip is well-retracted, and all Omicron components are shut down and powered off.
- Disconnect the following cables from the back of the Matrix box. See Fig. 2.6.
  - Red and blue BNC cables ($I_T$ and Bias) and 5 pin DN Pre-amp Control Cable from Preamp section.
  - Plug marked Piezo from right hand board.
  - Plug from Coarse Motion board
  - Two ground cables (blue and green/yellow) from near Coarse Motion plug.
  - 3 pin plug from power socket middle top of Matrix controller.
- Plug in STM control cables into the back of the analog box.
  - Attach blue bias BNC cable to the “Bias” BNC socket 6.
  - Attach red current BNC cable to the “Current” BNC socket 7, or to external filter box.
  - Attach 5 pin DN Pre-amp Control Cable to socket 8.
  - Attack 5 V pre-amp power supply to socket 9.
  - Attach DB-25 high voltage coarse positioning cable to socket 3.  Screw in strain relief bolts.
  - Attach 5 pin PIEZO plug to socket 4. Screw in strain relief.
- Plug-in to 110/230V power the digital control box, controller power supply, and any other system components as necessary.

**At this point, the system should be ready to operate the STM.**

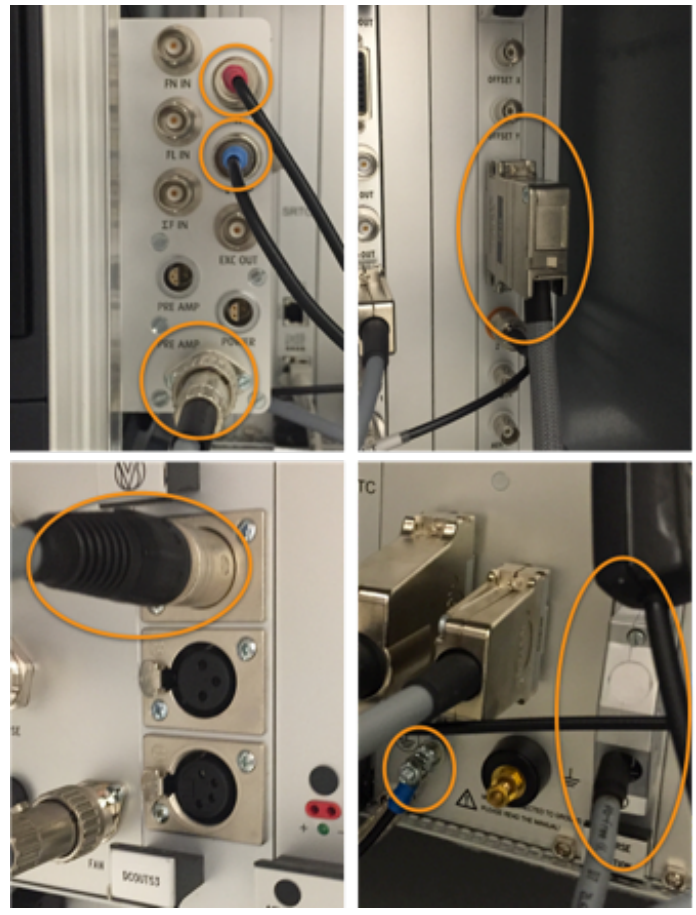


**Figure 2.6: Omicron connectors to be unplugged.**

**Upper Left: The Red and Blue BNC connectors and the 5-pin power and signal cable for the Omicron preamp.**

**Lower Left: The 3-pin power cable for the Omicron preamp.**

**Upper Right: Fine Piezo cable**

**Lower Right: Coarse Motion cable and two ground connection cables for preamp and piezo boxes.**

## Running ZyVector

The ZyVector analog box and DSP should be turned on before starting the software. Once it has finished booting, the DSP front panel will announce, “Waiting for PC...”. Double-click the ZyVector SCANZ icon on the Desktop. SCANZ should start up and quickly display the window in Figure 2.7 and start updating values such as tunneling current in the Status Panel. Messages in the Info Panel – the bottom middle window pane should indicate that the DSP has been found and that the analog box is connected. Otherwise, it should give an error message to indicate the source of the problem. Typically, if there is a problem with connecting to the DSP, Scanz automatically enters “Null DSP mode”. This is indicated by a message in the Info Panel. Restarting the DSP, waiting for 1 minute until it finishes booting, and then restarting the software will usually cure this issue. Also check the LED display on the front panel of the DSP. This should give some indication of any problem.

In the System tab, set the current user ID, and confirm that the Piezo box and Piezo motor settings are set correctly. For example, with the complete ZyVector system driving an Omicron VT, they should read “ScanZ Box” and “Omicron” respectively. UserID is used to identify the current user of ZyVector. Each user may have their own user profile, which saves and loads numerous setup parameters between runs. These profiles may be loaded, saved, or reset to default values by commands in the File menu.

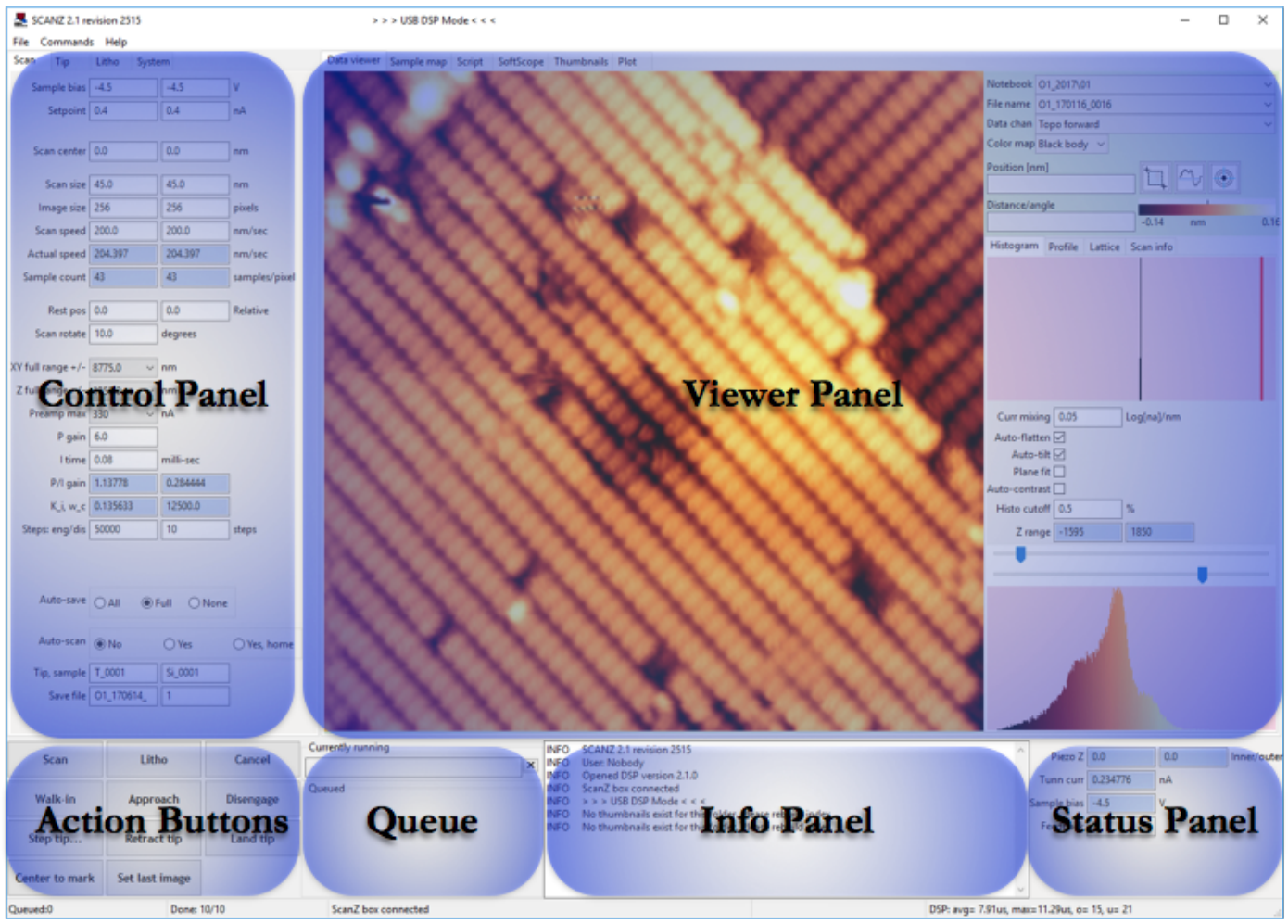



**Figure 2.7: ZyVector main window showing the various parts of the user interface shaded in blue.**

## Introduction to the User Interface

ZyVector SCANZ runs in one main window, shown in Figure 2.7. On the left side of the program window is the Control Panel, containing several tabs for different functions. The main Scan tab is meant to contain all the settings that the user will typically use during scanning. Some settings which are useful for the Auto Approach process are found in the Tip tab. Lithography settings are found in the Litho tab, which is discussed in the Lithography chapter. Other parameters, for piezo calibration, and creep and hysteresis correction, are found in the System tab (See Chapter 7).

The Viewer Panel contains several tabs, which can be moved to separate windows by right clicking on the tab and selecting “Move to new window”. The primary tab, which cannot be separated, is the Data Viewer tab, which shows the live STM image with a side panel containing various controls. The Sample Map shows the history of the current STM session, with thumbnails showing the location of each scan. Saved data can be viewed in the Thumbnails tab. The Script tab provides the main interface for automation of SCANZ. This, and other aspects of the program such as the Softscope and Plot tabs are addressed in their respective chapters.

Along the bottom of the program window are the Action Buttons, the Queue, the Info panel, and the Status panel. The Action buttons initiate actions such as coarse motion, auto approach, scanning, or lithography, following the settings made in the Control Panels. Commands, including parameter changes, Actions, and scripts, are all listed in order in the Queue. Some information about the ongoing actions is shown in the Info panel, which will record file saves, xy coarse motions, errors etc. The Status Panel in the lower right corner gives a live status of the bias voltage, tunnel current and xyz positions.

## Interface Threading

To provide an interactive and responsive application, ZyVector SCANZ has several interconnected threads of control running simultaneously. These communicate with one another to improve the user experience and to make ZyVector SCANZ's internal structure cleaner. To accomplish this, Scanz uses a queue system.

The User Interface thread handles all communication with the user and the computer's file system. This collects commands (such as Scan or Set bias voltage) from the user or from a script and sends those to the Background Scan thread. The Background Scan thread is responsible for all communication with the actual DSP hardware and STM controller. Some DSP operations take significant time to execute, so are run in their own thread to ensure that the User Interface thread is always responsive.

Two scanning modes are provided. Full Scan is used in scripts, and takes a whole image before accepting any inputs. This means that while an image is scanning, parameters can be modified but the changes will not be implemented until the scanning stops. In this mode, the parameter background will become yellow to indicate that the change is pending, and the change will also show up as a pending item in the Queue.

Live Scanning is an interactive scanning mode. In this mode, changes in bias, current or position etc. will be implemented during the scan. The only parameter which cannot be changed is the number of pixels in the image. The typical use of this mode is for tip conditioning, as voltage pulses can be applied to the tip during scanning, and the results are seen immediately.

## Units of measure in ZyVector

There are several unit systems used by ZyVector.

Times are measured in seconds.

Voltages are measured in volts, with occasional usage of *DAC units* (DACU).

Distances are usually measured in nanometers (nm), with occasional usage of *DAC units* (DACU). Pixels are used as a unit for lithography.

Current is usually measured in nanoamps (nA), with occasional usage of *DAC units* (DACU).

These values start out as an analog signal in the range of ±10V. This voltage is translated in the DSP to a 20-bit integer in the range ±1024576, which are referred to in the software and in the manual as DACU.

DACU values for parameters (e.g. volts) can be converted by the formula: *volts_out = 10.0 V * DAC_units / 1024576*

To calculate the voltage being sent to the STM piezos, there is also a multiplication in the analog box. The Matrix piezo controller uses ±135 V for the tube scanner and therefore the ZyVector analog box uses a ±10 V input multiplied by 13.5 to make 135 V. Values for fine X/Y/Z can also be scaled down by a multiplying DAC (MDAC) which multiplies them by a programmable factor less than one to provide a variable voltage range for fine positioning in X, Y, and Z. This scaling value is known as the *vernier* , with three scaling factors of 1, ¼, and $^{1}/_{14}$. This is important for the legacy 16-bit DSP. The 20-bit DSP can be run in the 1x scale at all times.

For automated AP lithography, we have introduced the litho pixel, which is a set of 4 atoms, or 2 dimers on a single dimer row of the Si(001) surface, making a 0.768 nm square. When writing patterns that require atomic precision, it is convenient to define patterns in pixels, not nm, and they will be written via tip vectors which run along the top of the center of the dimer rows.

## Orientation of tip/sample interaction

The operating principle of an STM is to bring a sharp tip into close proximity with a sample (typically with a tip-sample spacing of about one nanometer) to detect electron tunneling between tip and sample. A bias voltage is applied between the tip and sample. In the Omicron VT system, the sample is held at *virtual ground*, and the tip is biased as requested, from -10V to +10V. In Scanz, the sign of the bias voltage refers to the sample bias relative to the tip.

Coordinate measurements are made in a right-handed, tip-centric coordinate system. Imagine you are sitting astride the tip wire, looking down the tip at the sample: +X is to the right, +Y is straight up, and +Z is towards you. When zero voltage is applied to the piezo tube, the end of the tip is considered to be coordinate {0, 0, 0}. If the tip pulled away as far as it could go, its coordinate would be Z=100%. If it pushed into the sample as far as it could go, its coordinate would be Z=-100%. The actual distance for the tip away from neutral will then depend upon the system calibration and gain.

# Chapter 3: STM Imaging

While the focus of the SCANZ software is STM lithography, atomic-resolution imaging capability is an essential part of the writing process and can be carried out automatically as a part of a script or higher level function. In particular, images are required during lithography runs for:

- Pre-check of sample quality - terrace width, defect densities.
- Pre-check of tip quality.
- Determination of local phase and angle of the Si(001) dimer rows
- Optimization of parameters for AP and FE mode lithography
- Determination of surface position relative to fiducial marks, previously written patterns, etc.
- Check quality and accuracy of patterning after completing lithography to identify errors and perform error correction.

In this chapter, the operation of the STM as an imaging tool is described, followed by a summary of how various imaging tasks would be performed. The various Control Panel tabs are described, followed by the Viewer Panel tabs, followed by the other parts of the User Interface.

## The Scan tab

The Scan tab, shown in Figure 3.1, includes all the parameters which are frequently changed during a typical imaging session, such as the Sample bias, and the tunnel current Setpoint, and all the position settings and image parameters.

These values are changed by typing a new value into the box, and then either hitting Enter or accessing another field to activate the change. If an invalid value is entered, such as a Sample Bias voltage over 10 V, SCANZ will ignore the change. By hovering over a field, a tooltip will appear, describing what it does.

In some cases, there are two fields for each parameter. In some cases, these are linked so that changing the first one causes the second to change to the same value. The second may then be edited to a different value if desired.

The Scan Center fields may show yellow during large movements in xy which can take several seconds, to indicate that the tip is moving.

### Basic Scanning Controls

For the Sample bias, tunnel current Setpoint and the Scan speed, the two fields refer to the forward and reverse scan directions. For the position settings, such as Scan center, the two fields refer to the x and y directions.

The Scan size is the size of the image in nm, while Image size is the resolution of the STM image in pixels, (not to be confused with the lattice distance unit of pixel, described below), which can be set to any value, but is typically set to factors of two, usually 256 or 512. A different number of points along a scan line and number of scan lines down an image may also be set. The Scan speed is defined in nm/s, therefore a large value in the first Image size field will not slow down the scanning, but gives more resolution (but fewer data samples per point). More scan lines, defined in the second value of Image size, will increase the time taken to capture the whole image. Scanning too fast may cause artifacts in the image, as the feedback loop may respond too slowly to large surface features.

The Rest Position (Rest Pos) refers to the location on the image that the tip returns to at the end of a scan. This ranges from -1 to +1, in x and y, with (0,0) being the center of the image. Usually this is set to the top middle, (0,1). This setting can affect short-term creep during imaging or at the start of lithography.

The XY and Z full range fields define the maximum operational area of the STM, according to the nm/V calibration in the System tab. Due to creep and hysteresis corrections, the actual maximum scan size may be a little smaller than this.

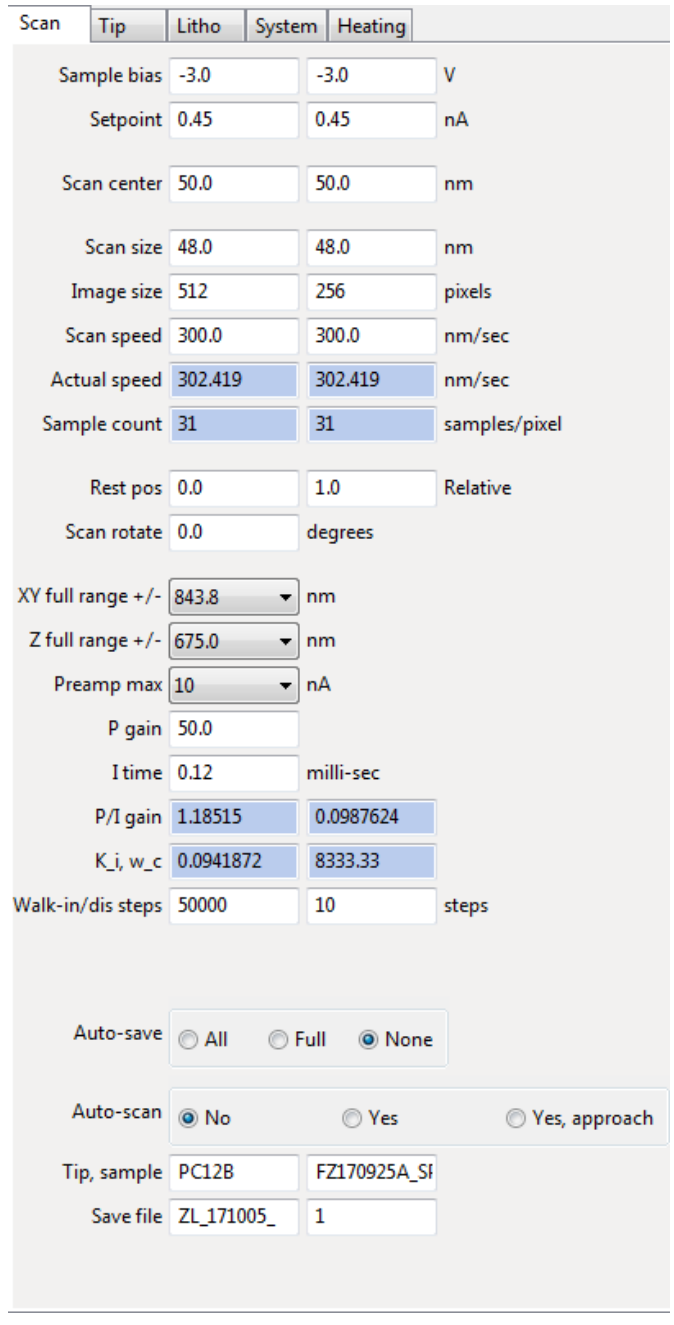



**Figure 3.1. The Scan Tab.**

## Feedback Loop Gain Settings

The sensitivity of the feedback loop is set by the product of the PI (Proportional and Integral gain) parameters and z range. It may be necessary to tweak these from the values set during installation. Values around 50 for P gain and 0.1 ms for the I time have been used successfully on an Omicron VT system. In general, a good way to optimize them is to increase the P value until the feedback loop starts to oscillate, as viewed on an oscilloscope or in Softscope, and then halve that value. Then repeat this process with the I parameter. This should give good sensitivity, without control loop overcorrection. Different values may also be required when operating in the 3.3 nA or 330 nA Preamp max ranges.

## Frames of References

The physical motions of the piezo scan tube define the Scanner frame of reference. Movements in x or y using Scan center follow this frame of reference. Scan rotate rotates the image relative to the scanner frame of reference. A positive angle rotates the scanning in a clockwise direction, so that surface features will appear to rotate in a counter-clockwise direction, as shown in Figure 3.2. The lattice directions use the same convention, so that a positive lattice angle will slope down to the right of the image. The lattice parameters are determined by the Fourier transform, based on the image frame of reference, however. Thus with an image rotation of 0°, the dimer row direction may be given as 10° while for an image rotation of 45°, it may be given as -35°. The default lattice parameters, however, are defined relative to the piezo frame of reference, so that the lattice is always correct for any image rotation angle. The sample frame of reference has primary directions parallel and perpendicular to the surface dimer rows, which are <110> crystal directions. In this frame of reference, we can use px as a unit. 1 px = 0.768 nm × 0.768 nm, aligned to two dimers on a single dimer row. This is not the same as the image pixels used in Image size.

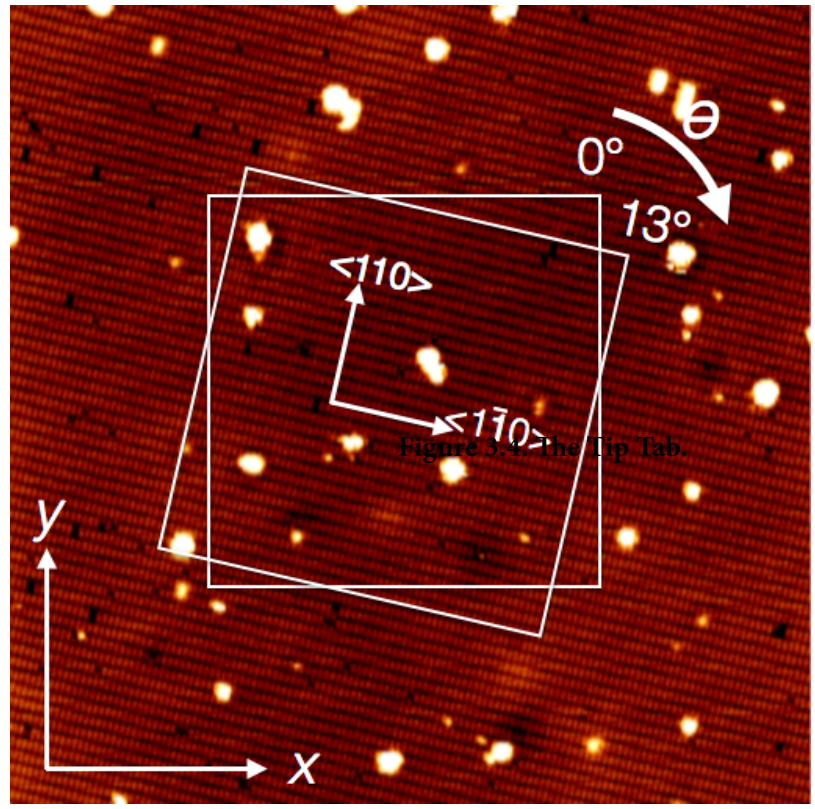



**Figure 3.2 Frames of Reference: x and y refer to the piezo directions. At 0° rotation, the tip scans along x, moving down the image line by line in a negative y direction. With a positive angle, the image rotates clockwise, so that the dimer rows appear to rotate counter-clockwise. A special angle is the lattice angle, here 13°, where the tip scans along a lattice direction.**

Figure 3.4: The Tip Tab.

## The Tip tab

The Tip tab, shown in Figure 3.3, contains controls which are less frequently accessed during scanning, but are accessed to modify the auto-approach process, or the speed of the motion of the tip across the surface.

Inner Speed refers to the speed at which the tip moves across the surface when not scanning. This can be different to the Scan Speed set in the Scan tab. The Outer slow fields control the tip speed during approaching and homing, and The Outer Fast fields during walk-in and disengage.

When approaching, the tip should move towards the sample slowly enough for the preamp to see tunneling current in time to stop moving, otherwise it will crash the tip. Walk in can go faster, since the higher voltage specified for walk-in allows a little more warning of impending tip touch. When disengaging, a high speed is safe.

Inner Home is the percentage of the full range within which the fine piezo is considered “home”. During Approach, SCANZ attempts to move the sample closer or further from the tip so that the fine piezo is within this range when it is tunneling with the desired Setpoint and Sample bias. By setting this range to negative values, such as -20% to -30%, we instruct SCANZ to move the coarse piezo such that the fine piezo has to reach out a little towards the sample to achieve tunneling. If power is lost, the piezos will then move to their zero position, meaning the fine piezo will move the tip away from the sample in this case, thus protecting the tip.

Tip protect sets the parameters used by the DSP to attempt to prevent tip crashes. If the DSP ever detects that the tunneling current is greater than the first number, or

the fine piezo is retracted by more than the second number, the DSP will automatically move the coarse piezo (and thus the tip) away from the sample the distance defined in Walk-in backup. Therefore even if SCANZ crashes or the computer locks up. So long as the DSP is running, it will protect the tip. To disable this protection, set the desired limit to a number greater than 100%.

Walk-in backup will move the coarse piezo away by the specified number of units when the walk-in current is exceeded. This helps protect the tip by quickly moving the sample away after overcurrent is detected. Picking a proper value needs care; too large a value (~5000 DACU) could cause inertial sliding of the tip.

Walk-in bias is the sample bias to be applied during walk-in. Setting this higher than the normal sample bias allows ZyVector SCANZ to detect tunneling sooner than it would with a lower voltage, protecting the tip during approaches. Walk-in curr controls the current limit during walk-in. If ZyVector SCANZ detects that the current has exceeded this threshold, the walk-in is terminated. This is separate from the normal tunneling current setpoint so that we can set a higher threshold for walk-in current to account for noise on the tunneling current signal related to fast moving of the piezo. Watching the coarse Z piezo and tunneling current in SoftScope allows tuning of this value; if the coarse Z steps are slow and irregular during walk-in, the Walk-In current limit may be set too low. The walk-in process slows the ramp down each time it sees tunneling above this threshold, then slowly ramps the speed back up to the goal value over several steps. Setting the threshold higher suppresses such false triggering, allowing smoother and faster ramps. In all cases, a minimum threshold current that avoids false positives should be chosen to minimize the potential for tip damage.

The Walk-In step and Approach Step controls determine the size of the slipstick motion during Walk-In and Approach. These can be set to the same parameters as used for coarse motion, but they are usually set to be smaller, especially for Approach Step, so that the step size is small relative to the range of the piezo. During Approach, the Info Panel (Fig. 3.9) will announce "surface has been found at -90%", -80% etc. until a percentage within the Inner home range has been reached. If the step is too large, or if the tip is drifting in, the Approach may overshoot the Inner Home range, and then outward steps are taken. Thus a good Approach Step value is a balance between fast approach, and overshooting.

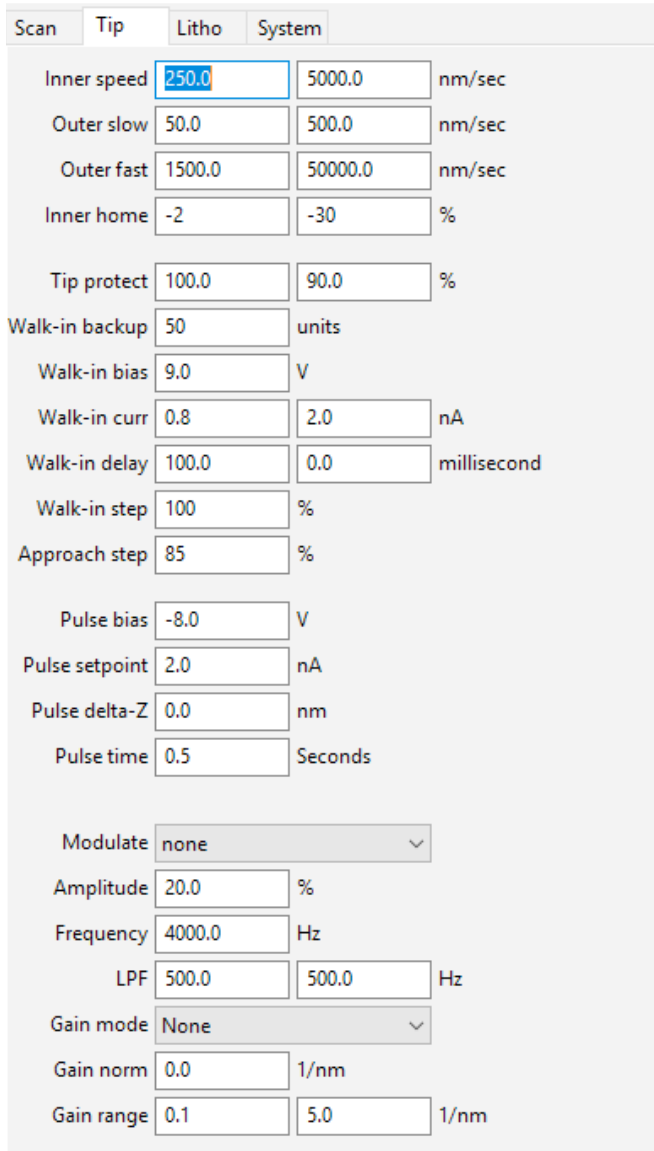



**Figure 3.3. The Tip Tab.**

Walk-in delay is a time built into the current detection to accommodate the current spike artifact caused by the slip-stick step. During this time, after the slip-stick step command is issued, the feedback circuit ignores the tunnel current signal. Typically the current spike is a few ms wide, so the normal setting of 5 ms should be enough.

The Tip pulse Action Button is used for tip conditioning. When the button is clicked, the bias and current settings are changed to those given in the Pulse bias and Pulse setpoint settings. The duration is given by the Pulse time. Alternatively, the tip can be moved towards the surface by the distance given in the Pulse delta-Z setting, in which case the Pulse setpoint is ignored.

## Active Feedback Loop Gain Control (LBH)

Since STM lithography occurs in more stringent tunnelling conditions, with higher voltages and currents, and because the results of tip instability (crashes, deposition of material from the tip) have much higher consequences than for ordinary imaging, ZyVector includes a unique capability - active measurement of the feedback loop gain, in order to allow higher sensitivity in the control of z-motion.

The DC gain of a scanning tunneling microscope (STM) feedback control loop is proportional to the square root of the local barrier height (LBH). LBH (i.e. DC gain of STM) may undergo significant variations during a scan and this can adversely affect the control loop stability if the controller parameters remain fixed. LBH variation is estimated by injecting a dither signal with fixed frequency to the feedback loop system at the set-point and tracking the amplitude of corresponding components at the Z channel and the log of the tunnel current. Then, LBH would be proportional to the ratio of log current amplitude to Z channel amplitude (d lnI / dZ) at the modulation frequency. A Lyapunov filter is used to track the amplitude of signals at the given frequency.

To use this capability, the settings at the bottom of the Tip tab can be used.

Modulate refers to the modulation channel. For the purpose of LBH estimation it should be set to set-point.

Amplitude sets the amplitude of modulation signal. It should be large enough to have a good signal to noise ratio. On the other hand, it should not be too high because it can have a negative impact on the imaging. By monitoring the FFT of current signal in SoftScope, an appropriate value can be selected. A Signal-to-noise ratio of higher than 10dB is recommended.

Frequency sets the frequency of modulation signal. It should be greater than the closed-loop bandwidth so it does not adversely affect the topography information at low frequencies. In addition, the tunnel current (It) should be small enough to avoid exciting resonance frequencies of the scanner. For example, suppose that we have a scanner which has its smallest resonant frequency near 8 kHz. For scanning speed of 200nm/sec or lower a closed-loop bandwidth of few hundred hertz is required. So, selecting a modulation frequency between 2 kHz and 6 kHz would be good.

LPF sets Lyapunov filter parameters. The first box is adaptive gain of Lyapunov filter which determines the convergence speed of the estimator. For large values the filter has a high-bandwidth and fast tracking which also lets more noise into the estimation, while a smaller value gives smoother but slower estimation. The second box determines 3dB passband width of the bandpass filter centered at the modulation frequency. The passband width should be chosen close to the topography imaging bandwidth, which is a few hundred Hz. Starting values for adaptive gain and passband width can be 500. Then, they should be fine tuned to obtain a good LBH image.

Gain norm shows a typical value of log current amplitude to Z channel amplitude at the modulation frequency which is proportional to the DC gain of the STM. This value can be obtained by monitoring d lnI/dZ signal in SoftScope while the STM tip is still. Approximately, the average value of monitored signal should be selected as the Gain Norm. Then, it will be used to change the controller gain adaptively to avoid instability. Gain adaptation would be disabled if Gain norm is set to zero. Otherwise, the controller gain (K) is adaptively changed to keep the overall loop gain constant.

$K_{new}$=(Gain norm)/ (d lnI/dZ) × K.

Gain range shows minimum and maximum values for d lnI/dZ to avoid large or small controller signals. Sometimes LBH estimation is not accurate and d lnI/dZ changes significantly. We don't want to increase or decrease the controller gain significantly, so the minimum and maximum values of d lnI/dZ should be specified. Usually, they are selected in such a way that 0.5<(Gain norm)/ (d lnI/dZ)<2.0. For example, if the gain norm is 10, it is good to select minimum as 5 and maximum as 20.

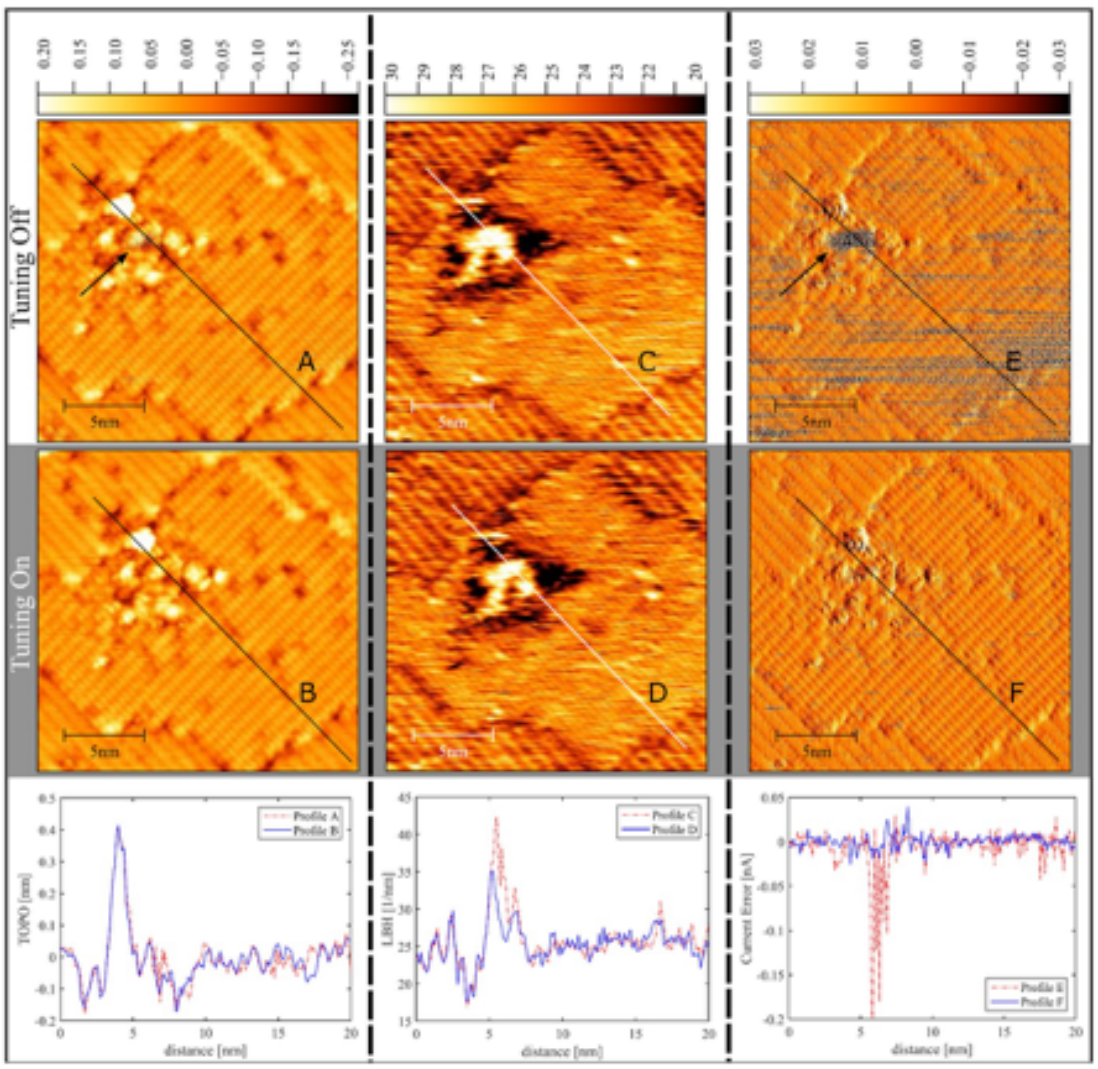



**Figure 3.5. Effect of PI tuning on the STM performance when controller gains are set to a high value which brings the feedback loop close to the stability margin. From [1].**

Tajaddodianfar et al. showed the effect of PI controller tuning on the STM imaging [1]. Fig. 3.5 shows two STM images obtained without (top row) and with (middle row) PI tuning. From profile E, it is visible that while the STM tip is passing over the high DC gain area (high dLn(I)/dZ), ringing appears on the current image and the feedback loop system experiences instability. By activating the PI tuning, the ringing is reduced significantly as can be seen in profile F. In addition, the artifacts shown in profile A have been removed by activating the PI tuning as shown in profile B.

[1] F. Tajaddodianfar, S. O. R. Moheimani, and J. N. Randall, "Scanning Tunneling Microscope Control: A Self-Tuning PI Controller Based on Online Local Barrier Height Estimation*," *IEEE Trans. Control Syst. Technol.*, pp. 1–12, 2018.

## The Data Viewer tab

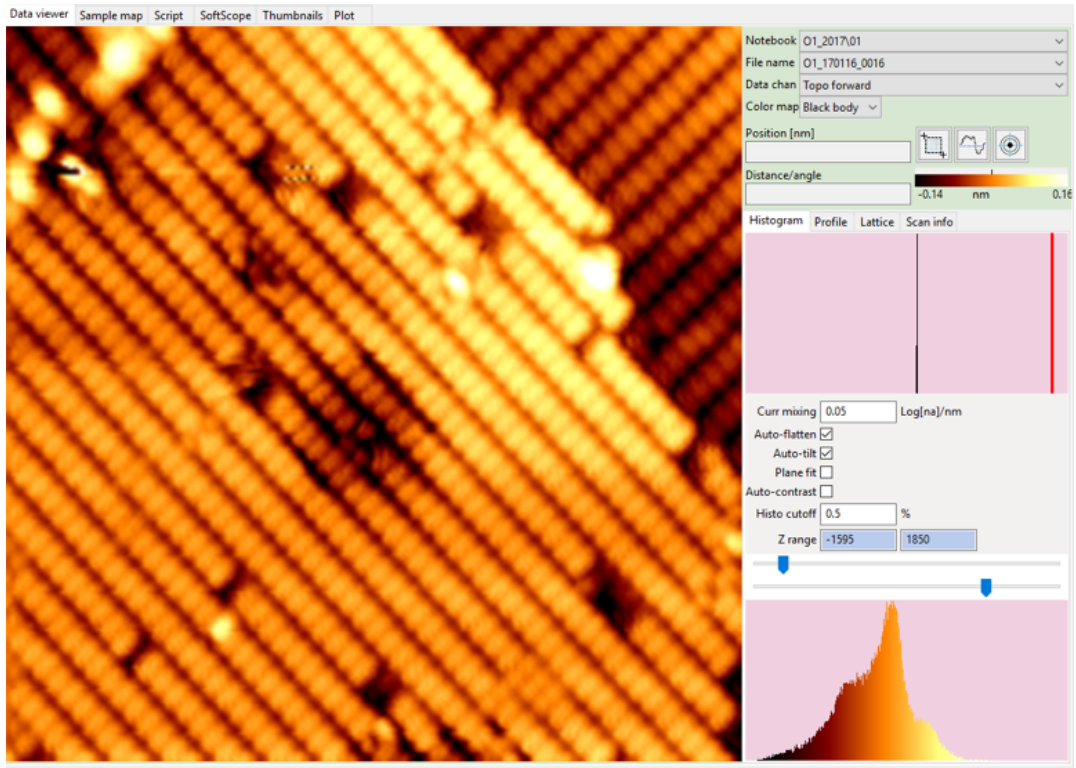



**Figure 3.5. The Data Viewer Tab.**

The main viewing tab used for STM, showing the live STM image, is shown in Figure 3.5. Unlike the other view tabs, it cannot be separated into a different window. The panel on the right hand side provides various controls, laid out in four Data tabs: Histogram, Profile, Lattice and Scan Info shown below in Figure 3.6.

A list of recent saved and unsaved images are listed in a drop down menu, identified by Notebook and File name, along with drop down menus to choose the Data channel displayed (topography or tunnel current, forward or backward scan) and the Color Map (Greyscale, Black Body).

Below this is a set of feature selection tools. If the Mark point button is clicked, the cursor can be moved across the image and the location read from the Position box. Clicking and dragging will also allow distance and angle to be read from the Distance/angle text field. Clicking and releasing causes a point to be marked on the image, and the image can be subsequently recentered to this location by clicking the Center to mark action button. Clicking the Profile button allows for a line to be dragged across the image. The positions and dimensions of this line can be read from the Position and Distance/angle fields. As a line is positioned on the image, the topographic profile across that line is shown in the Profile tab described below. Clicking the Rectangle area button allows for a rectangle to be drawn across the image in a similar manner.

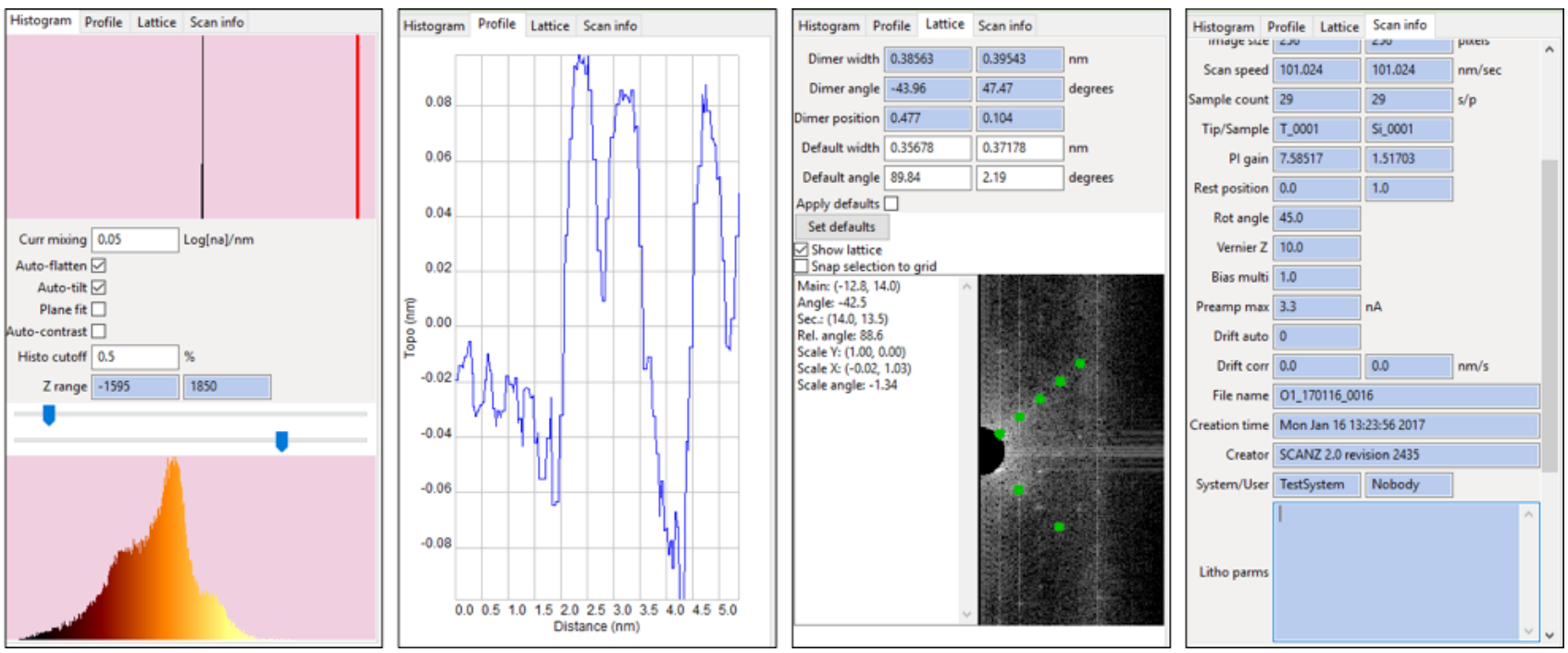



**Figure 3.6 The Histogram, Profile, Lattice and Scan Info tabs.**

## The Data Tabs

The four Data tabs are shown in Figure 3.6. The view of the current image can be adjusted using the options in the Histogram tab. The Profile tab shows a topographic contour of the surface, over an area defined by clicking the Profile line button. The Lattice tab shows a Fourier Transform of the current image, along with the local and default lattice parameters. Use of the default lattice parameters can be turned on an off here, and lattice data from an image can be applied to the default settings. The Scan info tab contains the scan parameters used by the scan currently being displayed in the Data Viewer tab. If that tab is showing the current scan, these parameters should be the same as the Scan tab. When viewing a scan captured at some other time, these parameters show the values used when that scan was captured.

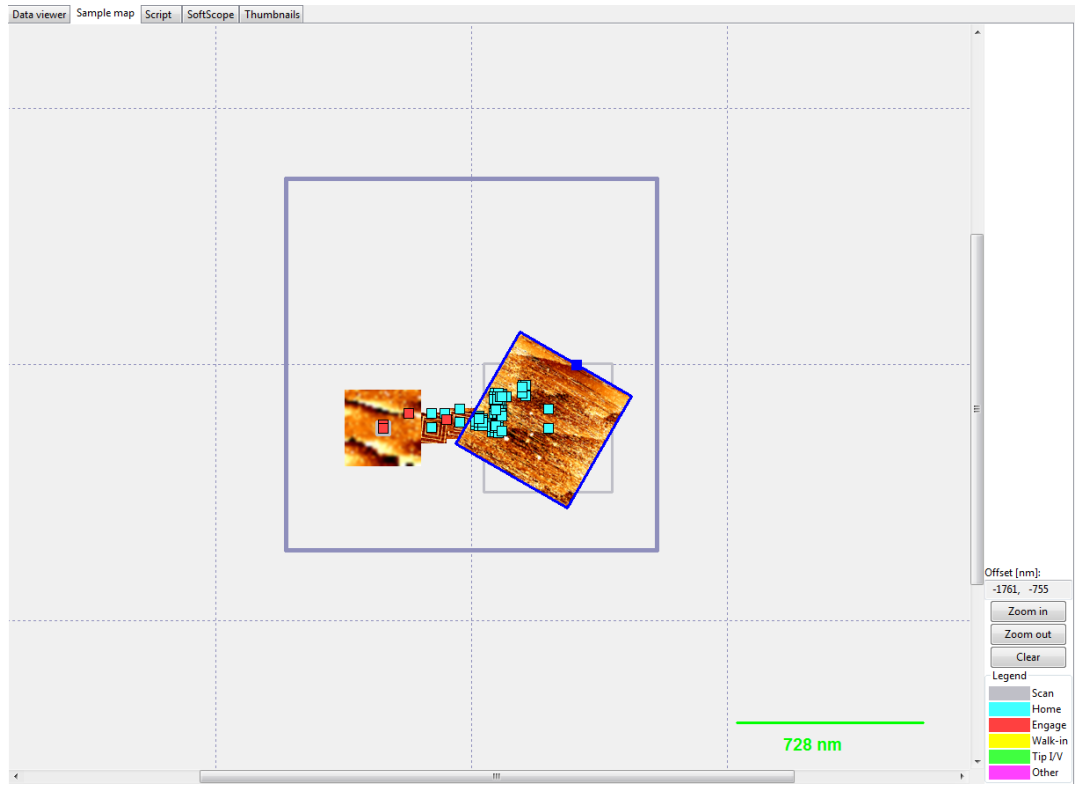



**Figure 3.7 The Sample Map tab**

## The Sample Map tab

The full available scan area is shown in the Sample Map tab (Figure 3.7). The size of the scan area depends on the calibration of the piezo-tubes voltage response and is typically about 10 µm for an Omicron VT STM. Scanned images are shown to-scale, unless they are very big or very small at the current zoom level, in which case only a gray outline is shown. Other events like tip Approach and Homing are marked with colored squares. The color scheme is defined by the legend to the right. The map is reset whenever the user moves the tip using Tip Step to a new area.

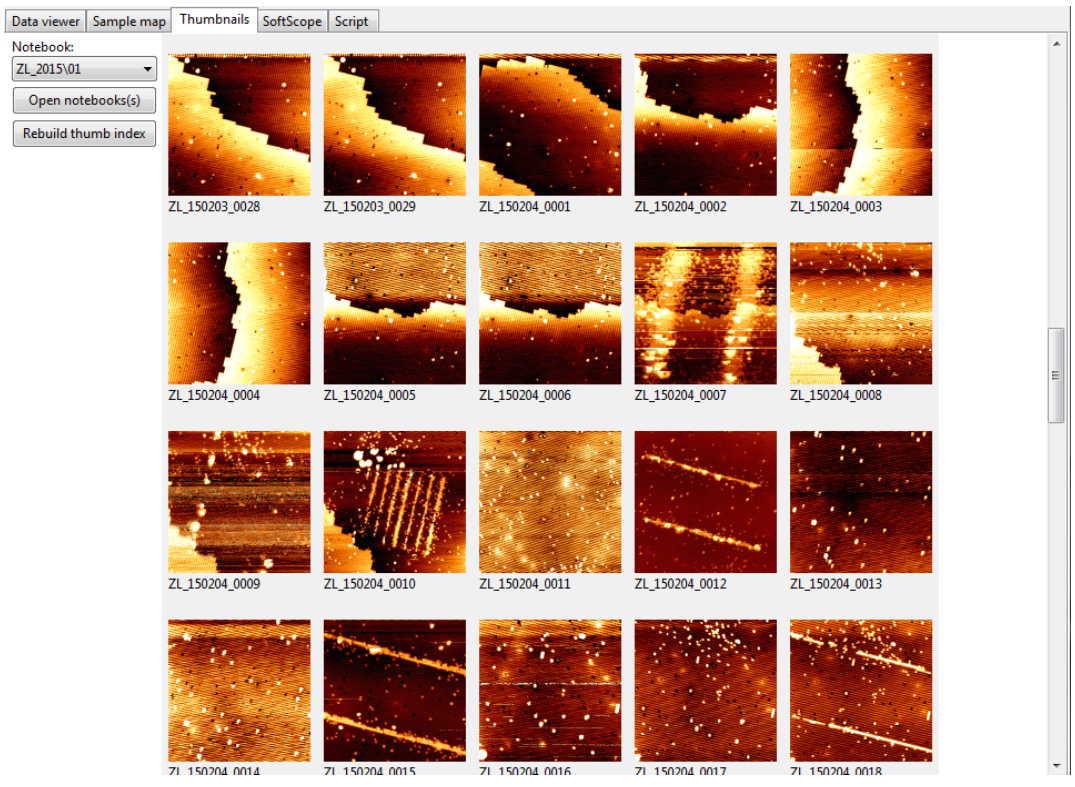



**Figure 3.8 The Thumbnails tab**

## The Thumbnails tab

Images are autosaved into folders corresponding to each month. These folders are listed under the “Open notebooks” button in the Thumbnails tab, as shown in Figure 3.8. The files can be opened by double-clicking. They are loaded into the DataViewer tab, and appear in the list of files there.


**Figure 3.9 The Action Buttons, Command Queue and the Info Panel**

## Action Buttons

An action, such as scanning an image, is initiated by clicking an Action Button.

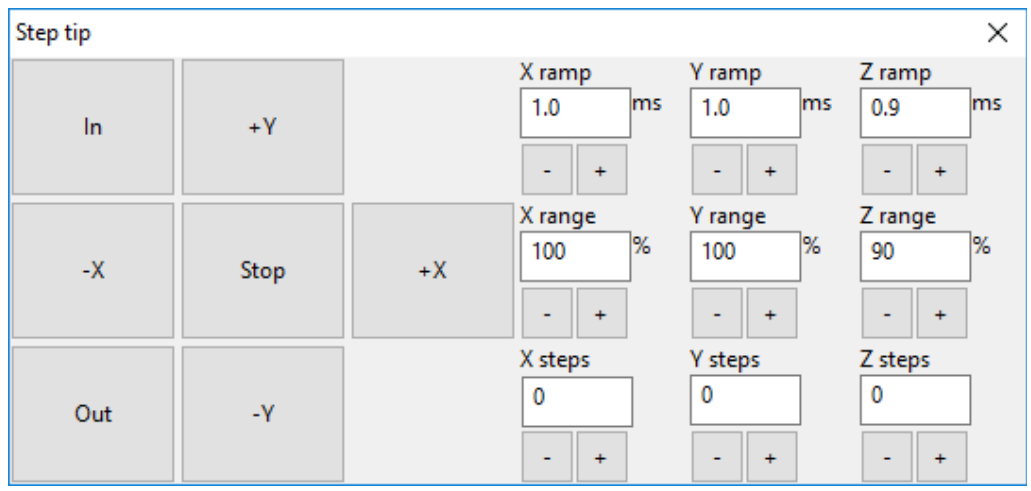



**Figure 3.10 The Step Tip window**

- Live Scanning initiates a scan with the current scan parameters. Parameters such as bias voltage, tunnel current, scan center, scan size can be changed during the scan. The only parameter which cannot be changed is the Image size, i.e. the number of pixels.
- Full Scan takes a complete image, and any parameter changes are queued until the end of the image. This is appropriate as part of scripted processes.
- Cancel sends a message to the background task to stop what it is doing. All actions and scripts in the Queue are cancelled, but changes to parameters, such as Sample Bias or Scan Center, will still be implemented.
- Litho performs lithography with the current parameters and patterns set in the Litho tab.
- Pulse tip applies the voltage, current, duration and delta-z settings as listed in the Tip tab. Tip pulsing is a typical way to improve the imaging by modifying the tip.
- Center to mark sets scan.center to the point specified on the image.
- Step tip opens up the Tip Step window, shown below in Figure 3.10. This allows for large-scale motion of the tip in X, Y, and Z. The tip should be disengaged at least 10 steps, and preferably more, before doing this.
- Walk-in does a fast approach with different parameters than Approach, in an attempt to detect tunneling before crashing the tip into the sample. If the sample is more than a few steps away, this command is far faster than the Approach command.
- Disengage retracts the tip from the sample, and walks the tip out the number of steps specified in the Disengage parameter of the Scan tab.
- Approach lands the tip onto the surface slowly by alternating coarse steps and fine piezo motion, and brings the inner piezo into the desired Home range. Approach will also step away from the surface if necessary to adjust the piezo position.
- Retract tip fully retracts the fine Z range, lifting the tip as far away from the surface as possible without coarse piezo steps. When the tip is fully retracted, the Actino Button changes to read Land tip.
- Land Tip brings the tip forward slowly until it reaches the surface, without taking any coarse steps.

## The Command Queue

It is possible to make several parameter changes and invoke several actions or scripts which will not immediately be implemented because the software is busy performing another task, for example while an image is being scanned. Instead these commands are added to the Queue located at the bottom of the software window, in between the Action Buttons and the Info panel, as shown in Figure 3.9. They will then be executed in turn. Hitting the Cancel Action Button will cancel all queued Actions, as well as the current item, but will not stop pending parameter changes. Individual queue items can be cancelled, including parameter changes, by clicking the 'x' box on the right side of each item. When a script is executed, all of the functionality of that script is encapsulated within a single queue item. When cancelled, the entire scripted procedure is cancelled. Information regarding the status of components within the script comes only from comments within the Info panel, as defined and output by the script itself.

## Imaging Tasks

Here some simple tasks for basic STM operations are outlined, including approaching the tip to the surface, taking an image, saving it, and then disengaging the tip so that it is a safe distance from the surface.

### Auto Approach of tip to surface

The vertical sensitivity of the STM is due to the exponential decay of the tunnel current signal with the distance of the tip above the surface. As a result, it cannot detect the surface's presence until the tip is within 1-2 nm of the surface. If the tip is moving towards the surface at a rate faster than 1-2 nm × current sensing bandwidth, it may be too late by then to avoid crashing into the surface once the tip gets into tunneling range. A safe way to bring the tip into tunneling range in a reasonable amount of time is therefore required. Three functions triggered by Action Buttons comprise the auto approach procedure, each of which will be subsequently described in detail:

- Step Tip – Coarse, manual approach, using a camera to bring the tip close to the sample. There is no tip protection in this mode.
- Walk In – High sensitivity stepping procedure, with tunneling current monitoring, which brings the tip into tunneling range
- Approach – Slow stepping procedure, which brings the piezo tube to within its desired operating range; also performs fine adjustment of the piezo tube position, to optimize tip-sample position.

**Coarse Motion using the Step Tip window**

The first stage of auto approach is to bring the tip close to the surface manually, using the stepper motors. The Step Tip window, shown in Figure 3.10, is opened by clicking the Step Tip Action Button. The speed of motion is controlled by the Ramp time, which determines the time to complete an entire sawtooth period. Typical ramp times are from 0.1-10 ms. Speed of motion is also controlled by the XYZ range settings. This sets the voltage applied to the piezo. There will be a minimum threshold voltage below which no slip-stick motion will occur. It is useful to determine this threshold in the coarse motion mode, so that appropriate voltages can be applied to the fields for the voltage used during Walk-In and Approach.

Clicking one of the direction buttons will move the tip a set number of steps as defined in the Step count fields. If Step count is set to zero, clicking a direction button will move the tip a set number of steps (100 for 1ms), or by clicking and holding, the tip will move continuously until the button is released. There is no surface feedback in this mode, and so the tip should be observed in a camera to avoid crashing into the surface. The manual approach should be stopped when a small gap is still visible between the tip and its reflection on the sample. The Step Tip window is also used to perform the operations required for tip exchange, following the standard Omicron process, as described in the Omicron VT STM manual.

**High-speed approach using the Walk-in Action Button**

With the tip close to the surface, the auto approach process can be started. The first stage is Walk-In. Here the piezo tube remains at maximum extension while the coarse piezos are stepped in. Feedback is switched on, so that the piezo will retract as soon as the surface is detected, and Walk-In will be terminated. The sample bias is set to a high value, so that the tip detects the surface at a greater separation, and stops without crashing into the surface. A message such as "`Walk-In done. 202 steps`." will appear in the Info Panel.

The speed of the approach process is controlled by the Outer Fast field in the Tip tab. A setting of 200 nm/s is typically slow enough to avoid crashing the tip. For a first approach to the surface for a new system, even slower values can be selected. The size of the slip-stick step is set by the Walk-In step field in the Tip tab. This can be set to 100% as the slip-stick step size is usually less than the range of the fine piezo Z motion.

**Fine Auto Approach using Approach**

The second stage is Approach, in which the piezo tube is retracted, the coarse piezo takes a step in, and then the presence of the surface is tested by moving the piezo tube forward slowly. This is a very similar process to that used by the Omicron Auto Approach. Here the speed of the piezo motion while testing the surface is controlled by the Outer slow field in the Tip tab. For general purposes, the size of the Approach steps should be made to be about 10% of the range of the piezo to allow for fine tuning of the tip extension when landed on the surface. For a first approach, a setting which is only slightly above the minimum voltage for Approach should be used, as the relative range will not be known until the tip has landed. For example, if the tip does not move with a 40% Z step size in coarse motion, then an Approach step of 50-60% could be used, even though this may cause the initial Approach process to be very slow. A more suitable step size can be set once the relative step size is known after first contact with the surface.

When the surface is first detected, the tip is likely to be close to the end of its extension range. A message such as "`Found surface at -84.7%`." will be seen on the Info Panel. However, the Inner Home range, set in the Tip tab is typically around -20% to -5%. Therefore, once the surface has been detected, further coarse steps are taken to bring the piezo tube into the Inner Home range. With each step, further messages such as "`Found surface at -73.4%`.", "`Found surface at -62.2%`." etc. will be seen, until the tip reaches the desired range. If the Approach step is too large, and the surface is encountered during the coarse step part of the Approach, a message, "`WARNING Approach found surface while stepping`" will appear on the Info Panel. This is likely to indicate a crash on landing. Thus the Approach step should be much smaller than the range of the piezo. Moreover, if the step size is larger than the Home range, it can overshoot, and the tip will be stepped back out. A dance back and forth of the tip to find the Home range will ensue, which can also result in a tip crash. Therefore, it is recommended to set the Approach step to be smaller than the width of the Inner Home range.

Because the homing procedure continues until the tip position has stabilised in Z, the number of steps required may become very large if there is significant drift in the Z direction. This often occurs immediately after the tip has approached the surface, so it is a good idea to allow the tip to settle after Walk-In, and then Approach. Vertical drift can also be due to a sample that is still cooling down from sample preparation. Alternatively, the Inner home range can be made larger, so that more drift can be accommodated. If the tip has drifted out of the Home range, Approach can also be used to relocate the tip extension to the Home range.

## First Image of a sample

To obtain an STM image, in the Scan tab, set desired STM parameters, for example 40 x 40 nm, at 256 x 256 px, with a typical sample bias for Si(001) of -2.5 V, and 0.15 nA. For a first attempt, use PI settings of 30% and 0.1 ms.

Click the Interactive Scan action button to start scanning. The live scan data will be shown in the Data Viewer tab. The appearance of the scan can be adjusted in the Histogram tab on the right side of the Data Viewer tab. Auto flatten and Tilt are usually on by default. The whole image can be flattened using Plane fit. The Profile tool can be used to draw a line across the image, and the corresponding profile will be shown in the Profile tab.

Assuming that the sample has good dimer row resolution, the position and direction of the Si surface lattice can be identified. In order to satisfy the Nyquist condition, a 48 nm image requires 256 px, for example, or >5.2 px/nm. The Fourier transform should show green dots if it has identified the lattice features. The corresponding numbers for the dimer dimensions are given in the Lattice tab within the Data viewer tab. Turn on Show Lattice to overlay the green lattice grid onto the image. The first lattice number in Fig. 3.6 refers to the slow scan direction down the image, and the second number refers to the fast scan direction across the image. These numbers are typically slightly different, and the angle between them is usually not quite 90°. The Fourier transform should show green dots if it has identified the lattice. Default lattice parameters are determined for use in lithography by using the SetDefaultLattice() script, as described below in the Lithography chapter or by using parameters from the most recent scan by pressing the Set Defaults button.

Next, move the tip position sideways using the Scan Center parameters. Move to (50, 50). The Info panel at the bottom will show the new position. Hit Scan once more, to take another picture.

Move the Rest position from (0,1) to (0,0) or (-1,-1). A message in the Info panel will show a change in the fine position, and the Sample Map will update to show the new tip position. Return to the normal Rest Position (0,1). Now change the Scan size from 40 x 40 nm, to 100 x 100 nm. The Info Panel will show that the tip moves up to the top of the bigger image.

Now, change the Scan rotation from 0 degrees to 90 degrees. The Info Panel will show that the tip moves with the top of the bigger image from (0,50) to (50,0), illustrating the Clockwise rotation of the scan direction.

## Disengaging and Shutdown

At the end of the STM session, or before any long-distance xy motion using the Step Tip command, the tip should be retracted from the surface in order to prevent any crash of the tip caused by a vibration, an electrical spike, or a power failure. The tip should be retracted by clicking the Disengage button, and the number of steps to be taken during the disengage process is configured from the Scan tab by configuring the second (right-most) text box under Steps: eng/dis. Disengaging 10 steps is usually enough, 100 may be safer. Note that the DSP will automatically disengage the tip by 10 steps if it loses communication with ZyVector SCANZ, thus protecting the tip even if ZyVector SCANZ quits without disengaging (or if the computer hangs up). Coarse motion using the Step Tip window can then retract the tip further.

## Saving and Exporting Images

After taking several images, the file list on the right side of the Data viewer tab, “File name” will have several items in it. This will contain all the recent scans, including unsaved images, and any partial images. Completed images are saved by default, partial images are not, but by selecting autosave all in the Control Panel, both complete and partial scans will be saved. The autosave behavior is controlled from the check boxes in the Scan tab. Saved images will now also be visible in the Thumbnails tab, and can be opened from there. Unsaved images can be saved manually as ZyVector format files (*.zad) using the Save scan command in the File menu, and can be saved automatically by changing the check box in the Scan tab from Full to All. The .zad files can also be opened in the open source image editor Gwyddion, using a special file import script provided with the SCANZ software installation. Gwyddion has much more sophisticated image manipulation routines than SCANZ. The installation of Gwyddion, and enabling it to read .zad files is described in Appendix 2. Images can be exported in .png or WinSXM format from the File Menu. The .png images produced include a title block, giving the file size, resolution etc.

## Tip Conditioning

In many cases, the first image of a new sample with a new tip will not give good resolution of the surface. Conditioning of the tip is commonly used to improve the quality of the STM images. There are various techniques used by different users, including high voltage pulsing, imaging at higher bias or high current, and even gentle crashing of the tip into the surface. In many cases, however, these techniques can lead to a tip which gives good images, but which is not stable to perform lithography. A typical method which produces a stable tip for lithography as well as for imaging, is to change the bias voltage to a large negative value, and back again. This can result in depositing material, including Si and H, onto the surface. On a H-terminated surface, it can result in the production of some dangling bonds, indicating the removal of H atoms. Therefore tip conditioning should be performed well away from the Area of Interest. Live tip conditioning can be achieved by using the Interactive Scan mode, and clicking the Tip Pulse Action Button. The settings for the tip pulse are listed in the Tip tab.

# Chapter 4: Scripting

The true power of the SCANZ software comes from its advanced scripting capability, giving the ability to automate almost any aspect of the imaging and lithography process so that operator time is minimized, and machine throughput is maximized. The scripts are written in the Python language with some modified syntax. The purpose of this chapter is not to teach the user how to write Python. There are many online resources for learning Python. This chapter will explain the process of writing and running a script, including the types of errors which are commonly encountered. Also, potentially useful sample scripts included with SCANZ are summarized.

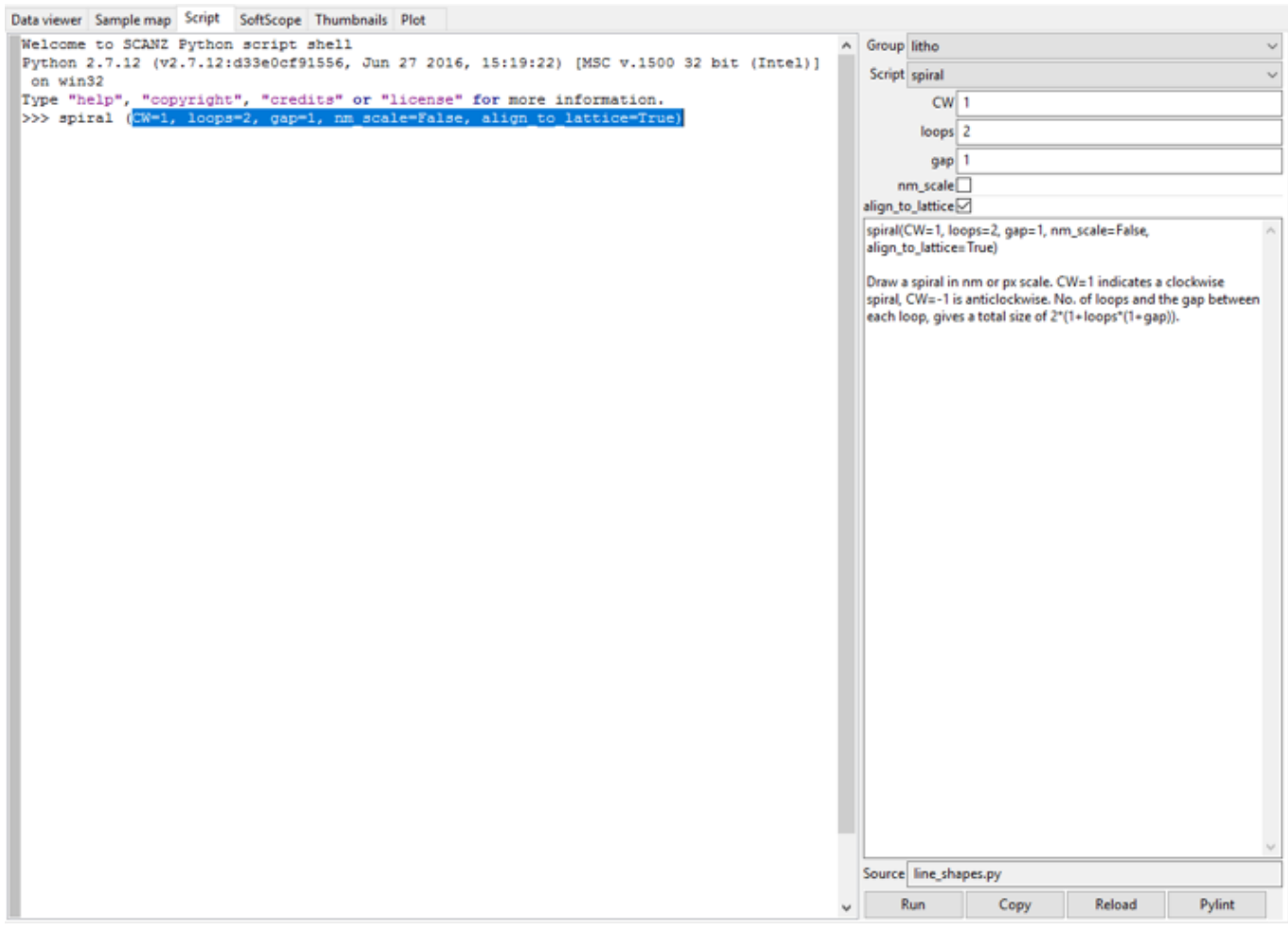



**Figure 4.1. The Scan Tab, comprising the command line interface and the Script Menu Panel.**

## The Script Tab

The Script tab, shown in Figure 4.1, has two main parts. To the left, the command line can be used to perform actions or run scripts by writing a command as an alternative to using the user interface. On the right is the Script Menu panel, which allows for the quick selection and execution of any script. All scripts in the *notebook/commands* folder, as well as those built into the software, are listed in the dropdown menu at the top of the menu panel, organized into groups by function. Below this are fields for all the parameters listed in the script, and a text box containing a description of the purpose of the script, and the effect of the parameters. At the bottom of the menu panel are Run, Copy and Reload buttons which are explained below.

## Command Line Interface

Nearly every parameter setting or action visible in the User Interface can potentially be called from the Command Line. A glossary of the most useful available commands is given in Appendix 1. A simple example of a parameter setting would be `scan.center = [100,100]`. This sets the position of the tip to (100,100). An example of an action would be `litho.start()`. A script is simply a set of commands, run as one command in the Queue. To activate a command or a script, simply type it into the command prompt, and press Return. This can be useful for building a script, as the set of actions and commands can be created and tested one by one in order on the command line. To reuse an executed command, click on it, and press Return. This will copy that command to the active command prompt, where it can be edited. Pressing Return once more, will then execute the command.

As an aid to typing more complex commands and discovery of available script options, an autofill capability is available. For example typing `scan.` and pausing will cause a menu to pop up, where both `center` and `start` will be found, along with several other options, as described in Appendix 1. To execute a script, the name of the script is typed, followed by an open parenthesis. Pausing at this point will cause all the parameters of the script to be filled in, as is shown in Fig. 4.1 for the command `spiral`, and again this is editable before execution. After selecting the relevant options, a closing parenthesis should be typed, and then pressing Return causes the script to run.

## Script Menu Panel

The Script Menu panel is designed to allow for the quick selection and execution of scripts from the dropdown list. The included scripts are separated into system, litho and sample groups, according to their expected use. User scripts can also be organised into groups, using the `__group__` parameter described below to define which group a script is placed into. Instead of the parameters within the script being listed in parentheses after the script name, fields for parameter input are provided for the currently selected script. The text box below shows any comment text given in the script, which is usually a brief description of the script, including a listing of the effect of each parameter.

To run a script using the menu panel, it should be selected from the list with any desired parameters filled in, and then the script is run using the Run button at the bottom of the panel. There are also buttons to Copy a script as generated in the menu panel, so that it can be pasted into the command line or pasted into a script file in an external program, and the Reload button, which loads all available scripts from the script folder into SCANZ. When writing a script, a new script needs to be loaded before it can be used. If there are any errors in the script, these will show up in the Info panel upon loading.

```
#-------------------------------------------------------------------------
# Name:         Parm_search.py
# Purpose:      Litho Parameter Determination
# Author:       <James Owen>
# Created:      2013/12/04
# Copyright:    (c) 2010, Zyvex Labs, LLC
# Licence:      <Zyvex proprietary>
#-------------------------------------------------------------------------
# The Script module must be imported to support all the default functionality.
from Script import *
from position_initialize import position_initialize
from setlithoparam import set_ap_litho_param
from setlithoparam import set_fe_litho_param
import datetime

# list of all exported objects from this module
__group__ = 'litho'
__all__ = ('AP_parm_search', 'FE_parm_search',)

def AP_parm_search(auto=Boolean(True), V_offset=0):
    """ writes lines with litho parameters from list.
    auto=True uses the global list from set_ap_litho_param.
    parmsAP = [[3.5, 2, 2], [4, 2, 2], [4.5, 2, 2], [3.5, 4, 2], [4, 4, 2],
               [4.5, 4, 2], [3.5, 4, 4], [4, 4, 4], [4.5, 4, 4]]
    auto=False uses local parmsAP list below."""
```

**Figure 4.2. An example script.**

## Writing a script

At its simplest, a script is a list of commands in order, as if they had been typed into the script command line and executed. However, a script can be much more than this, reading from external data files, or incorporating conditional statements, for example. A library of included scripts can be found in the *notebooks/commands* folder for reference and inspiration. Scripts are written in Python. There is a certain syntax required at the top of each script, as shown in Figure 4.2. The first line is often a comment to describe the script. The first active line (here from the text file `Parm_search.py`) will be:

```
from Script import *
```

If the script will call other scripts, it is necessary to import them here.

```
from setlithoparam import set_ap_litho_param
```

There will then be an optional group designation, and a list of exported objects, which lists all the executable scripts contained in the script text file, which are intended to be callable from the Script menu. Any subroutine scripts, which are not to be used outside a script within this text file, need not be listed.

```
__group__ = 'litho'
__all__ = ('AP_parm_search', 'FE_parm_search',)
```

The script itself begins with a definition of its name, and parameters:

```
def  AP_parm_search(auto=True, V_offset=0):
```

followed by the list of commands within the script.

When it is time to test out a script, it is loaded up into the software using the Reload button in the Script Menu Panel, or equivalently, the `load()` command on the command line. If there are any errors in the script, the `load()` process will often find them, and an error message will appear in the Info Panel. If there are no syntax errors (there can easily still be bugs in the script), there will just be a message "Scripts loaded".

## Useful Scripts

A list of script commands is given in the Glossary in Appendix 1. There are commands for virtually all the actions and settings available in the User Interface. ZyVector Scanz is installed with a number of useful scripts, which can be found in the *code/scanz/commands_sample* folder. To activate them, copy scripts to the *notebooks/commands/* folder. Within Scanz, these scripts are found in the system, litho and sample groups in the Script Menu Panel in the Script tab. A brief description of these scripts is given here.

### System Scripts

```
position_initialize(scan_size=12, image_size=64):
```

This script is designed to take a small image, usually just before performing lithography, so as realign the tip to the dimer rows. This is useful to zero out any small accumulated positioning errors.

```
SetDefaultLattice(wait=60, scansize=48, scanresfast=512, scanresslow=256, delta=0.002):
```

Sets values for the default lattice for patterning. The script takes a lattice number and angle from the fast scan direction. It scans at 0 and 90°, to get lattice data in both directions. It repeats scanning until the difference between data from successive scans is less than `delta`. It then saves the lattice data to the default lattice parameters, and to `latticedata.txt`. In order to get a good FFT, at least 2 image pixels per dimer row are required. For a 48 nm scan, 256 px is slightly more than this threshold, while 50 nm is slightly less.

```
tip_condition(V=-5, cur=.2, sspd=100, size=10, offset=None, image=32, threshold=7):
```

This script attempts to improve the tip condition by applying voltage pulses to clean the tip. The tip moves to one corner of the image area, and applies pulses. After each pulse, a small image is taken, and the corrugation of the dimer rows is calculated from the FFT. Once this corrugation reaches a threshold, the script stops and returns to the original position. This works well for blunt tips, but not so much for double tips.

```
creeptest, hysteresisjump
```

These scripts are used to calibrate creep and hysteresis position errors.

### Lithography Scripts

**Simple Shapes**

The script file `line_shapes.py` includes a number of scripts for writing various simple shapes, aligned to the lattice by default (this can be changed). Each shape script has multiple parameters, including the size, rotation relative to the lattice etc.

```
serpentine(width, height, spacing=1, line_width=None, offset=(0, 0), rotation=0,
relative_to_dimer_row=Boolean(True), nm_scale=Boolean(False), align_to_lattice=Boolean(True))
```

Writes a serpentine aligned to the lattice, with line width and line spacing set separately, so that either a filled rectangle or a serpentine shape can be produced.

```
rectangle(width, height, line_width=1, offset=(0, 0), rotation=0, start=0,
relative_to_dimer_row=Boolean(True), nm_scale=Boolean(False), align_to_lattice=Boolean(True))
```

Writes an empty rectangle aligned to the lattice. The start parameter specifies which corner the scripts begins with.

```
spiral(CW=1, loops=2, gap=1, nm_scale=Boolean(False), align_to_lattice=Boolean(True))
```

Writes a clockwise or anti-clockwise spiral, with the number of loops and the gap between the loops as parameters.

```
polygon(sides, height=20, nm_scale=Boolean(False), align_to_lattice=Boolean(True))
```

Writes a polygon, with the number of sides specified as a parameter.

```
star(points, height=20, nm_scale=Boolean(False), align_to_lattice=Boolean(True))
```

Writes a star. The number of points, specified as a parameter, must be a prime number.

```
star6(height=20, nm_scale=Boolean(False), align_to_lattice=Boolean(True))
```

Writes a filled Star of David

```
linewrite(start_pos, end_pos)
```

Writes a line from the start position to the end position, specified in px.

**Larger Patterns**

```
Multimode_VectorGen(filename='', sat_width=8, full_width=14, center_of_mass=False,
all_start_points=False, sigma=None)
```

This script takes a binary bitmap file as input, with black and white pixels. All the edge pixels must be black. It converts the white pixels of the bitmap file into a mixture of FE mode and AP mode tip vectors, and writes the result. The Preview version will perform the conversion and display the tip vectors on top of the image in sequence, but without actually writing the vectors.

```
Litho_array_lattice_mxn(widthx=80, widthy=80, m=4, n=4, serpsize=9, sizepx=512)
```

This script draws a grid of boxes, with a controllable pitch and array size, aligned to the lattice.

**Lithography Parameters**

```
AP_parm_search(auto=Boolean(True), V_offset=0)
```

```
FE_parm_search(auto=Boolean(True), V_offset=0)
```

These scripts draw a series of lines with a set of different lithography parameters, and from the resulting lines, the desired lithography parameters are selected. The `auto=True` option uses a global set of parameters, while `auto=False` allows for a manual set to be tested. If the global set of parameters is used, the scripts `set_ap_preset` and `set_ap_litho_param` (and similarly for FE mode) can be used to set the current lithography parameters and the preset shown in the Litho tab.

For small patterns, especially the 3-dimer pattern used for P qubit incorporation, different lithography parameters may be preferred over those used for writing long lines. Also, subpixel start and end points may be useful to write exactly 3 dimers with high yield. The following scripts allow for sets of different parameters to be tested to find the optimum conditions for writing 3-dimer patterns.

```
threedimer_parm_search()
```

```
threedimer_parm_search_vert(Shiftx=0)
```

```
threedimer_parm_search_horiz(Shifty=0)
```

Once these are defined, a 3-dimer pattern may be placed with precision between a source and drain electrode patch, or a grid of 3-dimer patterns can be laid out.

```
singleatomtransistor_1(Startx=-1.0, Endx=1.0, source_qubit=10, qubit_drain=10, number=1,
spacing=10)
```

```
single_pixel_grid(m = 2, n = 1, Spacingx = 10, Spacingy = 6, Shifty = 0)
```

## Fiducial Finding and Alignment Scripts

A number of routines allow the user to define a fiducial mark by drawing a mark on the surface, and then taking an image of it. This fiducial is then searched for within an image, and the `scan.center` can be relocated to the center of the fiducial mark. More explanation is given in Appendix 5.

```
fid = fiducial_define()
```

Defines a fiducial marker which will be utilized in future fiducial registration markers, returns a fiducial object which is passed subsequently to the `fiducial_relocate()` function. Works from the last image.

Outputs: It returns an object of type `fiducial_obj`, which is later passed to the `fiducial_relocate()` function. This object contains an image of the fiducial pattern, the x and y coordinates of the pattern (center of the representative image) and the width and height of the image.

```
fiducial_identify(fid, update_coords=False, goto=False, output_images=False)
```

This script locates the fiducial mark in both forward and backward buffers to double check the correlation. This function does not generally account for false positives, but returns the maximum cross correlation coefficient for later consideration.

```
fiducial_goto(fid, resize_image=False)
```

This script attempts to return to the location of this fiducial defined by `fid`. It sets `scan.center` to the last known coordinates of the fiducial, as located by `fiducial_identify`. If `resize_image` is set to `True`, it also resets the scan size and image size to those of the fiducial.

`fiducial_update_coords(fid,xcoord,ycoord)`

This script resets the stored x/y coordinates of the fiducial marker. Once you are confident that the fiducial has, in fact, been relocated using `fiducial_identify`, you can pass the fiducial coordinates returned by that function to this function in order to update the fiducial coordinates, but without moving `scan.center` to the fiducial location.

Inputs:

`fid` = the fiducial to recenter

`x_coord` = the new coordinate of the fiducial in x

`y_coord` = the new coordinate of the fiducial in y

## Other Sample Scripts

**Test patterns**

These scripts are designed to check the effectiveness of the creep and hysteresis correction, and measure the precision of the tip motion. For some results of these scripts with and without creep correction applied, see Figure 4.3.

`Test_boxes_1()`

This test script draws 5 concentric boxes, using the rectangle script. Each box is composed of two 1-px rectangles drawn next to each other. This is a stringent test of alignment of the two rectangles making up each box, and also the concentricity of the five boxes.

`Test_boxes_2()`

This test script draws 5 concentric boxes, using the bitmap script. First the left halves of the boxes are drawn, and then the tip is shifted and the right halves are drawn. This tests the alignment of the two halves, as well as the alignment of the vectors in each half. The result should be identical to the output of `Test_boxes_1`.

`Test_squares_n(step=20)`

This test script tests the jump accuracy. A serpentine is drawn at the origin, and then the tip makes jumps of size 1x to 5x step (default 20 nm) in two directions, drawing boxes at each location, and images the result. The precision of the jumps can then be checked by overlaying the lattice and counting the number of pixels between the written serpentines

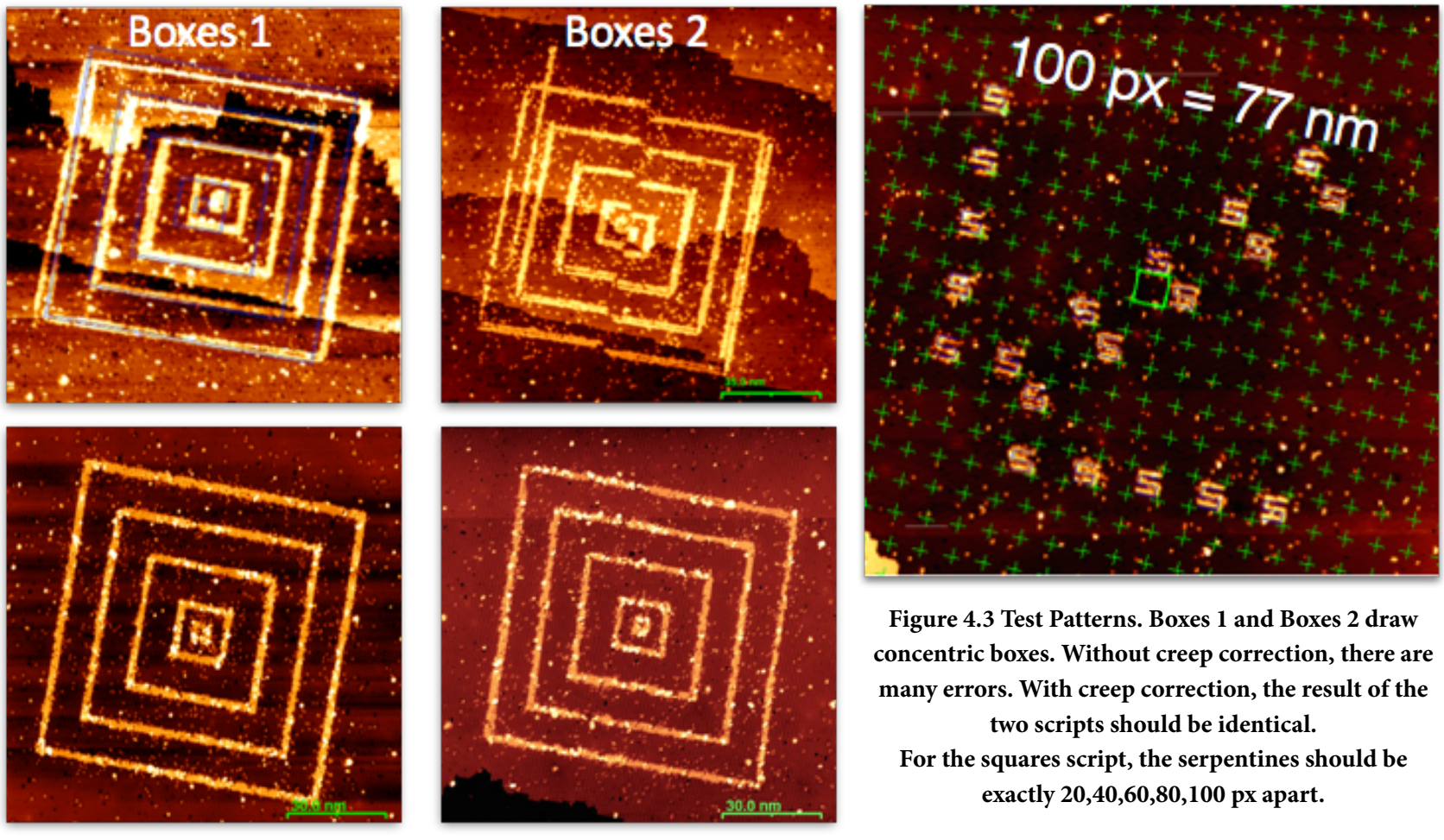



**Figure 4.3 Test Patterns. Boxes 1 and Boxes 2 draw concentric boxes. Without creep correction, there are many errors. With creep correction, the result of the two scripts should be identical.**
**For the squares script, the serpentines should be exactly 20,40,60,80,100 px apart.**

# Chapter 5: Lithography

There are three primary ways that lithography (litho for short) can be performed: using the user interface (UI), via scripting to call UI parameters, or via specialized scripts. In all cases except for the "Dot" pattern below, lithography is performed in a vector format, i.e. with the tip moving along a defined line of arbitrary length and angle at the defined litho bias and current settings.

In all cases, the required Litho parameters can be set in the Litho Setting tab. For simple patterns which are not automatically aligned to the lattice, the patterns defined in the Litho Pattern tab can be used. (Writing patterns which are automatically aligned to the lattice is performed via scripting.) By simply pressing the Litho Action Button, hydrogen depassivation lithography will be performed in accordance with the settings in the two Litho tabs.

## The Litho tab

The Litho tab, as shown in Figure 5.1, controls the bias, current, and dose for various modes of lithography. Presets for the Atomically Precise (AP) and Field Emission (FE) mode can be pre-determined, and these presets applied to the currently active settings by clicking the appropriate buttons. Also, this tab allows options for varying how ZyVector transitions the voltage and tunnel current between imaging and lithography modes.

### Definitions:

- Sample bias, Setpoint, Dose rate are the operational bias, current, and electron dose for lithography as performed. Note that the tip speed in nm/s is equal to Setpoint/Dose rate * 10 nm/s.
- The Apply preset buttons transfer the value of the preset litho conditions created from the `AP_Parm_search` and `FE_Parm_search` scripts to the operational condition fields, where each button transfers the values in the fields below it. The values in the preset fields correspond directly to those in the operational fields.
- Time delays: The first value sets the time that the tip waits before starting litho, after moving. It is also the time taken to write a Dot pattern. The second value is the time taken to switch from scanning conditions to lithography conditions.
- A set of simple geometric shapes can be drawn relative to the Image frame of reference (see below) using the radio buttons in the Litho tab. Shape scripts in the litho group of the Script Menu tab can also be used. Pattern scale defines the size of the basic pattern in horizontal and vertical directions, with units in nanometers. Pattern offset is the distance in nanometers of the individual pattern to be drawn relative to the centre of the image, or if Origin at Mark is checked, the mark location. Pattern angle is the rotation of the pattern relative to the Image frame of reference.
- The Show Litho check box will cause the tip path during lithography to be visible as lines on the image. Each line will show up on the screen in order as it is completed. By using Preview mode, the lines are drawn on the image, but the tip does not move. This is useful to check what will happen when the Litho Action Button is clicked.
- Litho trap defines a trigger to terminate the lithography. These settings are used in Feedback Controlled Lithography. When the trap is triggered, the tip is lifted by the amount given in Trap dZ, and the bias voltage and tunnel current are set to the values given in Trap V and Trap I.

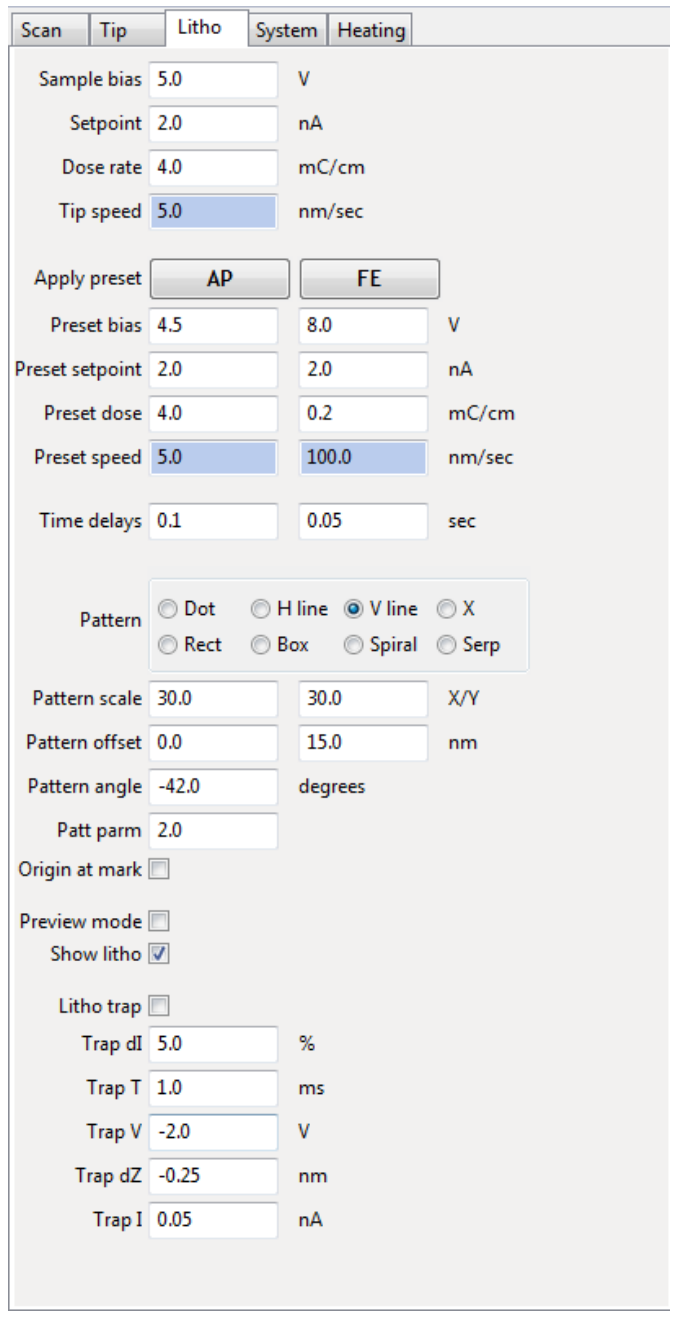



**Figure 5.1: The Litho tab**

## Frames of Reference

In using the STM for lithography, it is important to keep track of exactly what directions write vectors will follow. With that in mind, the following frames of reference are important to remember:

**Scanner frame of reference**. The fundamental direction of motion of independent motors or piezo tube quadrants. Under ideal conditions, this consists of two directions at right angles relative to each other.

**Image frame of reference**. The direction as defined by the acquired image, with fast scan direction and slow scan direction being the fundamental directions in this frame of reference. This frame of reference is defined relative to the scanner frame of reference by the Scan rotate field in the Scan tab.

**Sample frame of reference**. The direction as defined by the surface lattice. This software is tuned for Si(001), so the sample frame is composed of two directions at right angles relative to each other, along (110) directions, along and across the dimer rows. Typically the sample frame of reference is almost the same as the scanner frame of reference, depending on how the sample has been mounted into the sample holder.

## Choosing Lithography Parameters

In order to select proper lithography parameters for AP litho, a convenient script to use is `AP_parm_search` (and similarly for the FE litho parameters `FE_parm_search`), which draws a series of lines, cycling through a selection of possible effective AP litho parameters, as defined in the following table. After running `AP_parm_search`, the user can select the line that produces the desired linewidth. For the FE litho settings, either a set of lithography parameters can be chosen while noting the saturated and total linewidth, or else a line can be chosen based upon a preferred linewidth.

To use the chosen parameters, manually input them in the AP preset field in the Litho tab. A convenient alternative is to run the script `set_ap_preset(n)` to input the relevant litho parameters from the chosen nth line. On the command line, these values are defined for the AP and FE litho parameters as `litho_preset_values[0]` and `litho_preset_values[1]` respectively.

## Drawing Simple Patterns

- Dot cycles into and out of lithography conditions without moving the tip laterally. During lithography, the tip rests for the time defined by the first Time Delays value in the Litho tab.
- H line draws a horizontal line with length x nm.
- V Line draws a vertical line with length y nm.
- X draws an X shape of x by y nm.
- Rect draws an outline of a rectangle of x by y nm.
- Box draws a series of nominally horizontal lines to fill in a box of x by y nm, with a line by line pitch of Patt parm.
- Spiral draws a counter-clockwise pattern using Patt parm turns to make a shape with outer size of x by y nm.
- Serp draws a serpentine pattern of size x by y nm, with the primary line axis parallel to x at 0° rotation, and with a line pitch of Patt parm.

## Using Feedback Controlled Lithography

Feedback-Controlled Lithography is a useful technique for removing one or two H atoms at a time. A very low bias voltage is used, which gives a long time between depassivation events. To use this, select Litho trap in the Tip Tab. The settings allow for a trigger to be defined, usually the spike in current associated with H atom removal, which cause the tip to move out of litho conditions, either changing tip height or bias voltage of setpoint current. Trap dI sets the current trigger, Trap V and Trap I are the new tip conditions after the trigger, and Trap dZ allows for the tip to be retracted after the trigger. Useful example scripts using these parameters can be found in the commands folder, under *controlled_single_desorption.py*.

# Automated Lithography using Scripts

The main purpose of script-based lithography is to automate the process of writing complex patterns, containing many elements and perhaps using both AP and FE mode lithography, which is well beyond the scope of the commands available in the Litho tab. Scripts are also the only way to perform lithography aligned to the Si(001) lattice, where the dimensions of the lithography are defined in terms of surface litho pixels (px) rather than nm. In this way, patterns are automatically scaled to fit the surface lattice detected in the STM. Automating the lithography within a script allows for better alignment to the lattice, and automated correction of position errors. This is crucial for atomically-precise lithography. Equivalents to the simple pattern commands listed above, but designed for lattice-aligned lithography, are available in the Script Menu Panel (they are listed in the Scripting chapter). Many other commands and functions are available within ZyVector Scanz which can be run individually, or combined in larger scripts for performing specialised patterning tasks. For example, the tip may be instructed to move to a particular location, take an image to lock to the lattice, draw a pattern, move on elsewhere, check its location against a fiducial mark, write another pattern, etc. For a list of the preinstalled scripts, please refer to the Scripting chapter. A list of commands is given in the Glossary in Appendix 1.

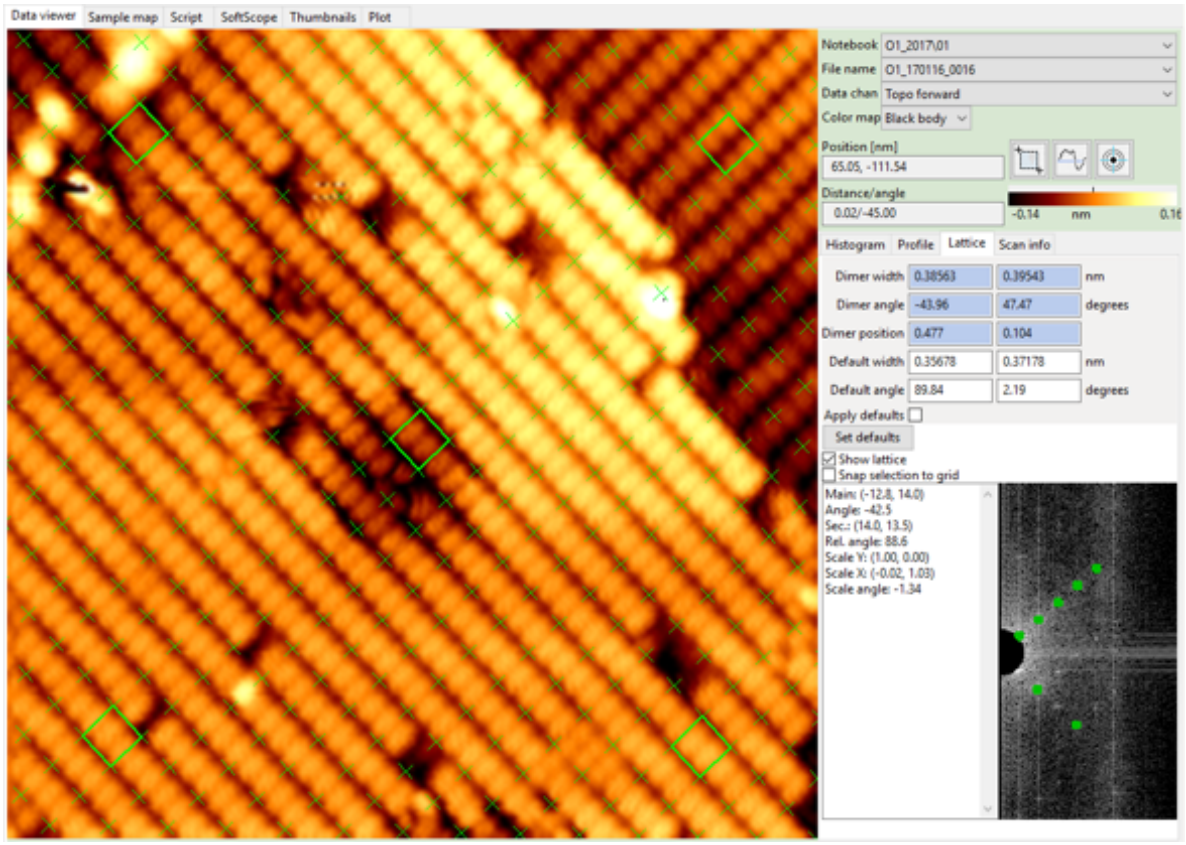



**Figure 5.2: The Lattice tab, with pixel grid overlay on the STM image.**

## Setting Lattice Parameters for Lithography

In order to perform lattice-aligned lithography, it is necessary to determine the location and direction of the Si(001) dimer rows. One benefit to performing lithography using scripts is the capability to use real-time data analysis to help define lithography vectors. More importantly, data analysis can determine dimer row directions for Si(001) as well as the phase offset of the periodic structure in order to align the vectors relative to the local surface lattice. Figure 5.2 shows an example of an image along with all of its lattice parameters listed in the Lattice tab. In the Lattice tab, 2-D Fourier Transform data, Dimer width, Dimer angle, and Dimer position for the image shown in the Data viewer tab are shown, relative to the Image frame of reference. Default width and Default angle represent stored data for those relevant parameters, which are defined globally relative to the Piezo frame of reference. Because the speed at which the tip travels across the image and down it during scanning is so different, the lattice numbers for the two directions will typically be rather different. Furthermore, drift or some residual creep will also affect the lattice data from an image. To set up global lattice parameters for use in lithography, therefore, it is better to run the `SetDefaultLattice()` script, which takes multiple images to check for consistent lattice data. The Default width and Default angle values can also be updated either by pressing Set defaults to use the values from the current image, or by manually entering values into the fields themselves. The Default values are used when the Apply Defaults radio button is selected in the Lattice tab, otherwise the values from the most recent image are used.

## Using bitmap input files

Complex patterns can be written by listing a sequence of simple patterns within a script. However, an alternative way to draw a complex pattern is to define the pattern as a black and white bitmap (bmp) or png file. Using this as an input, the script `Multimode_VectorGen` will read the file and convert the area defined by white pixels into lithography vectors. If the Apply defaults radio button is selected, the default parameters will be used for vector length, otherwise the parameters from the most recent image in the Data Viewer tab will be used. In nearly all cases, the default lattice dimension and direction parameters are preferable. However, the local dimer position information from the most recent image will always be used. Typically, images up to 256 pixels x 256 pixels (197 nm x 197 nm for Si(001)) can be patterned with little error, but larger patterns are certainly possible. Note that within the scripting window, the program searches for the bitmap to be produced with the root folder being the commands folder. To select a bitmap relative to the commands folder, enter `"subfolder\\bitmap_name.png"` with the quote symbols included. This is described in more detail in Appendix 4.

## Using pre-defined input vectors

In the case that the user would prefer to use an external tool for determining lithography vectors, ZyVector SCANZ has a facility for reading lists of vectors from text files. The script `draw_lines()` reads a list of vectors in units of px with AP or FE mode used for files with suffixes of "_AP" or "_FE", respectively, or as defined by the user.

## Handling SVG input files.

The Scanz software features builtin support for writing complex patterns generated using standard CAD packages. This is accomplished via the Scalable Vector Graphics (SVG), file format. If your pattern is currently in another format, such as GDSii, DWG, CIF, etc, you will need to export your design to SVG format. This feature is provided by most CAD packages, please check the manual for your particular package for instructions on how to accomplish this export. If your design includes multiple layers, export each layer as a separate SVG file. Details about converting an SVG input file into a SCANZ lithography script are given in Appendix 6.

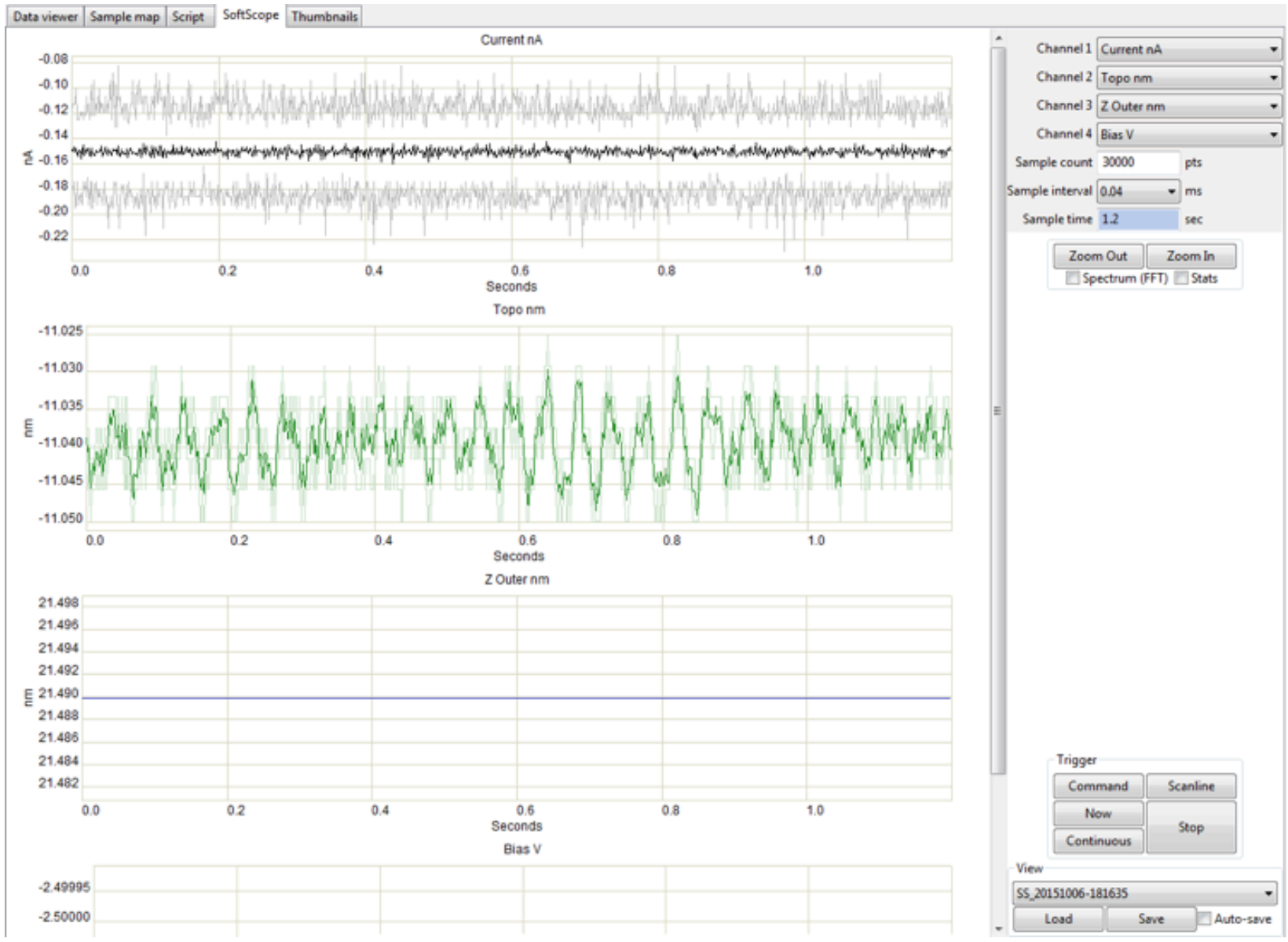



**Figure 6.1: Details of Softscope window.**

## Softscope

ZyVector SCANZ has built in flexibility to observe many types of signals either in real time or well after the fact. Softscope can read up to 4 separate signals at once, as shown in Figure 6.1. The signals include outputs from the DSP to the STM such as X fine, Y fine, Topography, X coarse, Y coarse, Z coarse, and Bias. Also included is the current input to the DSP, as well as metadata such as the instantaneous error function for the feedback. Besides displaying a binned average for each trace, the maxima and minima of each trace are displayed in lighter colored traces.

The channels to be viewed can simply be selected from the drop-down menu for each channel on the top right part of the Softscope tab. The sampling parameters are found below those the channel selectors. After a trace has been acquired, it is possible to zoom into and out of the desired data range by clicking the appropriate button. The zooming will be relative to the center of the displayed data range, and the vertical scale will autoscale. Loading a new Softscope trace will load the entire trace, not a zoomed trace. Oftentimes, it can be useful to observe a Fourier Transform of the acquired data which can be done by clicking the Spectrum (FFT) check box. The Fourier Transform can act on a static trace or can be updated continuously.

Softscope traces can be triggered in several ways. Clicking the Command button will prime Softscope to start data acquisition when the next command is executed, either in the user interface or script. Now and Continuous both start on the click of the button, with Now collecting a single line and Continuous collecting data until Stop is clicked. Scanline will display the forward and reverse topography and current. Deselecting Scanline will return to the previous sampling conditions.

Softscope traces can be loaded or saved by clicking the appropriate button at the bottom of the Softscope tab, with data saved in `*.csv` format. If operating in Continuous mode, only the most recent trace can be saved. If Auto-save is clicked, each trace will be saved into a new file.

# Chapter 7: Maintenance and Troubleshooting

In this chapter we discuss the various operations that are required to obtain the best possible lithography. Some operations are needed for each new sample put into the system, or after each bake, or should be performed at intervals so that ZyVector accurately represents the fine-tuned behavior of the STM system.

## The System Tab

Many settings, which are rarely changed, and some of which cannot be changed inside the software, are found in the System Tab shown in Figure 7.1. These settings are normally set during system installation to define the system name (which gets written into data files), and to set calibration scale factors for the STM scanner (i.e. the number of nm moved, for a 1 V change in piezo voltage). Most of these are set up by the site-specific file named `APE\config.py`.

User ID is the name of the operator, and is embedded within the file headers.

System ID is the identification for this system. This is written into every data file as an identifier specifying which system captured that data. This is read from the site configuration file and cannot be changed here. All fields, like this one, which cannot be changed in the User Interface appear blue in the software. The Scanner ID can also be set if desired, to identify the particular STM scanner.

The Piezo box and Piezo motor settings control configurations for different control boxes, and different STM hardware.

Calibration X/Y and Calibration Z define the calibration factor that relates the movement of the fine piezo X/Y and Z to the drive voltage. Once calibrated, this should not need to be changed. The initial value for this is read from the site configuration file `config.py`, and it is presented here for reference only.

After calibration of the creep and hysteresis, as described below, the relevant parameters are input here.

Bias multiplier supports an external sample bias amplifier. With some STM hardware the input from the DSP will be multiplied to give sample bias voltages greater than 10 V.

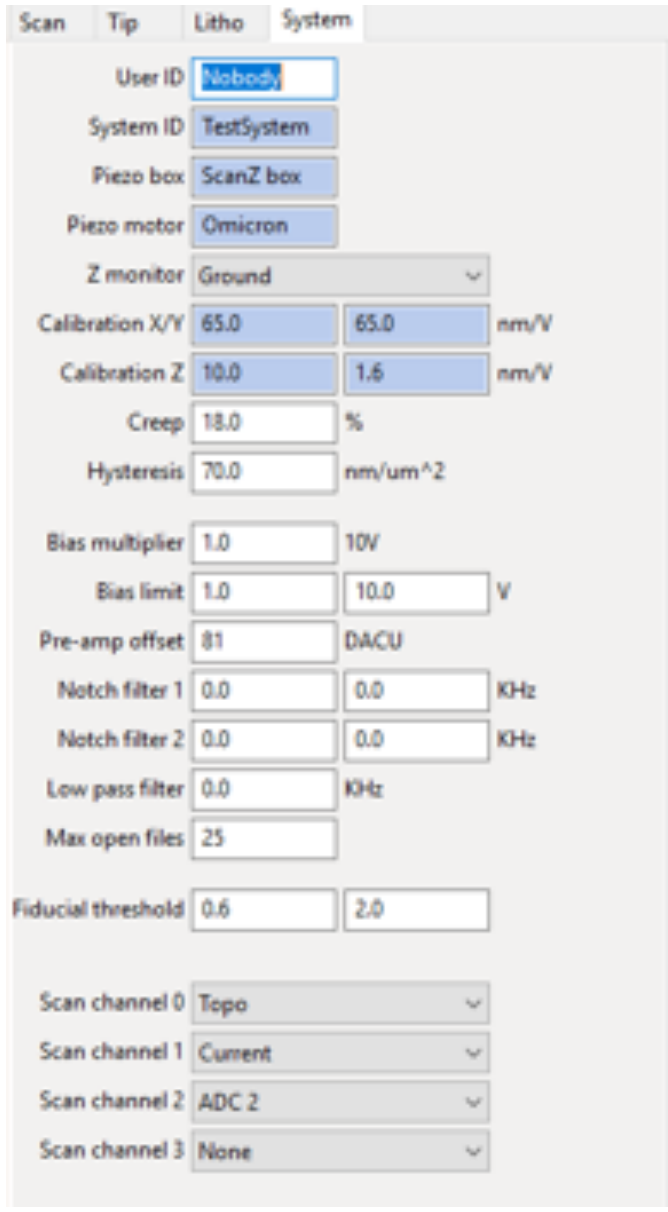



**Figure 7.1. The System Tab.**

## Calibrating Creep Parameters

One major source of position error is piezo creep. When a voltage is applied to the piezo tube, it will change its length, thus moving the tip in x,y, or z. In practice, only about 90% of the motion will occur when the voltage is first applied, and the last 10% will occur over some period of time, up to 10,000s. This last 10% motion is known as creep. On an uncorrected STM, if a large sideways movement is made, and then an image is immediately taken, the image will appear curved, as the tip continues to creep. An example of creep resulting from a 500 nm move is shown in Figure 7.2. Without creep correction, movements of the tip across the surface will not end up where they are supposed to, and any lithography will be written in the wrong place.

Furthermore, creep is also evident in the millisecond timescale, and causes distortion of distances along each scan line of an STM image. This is the cause of the offset typically seen between the forward and backwards scans of an STM image (often ascribed to hysteresis), and a distortion of the apparent dimer row width across the image. Correction of creep across all these timescales is necessary for atomically precise patterning.

The Creep parameter is set in the System tab. It can only be changed when the tip is disengaged from the surface. It comprises a ratio, given as a percentage, representing the size of the creep for the scanner. The creep parameter will have been set during the initial installation, but may not be optimal. Moreover, it is found that there is some change in the creep behavior of a piezo over time, and particularly after baking. Therefore, it is recommended to run the creep calibration script after each vacuum bake, and whenever there appears to be an issue with uncorrected creep.

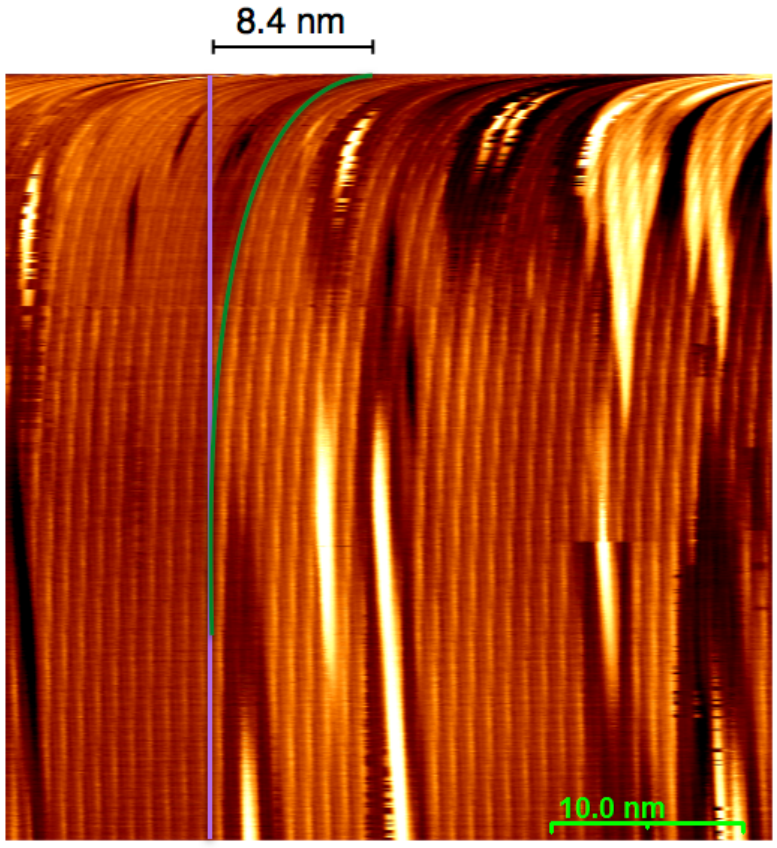



**Figure 7.2. STM data from a creep calibration test script. A 200 nm jump to the right has been made. As a result of creep undercorrection, the tip continues to move to the right. By scanning the same line over and over, the residual motion of the tip can be measured by the curvature of the dimer rows. The offset of the dimer rows over the course of the 1024-line image gives the required creep parameters. Here for a 200 nm jump, the uncorrected creep is 8.4 nm or 4.2 %.**

The main script for setting the creep parameter from scratch is `Creeptestmulti()`. The total offset of the dimer row during a 1024-line image is measured, and the ratio of this offset to the jump size is then input into the Creep parameter box. In practice, it is found that one pass at creep correction is never sufficient. Furthermore, some of the apparent creep at the beginning of the image is due to z motion of the tip. Therefore, since the creep correction is linear, a convenient process is to input half the apparent required correction, and rerun the creep script. Thus the effect of the z creep is reduced in successive runs, and the remaining creep can be measured and corrected with more precision.

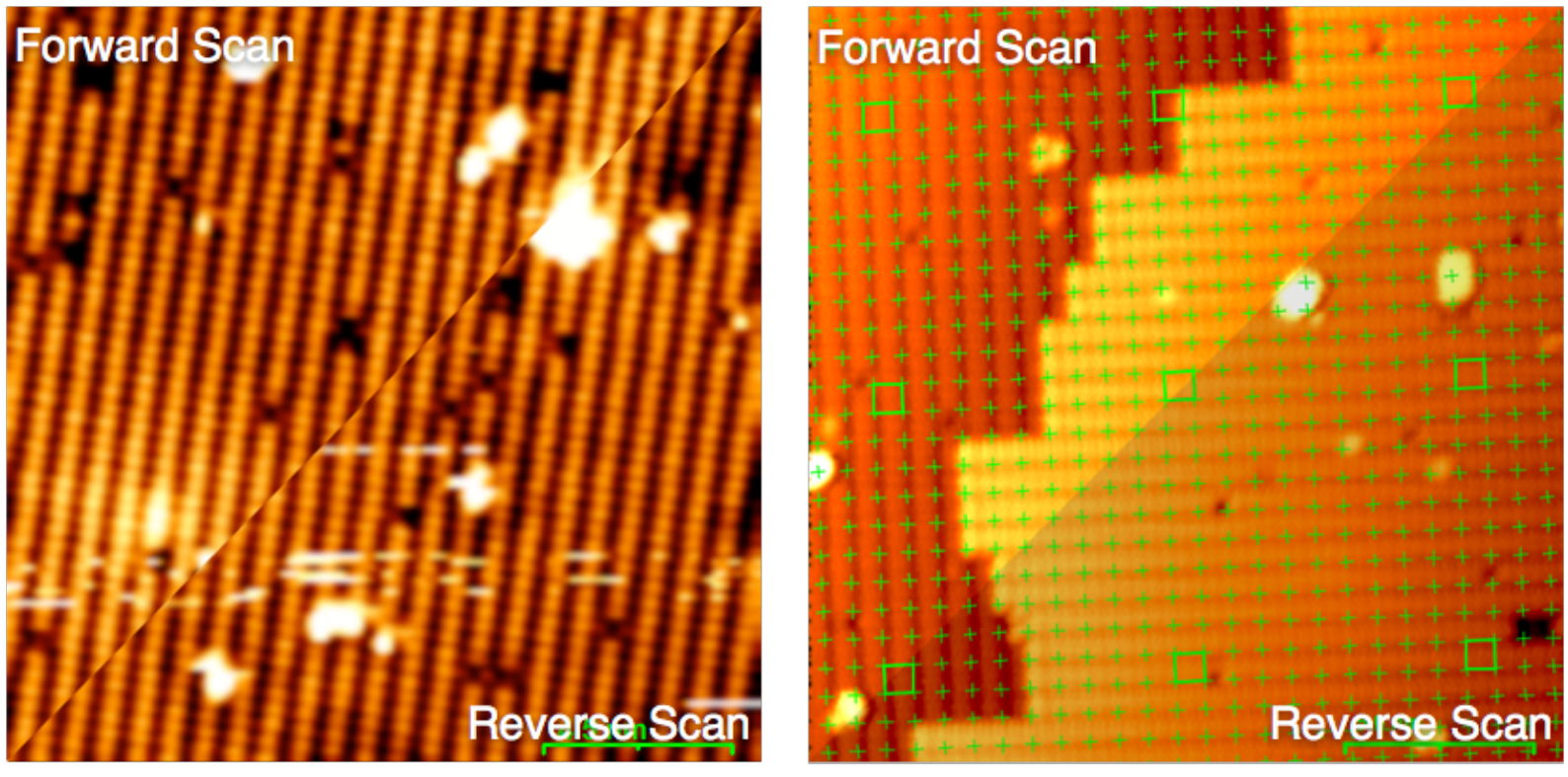



**Figure 7.3. Fast creep causes the offset between forward and reverse scans in STM. Left: Without creep correction, an offset of 3.6 Å, about half a dimer row, is visible between the two scan directions. Therefore, it is very difficult to position the tip along the top of the dimer row with precision. Right: With creep correction, the two scans overlap exactly, and the tip can be positioned on top of the dimer rows with confidence.**

## Calibrating Hysteresis Parameters

A second major source of position error is hysteresis in the piezo. When a voltage is applied to the piezo tube, it will change its length, thus moving the tip in x,y, or z. If the voltage is returned to zero, the tip will not be located at the original location. The offset is caused by hysteresis. Unlike creep, hysteresis is non-time-varying, but instead varies with the size and history of the piezo motion. Also unlike creep, hysteresis is not a linear function of the motion but quadratic, so that the errors resulting from hysteresis become proportionally bigger for large motions. Complicating the correction of hysteresis is the fact that the state of the hysteresis affects the size of the observed creep. Thus for accurate creep correction over larger distances, the hysteresis must be corrected at the same time. The way to do this is to draw a marker at (0,0). Turn off creep correction, which will confuse the results. Then move `scan.center` a large distance in one direction, e.g. to (500, 0) and then immediately return to (0,0). Image the area and measure the offset of the marker from the centre of the image. This will be the hysteresis. Repeat this for larger and larger steps, measuring the total offset each time. This should give a quadratic relationship of the hysteresis relative to the jump size. The parameter required to input in the Hysteresis field in the System tab is the hysteresis for a 1µm jump. There is a script, `hysteresisjumpx,` to perform this process automatically. It jumps back and forth a set distance, drawing boxes after each point. The offset between the boxes gives the creep value.

Currently, the combined Creep and Hysteresis Correction model tends to correct small motions resulting from scanning or moving across the surface very well, but overcorrects large motions, such as landing a tip.

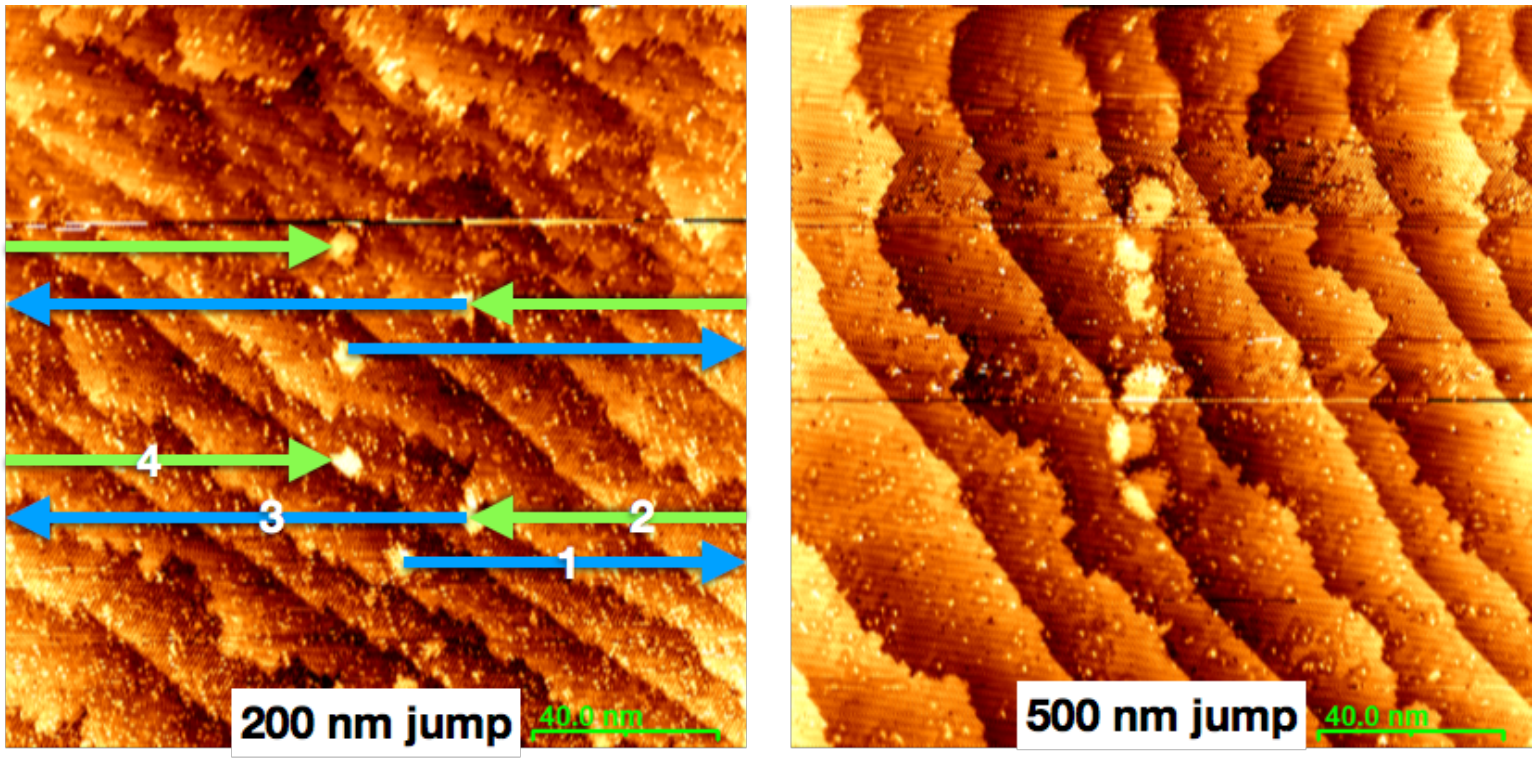



**Figure 7.4. STM data from the hysteresis calibration test script, `hysteresisjumpx`.**

**Left: A box is drawn, then a 200 nm jump has been made to the left and back again (arrows 1 & 2) and a second box drawn. Then a 200 nm jump is made to the left and back (arrows 3&4) and a third box is drawn. This process is repeated a second time. The hysteresis offset is measured as the distance between the second and third boxes.**

**Right: A similar data set after applying hysteresis correction. Now the boxes are all almost exactly aligned, even after a larger 500 nm jump.**

## Calibrating Lattice Parameters

There are a number of reasons why the lattice parameters should be calibrated. Accurate motion across the surface requires a good calibration of the scanner, as well as creep correction. Using the imaging mode of the STM, it makes sense to calibrate the STM against the known atomic crystal periodicity of the Si(001)-2×1 reconstruction.

While the STM should be calibrated to give the correct lattice dimension at (0,0), the apparent local periodicity of the dimer rows will vary with the xy position of the scanner, particularly at the outer edges of the scan range. Moreover, each new silicon sample will have been mounted in a slightly different way, and therefore the precise angle of the dimer rows will be different.

Furthermore, in order to perform lithography aligned to the surface lattice, the local position and the periodicity of the dimer rows must be determined, as the yield of the lithography can vary with the precise location relative to the center of the dimer row. For each new sample, therefore, and if the tip location is well away from the Centre of the scan range, the `SetDefaultLattice()` script should be run to determine the local default lattice parameters to be used in lithography. As a result of tweaking the creep calibration, it is likely that the xy calibration is also slightly different.

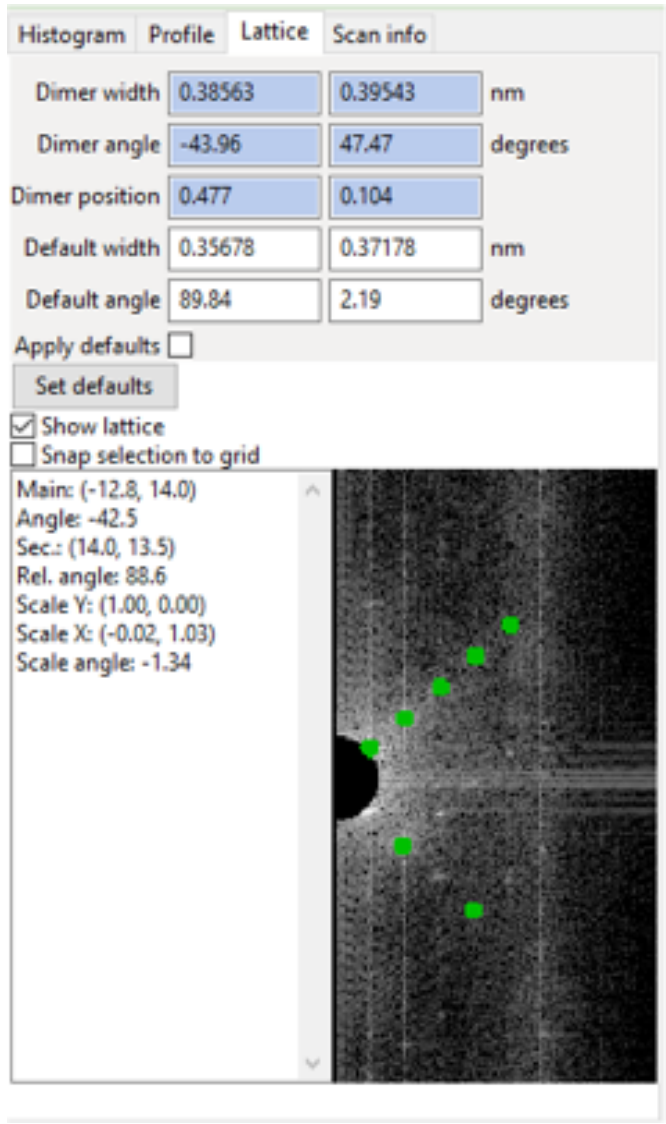



**Figure 7.4: The Lattice Tab.**

## Calibrating XYZ piezos

Once the lattice parameters are well known from the `SetDefaultLattice()` script, the piezos may be accurately calibrated since the actual dimer row width is known to be 0.768 nm. To achieve the best piezo calibration, the Si(001) sample should be mounted so that the dimer rows are approximately aligned with the x and y piezos, otherwise the x and y calibrations will not be entirely separable.

To reset the calibration, open `APE/config.py` in a text editor. Next, in the section with the heading "`#SCANZ`" find the line `config['Calibration'] = [X, Y, Zfine, Zcoarse]` (note that the calibration parameters will come preloaded with approximate values).

- To update the X value apply the formula `X * 0.384/Default width(0)` where `Default width(0)` is the left field of the Default width line in the Lattice tab.
- The formula for Y is `Y*0.384/Default width(1)`, which is the right field.
- To update Z, find a step edge of Si(001) and measure its height using the Profile tab, then apply the formula `Zfine*0.135/measured step height(nm)`.
- Save the `config.py` file and restart SCANZ to enable the new values.
- Rerun the `SetDefaultLattice()` script to check that the lattice dimensions are now 0.384 nm.

# Appendix 1: Glossary of some commands and useful scripts:

While each of the following parameters can be set from within the user interface, they can also be set on the command line, or within a larger script via the following commands. This is not an exhaustive list of all possible commands; they are the most useful. A complete list of available commands can be accessed from the command line in the Script tab, by typing the first part of a command, and then the dot, and then pausing. A menu will appear with the available options. However, many of these options are for internal use, and are not appropriate for use in scripts.

## Scan Parameters

Forward (Vf) and Backward (Vr) scan bias voltages

```
scan.bias = [Vf, Vr]
```

Forward (If) and Backward (Ir) current set point in nA

```
scan.set_point = [If, Ir]
```

X/Y Location of image center in nm

```
scan.center = [x, y]
```

Size of the scan area  in nm

```
scan.size = [x, y]
```

Number of pixels in an image, usually in powers of 2 (e.g. 128, 256, 512)

```
scan.image_size = [x, y]
```

Forward (vf) and Backward (vr) tip velocity in nm/s

```
scan.speed = [vf, vr]
```

Rest position of the tip within scan window. Typically top middle [0,1] or center [0,0]

```
scan.rest_position = [x, y]
```

Rotation angle of the scan window in CW degrees. Apparent rotation of surface is CCW.

```
scan.rotation = theta
```

Returns the pixel center closest to the center of the image

```
scan.lattice_center
```

Moves scan.center by a distance defined in pixels

```
scan.lattice_recenter(deltax, deltay)
```

Full scan range in X/Y in nm. Available full ranges: 675, 2362.5, 9450 nm

```
scan.xy_full_range = 675
```

Full scan range of Z in nm. Available full ranges: 96.4, 337.5, 1350 nm

```
stm.z_full_range = 337.5
```

Sets tunnel current range: 3.3 or 330 nA

```
stm.preamp_max_curr = 330
```

Command to initiate a scan using previously defined image parameters. Default is for one image to be taken (allow_auto_scan=False). For (allow_auto_scan=True), system will continue to image until the Cancel button is pressed.

```
scan.start()
```

## Lithography Parameters

```
Bias voltage used during lithography in V
```

```
litho.bias = 5.5
```

Current set point during lithography in nA

```
litho.set_point = 2
```

Electron dose during lithography in mC/cm

```
litho.dose = 0.2
```

Sets lithography conditions to presets. (0) applies AP preset conditions. (1) uses FE preset conditions.

```
litho.apply_preset(0)
```

Defines lithography preset bias voltages, both AP (VAP) and FE (VFE) in V

```
litho.preset_bias = [VAP, VFE]
```

Defines lithography preset current setpoints, both AP (IAP) and FE (IFE) in nA

```
litho.preset_set_point = [IAP, IFE]
```

Defines lithography preset electron dose, both AP (DAP) and FE (DFE) in mC/cm

```
litho.preset_dose = [DAP, DFE]
```

Initiate lithography procedure from pre-configured routine (non-lattice patterns, described below)

```
litho.start()
```

## Setting parameters for non-lattice patterns:

```
Defining one of the available generic patterns to be written.  PATTERN is selected from the
following list: BOX, DOT, H_LINE, RECT, SERPENTINE, SPIRAL, V_LINE, X-CROSS
```

```
lp.Pattern = litho.PATTERN
```

```
X/Y dimensions of the pattern in nm
```

```
lp.PatScale = [x, y]
```

```
X/Y offset of the pattern from the image center
```

```
lp.PatOffset = [x, y]
```

```
Angle of pattern with respect to piezo x direction
```

```
lp.PatAngle = theta
```

```
Parameter required by the pattern. These parameters will vary between patterns. E.g. line spacing
of the serpentine.
```

```
lp.Parm1 = 1
```

```
Integer number of repetitions of the pattern in X/Y directions
```

```
lp.ArrayCount = [x, y]
```

```
Step size between repetitions of the pattern in X/Y directions (in nm)
```

```
lp.ArrayStep = [x, y]
```

```
Origin of the pattern array
```

```
lp.ArrayOrigin = [x, y]
```

```
Angle of the pattern array relative to the image
```

```
lp.ArrayAngle = theta
```

```
True (or False) to turn on (or off) Preview mode
```

```
lp.PreviewLitho = True (or False)
```

## Setting parameters for generic lattice patterns

Writes lithography based on PATTERN, where PATTERN is a list of lines. Each line is defined by a start point and end point, which are (x, y) pairs. E.g. to write a single line:

PATTERN = [((x1, y1), (x2, y2))]

```
litho.lattice_pattern(pattern)
```

# Appendix 2: Installing Gwyddion

Gwyddion (found at gwyddion.net) is a useful open-source image manipulation program for STM and AFM data. The installation of Gwyddion will normally occur during installation. The latest 32-bit version (currently 2.39) is compatible with the 32-bit Python installed, but the 64-bit version is not. A special python script zad_gwy.py needs to be installed in order for Gwyddion to open .zad files. Starting from scratch on a Windows PC, the following programs should be installed from the software DVD:

- `Gwyddion-2.39.win32.exe`
- `pycairo-1.8.10.win32-py2.7.msi`
- `pygobject-2.28.3.win32-py2.7.msi`
- `pygtk-2.24.0.win32-py2.7.msi`

After installing these items, run Gwyddion for the first time. This creates a Gwyddion folder in the Users folder. Then, to install the plugin, copy:

- `c:\APE\code\SCANZ\zad_gwy.py to:`
- `c:\documents and settings\USER\gwyddion\pygwy\`

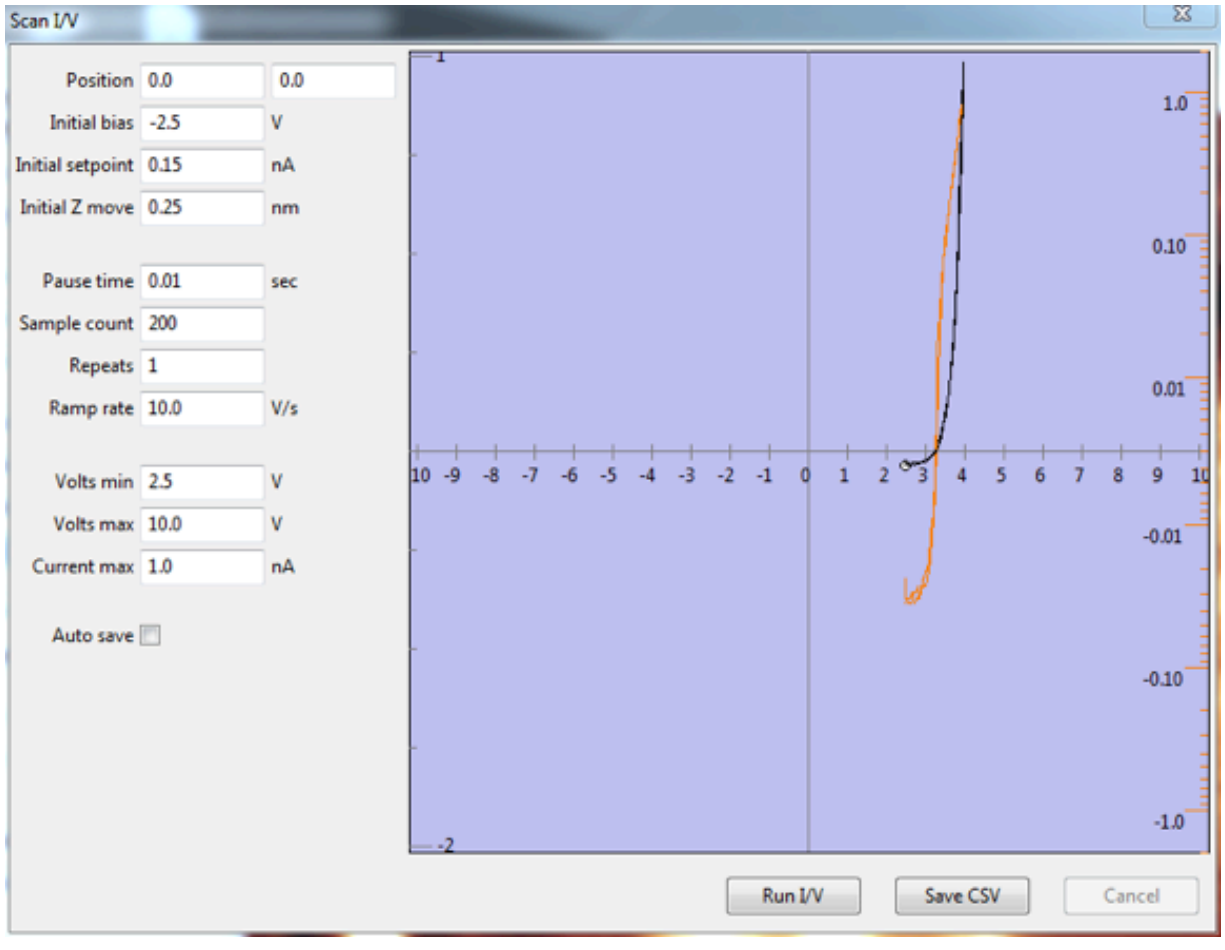



**Figure 8.1: I/V plot dialog**

# Appendix 3: Scanning Tunneling Spectroscopy

ZyVector SCANZ offers a basic Scanning Tunneling Spectroscopy (STS) capability, which is accessed by selecting "I/V Plot" from the Commands menu. The I/V Plot window is shown in Figure 8.1. In this mode, the tip is translated to the desired point on the surface, the feedback control system is temporarily disabled, the voltage is swept over a specified range while measuring the change in tunneling current, and finally the feedback control loop is reactivated. The resulting data can be plotted directly as an I/V curve, or as a calculated dI/dV curve, from which information about the surface's electronic density of states can be extracted. Figure 8.1 shows a representative STS data set. The tunneling data is shown in blue, with the associated dI/dV shown in orange. The horizontal axis here is in units of volts, with the vertical axis showing nA on the center scale, and dI/dV values on the outer scale.

This I/V Plot allows numerous parameters to control it.

- Position: before doing any I/V changes, ZyVector moves to this location
- Initial bias: after moving to the starting point, ZyVector sets the bias to this value
- Initial setpoint: ZyVector sets the setpoint current to this value. The initial tip Z position is controlled by the choice of initial bias and setpoint.
- Initial Z move: after pausing briefly to stabilize the tip, ZyVector turns off feedback and adds this value to the tip Z position.
- Pause time: this specifies the delay time for various parts of the I/V curve analysis process
- Sample count: specifies how many times to oversample each I/V value
- Repeats: specifies how many times to repeat the full I/V curve and plot the results
- Ramp rate: specifies the speed at which to ramp the voltage during the I/V calculation
- I/V volts min: specifies the minimum voltage for the voltage sweep
- I/V volts max: specifies the maximum voltage for the voltage sweep
- I/V curr max: specifies the maximum current allowed during the I/V curve. If the current exceeds this value, that I/V curve is stopped, and the voltage ramped back towards zero

The process for capturing tunneling spectra is as follows:

- Move to the starting X/Y position with the current setpoint, bias, and move speed specified for scanning
- Set the initial bias voltage (feedback loop is on)
- Set the initial setpoint current (feedback loop is on)
- Turn feedback off to maintain Z position of tip.
- Apply the delta Z, if any
- Sleep for one pause time
- Begin capturing I/V data into DSP buffers
- Pause to capture one data point at the starting bias voltage
- Ramp to the maximum positive voltage, stopping if the maximum current is exceeded
- Ramp to the starting voltage with no maximum current check
- Ramp to the maximum negative voltage, stopping if the maximum current exceeded
- Ramp to the starting voltage with no maximum current check
- Sleep for one pause time
- Turn feedback on
- Sleep for one pause time
- Read back the captured data buffers from the DSP
- Loop back to step 4 for the specified number of cycles

# Appendix 4: Multimode Lithography summary

For Multi-mode HDL, more than one mode of lithography must be chosen. In Figure. 8.2(A) a 4 pixel x 4 pixel square has been written on the surface by HDL. This pattern was made using AP mode with a tip-sample bias of 4 V, a current of 4 nA, and a linear dose of 4 mC/cm (10 nm/s tip velocity). In order to aid the determination of HDL parameters that correspond not only to AP mode but also to FE mode, parameter space searches were performed in order to find lithography conditions that produce linewidths of approximately 1 pixel and 10 pixels, respectively. In order to gain an understanding of the robustness of AP mode parameters, the parameter space study was performed varying the voltage from 2.5 V to 5 V, varying the current from 1 nA to 30 nA, and varying the line dose from 1 mC/cm to 30 mC/cm. In this regime, it is believed that the HDL occurs via a multi-electron ladder climbing process which results in very thin lines. Figure 8.2(B) shows one such parameter variation where the two rows of lines were written with the same conditions of 2.5 V to 5 V from left to right, a current of 5 nA, and a line dose of 1 mC/cm. Repeated experiments with many tips shows a threshold of approximately 4 V, 2 nA, and 2 mC/cm for fully depassivated single dimer row linewidth, although the threshold robustness depends on all three parameters. A similar but less comprehensive parameter space search was performed to find adequate FE mode conditions. It has been shown that above 7V, HDL likely proceeds via a single-electron excitation to a dissociative electronic state with a yield that is relatively independent of voltage and current. Figure 8.2(C) shows an example a subset of a parameter space search, with a tip sample bias of 8 V, current of 1 nA, and line dose of 0.05 mC/cm to 0.3 mC/cm from left to right. The main trend seen in this image is that an increase in line dose results in an increase in linewidth while maintaining a relatively constant diffuse, or "unsaturated," HDL outside the main line. In this regime, the tip shape plays a large role in the HDL lineshape, so the lithography parameters and lineshape are quickly characterized for each tip.To perform Multi-mode lithography, the parameter space search results were used to choose approximate Multi-mode HDL parameters, where the AP mode and FE mode writing behaviors are examined, shown in Figure 8.3. Figure 8.3(A) shows two lines that were written with parameters that yield a linewidth of slightly more than 1 pixel. For the tip used in this case, the lithography parameters were 4.5 V, 3 nA, and 1.5 mC/cm (20 nm/s). Whereas a typical single dimer row wide line can be produced with 4 V, 3 nA, and 1.5 mC/cm, 4.5 V was used to assure that any misalignment with the lattice resulted in depassivation instead of a non-written line. Using the same tip in a different location, HDL was performed using FE mode, with lithography conditions of 8 V, 1 nA, and 0.2 mC/cm (50 nm/s tip velocity). Figure 8.3(B) shows the relevant linewidth parameters for these patterning conditions. Where all of the hydrogen has been removed from the surface a linewidth saturated was determined to be approximately 9 pixels, as indicated by the solid double-arrow line. Outside this area, there is still significant depassivation due to direct electron

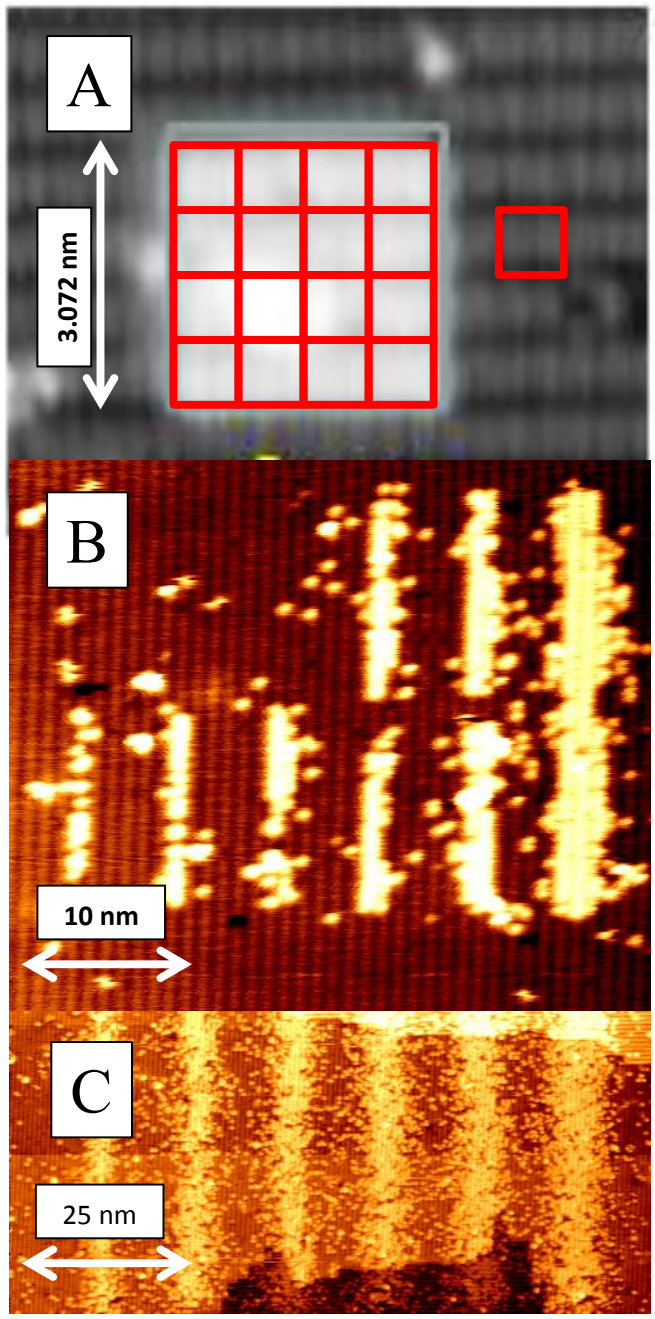



**Figure 8.2: Capabilities of hydrogen depassivation lithography: A: 4.5 nm x 7.5 nm image of the Si(001)-H surface after an 8 dimer x 4 dimer row region has been depassivated. The small red squares indicate single pixels consisting of two adjacent dimers on a single dimer row. The bright areas are dangling bonds where H has been removed, and they have an approximate apparent height of 0.15 nm. B: Sample dataset for finding robust AP mode depassivation lithography conditions. Both rows of lines were written using 5 nA and 0.1 mC/cm. From left to right, the sample bias for each line is 2.5 V to 5 V in 0.5 V increments. Notice the HDL inconsistency below 4 V. C: Sample dataset for finding lithography conditions in FE mode. All lines were produced using 8 V and 1 nA. From left to right, the line dose increases from 0.05 mC/cm to 0.3 mC/cm in 0.05 mC/cm increments.**

interactions with the surface but without removing all of the hydrogen. The dashed double-arrow line indicates the total—saturated plus unsaturated—linewidth and is 13 pixels wide. Once a set of parameters with a saturated linewidth of more than 50% of the total linewidth are found, the lithography parameters and linewidth data are recorded. These observed parameters will subsequently be used as an input to a Multi-mode lithography compiler ZyVectorGenerator to determine optimized tip paths for AP mode and FE mode to minimize write time while maintaining atomic precision edges.

## Vector Generation by Multimode_VectorGen

After the parameters for AP mode lithography have been established and the FE mode linewidth has been characterized, Multimode_VectorGen can convert an input bitmap into AP mode and FE mode write vectors. A square pattern of 100 pixels x 100 pixels is used to demonstrate the procedure for vector generation due to its simplicity, as shown in Figure 8.4. More complex patterns can always be broken down into a series of rectangular areas, so this shape is a fundamental building block. First, the input bitmap is converted into a binary pattern with values of 1 indicating where an exposed pixel is desired, and a value of zero everywhere else. Next, the pattern is shrunk by half of the total FE linewidth (in this case 6 pixels along each edge) by applying a smoothing and thresholding operation. This returns any area where FE lithography can be performed without directly exposing any pixels outside the desired area [Figure 8.4(B)]. Within this area, all possible horizontal and vertical write vectors are calculated. Starting with the longest vector, vectors are put into the vector writing list until an intersection occurs, and the expected saturated area from those vectors is subtracted from the desired FE area to be written. This procedure is repeated until all of the FE area is written. After all FE vectors are determined [Figure 8.4(C)], the expected write area is subtracted from the original image, leaving the portion of the pattern to be written with AP mode as shown in Figure 8.4(D). This area has a width that generally corresponds to the width of the unsaturated region of the FE line, although this region can be much larger, depending upon the feature shape. The AP mode vectors are generated in a similar fashion [Figure 8.4(E)], except without the additional edge operation.

The two vector sets are integrated into a single routine which includes both vector lists, as shown in Figure 8.4(F), with all AP mode vectors written first followed by all FE mode vectors. The longest vector in each AP mode or FE mode vector list is written first, starting from the end nearest the tip rest position. The next vector to be written is the one with a start or endpoint nearest the endpoint of the just written vector, with this procedure repeated until all vectors are properly placed in their list. While not directly affecting lithography write times, this vector ordering procedure reduces tip transit time during the entire lithography process

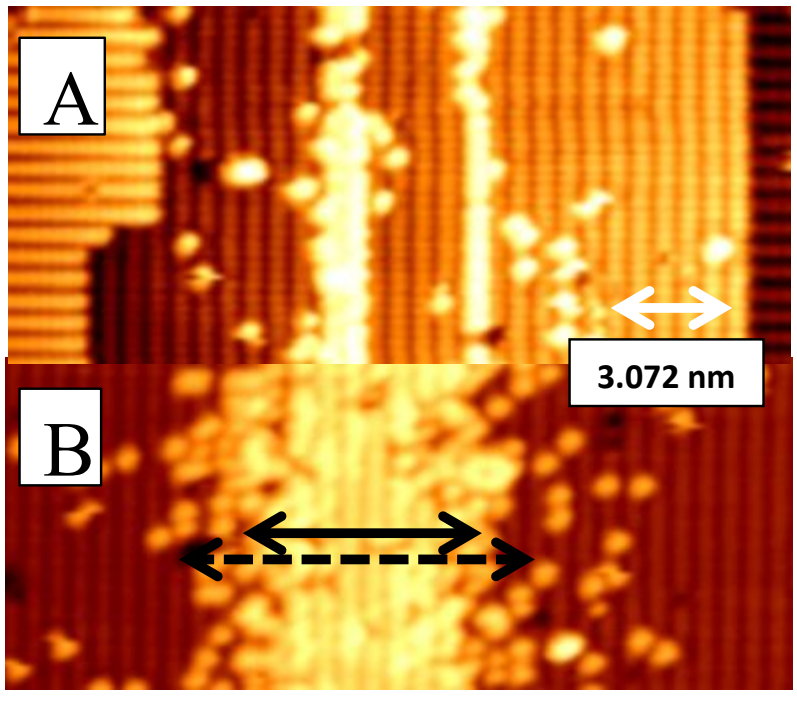



**Figure 8.3: Fundamental linewidth inputs to multi-mode lithography image compiler, ZyVectorGenerator. A: Two AP mode lines were written parallel to dimer rows using the conditions of 4.5 V, 3 nA, and 1.5 mC/cm (20 nm/s). Note that the effective linewidth is between 1 and 2 pixels. Visible in this image are three terraces with step heights of 0.15 nm. C: A single line was written using FE mode using 8 V, 1 nA, and 0.2 mC/cm (50 nm/s). The saturated linewidth, where almost all hydrogen has been removed, is indicated by the 7 pixel wide solid double arrow line, and approximate total linewidth is indicated by the 13 pixel wide dotted double arrow line.**

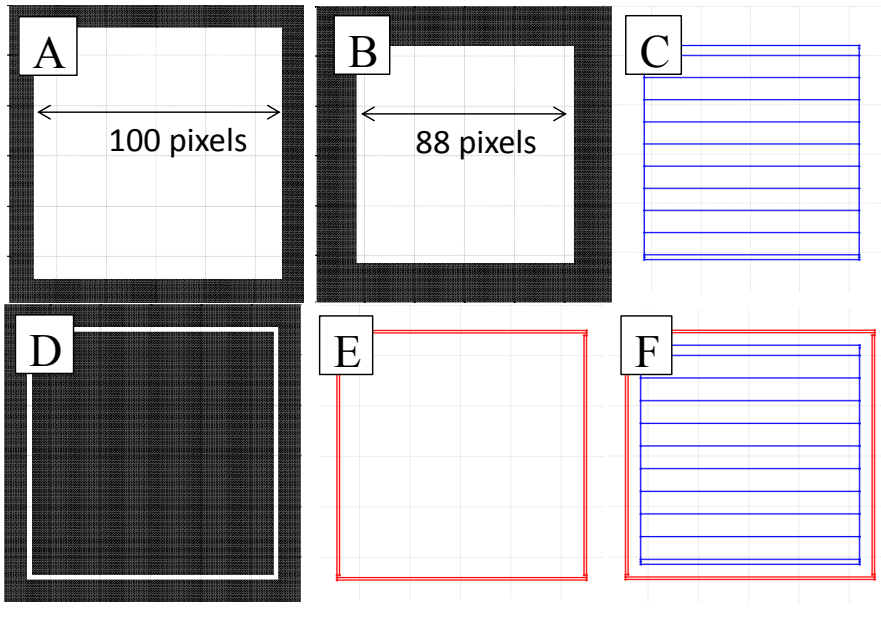



**Figure 8.4: Reference bitmaps and write vectors. A: 120 pixel x 120 pixel input bitmap. White regions are to be written. B: Area to be written by FE mode. C: FE mode vectors. D: Area to be written using AP mode. E: AP mode vectors. F: combination of AP mode and FE mode vectors.**

# Appendix 5: Using Fiducial marks for improved position precision

ZyVector allows for the creation and relocation of fiducial marks for registration following lithographic procedures. Here, we provide a short description of how the fiducial registration process is designed, and provide examples of its use. The basic process is as follows:

1. Write a pattern
2. Image the pattern and store as a fiducial
3. Come back later and collect a large image
4. Relocate the fiducial within that image
5. Store the new location and re-zero to that location

## Structure of Fiducial Find

Fiducial Find provides four key functions, which are combined in functional examples in the following section.

`fid = fiducial_define()`

This function generates a "fiducial" object of the type `fiducial_obj` from the last STM image collected by ZyVector (Figure 1). The fiducial, which is returned by this function, is simply a box containing a copy of the image and relevant information about it. The image itself becomes the fiducial mark. The advantage of this is that additional, otherwise irrelevant, features such as step edges, lattice defects, adsorbates become a part of the fiducial mark, allowing for more reliable feature relocation. It has the drawback that the fiducial mark may not be located precisely at the centre of the image. Generally, it is not necessary to consider the internal structure of the fiducial; it is simply passed as a parameter to the other fiducial finding functions. In certain special cases, it may be helpful to access internal parameters of the fiducial within scripts:

Both the forward and reverse buffers of the fiducial image are stored. In this way, during subsequent calls to `fiducial_relocate`, forward buffers can be compared, and reverse buffers compared independently.

| | |
|---|---|
| `scansize_y` | Height of the fiducial image in nanometers |
| `orig_x_coord` | Original x-coordinate of the fiducial (`scan center` when image was first collected) |
| `orig_y_coord` | Original y-coordinate of the fiducial (`scan center` when image was first collected) |
| `x_coord` | Current x-coordinate of the fiducial (as adjusted after further calls to the `fiducial_recenter` function) |
| `y_coord` | Current y-coordinate of the fiducial (as adjusted after further calls to the `fiducial_relocate` function) |
| `angle` | Scan angle of the fiducial image. Currently irrelevant for fiducial finding, as fiducial image and later fiducial relocation images must utilize the same scan angle. In future versions this may be used to correct for misorientation angles between images. |

- The scan angle of the fiducial image does not have to be zero, but it must match that of the subsequent image in which you are seeking the fiducial mark. In its current form, the script does not accommodate scan angle misorientation.
- Images do not need to have identical resolution in the vertical and horizontal directions. The fiducial finding algorithm will compensate for these variations. For example, it is not uncommon to collect additional data in the fast scan direction, while limiting the number of scan lines in the slow scan direction. In fact, this reduction in scan lines is a good way to minimize the time required for fiducial relocation. However, some calibration (and optimization of your specific fiducial mark) may be required to determine an appropriate line resolution for reliable fiducial finding.
- As implemented, the fiducial finding algorithm does not intelligently seek the mark outside of the current scan if relocation fails. However, it is designed for such extensibility. The (private) function `fiducial_identify()` seeks the fiducial mark within a single image and returns a cross correlation value. In its present implementation, the (public) `fiducial_relocate()` function simply calls `fiducial_identify()`, but can be further developed to suit the requirements of the specific user. For example, it could be designed to expand the scan size of the fiducial if it is not found on the first attempt, or make intelligent predictions of which direction the fiducial may be found, perhaps due to a predictable error such as hysteresis.

`fiducial_relocate(fid, update_coords=False, goto=False, output_images=False)`

This function is the core of fiducial registration. After an image is collected, we desire to know whether and where the fiducial mark is located within the scan field. This is accomplished by calling `fiducial_relocate()` and providing the fiducial (fid) which was defined earlier. The function seeks a representation of the fiducial within the last image collected by the STM. It returns a correlation value which indicates confidence, and the location of the fiducial in both pixels (relative to the upper left corner of the image) and nanometers (relative to the zero position of the microscope). There are two failure modes coded into the function. If the last collected image is smaller than the stored fiducial mark, relocation cannot occur. In that case, an error message is displayed and the function returns a correlation value of zero.

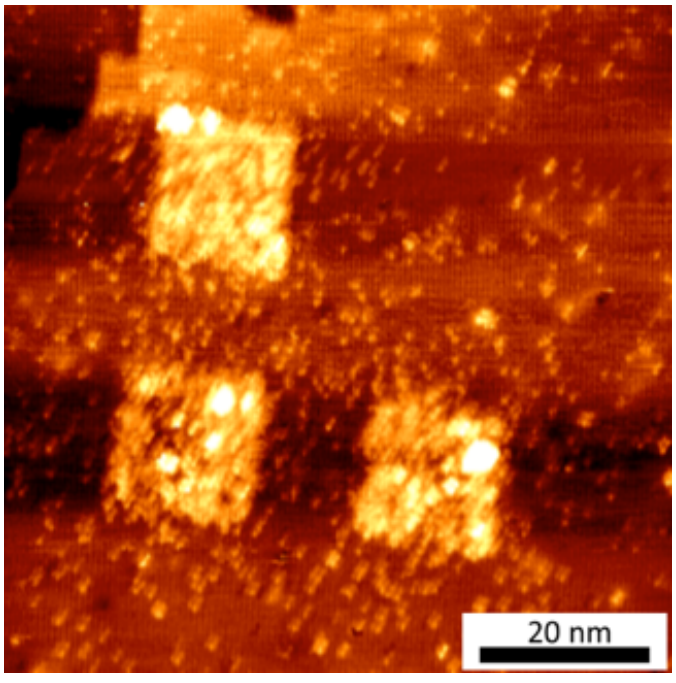



**Figure 1: Representative fiducial mark comprising three squares.**

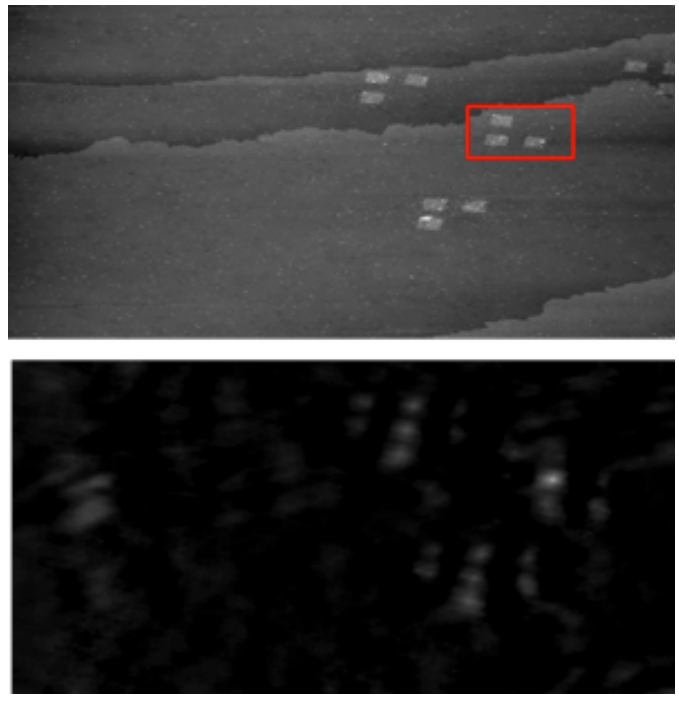


**Figure 2: Results of fiducial relocation. Upper image is an STM topograph with the identified fiducial location marked with a red box. Lower image is a 2D plot of correlation versus position. The brightest spot within the image corresponds to the fiducial.**

If both forward and reverse scan buffers are collected, an additional check is performed to confirm successful identification of the fiducial mark. After searching for the fiducial in both forward and reverse scans, the most probable location in each buffer is compared. If there appears to be an offset between the fiducial marks of more than half the lateral extent of the fiducial, we assume that this is a false positive. In this case, the correlation value is set to zero, but the locations returned by the function correspond to the location reported for the forward scan direction.

Assuming forward and reverse buffers are in close agreement, the position reported for the fiducial mark is the average of these two results. In general, for creep-corrected images, we are seeing offsets of ~1 pixel in the fast scan (horizontal) direction, and perfect agreement in the slow scan (vertical) direction. The remaining discrepancy is likely related to uncorrected fast creep, and thus we average forward and reverse results to provide a best estimate of fiducial location.

There are several flags which can be utilized to extend functionality. If `update_coords` is set to `True`, after the fiducial is relocated, the new location is stored within the fiducial. This is equivalent to calling `fiducial_update_coords()` after relocation completes. Also, if `goto` is set to `True`, the `scan.center` will be changed to the center of the fiducial mark after relocation. This is equivalent to calling the `fiducial_goto()` function after relocation, and can be helpful in cases where future patterning locations are reached by relative motion from the fiducial mark. If `output_images` is set to `True`, four png images are written during the fiducial relocation process for troubleshooting purposes (shown in Figure 2). These include two 2-D crosscorrelation maps which show correlation at each position across the image (one for forward and one for reverse). The best guess at the fiducial location corresponds to the global maximum of this map. Also, two copies of the STM topograph are saved, each with a red box drawn to indicate the location at which the fiducial mark was located (one for forward, one for reverse). These images are saved to the notebooks directory.

```
fiducial_goto(fid, resize_image=False)
```

This function attempts to return to the location of the fiducial mark. It does this simply by returning the `scan.center` to the last known coordinates of the fiducial. If `resize_image` is set to `True`, this function will set the scan size to match that of the original fiducial mark. This functionality can be handled directly by `fiducial_relocate()` if the appropriate flag is set.

```
fiducial_update_coords(fid, xcoord, ycoord)
```

This function resets the stored x and y coordinates of the fiducial mark. This is typically called after the `fiducial_relocate()` function completes successfully. Once you are confident that the fiducial has been successfully located, the coordinates returned by `fiducial_relocate()` can be fed directly to the `fiducial_update_coords()` function. This functionality can be handled directly by `fiducial_relocate()` if the appropriate flag is set, as described above.

## Example Usage of Fiducial Find

Several simple test scripts are provided in `Fiducial_test.py`. Here is one example script for a common approach to fiducial finding.

```
def fiducial_test1(litho_function=None):
    """Simple test script for fiducial finding
    """
    # Lithography function to be used may be provided to the function.
    # If no litho function is provided, use the three squares function.
    # This sample lithography function is provided in fiducial_find script.
    # The lithography function used must collect an image of the pattern.
    if(litho_function is None):
        sample_fiducial_write()
    else:
        litho_function()

    # Starting from the final image collected during sample_fiducial_write,
    # define a fiducial pattern by calling fiducial_define()
    fid = fiducial_define()

    # Store our location, we'll come back to it later
    zero_point = scan.center

    # Now we will collect a large image in which to search for the fiducial
    # 500x500 seems large enough.
    xsize = 500
    ysize = 500
    # But, if somebody wrote a really big fiducial, we'll need to go bigger
    # If the fiducial is larger than 400 nm, add another 400 on top of that.
    if(fid.scansize_x > 400):
        xsize = 400+fid.scansize_x
    if(fid.scansize_y > 400):
        ysize = 400+fid.scansize_y
    scan.size = [xsize, ysize]
    scan.image_size = [512, 256]

    # Add a random offset to our location, as though we had returned from a
    # writing procedure.
    scan.center = [zero_point[0] + np.random.randint(-100,100) , zero_point[1] +
np.random.randint(-100,100)]

    # Now we will take a large scan
    tip.home()
    scan.start()
```

```
# Now we search that large image for the fiducial mark
max_cor,cor_center,fid_center = fiducial_relocate(fid)
print “Max Correlation=”, max_cor,
print “cor_center=”, cor_center,
print “fid_center=”, fid_center,

# Store the new location of the fiducial
fiducial_update_coords(fid,fid_center[0],fid_center[1])

# Take the microscope back to that new location
# Also resize the scan to match the fiducial mark
fiducial_goto(fid, resize_image=True)

# Alternatively, perform the previous two steps in one go,
# and output images to check the correlation.
fiducial_relocate(fid, update_coords=True, goto=True, output_images=True)

# Collect another image to confirm that the fiducial is there
tip.home()
scan.start()
return
```

# Appendix 6: Working with SVG input files using Inkscape

Please visit the Inkscape website, https://inkscape.org/en/release/0.92.3/, to download the latest version of the Inkscape SVG editing software for your platform. This page provides for installation instruction for all supported OSs.

## Handling SVG files.

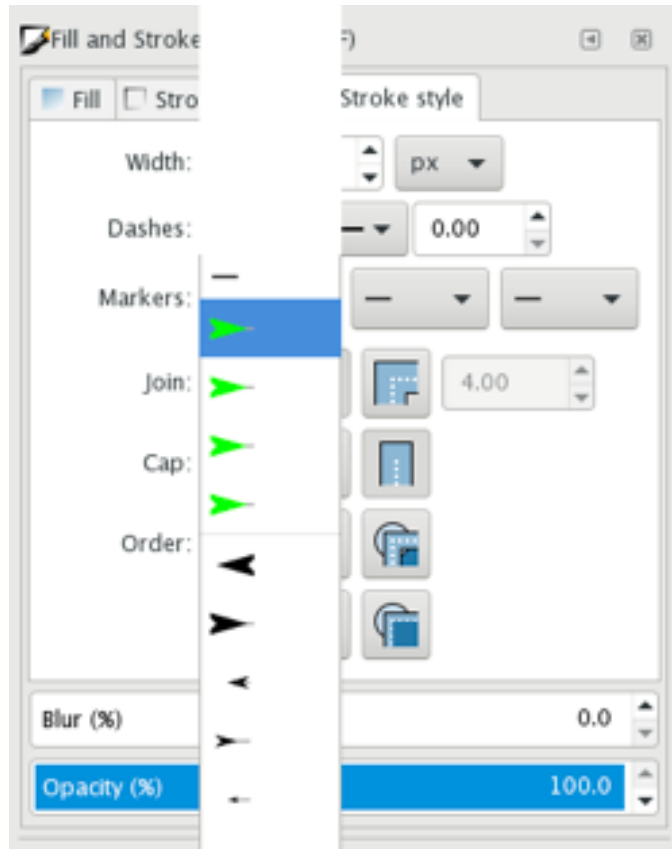
The Scanz software features builtin support for writing complex patterns generated using standard CAD packages. This is accomplished via the Scalable Vector Graphics (SVG) file format. If your pattern is currently in another format, such as GDSii, DWG, CIF, etc, you will need to export your design as SVG format. This feature is provided by most CAD packages, please check the manual for your particular package for instructions on how to accomplish this export. If your design includes multiple layers, export each layer as a separate SVG file and process each following the steps below.

## Verify your Design after Export

Once you have your design in SVG format, open the file in Inkscape. If your design was exported from a CAD package, you will need to verify that the dimensions were maintained during the export. Locate a feature and select it. Verify the Width and Height measurements in the files located above the drawing, as shown in the figure below. If this is incorrect after exporting you need to calculate the scaling factor, Target Size/Current Size. Select all elements in your design and use Object - Transform, Scale Tab to resize your pattern.


## Verify Path

SVG files provide a starting point and direction for the vectors used to draw the shapes in your pattern. This can be arbitrary when the result is to be pixels on a screen, but for directing tip movement, you should verify that these make sense for moving a physical tip over a surface. For example, the start point may need to be close to the centre of the image, or to the Rest Position of the tip. Open the XML Editor in Inkscape, Edit - XML Editor, This brings up a listing of all the elements in your design and the XML code used to define them.



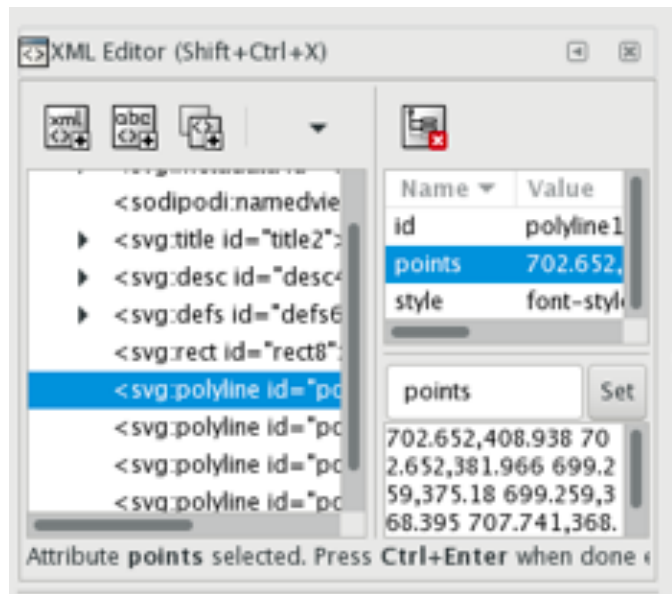

## Example

In order to verify the path starting points, they must first be visible. Select all the objects in your pattern and open Object - Fill and Stroke, Stroke Style Tab, Markers, as in the figure shown, and select the top arrow.

This will result in an arrow being added to all objects for start point and direction. However, the automatic path creation process will not always result in the most efficient path for writing. Usually this can be fixed by changing the start point of the writing. To change the start point, select the object, go to the XML Editor and select Points, as shown at right.

In order to close the path, the first and last coordinate in the Points list must be the same. To change the start point, delete the current last coordinate and copy the next to last coordinate to the first position in the list.

This shifts the starting point as well as the direction of the starting arrow as seen in the figure below.




**As exported :**
**P1, P2….PX, P1**

**After correction:**
**PX, P1, P2….PX**

## Setting Litho Line Spacing

Each shape in your pattern needs to have the line spacing defined for litho. For AP litho, this is set to 1 px, for FE this is set to the desired line spacing for the desired fill condition.

Select all of the objects in your pattern with the same litho spacing. Open Object - Object Properties. In the Description field, enter `litho <spacing>` (use 1 for AP) and click the Set button.

Repeat for each desired spacing. In most patterns, there are likely to be both AP and FE elements.

## Defining the Center

In order to place your pattern correctly within the write field, a center location must be defined. This is done by defining any closed polygon made with a non-intersecting polyline as the center of the field. The object defined as Center will not be written, so you cannot use an existing shape from your pattern. The center object can either be added before export from your CAD package or added with Inkscape's built in drawing capabilities.

To define the Center object, select the object and go to Object Properties, Object - Object Properties. In the Description field, enter "center" and click the Set button.

Save these changes and make this file available to Scanz.

## Preparing for Writing

In Scanz, run the script "`svg_generate_serpentine`" using the file generated above as the input. This will generate appropriate lithography vector paths for the STM tip based on the litho spacing that you defined. This will generate a new file with `_litho` appended to your filename.

## Write the Pattern

Once the vector paths have been generated above, run the script "`svg_apply_litho`" on the `filename_litho` file generated in the previous step to write the pattern to the surface. Use the same file used in "`svg_generate_serpentine`" as input. There are options for scale, rotation, nm_scale and align to lattice, select these as appropriate for your pattern.