*Appendix E*

# Two-Color Tip Enhanced Fluorescence Microscope Hardware, Software and Electronics

## Design Overview

The basic design of the two color TEFM couples an AFM[1] with a homebuilt optical microscope. The system description will be broken into five parts. The first will deal with the central Microscope Hardware. The second section will define all of the filters and operating configurations. The third section shows the lasers and their associated optics. The fourth section describes the Data Acquisition and Control electronics and software (DAC). The fifth section presents the Matlab scripts used for image processing.

## Microscope Hardware:

The assembled microscope is shown in Figure E.1. Figure E.2 shows the section of the microscope where the beams are combined, and then expanded. Figure E.3 shows both optical paths for achieving p-polarization of the evanescent field. One of the paths relies on an optical mask in a similar fashion to that used in the dry TEFM and single-color TEFM. The second incorporates a radial polarization filter, 1:1 telescope and a pinhole that results in a single radially polarized mode being propagated to the sample. Figure E.4 shows the dichroic wheel used to separate the excitation light from the emissed signal. It also shows a tip-tilt mirror used for alignment. Figures E.5 to E.7 show the AFM, sample scanning stage and optical baseplate arrangement. Also included are the 1:1 telescope used in combination with the tip-tilt mirror for beam steering during alignment. Figures E.8 and E.9 show the two optical detection pathways. They also include an optical video camera used for AFM probe-excitation beam alignment.

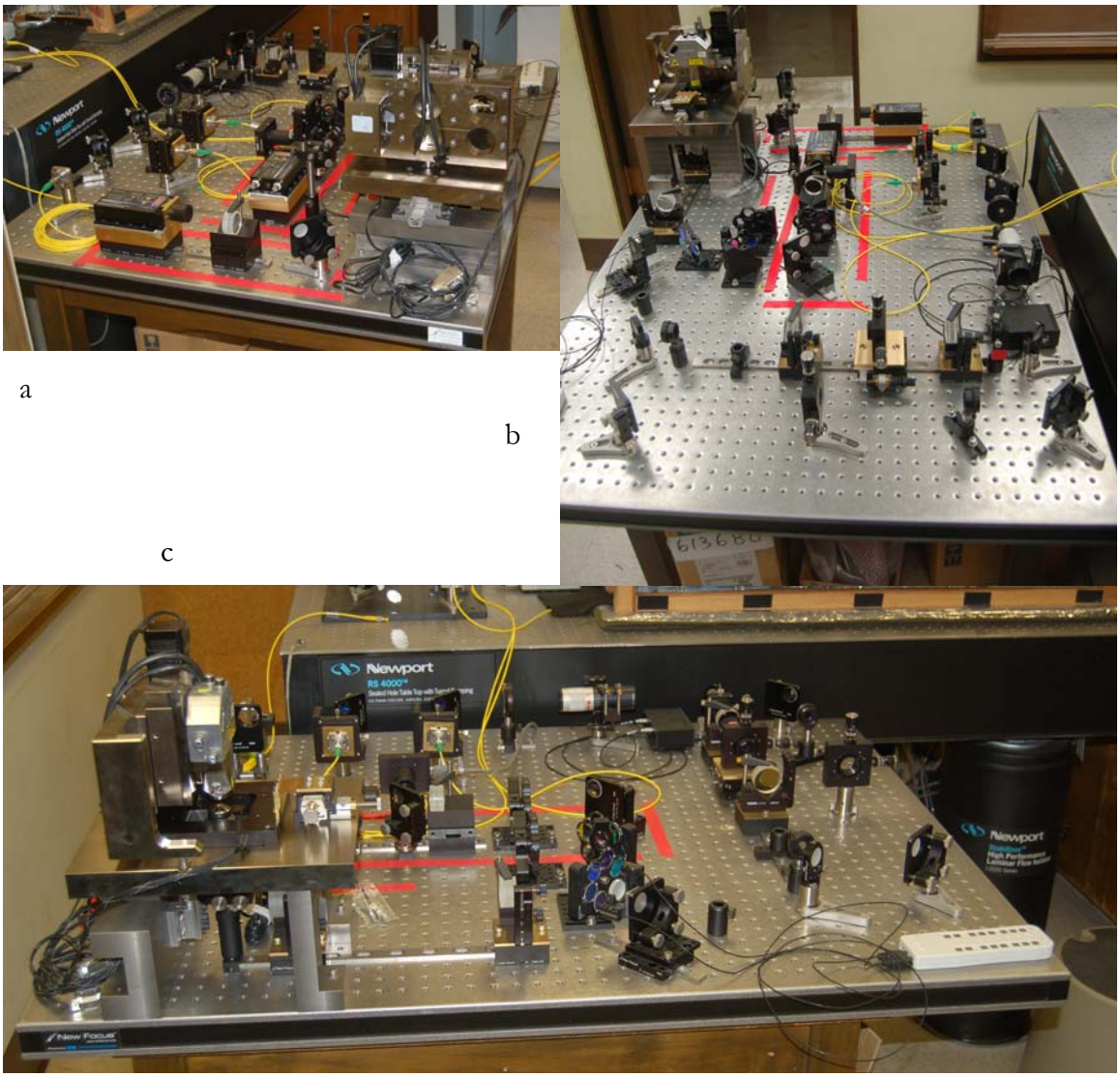



Figure E.1 a, b and c show the complete microscope from three sides.

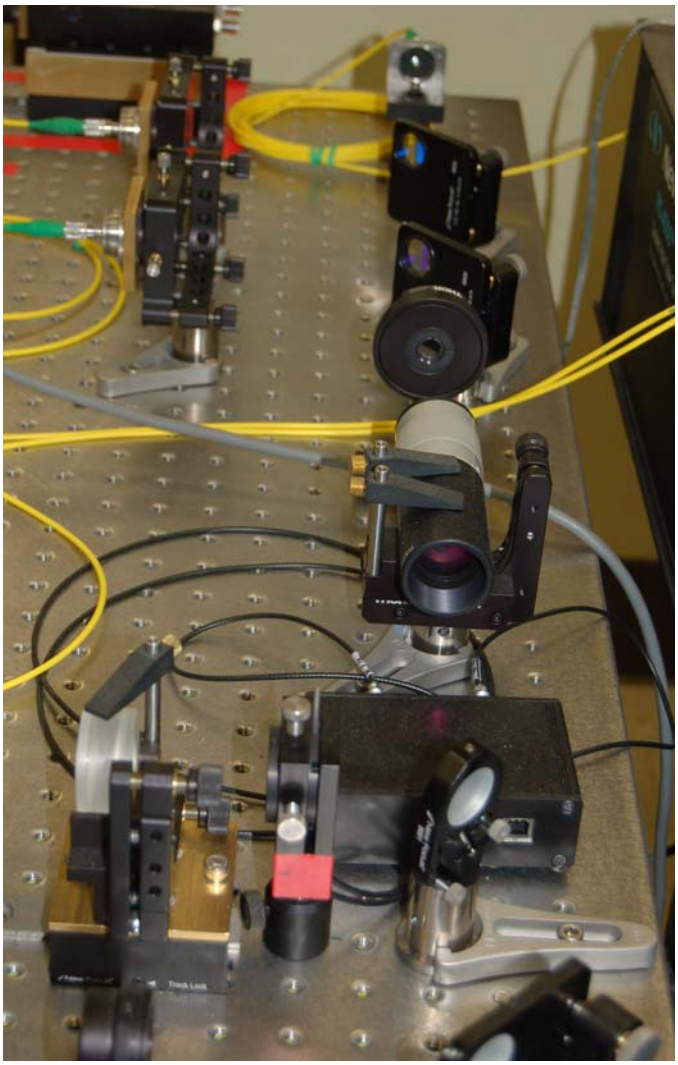


Figure E.2 Three excitation lasers are combined and expanded above. The 543 laser is directly in the beam path. The 502/514 laser is combined next. The 442 nm laser is the last one combined (closest). An apochromatic half-waveplate is provided to ensure uniform linear polarization. A 10x beam expander is also shown.

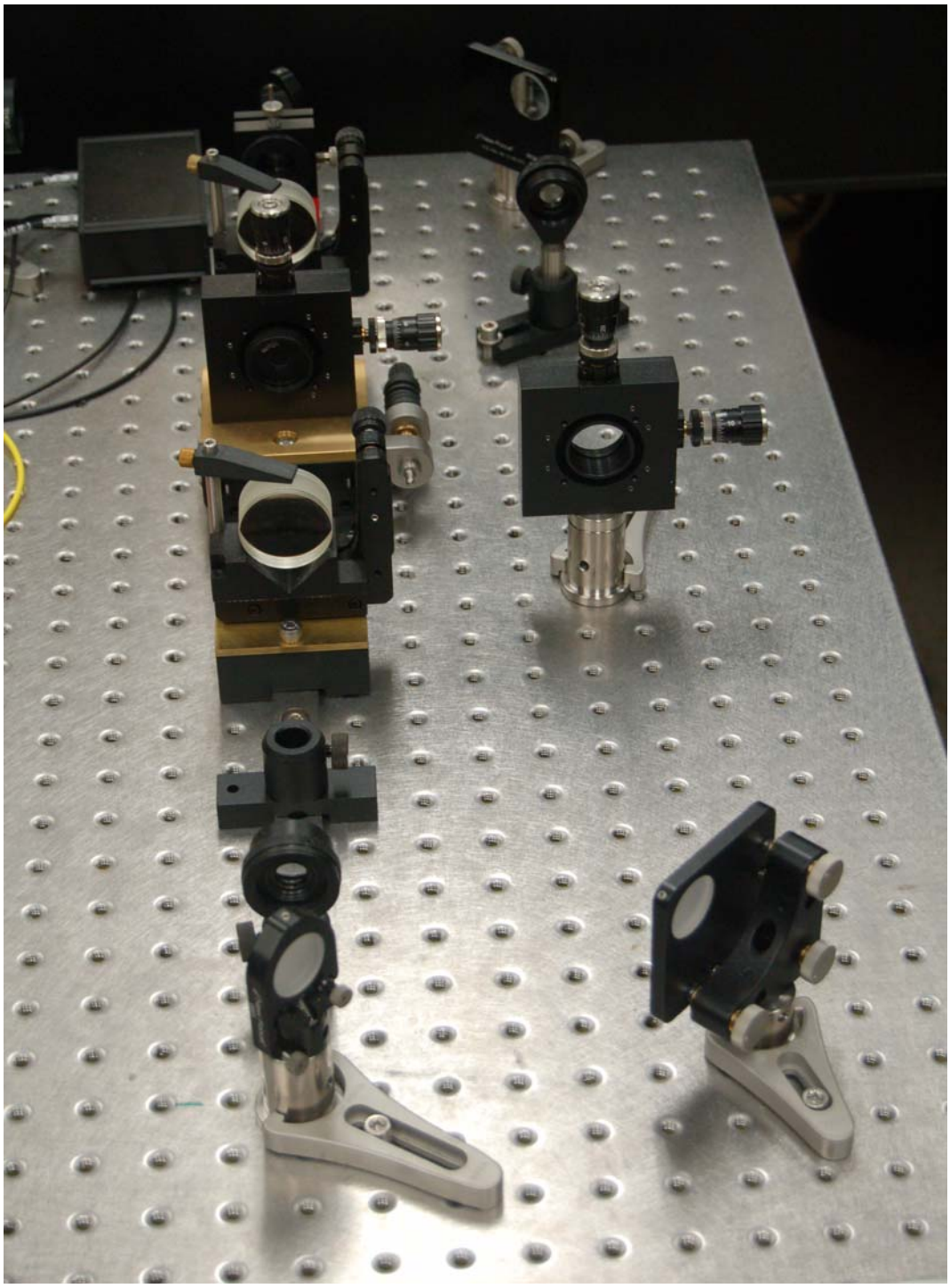


Figure E.3 Two methods for achieving p-polarization of the evanescent field are provided. On the left is the optical path for radially polarizing the excitation light. On the right is the optical path for masking the beam so that only a wedge shaped portion of the back aperture of the microscope objective is illuminated.

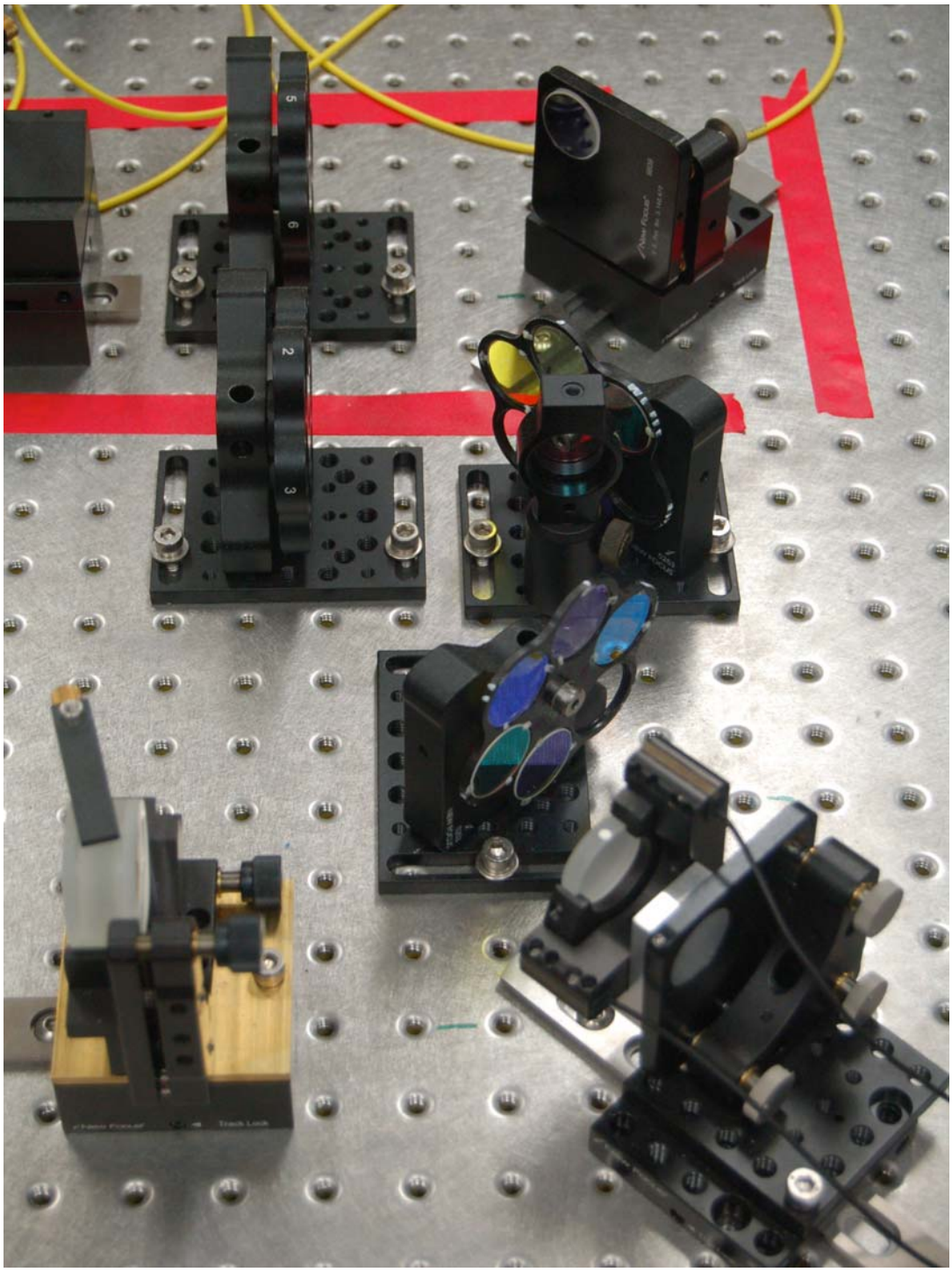


Figure E.4 In the foreground is the dichroic wheel #2 used to separate the excitation beam from the sample emission. The excitation beam then comes forward and reflects off of a dielectric mirror mounted on an actuated tip-tilt fixture. To the left is the first element of the 1:1 telescope that leads to the microscope objective and sample. In the background is the dichroic that splits the emission signal between the two APDs and also the emission filters.

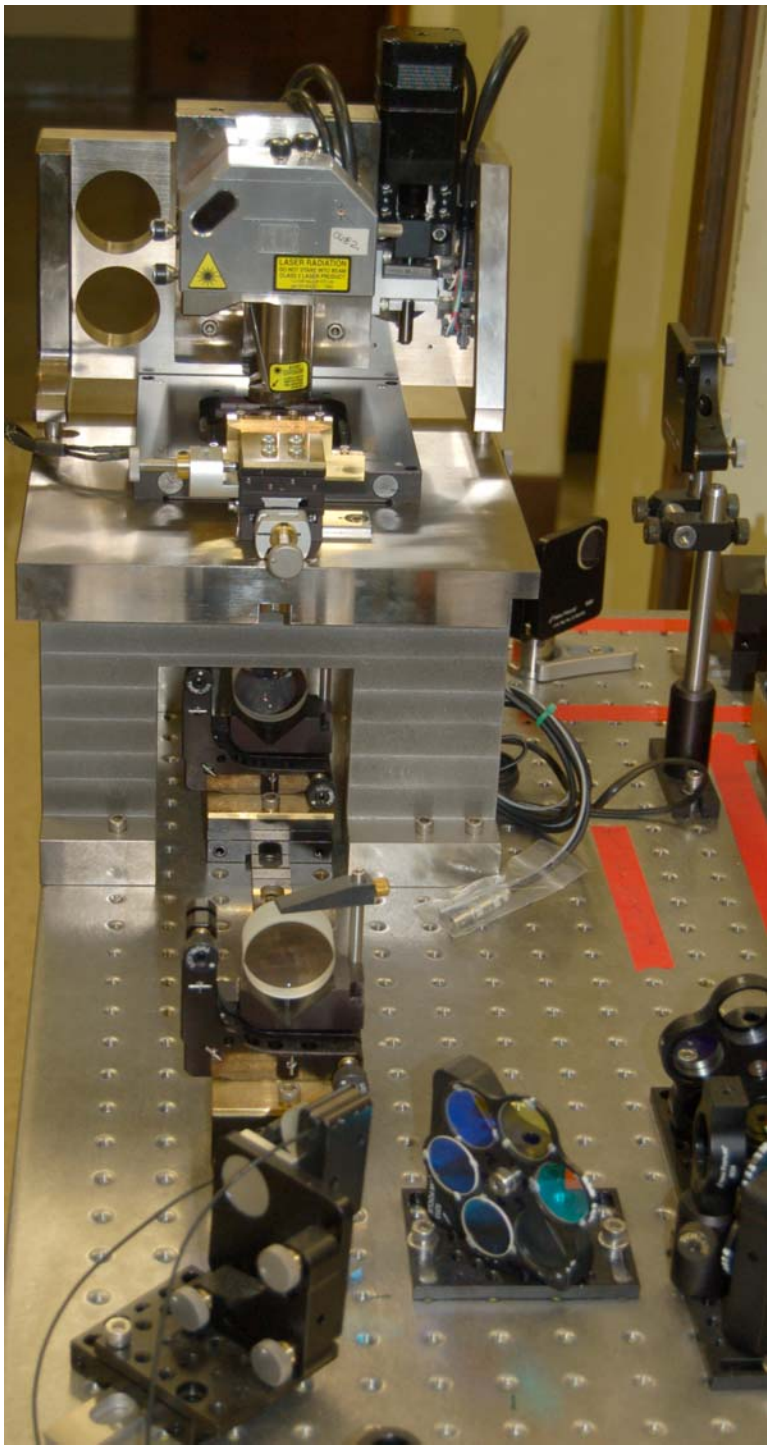



Figure E.5 shows the sample stage, AMF and TIRF microscope. Just to the right of the sample stage is the holder for the laser used to illuminate the AFM probe for alignment.

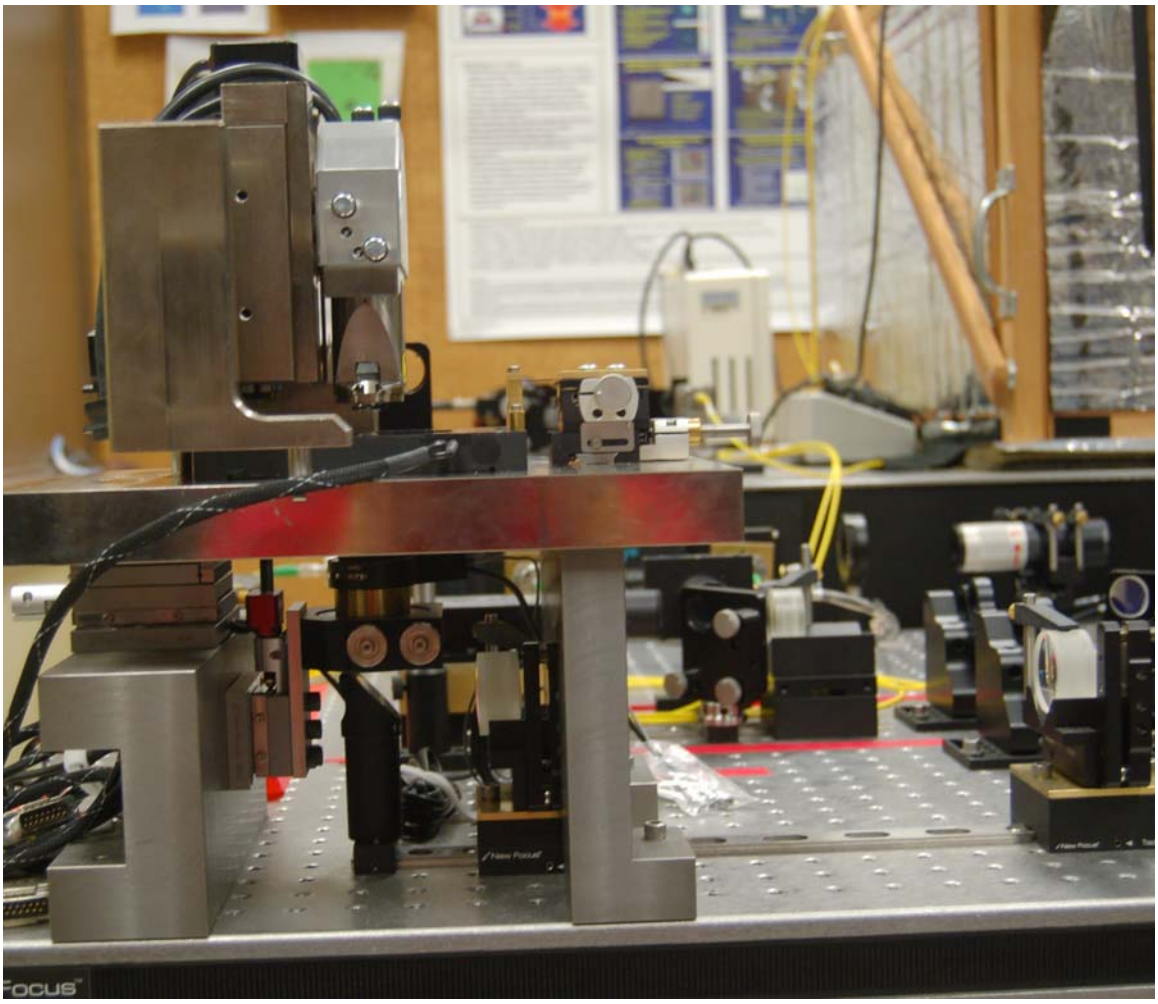


Figure E.6 Shows the AFM and TIRFM from the side.

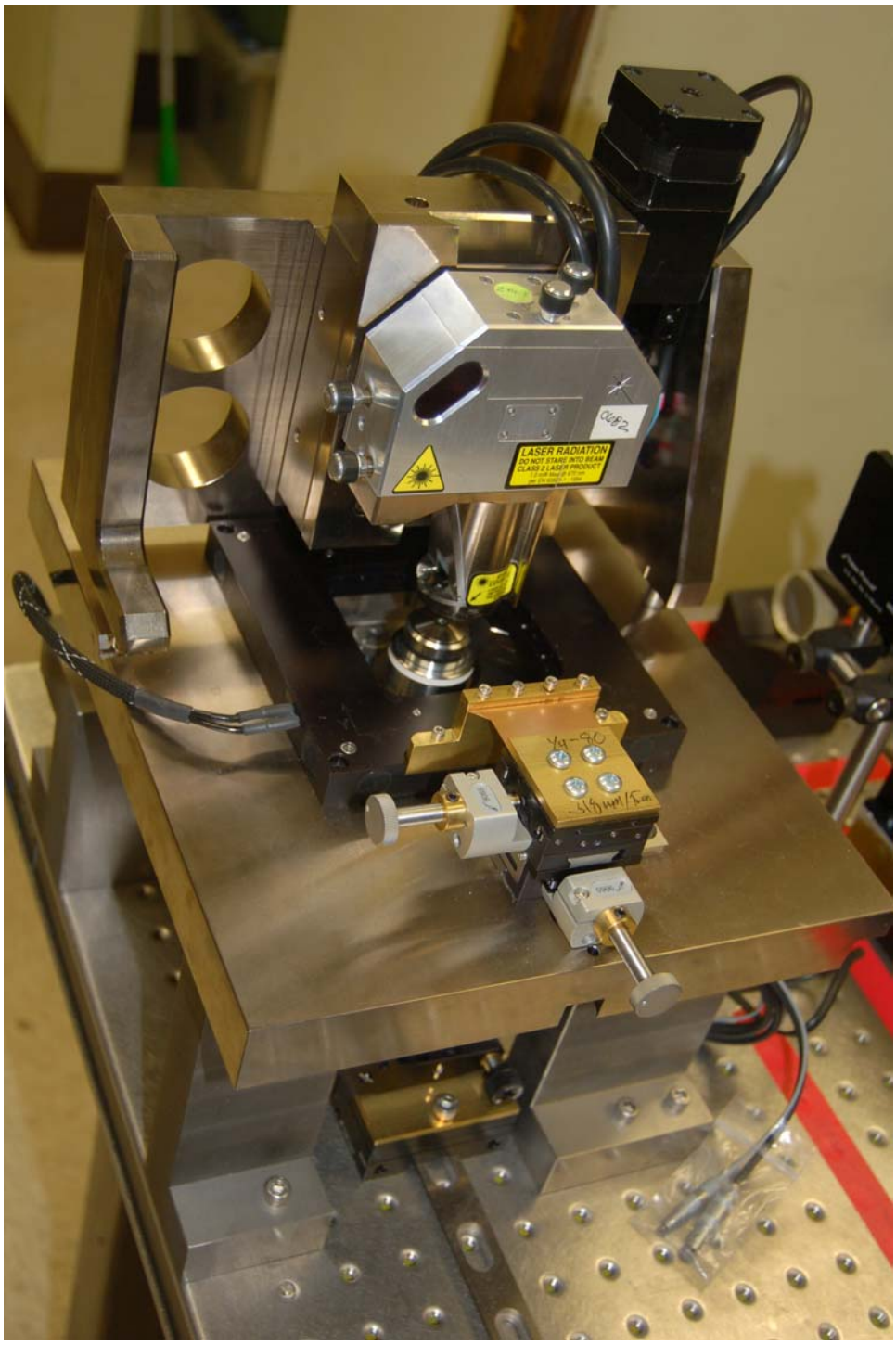



Figure E.7 shows the microscope objective, sample scanning stages (fine and coarse), and the AFM.

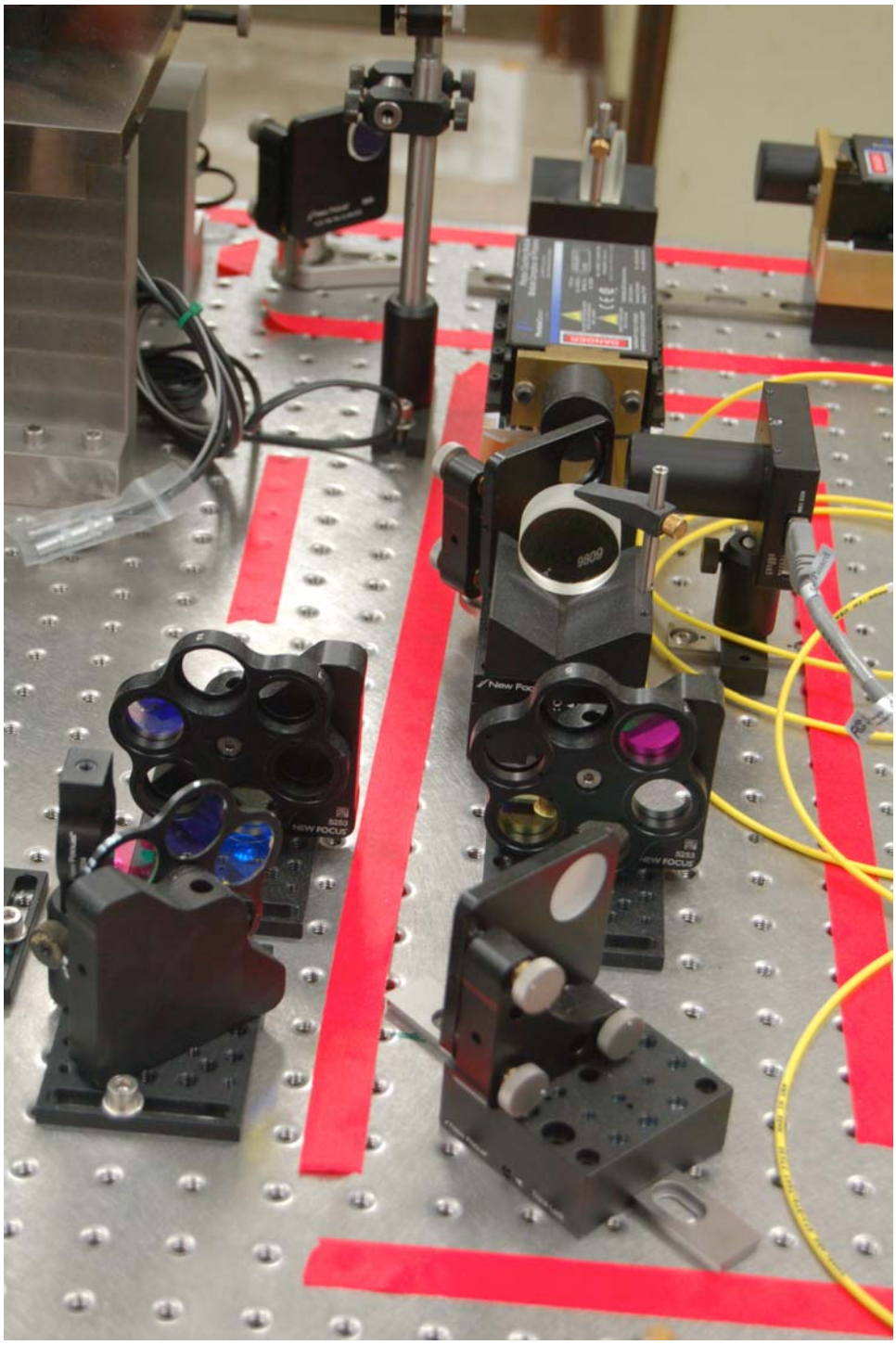


Figure E.8 Dichroic wheels. On the left is shown (starting from the bottom) the dichroic wheel that splits the emission signal between the APD #1 and APD #2. Just above the dichroic wheel is the emission filter wheel for APD #1. At the top of the picture are shown a mirror, a focusing lens, and APD #1. In the center/right you can see (from the bottom to the top) a mirror, an emission filterwheel, a focusing lens, then a clear coverslip used to split ~2% of the light to the right into a camera used for alignment. Just behind/above that 2% beamsplitter is APD #2.

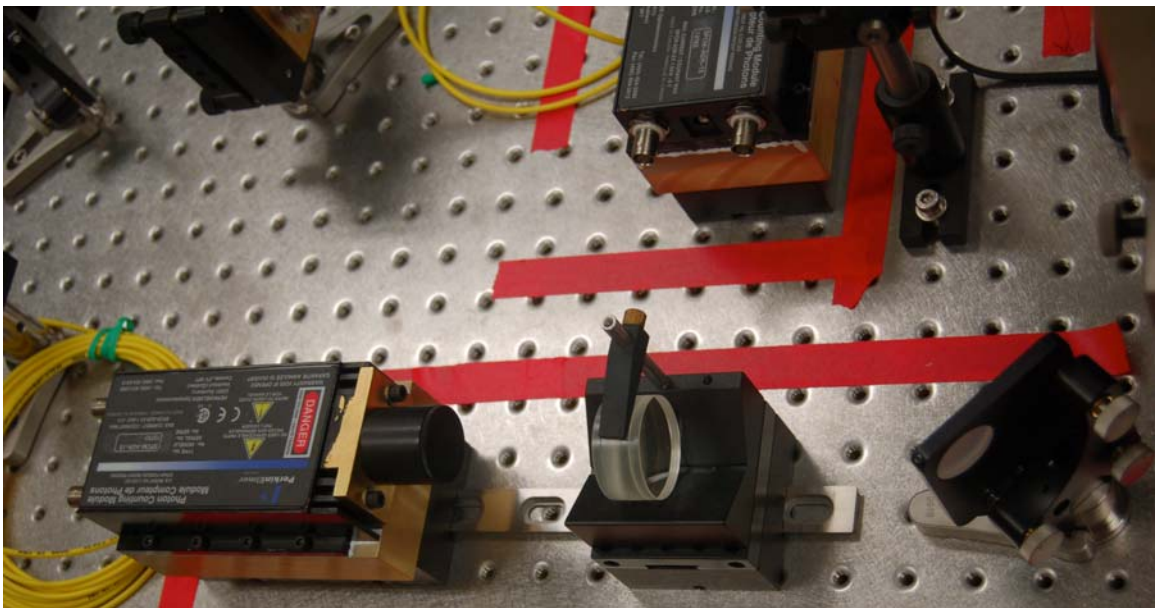



Figure E.9 APD #1 is shown in the bottom left. The focusing lens and a mirror (bottom right) can also be seen. I, the center/top can be seen the back half of APD #2.

**Microscope Operating Wavelengths**

This microscope was designed to simultaneously detect emissed light in two colors. Three excitation wavelengths are provided. The filters were all custom made by Chroma. The spectral bands were selected to minimize coupling of the targeted fluorophores to more than one of the APDs.

The filter set was designed to accommodate 6 different imaging configurations:

Configuration 1 is for imaging just CFP. The excitation wavelength is 442 nm. The emission band is 460-550 nm.

Configuration 2 is for imaging just YFP. The excitation wavelength is 502 nm. The emission spectral band is 520-600 nm.

Configuration 3 is designed for 2-color excitation and 2-color detection of CFP and YFP. Excitation is provided at 442 nm and 502 nm. The emission spectral bands are 460-485nm and 515-575 nm.

Configuration 4 is designed for 1-color excitation at 514 nm with an emission spectral band of 520-600 nm.

Configuration 5 is designed for 2-color excitation and 2-color detection of CFP and YFP. Excitation wavelengths are 442 nm and 514 nm. The emission spectral bands are 460-500 nm and 525-575.

Configuration 6 is designed for 1-color excitation and detection of mCherry. Excitation is at 543 nm and the emission spectral band is 575-625 nm.

Below are the specific filters (Chroma part numbers) used.

Four excitation filters are provided for the three lasers:

Z442/10X for 442 nm

Z502/10X for 502 nm

Z514/10X for 514 nm

Z543/10X for 543 nm

For excitation beam combination two filters are required. The 543 nm laser is injected directly into the light path. A ZT502RDC dichroic is used to bring 502 and 514 nm lasers in from the side. A ZT442RDC dichroic is used to bring 442 nm laser in from the side.

Dichroic Wheel 2 separates emission signal from excitation signal. This wheel is numbered by position. The filters in each position are:

Pos 1: ZT442RDC used for configuration 1

Pos 2: ZT502RDC used for configuration 2

Pos 3: ZT442/502RPC used for configuration 3

Pos 4: ZT442/514RPC used for configurations 4 and 5

Pos 6: ZT543RDC used for configuration 6

In addition, between the dichroic wheel 2 and the detectors is a 692 blocking filter (to stop the AFM laser): Z692NFRB.

Dichroic Wheel 3 separates longer wavelength emission (to detector 2) from shorter wavelength emission (to detector 1). This filter wheel is also numbered by position:

Pos 1: Z442RDC used for configuration 1

Pos 2: T505LP used for configurations 2 and 3

Pos 4: 515DCLP-XT used for configurations 4 and 5

Pos 6: Z543RDC used for configuration 6

An emission filter wheel is provided for imaging via APD #1. A star by the configuration number indicates the optimal filter choice for that configuration. The filters in the numbered positions are:

Filterwheel Pos 1: ET470/25M for configuration 3*

Filterwheel Pos 3: ET480/40M for configuration 5*

In front of APD #2 an emission filter wheel. A star by the configuration number indicates the optimal filter choice for that configuration. The filters contained in this wheel are:

Pos 1: ET510/80M for configuration 1* (470-550)

Pos 2: ET535/50M for configurations 3 (510-560)

Pos 3: ET540/30M for configurations 2, 3, 4, and 5 (525-555)

Pos 4: ET560/80M for configurations 2*, 3*, 5* (520-600)

Pos 5: ET 595/50M for configurations 1, 4, 6* (570-620)

In addition, just in front of each detector is a filter designed to block the tip illumination laser: ET670SP.

**Lasers**

The laser assemblies are shown in Figures E.10 - E.12. The three lasers are a 442 nm HeCd laser, an Argon ion laser (selectable for 502 and 514 nm) and a HeNe that operates at 543 nm. Each laser has an Acousto-Optic Modulator (AOM) and a neutral density double optical wheel. Two tip-tilt mirrors are used for coupling into the laser-coupler.

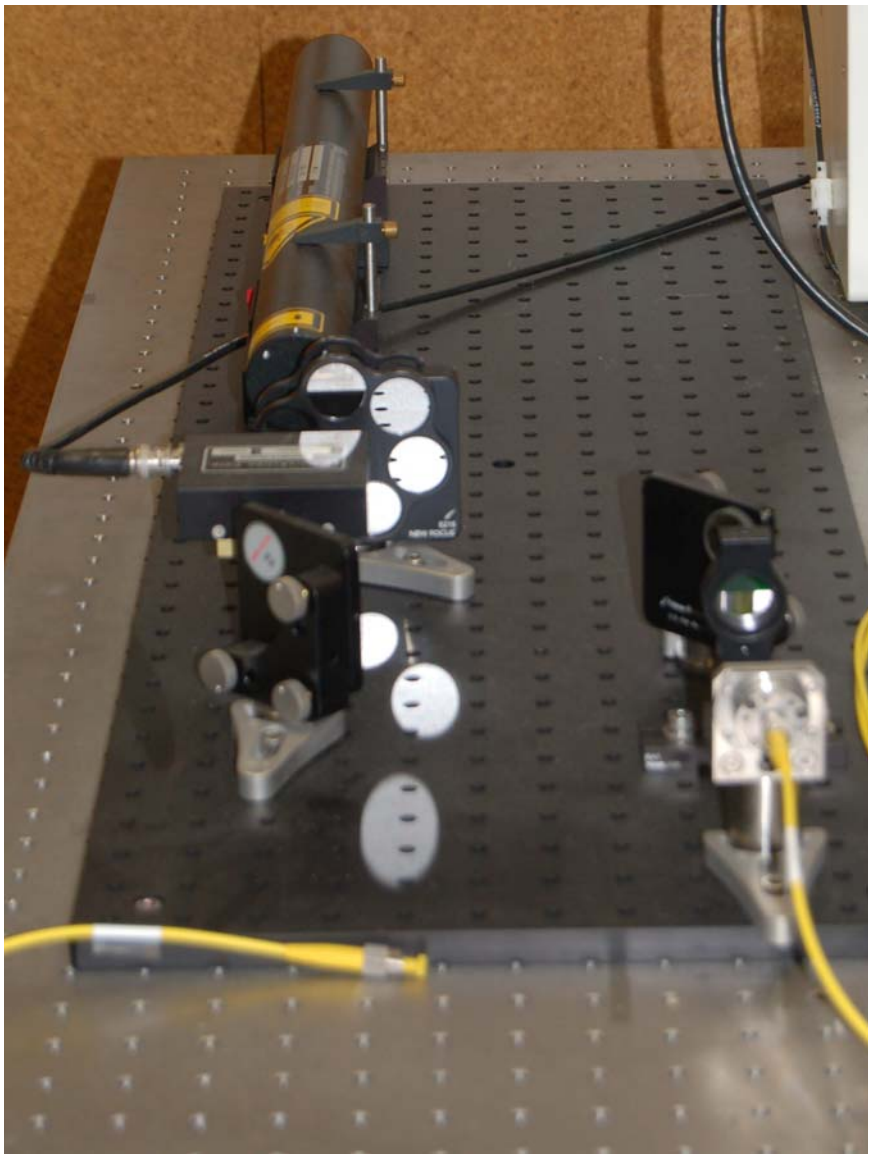


Figure E.10 This shows the 543 nm HeNe laser, AOM, neutral density filter wheels and fiber coupling optics.

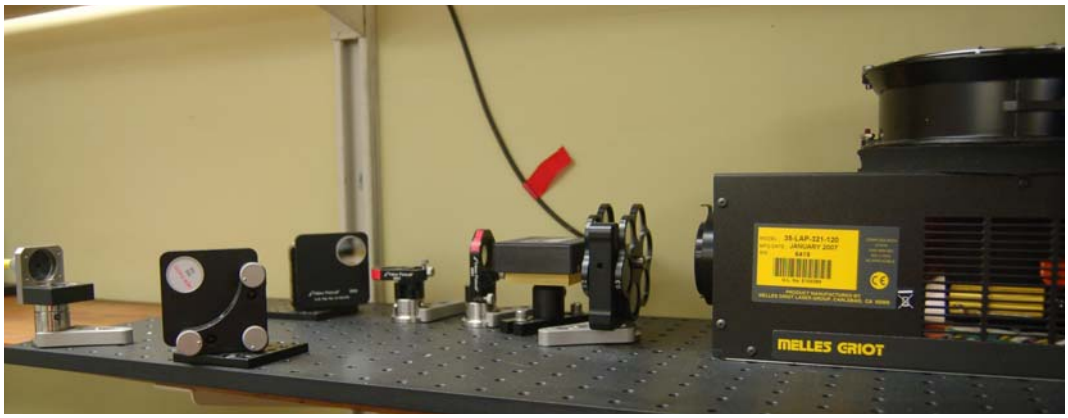



Figure E.11 The 502/514 nm adjustable Argon Ion laser, AOM, neutral density filter wheels and fiber coupling optics.

Figure E.12 The 442 nm HeCd laser, AOM, neutral density filter wheels and fiber coupling optics.

**Data Acquisition and Control Electronics and Software**

The DAC was designed by me and fabricated at JPL. Below is a report that provides a detailed description of the DAC design. The DAC hardware is shown in Figures E.13-E.15.

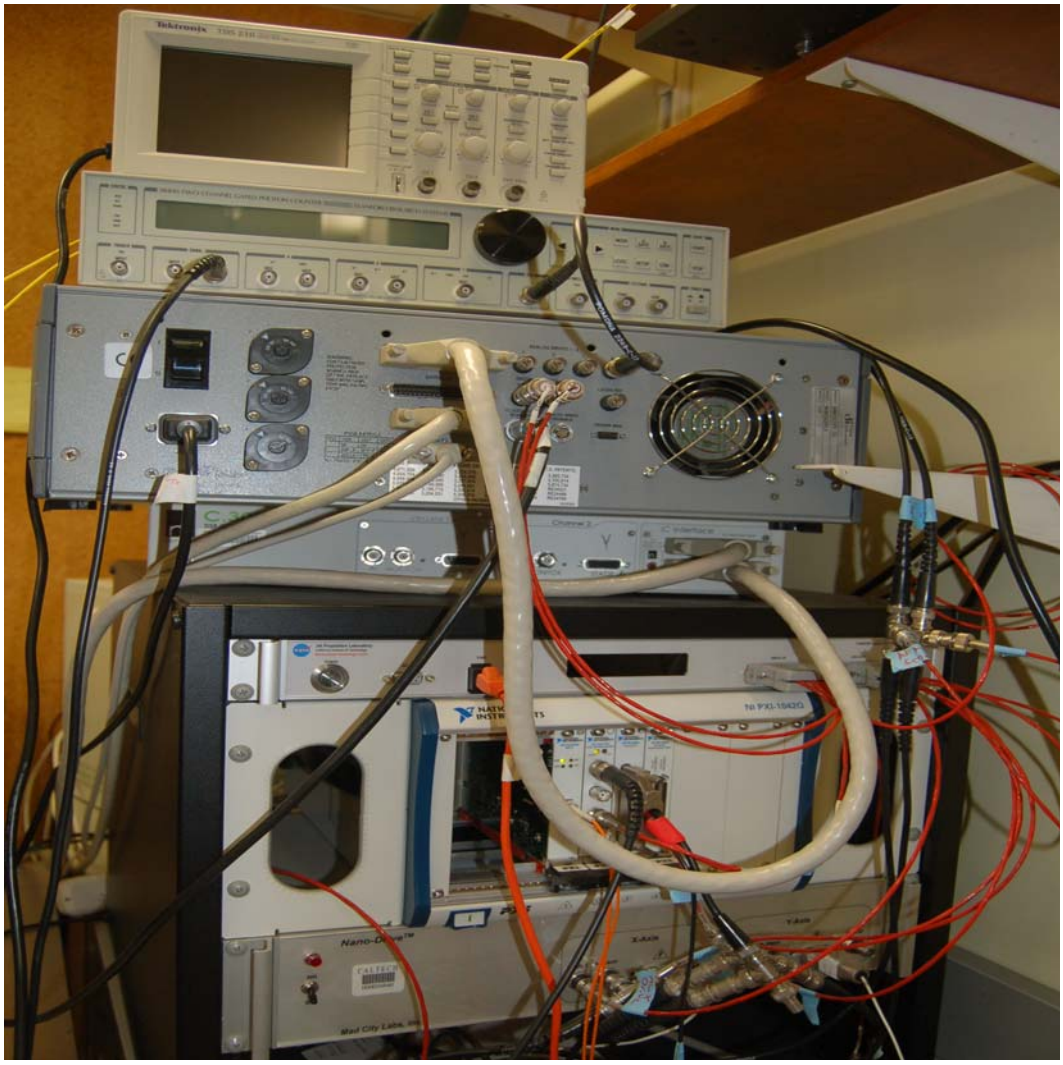


Figure E.13 Several of the DAC electronics boxes are shown above. At the top is an oscilloscope, below that is a SRS400 Gated Photon Counter, then the back of the Nanoscope IV AFM controller, then the nPoint C300 DSP controller (for the fine sample scanning stage), then Box H made by JPL, then the National instruments modules and at the bottom is the Mad City Labs controller for the tip-tilt mirror.

Figure E.14 Top right can be seen the HeCd power supply. In the center can be seen two AOM controllers. Just to the right of the AOM controllers is the side of the Nanoscope IV AFM controller with the gated photon counter on top.

Figure E.15 The tip-excitation laser alignment monitor (top left). Below it is the DAC monitor. The center bottom two monitors are for the AFM computer. The bottom right monitor is for the image-processing monitor.

The next section of this Appendix is a document authored by Brian Franklin and Erik Peterson of the Jet Propulsion Laboratory: 'FANSOM Measurement Technology Center (MTC) Delivery Package.' This document describes the architecture of the data acquisition and control system used in the wet TEFM imaging effort described here. The electronics (pictured in Figure 3.13) were designed and built by Brian Franklin. Erik Peterson wrote the software described in this document. The 'FANSOM Measurement Technology Center (MTC) Delivery Package' is included so that the details of the TEFM would be available to anyone trying to replicate or extend the work reported herein. My role in this work was to define all electrical interfaces, operational modes and the required electronics and software functionalities. I also helped with debugging the system. In addition I funded and directed this work through a JPL Research and Technology Development grant for which I was Principal Investigator.

Later, the data acquisition software described above was further debugged by Mike Gordon (then Caltech and now UCSB) and then significantly upgraded and modified by Jian Li (previously of JPL).

None of the image processing software written by Erik and described in the design delivery package was used for the work reported in this thesis. Instead I used software written by Dan Lo (then a Caltech undergraduate student), which was later replaced by software written by Eyal Shafran. Eyal's software, slightly modified by me, was used to process all of the results included in Chapter 3 and so is included at the end of this Appendix.

# FANSOM
# Measurement Technology Center (MTC)
# Delivery Package



Date: August 10, 2005

Authors: Brian Franklin
Erik Peterson

# TABLE OF CONTENTS

**TABLE OF CONTENTS** ..... **2**

**TABLE OF FIGURES** ..... **4**

**1 INITIAL QUOTE** ..... **5**

**2 SCHEDULE** ..... **10**

2.1 RECAP ..... 10
2.2 MAJOR MILESTONES ..... 10

**3 SYSTEM SETUP** ..... **11**

3.1 DIAGRAM ..... 11

**4 HARDWARE** ..... **12**

4.1 DETAILED DESCRIPTION ..... 12
*4.1.1 Motorola Coldfire Microprocessor* ..... *12*
*4.1.2 CPLD* ..... *12*
*4.1.3 LCD display* ..... *12*
*4.1.4 Analog-to-Digital Converters* ..... *12*
*4.1.5 Digital-to-Analog Converters* ..... *12*
*4.1.6 Power Supplies* ..... *12*
4.2 SCHEMATICS/DIAGRAMS ..... 13
*4.2.1 Chassis Wiring Schematic* ..... *13*
*4.2.2 Other Schematics* ..... *14*
*4.2.3 Firmware Activity Diagram* ..... *14*
*4.2.4 Signal Connections* ..... *15*
4.3 HARDWARE INTERFACE DESCRIPTION ..... 16
*4.3.1 Power Connectors* ..... *16*
*4.3.2 Analog Signal Connector* ..... *16*
*4.3.3 Digital Signal Connector* ..... *16*
*4.3.4 Ethernet Port* ..... *17*

**5 SOFTWARE** ..... **18**

5.1 HIGH LEVEL SOFTWARE ..... 18
5.2 DATA ACQUISITION ..... 19
*5.2.1 Optical Stationary* ..... *19*
*5.2.2 Optical Raster* ..... *20*
*5.2.3 Synchronous FANSOM* ..... *21*
*5.2.4 Stationary FANSOM* ..... *22*
*5.2.5 FANSOM Raster* ..... *23*
5.3 ANALYSIS ..... 24
*5.3.1 Evenly-binned photon counts* ..... *24*
*5.3.2 FANSOM-binned photon counts* ..... *24*
*5.3.3 AFM Amplitude Bins* ..... *25*
5.4 USER INTERFACE PANELS ..... 26

**6 OPERATING INSTRUCTIONS** ..... **29**

6.1 OPERATIONAL OUTLINE ..... 29
6.2 FOCUSING THE OBJECTIVE AND ENSURING OPERATION ..... 29
6.3 CALIBRATING ..... 32
6.4 CAPTURE DATA ..... 33
6.5 ANALYZING ARCHIVED DATA ..... 34
6.6 DISPLAYING DATA ..... 36
*6.6.1 Stationary Optical Data Display* ..... *36*

*6.6.2 Optical Raster Data Display ..... 37*
*6.6.3 FANSOM Data Display ..... 38*
**7 SYSTEM CALIBRATION REQUIREMENTS ..... 41**
**8 SYSTEM VALIDATION ..... 42**
8.1 VALIDATION DESCRIPTION ..... 42
**9 ISO COMPLIANCE STATEMENT ..... 43**
**10 COMPLIANCE MATRIX ..... 44**
**11 TOTAL COST REPORT ..... 45**
**12 EVALUATION OF SCHEDULE AND COST ..... 46**
**APPENDIX A – BOX H ETHERNET COMMANDS ..... 47**
**APPENDIX B – HARDWARE SCHEMATICS ..... 49**
**APPENDIX C – PREVIOUS SYSTEM ..... 51**

# TABLE OF FIGURES

FIGURE 1 - SYSTEM DIAGRAM....11
FIGURE 2 - CHASSIS WIRING SCHEMATIC....13
FIGURE 3 - BOX H FIRMWARE ACTIVITY DIAGRAM....14
FIGURE 4 - SCOPE CARD CONNECTIONS....15
FIGURE 5 - ANALOG AND DIGITAL SIGNAL CONNECTORS....15
FIGURE 6 - DEBUG SIGNAL CONNECTIONS....16
FIGURE 7 - HIGH LEVEL SOFTWARE ACTIVITY DIAGRAM....18
FIGURE 8 - STATIONARY OPTICAL DAQ ACTIVITY DIAGRAM....19
FIGURE 9 - OPTICAL RASTER DAQ ACTIVITY DIAGRAM....20
FIGURE 10 - SYNCHRONOUS FANSOM DAQ ACTIVITY DIAGRAM....21
FIGURE 11 – STATIONARY FANSOM DAQ ACTIVITY DIAGRAM....22
FIGURE 12 - FANSOM RASTER DAQ ACTIVITY DIAGRAM....23
FIGURE 13 - MAIN FANSOM USER INTERFACE PANEL....26
FIGURE 14 - SANITY CHECK FAILURE BOX....26
FIGURE 15 - FANSOM ABOUT BOX....26
FIGURE 16 - FANSOM WIZARD PANEL....27
FIGURE 17 - FANSOM DEBUG SCREEN....27
FIGURE 18 - FANSOM CALIBRATION WIZARD....28
FIGURE 19 - FANSOM CONFIGURATION....28
FIGURE 20 - MAIN PCB SCHEMATIC #1....49
FIGURE 21 - MAIN PCB SCHEMATIC #2....49
FIGURE 22 - MAIN PCB SCHEMATIC #3....50
FIGURE 23 - CPLD BOARD SCHEMATIC....50
FIGURE 24 - PREVIOUS SYSTEM BLOCK DIAGRAM....51

# 1 INITIAL QUOTE

**Task Name** FANSOM System Automation
**Customer** Larry Wade, sect. 3543
**Phone** 4.2272
**Cell** 818.642.1798
**SW Cog. E.** Erik Peterson & Brian Franklin

**Description**
The Fluorescence Apertureless Near-Field Scanning Optical Microscope (FANSOM) system is a novel microscopy technique achieving 10 nm resolution using an apertureless beam forming probe to excite the object being observed at a molecular level.

**Requirements**

1. Software – The MTC will evaluate all software components of the system and plan an integrated software approach using LabVIEW for the user interface and control software. Some existing code may be integrated into the final product using CIN or functional calls to C. The basic approach will be identified and presented to the customer by 1-2-2005. A data flow diagram and UML model of the current SW and the proposed SW will be presented. Drivers for the commercial hardware will be found and evaluated if possible. Once the detailed plan is presented to and accepted by the customer, development will begin in early January of 2005.
2. Hardware – The MTC will evaluate all commercial and custom hardware and will make a system block diagram of the current system. The Commercial hardware will be re-used. All custom hardware will be evaluated for functionality with a goal of integrating the hardware and providing automated control of the system. New computers and commercial hardware needed to automate the system will be identified and a plan for an integrated and automated system will be produced by 1-2-2005. A system block diagram, description of proposed custom hardware, and a list of needed commercial hardware will be generated as part of the planning process. Development of the integrated system can begin in early 2005 based on acceptance of the plan by the customer.
3. Procurements – The customer prefers to place orders for commercial hardware to allow for development in early January 2005. The team will identify the needed hardware as early as possible to allow for the orders to be processed through Caltech.

**Technical Approach**

1. Software
    a. Identify current components and functions
    b. Identify controls, measurements, and processing
    c. Produce Data Flow Diagram (or UML)
    d. Define or estimate data processing loads, examine current approach
2. Hardware
    a. Identify current components and functions
    b. Secure Tech data for commercial components

   c. ID functions of all custom components
   d. Produce system block diagram including interconnects
3. System Planning
   a. Define Integrated Software Requirements
   b. S/W development plan (module level)
   c. Define Integrated Hardware requirements (Box Level)
   d. Define Commercial HW list, and custom development plan
   e. Prepare orders for new commercial HW

**Acceptance Criteria**
The acceptance criterion is split into three parts. The first phase will be for learning the system and proposing the integrated/automated system. The first phase will be completed in early January 2005. The Second phase will be the development of the new software and hardware components. This is expected in late February, but detailed planning is needed to confirm this. The third phase will be the integration and test phase where the new components will be integrated into the system, and tested and refined as needed. This phase is expected to take approximately one work month and will require a high level of customer support.

**Deliverables**
1. Phase 1 – The Plan – requirements, diagrams, HW and SW plans.
2. Phase 2 – Working Components and Software.
3. Phase 3 – A fully integrated, tested, and documented system.

**Schedule**
Planning will last until January 2, 2005. Development is expected to be approximately 2 calendar months (TBC). Integration and testing is expected to be 2 weeks. Task completion is currently estimated to be mid March 2005.

**Cost**
Total cost is expected to be $80-100K with the following breakdown:

Procurements, Parts, Services: $30-40K
Labor (3-4 WM shared) $50-70K

These amounts are preliminary, and are to be verified and agreed upon at the time development begins.

If changes are necessary, the Customer or the Cognizant Engineer, whoever is initiating the change, will notify the other as soon as possible, and this document will be amended.

**Appendix A – Preliminary Commercial Equipment List**
- SR400 Stanford Research Photon Counter
- Nanoscope IV, Digital Instrument
- MAD CITY Labs Nanodrive

- Bioscope Atomic Force Microscope (cannot be integrated?)
- Uni-phase HV Source Model 1208-1
- Various NI DAC HW

**Appendix B – Proposed Software**

1. Requirements – The new software shall mimic all features available with the current software with the added feature of capturing AFM tip height waveforms alongside the digital data in Synchronous Mode:
    a. The new software shall be capable of recording photon timestamps in any of the three modes:
        i. Stationary, where photons are captured from a single location for a period of time.
        ii. Optical, where photons are captured over an area of the sample by rastering the laser over the sample and counting photons.
        iii. Synchronous, where photon captures are synchronized with the movement of the AFM head.
    b. Data shall be archived
    c. Analysis code shall be provided to create visual representations of the data files.
    d. Driver software necessary to control the Mad City Labs tip/tilt mirror, the AOMs, and any additional custom hardware shall be provided and integrated. Also of immediate importance is building the software with the intention to add additional channels of DAQ and control, as well as maintenance algorithms to improve the reliability of Synchronous Mode. These features will not be implemented in this phase, but their effect on the architecture shall be considered.
2. Approach – For the new version of the software, the C++ code will be replaced with LabVIEW code and the use of the DAQ hardware will be optimized. All DAQ will be performed in a PXI chassis. The existing PCI NI6602 counter card will be replaced with the PXI equivalent for counting photons, the NI6502 E-series DAQ card will need to be replaced by a PXI M-series card offering additional analog output lines (e.g. NI-6229M) and a scope card (NI-5122) will need to be added to capture the AFM height waveform with high resolution. The following is the planned usage of DAQ channels:

| NI 6602 | | | M-series MIO card | | | Scope Card | | |
|---|---|---|---|---|---|---|---|---|
| Function | Used | Free | Function | Used | Free | Function | Used | Free |
| Counters | 6 | 2 | Analog Inputs | 5 | 12 DI | Analog Input | 1 | 1 |
| | | | Analog Outputs | 3 | 1 | | | |
| | | | Digital I/O | 6 | 42 | | | |

The LabVIEW code will still run independent of the AFM control code, which will need to be set up and run prior to image capture in Synchronous Mode. An improved interface will be developed that will present a separate set of controls and indicators, depending on which of the three operating modes (i.e. Stationary, Optical Raster, or Synchronous) the FANSOM is imaging in. All data captures will be archived on an image by image basis and the analysis software will be integrated into the LabVIEW code, allowing for immediate display of captured

data, or for review of previous data sets, all within the same interface. Analysis code will be rewritten and optimized in LabVIEW. If equivalent or better performance cannot be achieved with LabVIEW, the analysis code will be returned to Matlab and called from LabVIEW.

3. Schedule – Three weeks will be required to write and debug LabVIEW DAQ software with the same functionality of the existing C++ software. The addition of the scope card to capture AFM tip height waveforms will require another two weeks of work and debug. Rewriting the existing analysis code in LabVIEW will take a week. Writing new analysis code in LabVIEW to account for the AFM tip height waveforms will take another week, assuming that the algorithms are specified beforehand. Additional time may be required to create analysis code capable of working in water if the algorithms are not already known. As a total, expect completion after seven weeks of 30hrs/week work. Additional features will be added as requested and are not covered by this plan.

**Appendix C – Proposed Hardware**

1. Requirements and Purpose – FANSOM project requires upgraded data acquisition hardware and software to enable proper imaging of samples underwater. In addition, the current system is undocumented – Documentation needs to be generated to enable future upgrades and transfer-of-knowledge to future researchers. Also, the current custom electronics are outdated and crudely fabricated. It is desired to replace them in order to improve performance, reliability and future upgradeability, rather than reverse-engineer and repair them.
2. Future Upgrade Requirements – It is desired to add one more laser and three more photon counters in the future. Also, the current system suffers from mechanical and thermal drift problems, and poor coupling between the laser and AFM tip. Future upgrades to address these issues include an automated polarization tune-up algorithm, an automated drift control algorithm, and a moving-stage/fixed-laser-and-optics approach to translating the sample.
3. Existing System Evaluation – Current commercial and custom hardware used by the FANSOM project have been evaluated and an existing system block diagram has been generated as shown in Figure 1. In the upgraded system, all items in blue will be reused and all items in red will be changed.
4. Development Plan – Figure 2 shows the proposed system diagram with incidental hardware removed and future add-ons marked in green. Items in blue are left unchanged and red items are new or rebuilt. Schedule for the following tasks is shown in Figure 3.
    a. Rebuild Box A. Upgrade analog control loop to digital using microcontroller and serial communications link to PC 1.
    b. Replace Box C (comparator) with NI-5122 Scope-Card (100 MS/s, 14-bit, 32MB, $6500). Capture entire z-position waveform from BioScope Electronics Box.
    c. Remove Boxes B, D, E, F, and G. Consolidate into a localized power supply solution.

5. Cost and Schedule – As shown in Figure 3, Total billable hours is estimated to be 23 days @ $120/hr. or $22,080.00 and date of completion is estimated to be March 14, 2005 for a total duration of 7 wks

# 2 SCHEDULE

## 2.1 RECAP

After the initial meeting regarding the project in October of 2004, two months were spent coming up with a design that would incorporate the various components from the old system into a new, unified whole. The first cut at the software was completed in early March of 2005, with Box H and its firmware completed later in the month. On the 24th of March the system was integrated at Caltech, where work continued for four more months on the addition of many new features as well as tweaks to original features. As of the end of July, 2005, the system reached a stable, functional state that would be considered complete for Fiscal Year 2005.

## 2.2 MAJOR MILESTONES

10.28.04 - First meeting

11.04.04 - First trip to Caltech

02.14.05 - Delay of NI order

02.16.05 - Software architecture complete

03.01.05 - First cut of LabVIEW code complete

03.08.05 - NI hardware delivered

03.24.05 - FANSOM system delivered to Caltech

04.06.05 - Budget meeting

06.01.05 - Budget/Progress meeting

06.20.05 - Bug/Feature Meeting

07.05.05 - Box H back at Caltech and fixed

07.06.05 - First run ‘wet’

07.08.05 - Fixed afm image corruptions.

07.19.05 - Got new monitors.

07.20.05 - First full wet run w/ analysis.

07.29.05 - Meeting to discuss final FY05 tasks and documentation

# 3 SYSTEM SETUP

## 3.1 DIAGRAM

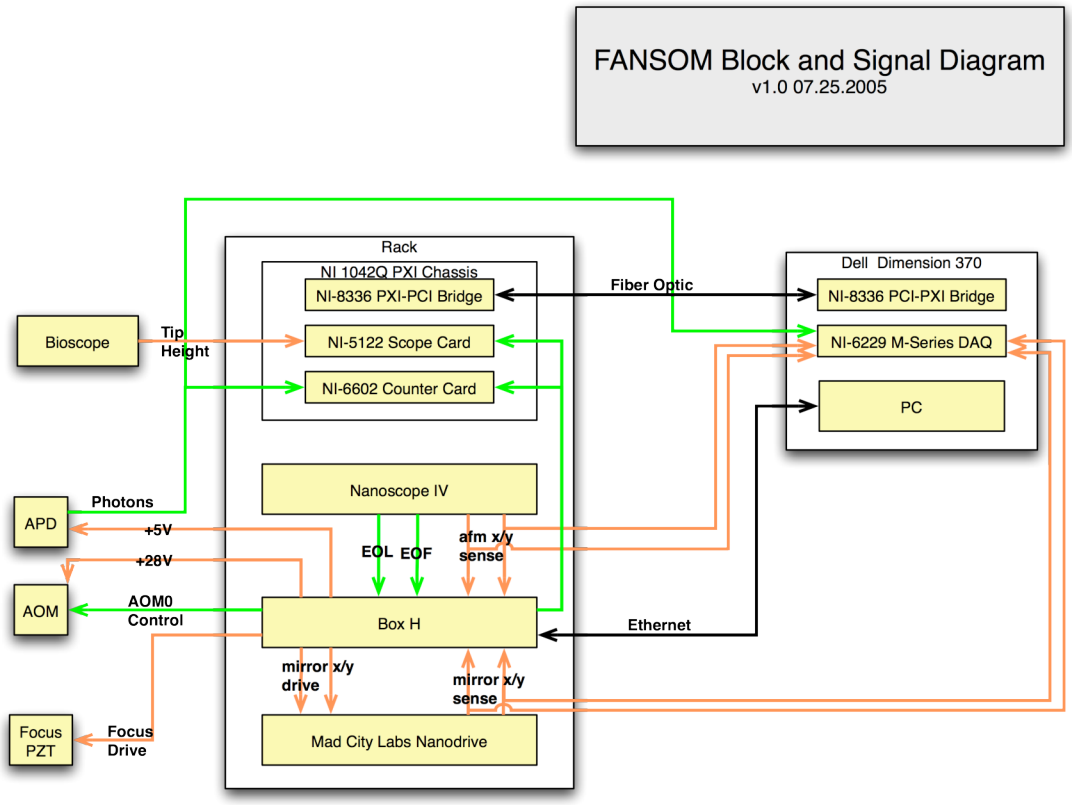



**Figure 1 - System Diagram**

# 4 HARDWARE

## 4.1 DETAILED DESCRIPTION

The custom hardware in the system takes the form of Box H, a 1U rackmount chassis incorporating and improving upon all functionality of the custom hardware in the previous system. It contains a Motorola Coldfire microprocessor, a Xilinx CPLD, a two-line blue LED-backlit LCD display, four ADCs, three DACs, and 6 power supplies.

### 4.1.1 Motorola Coldfire Microprocessor

The Motorola Coldfire Microprocessor, included in a Netburner package, provides:

- computer-based control of all aspects of Box H's operation through the Ethernet
- a digital control loop used to synchronize the movement of the tip-tilt mirror with the afm
- fast-scan capabilities of the mirror x-axis at user-configurable rate
- programming for the LCD display

### 4.1.2 CPLD

All digital output and input to and from the Coldfire is passed through an CPLD. This provides several features currently, and will expedite the addition of other features at a later time. The most important feature of the CPLD is to combine the EndOfLine and EndOfFrame signals from the Nanoscope into a LINE signal capable of triggering all of the DAQ hardware in sync with the AFM's movement.

### 4.1.3 LCD display

The LED display provides information during debug operation of Box H, as well as indication that the Box is running in the form of the word 'FANSOM' spelled across it.

### 4.1.4 Analog-to-Digital Converters

The four ADCs digitize the analog representation of the mirror and AFM x- and y-axes for use in the digital control loop and reporting back to the host computer.

### 4.1.5 Digital-to-Analog Converters

Two of the DACs drive the position of the tip-tilt mirror, either in response to direct host computer command or as the result of the digital control loop. The output from a third DAC is amplified to provided the drive voltage for the fine focusing PZT in the microscope's optical system.

### 4.1.6 Power Supplies

The power supplied provide power internally for Box H's operation and externally to power the AOM and photon detector. The photon detector's power can be enabled or disabled by command from the host computer while the AOM's power is output as long as Box H is powered on.

## 4.2 SCHEMATICS/DIAGRAMS

### 4.2.1 Chassis Wiring Schematic

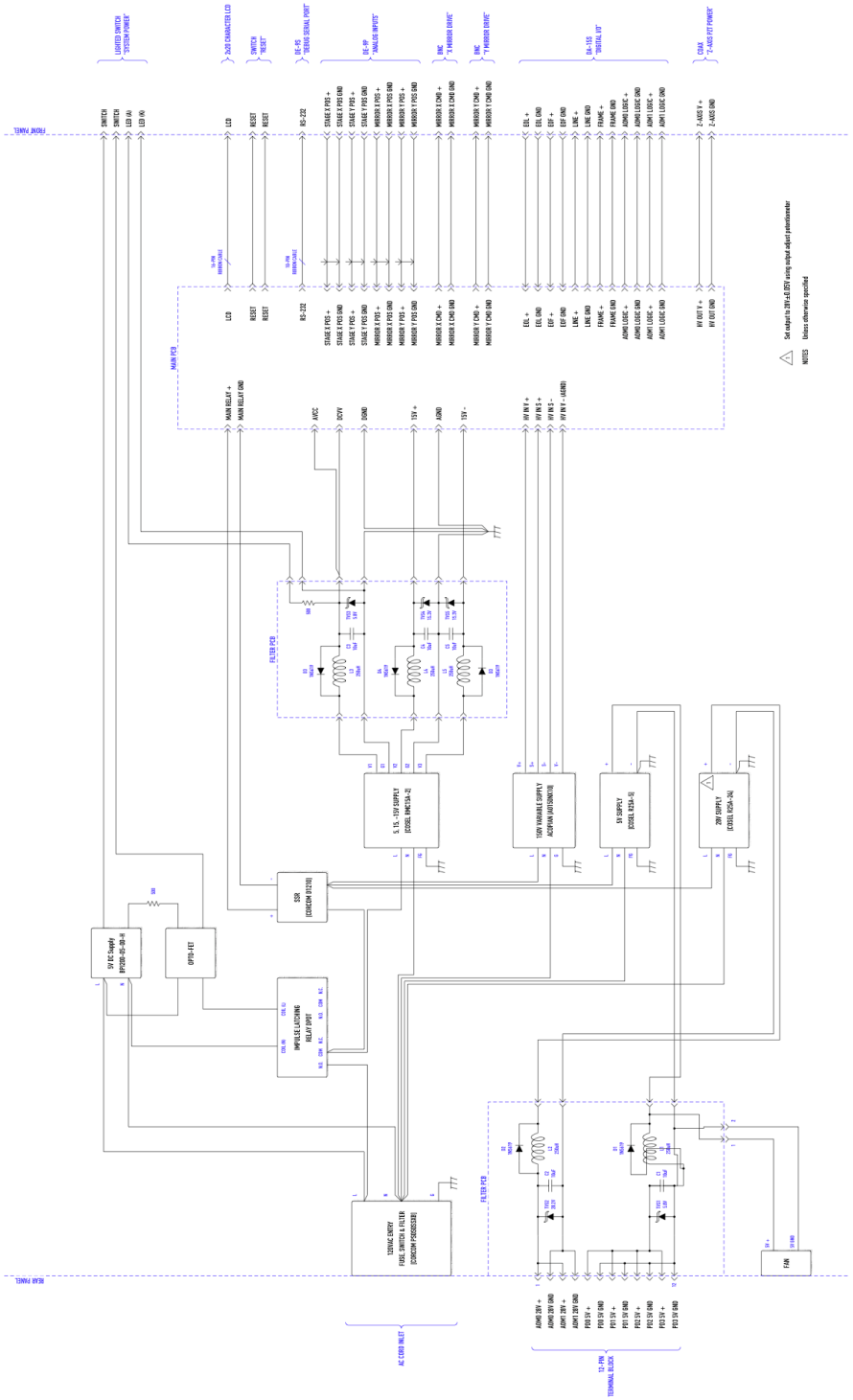


**Figure 2 - Chassis Wiring Schematic**

### 4.2.2 Other Schematics

Other hardware schematics can be found in Appendix B – Hardware Schematics.

### 4.2.3 Firmware Activity Diagram

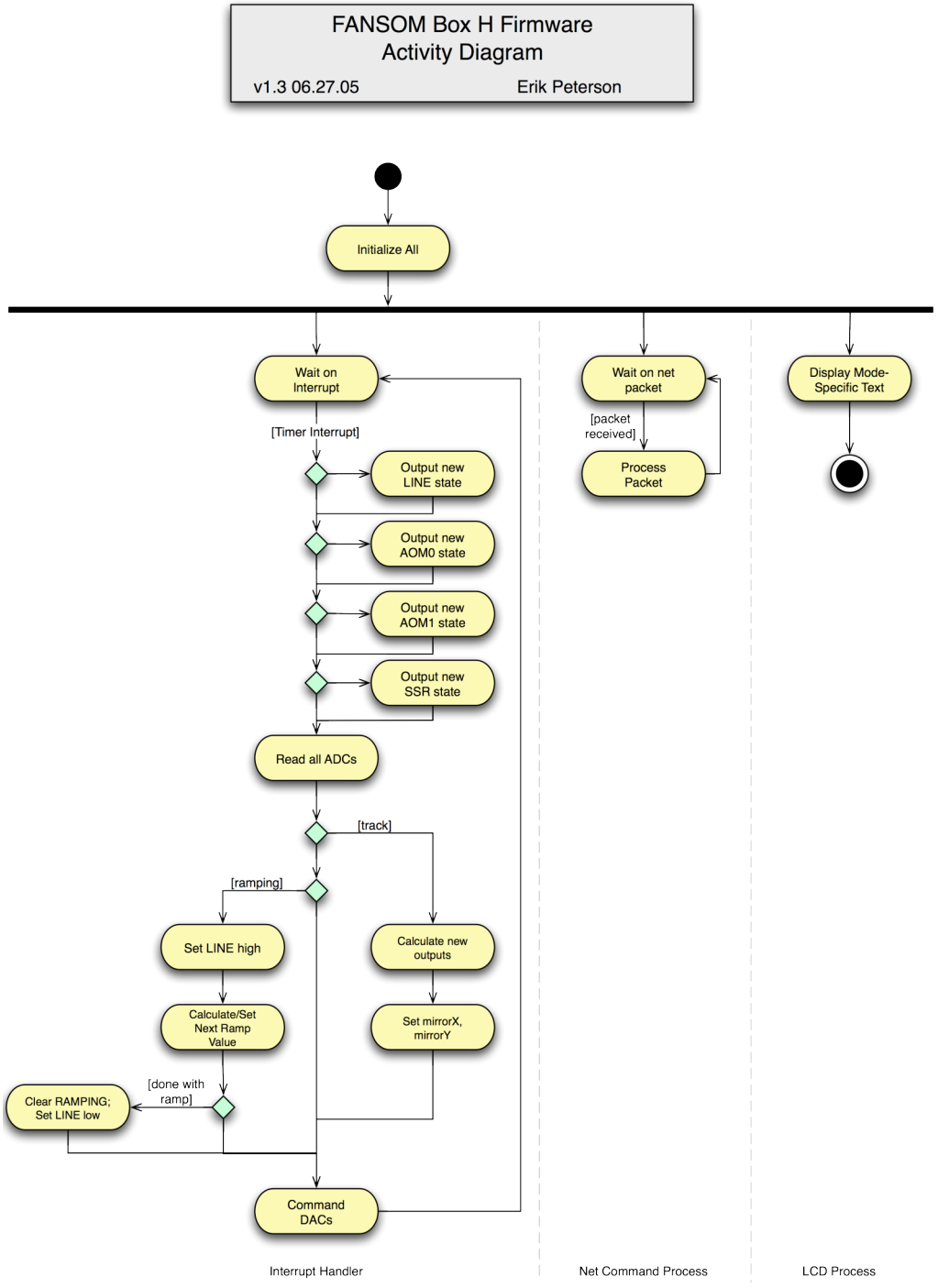



**Figure 3 - Box H Firmware Activity Diagram**

### 4.2.4 Signal Connections

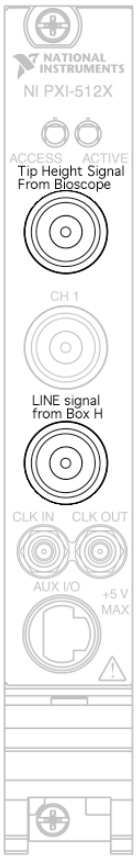
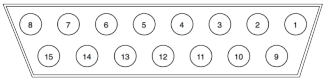



**Figure 4 - Scope Card Connections**

Analog Signal Connector



| | | | |
|---|---|---|---|
| 1 | AFM X Sense | 9 | GND |
| 2 | AFM Y Sense | 10 | GND |
| 3 | Mirror X Sense | 11 | GND |
| 4 | Mirror Y Sense | 12 | GND |
| 5 | | 13 | GND |
| 6 | Mirror X Drive | 14 | GND |
| 7 | Mirror Y Drive | 15 | GND |
| 8 | | | |

Digital Signal Connector

| | | | |
|---|---|---|---|
| 1 | | 14 | GND |
| 2 | | 15 | GND |
| 3 | EOL | 16 | GND |
| 4 | EOF | 17 | GND |
| 5 | | 18 | GND |
| 6 | | 19 | GND |
| 7 | | 20 | GND |
| 8 | | 21 | GND |
| 9 | | 22 | GND |
| 10 | LINE | 23 | GND |
| 11 | FRAME | 24 | GND |
| 12 | AOM0 | 25 | GND |
| 13 | | | |

**Figure 5 - Analog and Digital Signal Connectors**

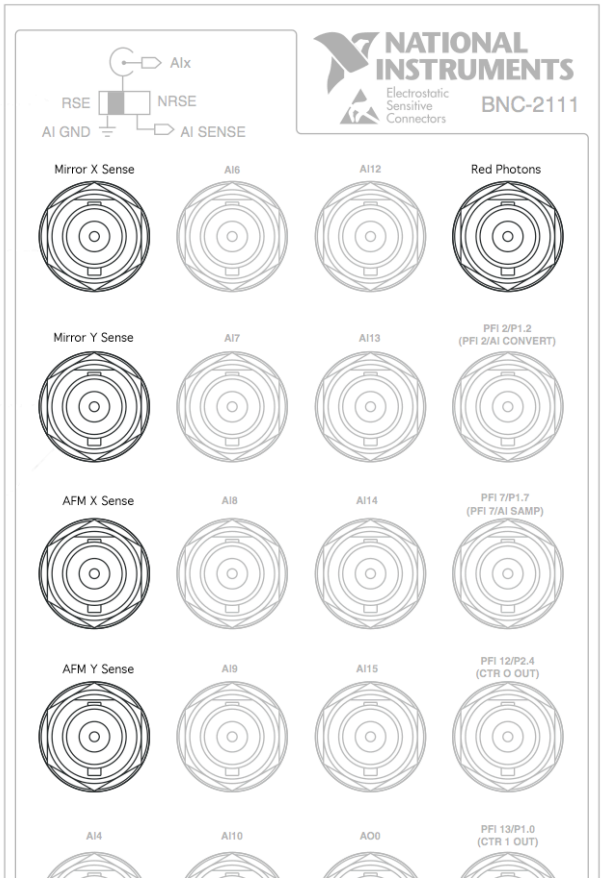



**Figure 6 - Debug Signal Connections**

## 4.3 HARDWARE INTERFACE DESCRIPTION

The external interfaces for Box H are: power connectors, an analog signal connector, a digital signal connector, and the Ethernet port.

### 4.3.1 Power Connectors

On the back of Box H are several connectors for externally-powered devices. On a terminal block there are four connectors, which provide +5V, and two connectors, which provide +28V. Currently, one of the +5V connectors powers the photon detector and one of the +28V connectors powers the AOM. There is also a BNC, which connects to the focus PZT and provides 0-150V.

### 4.3.2 Analog Signal Connector

On the front of Box H is a 15-pin DSUB connector, which provides input and output of analog signals for Box H.

### 4.3.3 Digital Signal Connector

On the front of Box H is a 25-pin DSUB connector, which provides input and output of digital signals for Box H. All of these signals pass through the CPLD.

### 4.3.4 Ethernet Port

On the front of Box H is an Ethernet connection, which provides the command and data gathering capabilities of Box H. A host computer can connect to TCP port 10001 at IP address 192.168.1.5 and send ASCII commands, all terminated with the '\n' character (ASCII 13). A complete listing of the commands available over Ethernet can be found in Appendix A – Box H Ethernet commands

# 5 SOFTWARE

## 5.1 HIGH LEVEL SOFTWARE

The main software of the system is written in LabVIEW and offers a user interface for data acquisition, data analysis, and general configuration of the FANSOM system. Separate screens handle configuration and calibration, and a custom wizard handles the preparation of data acquisition or analysis. Once the wizard is completed with a valid set of configuration parameters, several parallel processes are initiated to handle the tasks: the DAQ process, the Archive process, and the Analysis & Display process.

In the case of capturing a new dataset, the DAQ process takes the results of the FANSOM Wizard and preps and executes the necessary data acquisition hardware operations. If archived data is being used the DAQ process is not initiated.

In the case of capturing a new dataset, the Archive process waits for each line of data, in the case of Optical Raster or the FANSOM modes, or the whole duration, in the case of Stationary Optical, to become available. The Archive process next writes the data to a file in the appropriate place in the filesystem. In the case of both new and archived data operations, the Archive process waits for data requests from the Analysis & Display process, which it fills as the data become available in the filesystem.

In the case of capturing a new dataset, the Analysis & Display process requests data lines, performs the necessary analysis on the data, and then displays the data to the user in one of several graph formats.

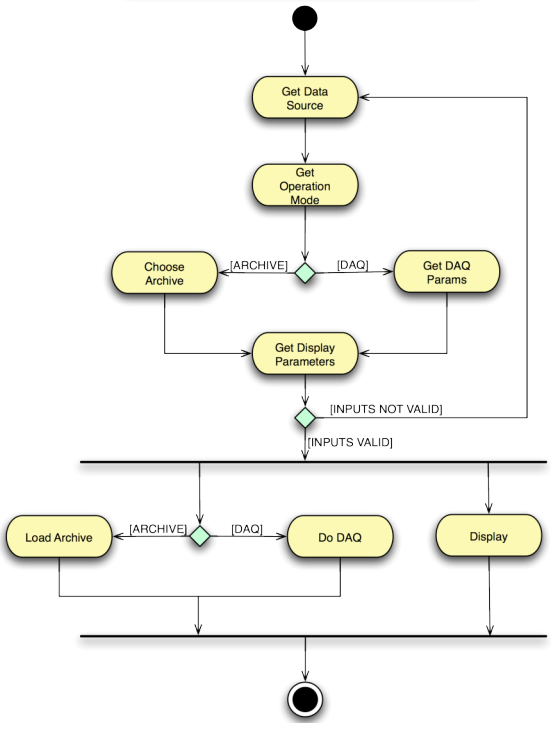



**Figure 7 - High Level Software Activity Diagram**

## 5.2 DATA ACQUISITION

Each software mode performs its data acquisition in a slightly different way. There are a few common aspects, which involve how the actual data are taken. In all modes, photons are counted by NI general-purpose counter hardware in Buffered Event Counting mode. In this mode, the counter is constantly being incremented by an internal 20MHz clock. At each photon trigger, the contents of the counter are dumped into a buffer. Thus, at the end of acquisition, the result is a series of photon timestamps with 1/20,000,000 time resolution.

In the FANSOM modes, an NI scope card captures the tip height waveforms. The scope card's buffer is set up based the total amount of time to capture and a 1Megasample/second capture rate.

In all modes but optical stationary, all data acquisition is triggered by LINE triggers coming from Box H.

### 5.2.1 Optical Stationary

In Optical Stationary Mode, the stage is moved into position, the data acquisition hardware is configured to capture photon timestamps on a software trigger, the AOM is enabled, the software trigger is fired, and the software waits for the specified time. Every 20 ms the software empties the hardware capture buffer into a local array until the acquisition completes, when the data are sent to the Archiver and the stage is returned to (0, 0).

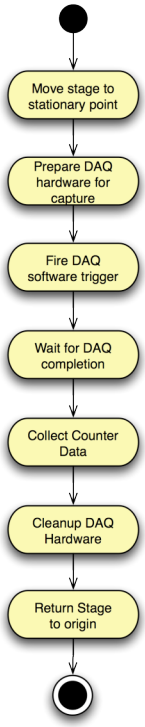



**Figure 8 - Stationary Optical DAQ Activity Diagram**

### 5.2.2 Optical Raster

In Optical Raster Mode, first the stage is moved to the beginning of the first line in the image. Then, for each line, the data acquisition hardware is configured to capture photons after a trigger from Box H, the AOM is enabled, and Box H is commanded to perform a ramp along the x-axis at the given rate. When Box H executes the ramp it also sets LINE high, which triggers the data acquisition hardware. Meanwhile, the software polls LINE waiting for it to go from high to low. When this occurs the software empties the data acquisition buffer and sends the data to the Archiver.

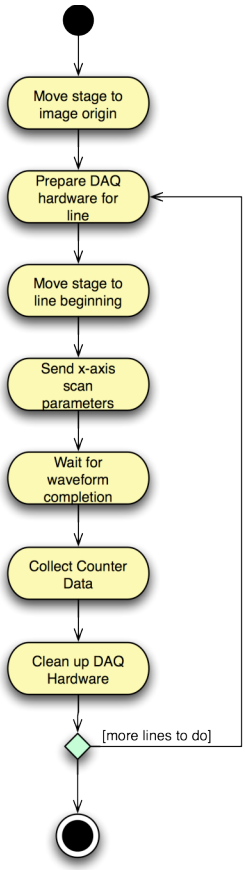



**Figure 9 - Optical Raster DAQ Activity Diagram**

### 5.2.3 Synchronous FANSOM

In Synchronous FANSOM Mode, first tip tracking and LINE toggling are enabled in Box H. This forces the mirror to follow the AFM tip (if calibrated correctly) and forces LINE to toggle based on the EOL trigger from the Nanoscope, starting at the top of the image. The AOM, too, toggles in sync with the movement of the AFM tip. Then, for each line the data acquisition hardware is configured to capture both photons and tip heights on LINE triggers. The software waits to see LINE toggle from low to high and back to low again, signaling a capture line has completed, and then empties the hardware buffers, sends the line data to the Archiver, and prepares for the next line. At the end of the image the tip tracking and LINE toggling are both disabled.

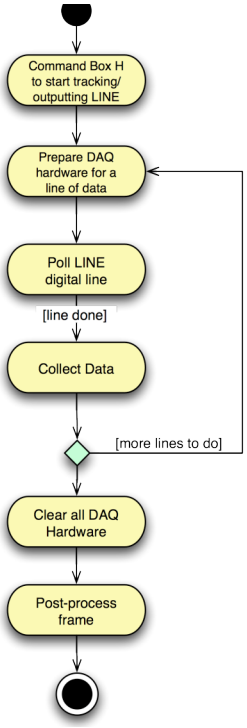



**Figure 10 - Synchronous FANSOM DAQ Activity Diagram**

### 5.2.4 Stationary FANSOM

The data acquisition in Stationary FANSOM Mode is the same as it is for Synchronous FANSOM mode, but there is some setup that differs. First, since the mirror remains motionless during this mode, Tip Tracking is not enabled during data acquistion. Instead, the user is prompted to place the AFM in the position where the laser should remain. The Tip Tracking is enabled just long enough for the mirror to orient itself properly. Then Tip Tracking is turned back off, the user is prompted to start the AFM moving again, and data acquisition occurs, line-by-line, the same as in Synchronous FANSOM Mode.

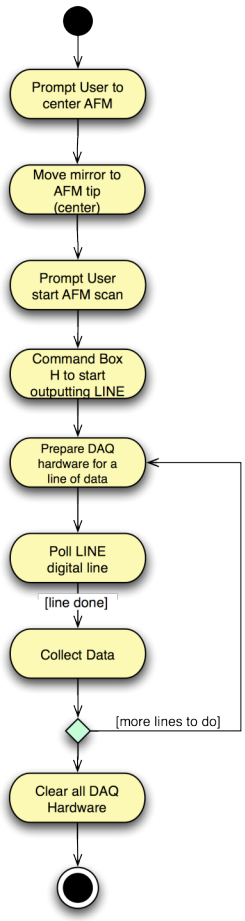



**Figure 11 – Stationary FANSOM DAQ Activity Diagram**

### 5.2.5 FANSOM Raster

The data acquisition in FANSOM Raster Mode is very similar to Optical Raster mode, except for two notable differences. First, tip heights are collected as well as photons during each line. Second, the location of the center of the scan is derived from the user-placed position of the AFM tip, which remains stationary during the data acquisition.

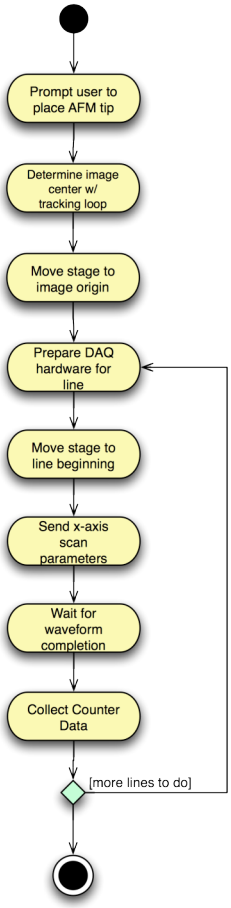



**Figure 12 - FANSOM Raster DAQ Activity Diagram**

## 5.3 ANALYSIS

There are essentially three data products in the FANSOM software. They are: evenly-binned photon counts, FANSOM-binned photon counts, and AFM amplitude bins.

### 5.3.1 Evenly-binned photon counts

This data product is generated for the Stationary Optical Mode graph, as well as for every line in an Optical Raster Mode graph or a Total Power graph for a FANSOM mode. It is generated by taking the total length of time of the group of photons, splitting that time period into as many bins as are requested, and then counting the number of photons (i.e. timestamps) in the group in each bin.

In the case of Stationary Optical Mode, this is displayed as a waveform graph with the x-axis being time and the y-axis being number of photons in the time bin.

In the case of Optical Raster Mode and the Total Power graph for a FANSOM mode, this is displayed as a line in an intensity graph with the x- and y-axes being physical distance and the intensity being number of photons.

These data can be further analyzed in the Optical Raster graph with the Profile feature, whereby a cross-section of an intensity graph is generated between two cursors on the graph by taking the total length of the profile, splitting it up based on the x- and y-axis scales, and then moving along the profile, from one cursor to the other, reading the intensity values and displaying them in a waveform graph.

### 5.3.2 FANSOM-binned photon counts

The most complicated of the data products is the FANSOM-binned photon count. The main portion of the algorithm starts with a line of photons and tip heights. It then splits both the tip heights and the photons into bins representing pixels in the final intensity plot. Each bin is then processed individually.

For the bin, first the tapping frequency is extracted with a call to the LabVIEW library VI Extract Single Tone Information.vi. With the tapping frequency, then five periods of the waveform are averaged together and the phase offset is determined, i.e. the number of samples from the beginning of the bin's tip height waveform until the first 'zero crossing.' Knowing the first zero crossing and the frequency makes it possible to extrapolate the zero crossings for the entire bin. Then the midpoint between every two zero crossing timestamps is calculated, with this timestamp going into the list of either peaks or troughs. Before the next step, if this is the first bin on the first line of an image, a dialog appears that shows the user a couple of periods of the tip height waveform and the calculated peak and trough within this range. The user can then set three parameters: the window around the peak that should be considered for counting photons, the window around the trough that should be considered when counting photons, and an offset value that will shift all of the peaks and troughs forward or backward in time by some amount. Then the actual counting takes place. The code iterates through each peak and counts the photons with timestamps within the window around each peak. It then does the same for

the troughs in the bin. The last calculation is, if the peak and trough windows are not equal, then the total peak photon count is scaled by the ratio between the two windows.

The bins of peaks and the bins of troughs are generated for the whole image. The final answer generated by subtracting the peaks, i.e. the background photons, from the troughs, or the interaction photons. To try to beat down the background noise one final bit of analysis can be undertaken. A dialog will be displayed for the user to choose which bins to average together for every bin in the image. The pattern is chosen and the software iterates through each pixel in the background, adds together all of the background photons in the pattern, and then divides by number of bins used (to account for coming up against the side of the image and having less than the optimal number of bins).

Then the background signal is subtracted from the foreground signal and the result is displayed as an intensity graph. It can also be sliced into profiles or shown as a 3D representation.

### 5.3.3 AFM Amplitude Bins

In all FANSOM modes that involve a rastering AFM tip, AFM Amplitude Bins can be generated by taking the full length of a line of tip height data, splitting it up into the proper number of bins, and then taking the amplitude of the waveform within each bin and graphing it as a pixel on an intensity plot with the intensity corresponding to the voltage representation of the tip height. The average amplitude is generated by calling a LabVIEW routine Amplitude and Levels.vi, which returns the amplitude of a waveform.

## 5.4 USER INTERFACE PANELS

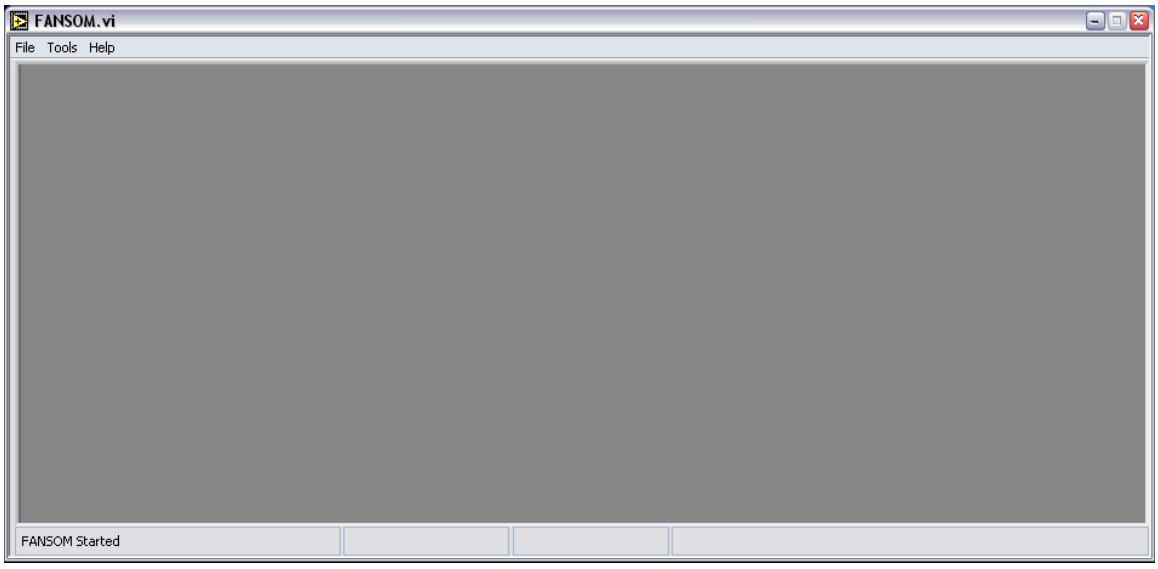



**Figure 13 - Main FANSOM User Interface Panel**

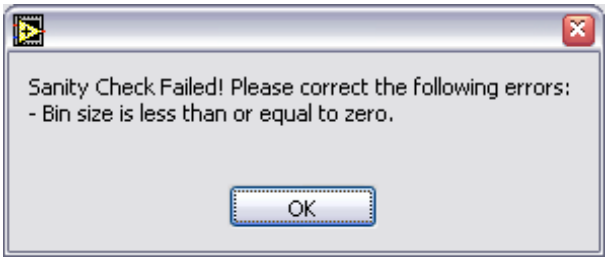



**Figure 14 - Sanity Check Failure Box**

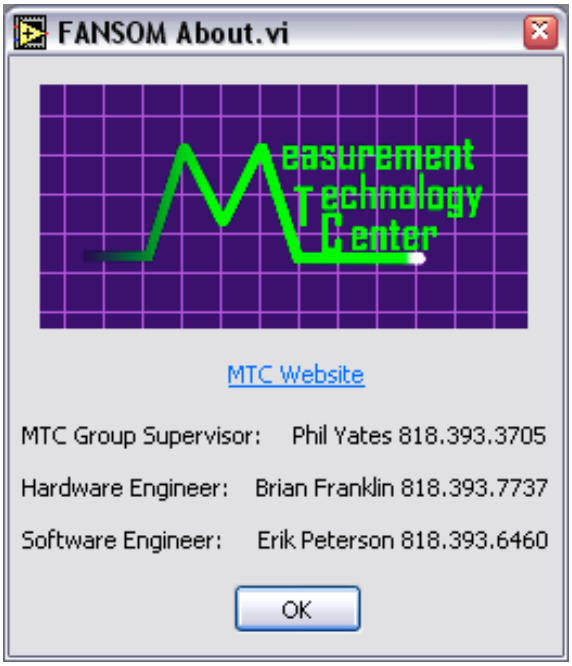



**Figure 15 - FANSOM About box**


**Figure 16 - FANSOM Wizard Panel**

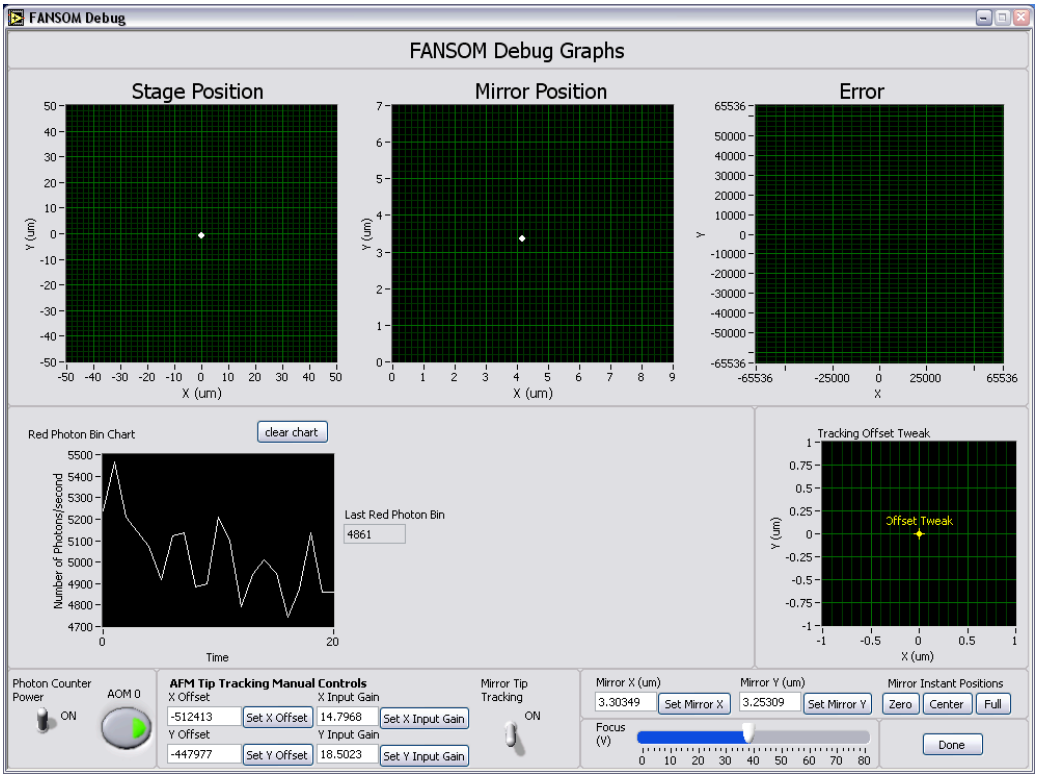



**Figure 17 - FANSOM Debug Screen**


**Figure 18 - FANSOM Calibration Wizard**

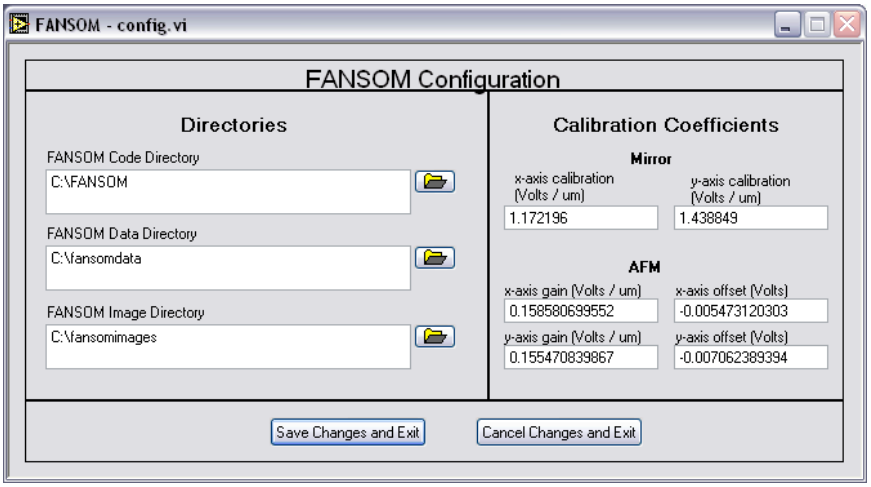



**Figure 19 - FANSOM Configuration**

(Note: Accessible from the FANSOM Start Menu folder)

# 6 OPERATING INSTRUCTIONS

## 6.1 OPERATIONAL OUTLINE

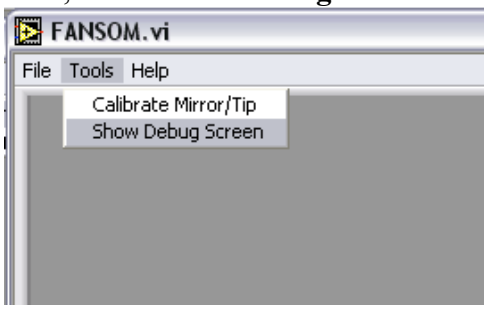
The basic operation of the FANSOM software follows the following scheme:

1. *Focus the objective, ensure operation* - With the debug screen, the user must focus the objective, ensure that photons are being captured, and verify network communication with 'Box H'.
2. *Calibrate tip/mirror control loop* - This must be done once per data acquisition session, or more often as desired.
3. *Capture data* - May it be optical raster images, optical stationary profiles, or synchronous FANSOM images, this involves performing the applicable hardware operations to generate the desired dataset.
4. *Analyze data* - In the case of optical raster and stationary modes, data analysis is performed concurrently with the data acquisition.
5. *Display data* - For optical modes, data are displayed immediately after performing a daq operation. For all modes, data can be loaded to be analyzed and displayed at a later time from archive files.

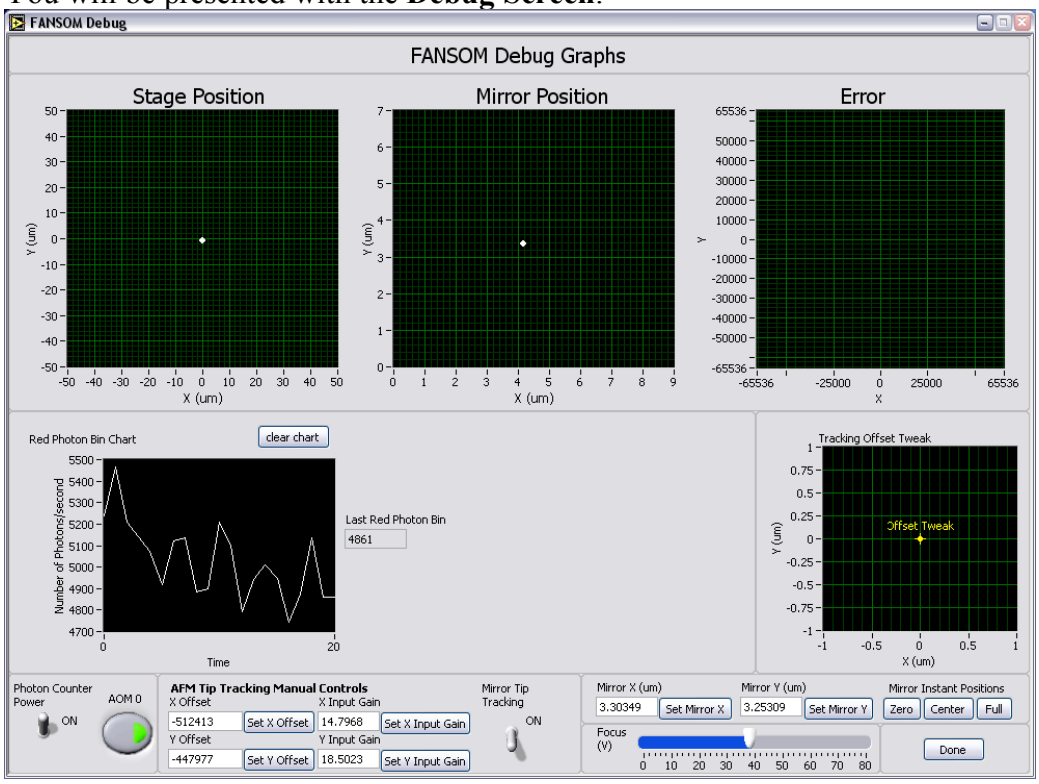
## 6.2 FOCUSING THE OBJECTIVE AND ENSURING OPERATION

Before calibrating and using the FANSOM system, it is necessary to focus the objective and ensure that all aspects of the system are operational. These tasks are accomplished on the Debug Screen.

1. First, select **Show Debug Screen** from the **Tools** menu.



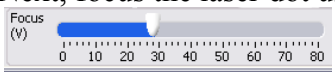
You will be presented with the **Debug Screen**.




2. Next, ensure that the communication with Box H is operational. Enable laser with the **AOM 0** button.


The laser dot should appear on the tv screen or on the vision system screen on the AFM control computer.

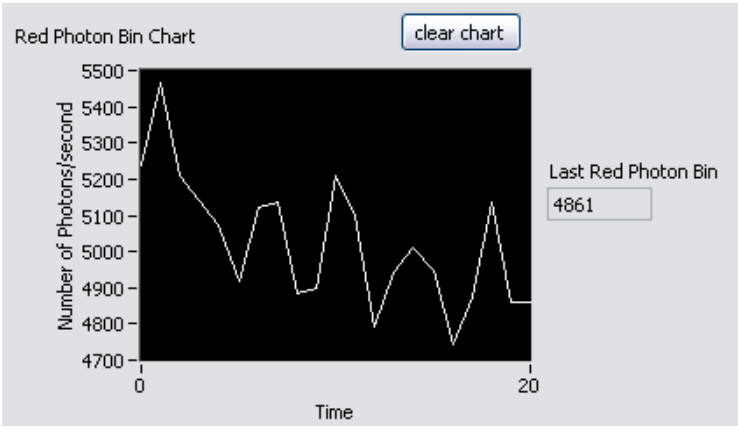
3. Next, focus the laser dot using the **Focus** slider.




4. Finally, make sure that the **Photon Counter Power** switch is at the ON position


and check to make sure that photons are being collected on the **Red Photon Bin**

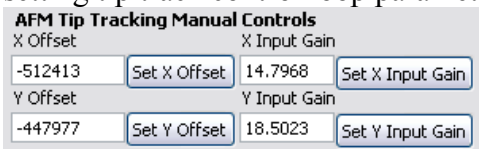
**Chart**.



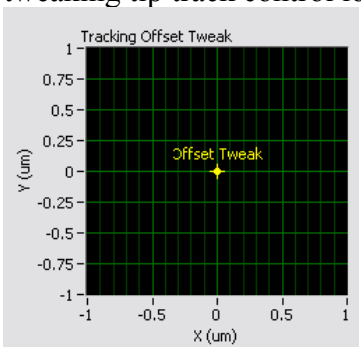

Additional functions on the Debug Screen are:

- activating/deactivating the Mirror Tip Tracking

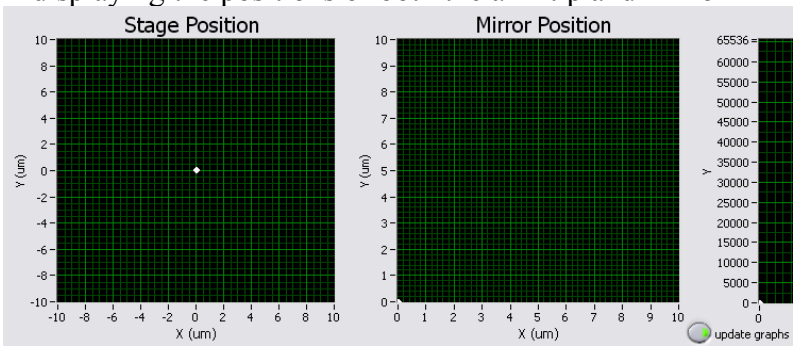

- setting tip track control loop parameters



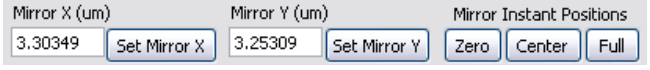

- tweaking tip track control loop offset




- displaying the positions of both the afm tip and mirror

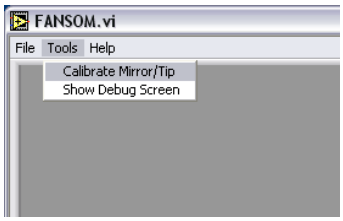



- commanding the mirror to an absolute position



## 6.3 CALIBRATING

1. Open the calibration wizard by selecting the **Calibrate Mirror/Tip** item from the **Tools** menu.



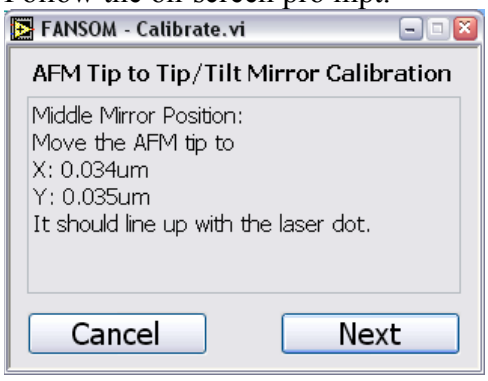

2. Follow the on-screen prompt.


From the Nanoscope control software, using the vision system, align the afm with the location of the blinking laser dot. Click **Next->**

3. Follow the on-screen prompt.


From the Nanoscope control sotware, using the vision system, align the afm with the location of the blinking laser dot. A suggested location will be given on-screen. Click **Next->**

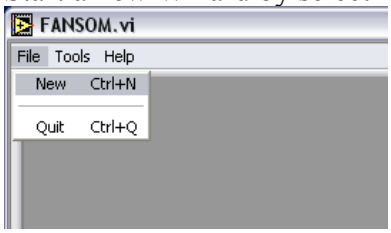
4. Follow the on-screen pro mpt.



Move the afm to the location specified. If the dot and the afm line up, then the system is calibrated, click **Next->**. If the dot and afm do not line up, click **Cancel** and try again.

## 6.4 CAPTURE DATA

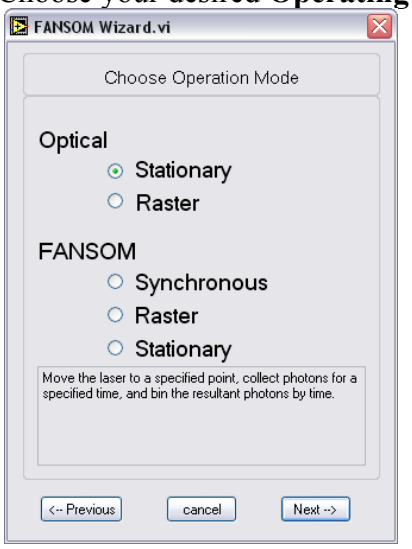
1. Start a new Wizard by selecting the **New** item from the **File** menu.




2. Choose **DAQ** as your **Data Source**.

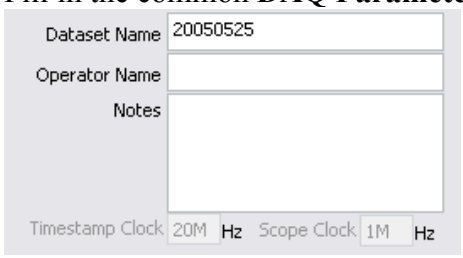

Click **Next->**

3. Choose your desired **Operating Mode**.

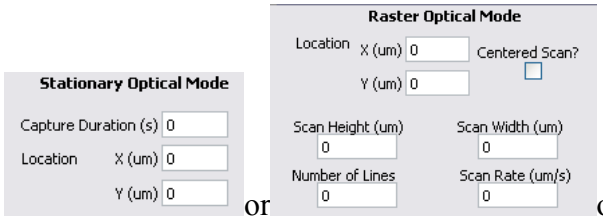



Click **Next->**

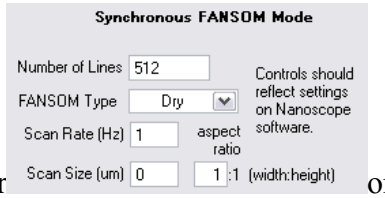
4. Fill in the common **DAQ Parameters**.

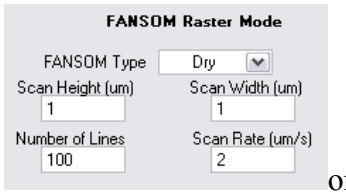


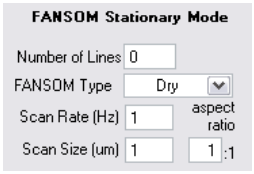

Fill in the Operating Mode-specific **DAQ Parameters**.



or

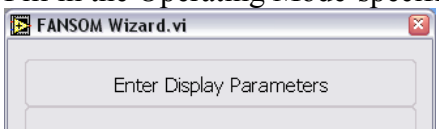

or


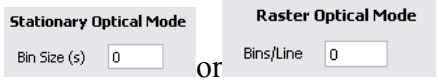
or


or



Click **Next->**

5. Fill in the Operating Mode-specific **Display Parameters**.



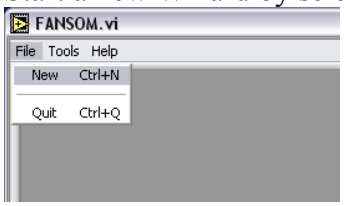
or

(If you choose a FANSOM mode you will not be prompted for Display Parameters because FANSOM mode must be post-processed). Click **Next->** to being the DAQ and, in the case of the optical modes, display of your data

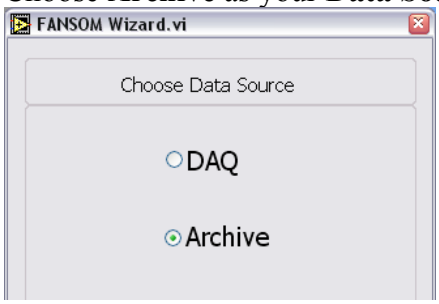
Note: In the case of Synchronous FANSOM and Stationary FANSOM operation, after the Wizard is completed the AFM should be commanded to start from the top of the image.

## 6.5 ANALYZING ARCHIVED DATA

1. Start a new Wizard by selecting the **New** item from the **File** menu.

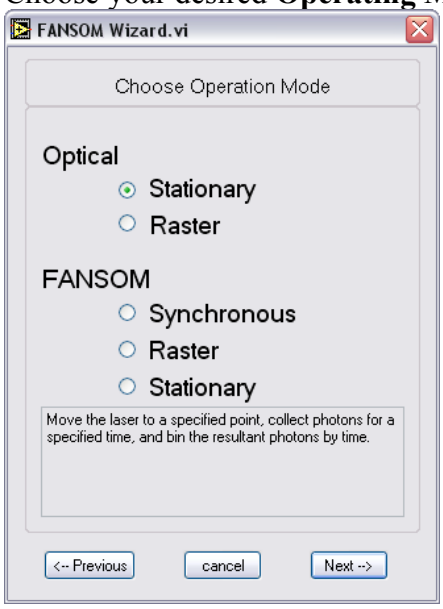



2. Choose **Archive** as your **Data Source**.



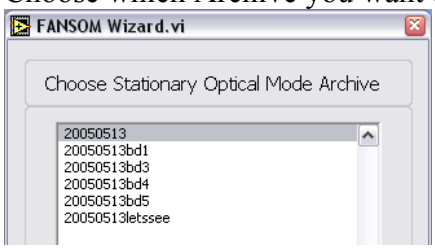

Click **Next->**

3. Choose your desired **Operating Mode**.



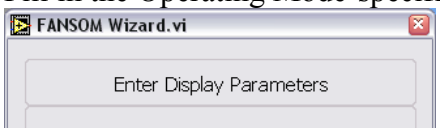

Click **Next->**

4. Choose which Archive you want to open.



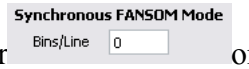

Click **Next->**

5. Fill in the Operating Mode-specific **Display Parameters**.






or


or


or

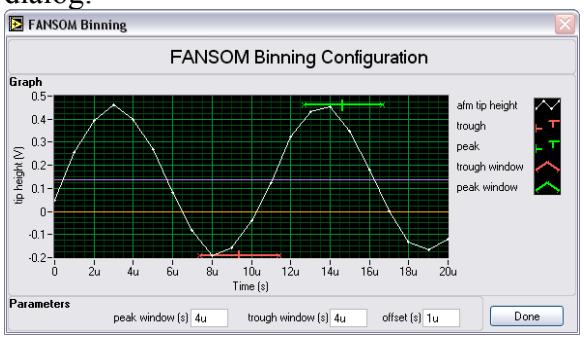

or



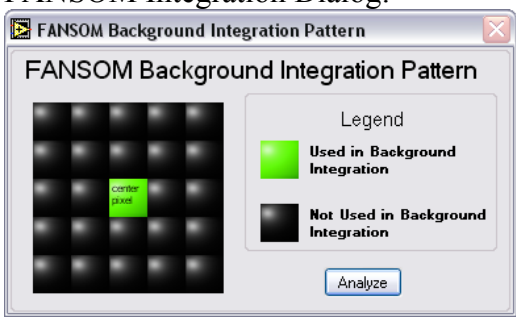
Click **Next->** to being the processing and displaying of your data.

6. For the three FANSOM modes, the user will have to choose the windows around the peaks and troughs where photons should be counted, as well as an offset for all peaks and troughs, if desired. This is accomplished in the FANSOM Binning

dialog:




7. Also in the FANSOM modes, when initial analysis has completed, the user will have to define a region for background signal integration. This is done in the FANSOM Integration Dialog:



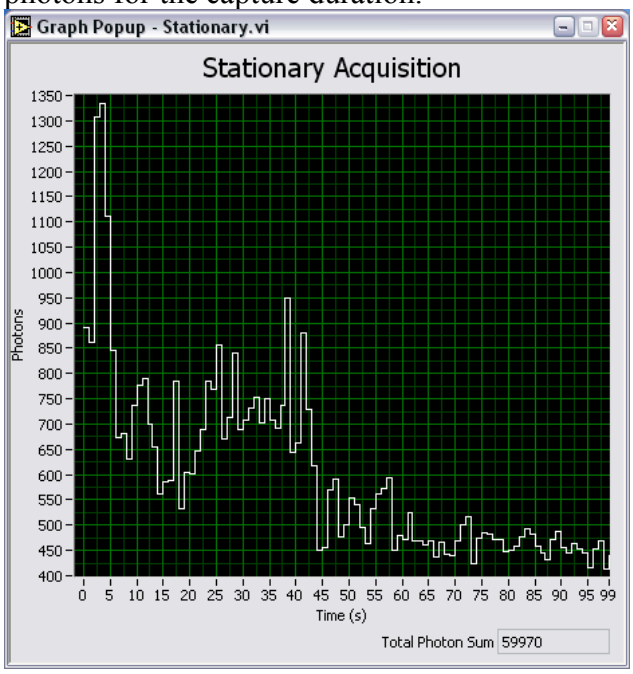

## 6.6 DISPLAYING DATA

Once analysis has completed, the data will be displayed. The manner in which the data are displayed will vary by mode.

### 6.6.1 Stationary Optical Data Display

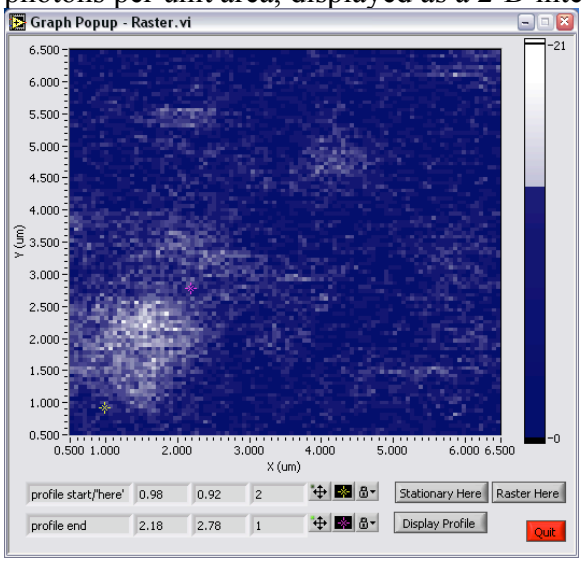
The data product of the Stationary Optical mode is a profile of number of photons vs. time, binned according to a user-specified value. Also displayed is the total number of

photons for the capture duration.




## 6.6.2 Optical Raster Data Display

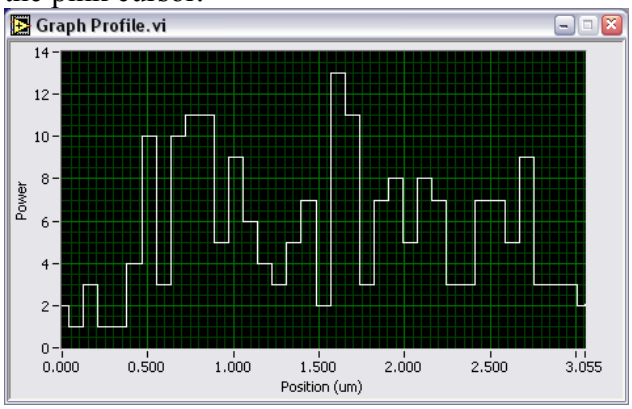
The data product of the Optical Raster mode is a three dimensional plot of number of photons per unit area, displayed as a 2-D intensity graph.




Several additional functions are available once the plot is complete:

- Stationary Here - starts a new DAQ Wizard with Stationary settings at the location of the yellow cursor.
- Raster Here - starts a new DAQ Wizard with Raster settings centered at the yellow cursor.

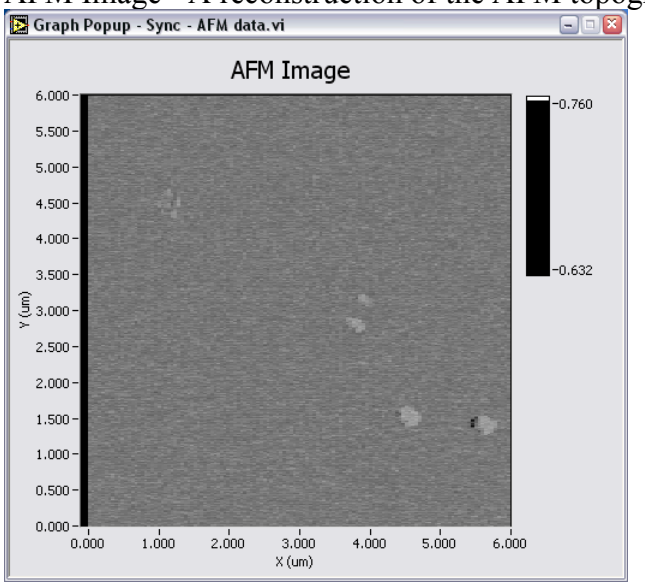
- Display Profile - displays a 'slice' of the intensity graph from the yellow cursor to the pink cursor.



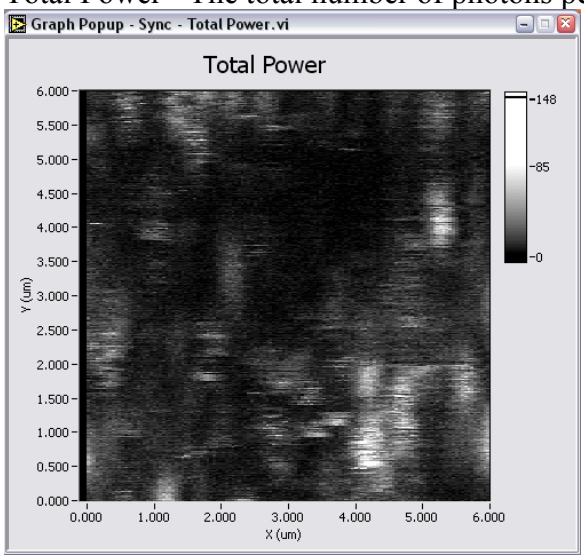

### 6.6.3 FANSOM Data Display

FANSOM modes offers up three data products. They are:

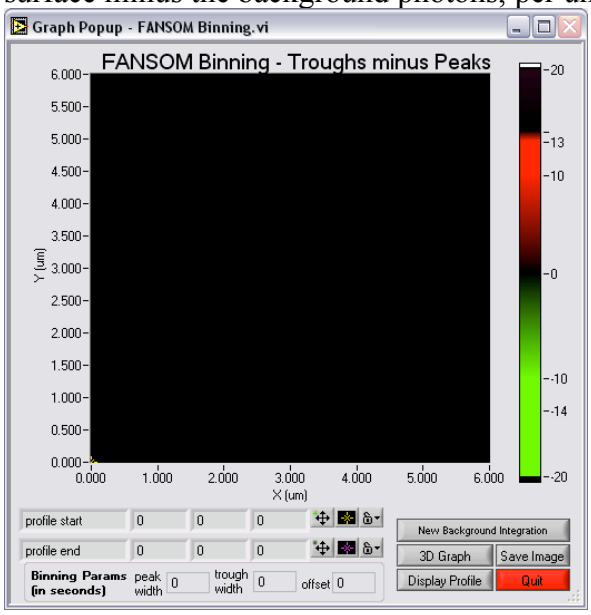
- AFM Image - A reconstruction of the AFM topographical data.



- Total Power - The total number of photons per unit area.

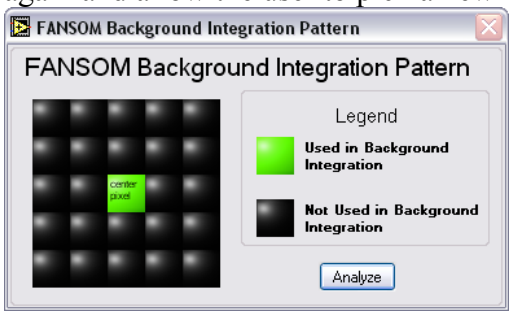



- FANSOM Binning - The total number of photons when the tip is close to the surface minus the background photons, per unit area.

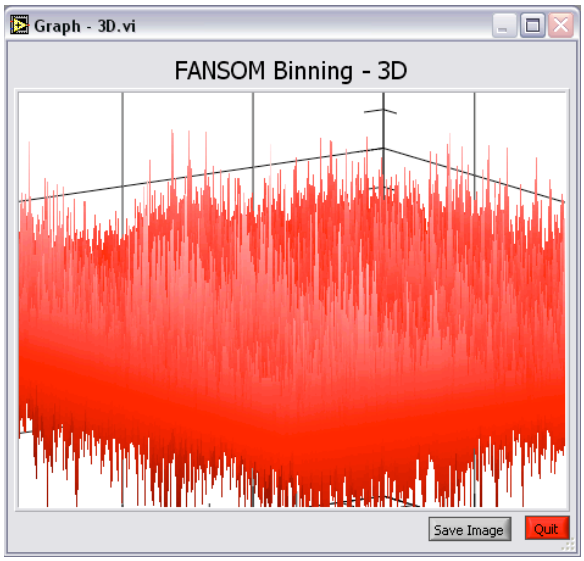



When the plot is complete, clicking…

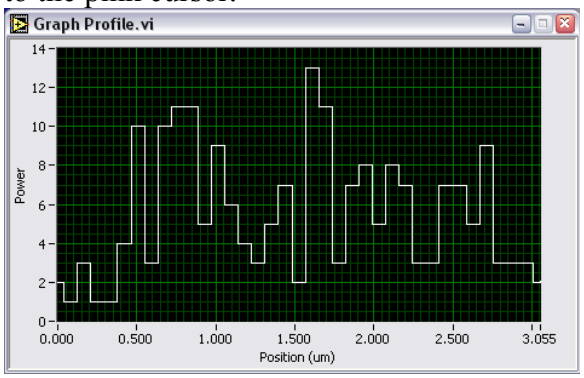
**New Background Integration** will display the FANSOM Integration Dialog again and allow the user to pick a new integration pattern




**3D Graph** will Pops up a 3D version of the binning graph




**Save Image** will allow the user to save another copy of the graph.
**Display Profile** will display a 'slice' of the intensity graph from the yellow cursor to the pink cursor.



# 7 SYSTEM CALIBRATION REQUIREMENTS

The software had to be originally calibrated to convert between the voltage representations of the mirror and AFM positions into actual, physical position units. Since the positions are represented as proportional to these voltages, it was a simple matter of moving each to two different positions and doing a linear fit for x and y in terms of voltage.

For the AFM, this is a matter of commanding the tip to two positions with a DMM connected to the sense coax cables. With the commanded positions and the corresponding voltages in hand, perform a linear fit and place the resultant gain and offset values in the FANSOM Configuration screen shown in Figure 19.

In some ways calibrating the Mirror is easier because its offset is zero (i.e. 0 voltage in both axes translates to 0 displacement in both axes). Thus it is only necessary to measure one point, preferably at a known voltage. The method that I used was to drive the mirror full scale (0V-10V) in both axes (with the Full button on the Debug screen), and then use the AFM and the optical screen to measure its travel. Again, a linear fit can be done with the points and the results entered in the FANSOM Configuration screen shown in Figure 19.

If the Nanoscope is recalibrated internally, or the nanodrive is replaced, then these calibrations ought to be performed again. These two calibrations, combined with the calibration described in section 6.3, represent all of the necessary system calibrations.

# 8 SYSTEM VALIDATION

## 8.1 VALIDATION DESCRIPTION

The completed system was validated by operation. It was deemed correct by the customer when it was operated in modes, which existed in the previous system, and found to get similar results. Other, secondary indicators also existed. For example, the tip height waveform capture could be shown to be correct by comparing the results with the results captured by the commercial AFM software doing the same analysis. The photon power was shown to be correct by showing, by inspection that the power graph lines up well with the expected locations seen on the AFM image.

# 9 ISO COMPLIANCE STATEMENT

This system is compliant with JPL's ISO procedure as described in JPL Procedure: Use of Inspection, Measuring and Test Equipment, Rev. 5 (http://dmie.jpl.nasa.gov/cgi/doc-gw.pl?DocID=30312)

# 10 COMPLIANCE MATRIX

| Requirement | System as Delivered Complies? |
|---|---|
| - System retains the functionality of the previous system. | ✓ |
| - Analysis software converted from Matlab to LabVIEW. | ✓ |
| - All custom hardware redesigned and combined with placeholders kept for future additions. | ✓ |
| - DAQ and interface software redesign with placeholders kept to support future hardware additions. | ✓ |
| - Use of COTS hardware optimized. | ✓ |

# 11 TOTAL COST REPORT

| Type of Labor | Hours | Cost/hour | Total |
|---|---|---|---|
| Technician | 124.5 | $88.00 | $10,956.00 |
| Associate | 474 | $93.00 | $44,082.00 |
| Staff | 288 | $111.00 | $31,968.00 |
| Senior | 4 | $130.00 | $ 520.00 |

Total Hours: 890.5 Labor Total: $87,526.00

(an additional 27 Staff hours and 171 Associate hours were worked and written off as training on several new technologies employed on this task)

| Equipment | Quantity | Cost/unit | Total |
|---|---|---|---|
| Adobe Acrobat | 1 | $190.46 | $ 190.46 |
| NI PXI-5211 | 1 | $6,977.62 | $6,977.62 |
| NI PXI-1042Q | 1 | $2517.31 | $2,517.31 |
| NI PXI-6229 | 1 | $831.95 | $ 831.95 |
| NI PXIPCI-8336 MXI | 1 | $3107.72 | $3,107.72 |
| NI PXI-6602 | 1 | $853.42 | $ 853.42 |
| LabVIEW 7.1 | 1 | $2, 141.59 | $ 143.59 |
| Tax,S&H | 1 | $426.43 | $ 426.43 |
| Teflon Wire | 1 | $405.76 | $ 405.76 |
| | 1 | $60.53 | $ 60.53 |
| Coldfire/Dev Kit | 1 | $569.86 | $ 569.86 |
| | 1 | $56.91 | $ 56.91 |
| | 1 | $65.07 | $ 65.07 |
| | 1 | $65.07 | $ 65.07 |
| 10U Rack | 1 | $167.45 | $ 167.45 |
| | 1 | $336.06 | $ 336.06 |
| Dell Precision 370 | 1 | $2148.10 | $2,148.10 |
| | 1 | $9.71 | $ 9.71 |
| | 1 | $9.71 | $ 9.71 |
| | 1 | $4825.15 | $4,825.15 |
| | | Procurements Total: | $23,767.88 |

# 12 EVALUATION OF SCHEDULE AND COST

The project ended up 10% over the original cost estimate and 166% over the original schedule. There are several factors related to the overrun on each of these metrics.

Schedule, for example, was initially influenced by difficulties with the procurement of all of the National Instruments hardware. Initial delivery of the system to Caltech was completed in a timely fashion, taking this delay into account. Since the March 24 delivery to Caltech, schedule has been influenced by a combination of bug fixing and the addition of new features.

The overruns in cost were due initially to surprises in the implementation of the software and hardware designs, resulting in an early hit to our budget, and later about half and half with the chasing of elusive system bugs and adding new features.

Some of the features added which contribute to the later overruns (estimated to ~10% of the final cost) were not included in the original design and estimate, but were added to facilitate the completion of a truly usable product.

# APPENDIX A – BOX H ETHERNET COMMANDS

**AOM0 ON|OFF** – Controls whether the AOM allows the laser to pass through or not (AOM0 ON : Laser passes through to optical system).

**DBG?** – Requests a debug packet in the format:
[xxxx,xxxx,xxxx,xxxx,f.fff,f.fff]
With the parameters:
Stage X – the x position of the AFM stage as a four-digit hex value
Stage Y – the y position of the AFM stage as a four-digit hex value
Mirror X – the x position of the mirror as a four-digit hex value
Mirror Y – the y position of the mirror as a four-digit hex value
Error X – the x error from the control loop as a five digit float
Error Y – the y error from the control loop as a five digit float

**DISP #** - Changes the display mode on the LCD. [Currently inactive]

**DUMP** – Dumps a full OS state of the Coldfire to the serial port. [Currently inactive]

**FOCS xxxx** – Sets the Focus PZT DAC output to the value represented as the four character hexidecimal representation xxxx (FOCS 0000 : 0V, FOCS : 140V).

**GNNX f.ffff** – Sets the digital control loop's AFM input gain to the value represented by the floating-point string f.ffff (GNNX 0.0000 : 0 input gain). **GNNY f.ffff** does the same for the y-axis.

**GNDX f.ffff** – Sets the digital control loop's D gain to the value represented by the floating-point string f.ffff (GNDX 0.0000 : 0 p gain). **GNDX f.ffff** does the same for the y-axis.

**GNIX f.ffff** – Sets the digital control loop's I gain to the value represented by the floating-point string f.ffff (GNIX 0.0000 : 0 p gain). **GNIX f.ffff** does the same for the y-axis.

**GNPX f.ffff** – Sets the digital control loop's P gain to the value represented by the floating-point string f.ffff (GNPX 0.0000 : 0 p gain). **GNPY f.ffff** does the same for the y-axis.

**LINE ON|OFF** – Controls whether the Coldfire toggles on the LINE pin during each line of the afm's movement (LINE ON : LINE toggles at each line transition).

**MIRX xxxx** – Sets the DAC output for the mirror's x-axis to the value represented by the four digit hex number xxxx (MIRX 0000 : 0V out). **MIRY xxxx** does the same thing for the y-axis.

**OFFX f.ffff** – Sets the digital control loop's AFM x-offset to the value represented by the floating-point string f.ffff (OFFX 0.0000 : 0 offset). **OFFY f.ffff** does the same for the y-axis.

**POWR ON|OFF** – Controls whether the +5V power is output to the photon detector (POWR ON : Power is output).

**RMPX xxxx yyyy zzzz** – Starts an x-axis ramp starting from the DAC output represented by the four-digit hex value xxxx, going to the DAC output represented by the four-digit hex value yyyy, with as many updates per second as represented by the four-digit hex value zzzz (RMPX 0000 FFFF 1111 : a ramp from 0 to full scale lasting 15 seconds).

**RSET** : Resets Box H.

**RVAR** : Resets debug boundary variables. [Currently inactive]

**STA?** – Requests a state packet in the format:

[f.fff,f.fff,f.fff,f.fff,f.fff,f.fff,f.fff,f.fff,f.fff,f.fff,xxxx,x,x]

With the parameters:

Offset X – the x offset for the afm position as a five digit float
Offset Y – the y offset for the afm position as a five digit float
Gain N X – the x input gain for the afm position as a five digit float
Gain N Y – the y input gain for the afm position as a five digit float
Gain P X – the x P gain for the digital control loop as a five digit float
Gain P Y – the y P gain for the digital control loop as a five digit float
Gain I X – the x I gain for the digital control loop as a five digit float
Gain I Y – the y I gain for the digital control loop as a five digit float
Gain D X – the x D gain for the digital control loop as a five digit float
Gain D Y – the y D gain for the digital control loop as a five digit float
Focus – the focus DAC value as a four digit hex number
Flags1 – a single hex digit whose bits mean
(1 : tracking loop on; 4 : aom0 on; 8 : pd power on)
Flags2 – a single hex digit whose bits mean
(1 : line output high; 2 : frame output high)

**TICKS?** – Returns the current number of interrupt ticks. Useful for debugging the rate of interrupt of Box G.

**TRAK ON|OFF** – Controls whether Box H controls the position of the mirror with the digital control loop (TRAK ON : Box H controls the mirror).

# APPENDIX B – HARDWARE SCHEMATICS

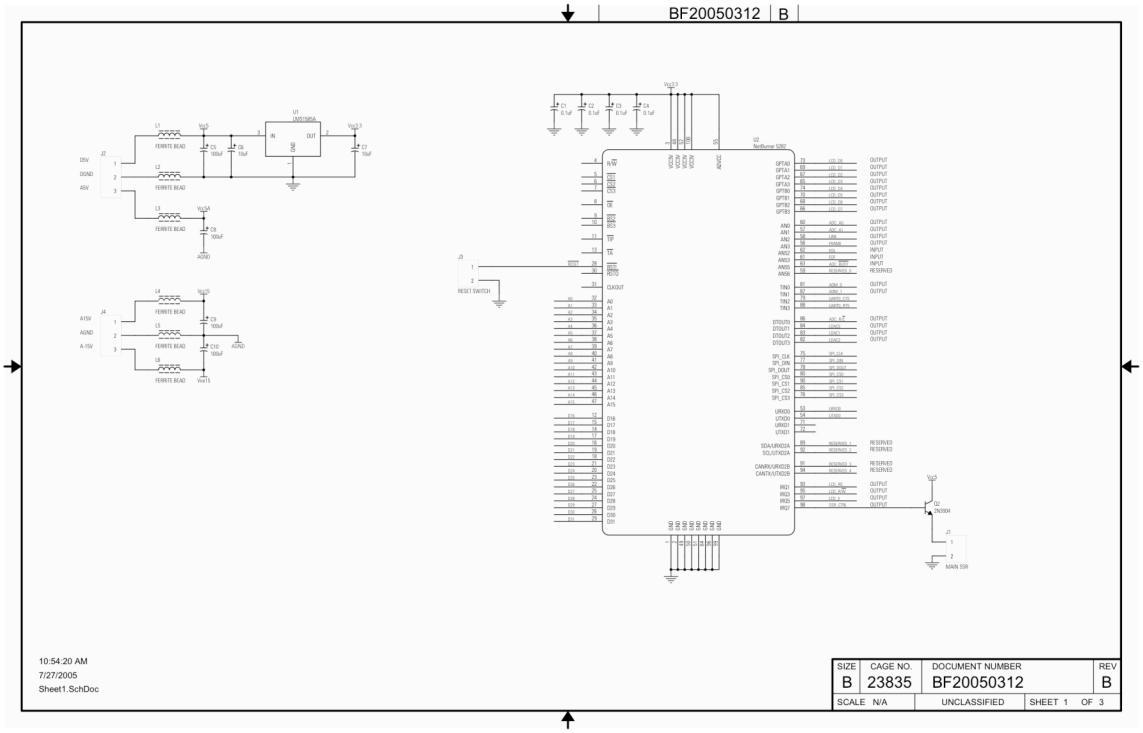



**Figure 20 - Main PCB Schematic #1**

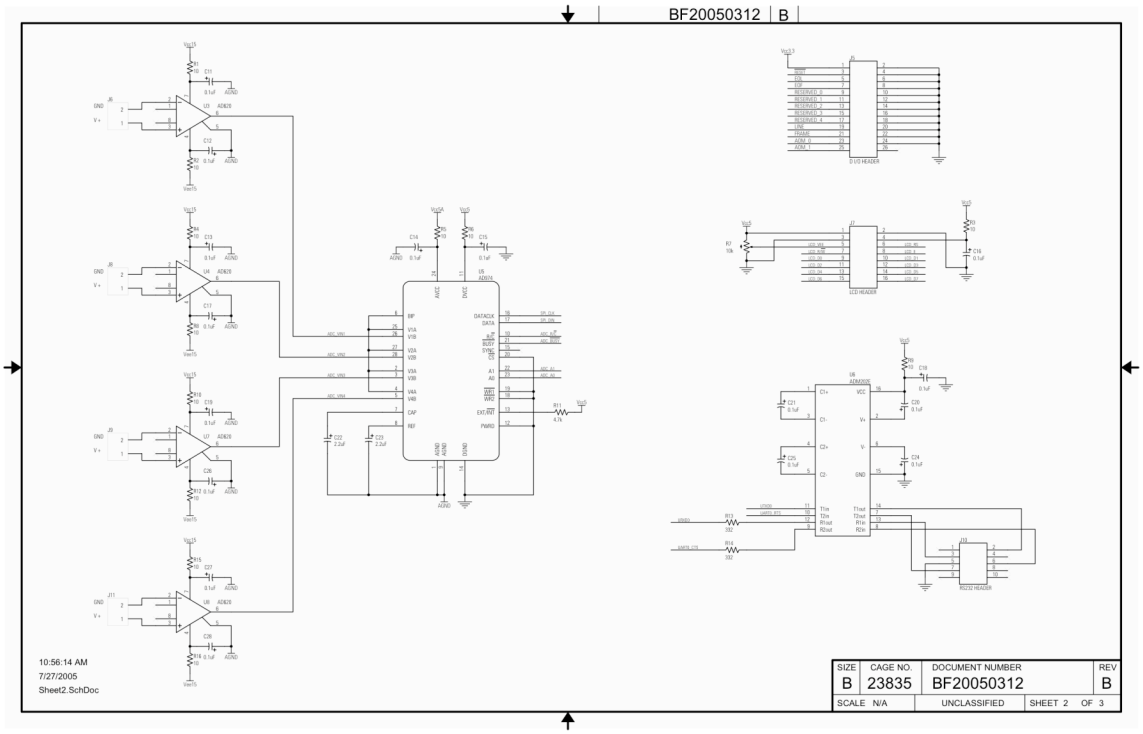



**Figure 21 - Main PCB Schematic #2**

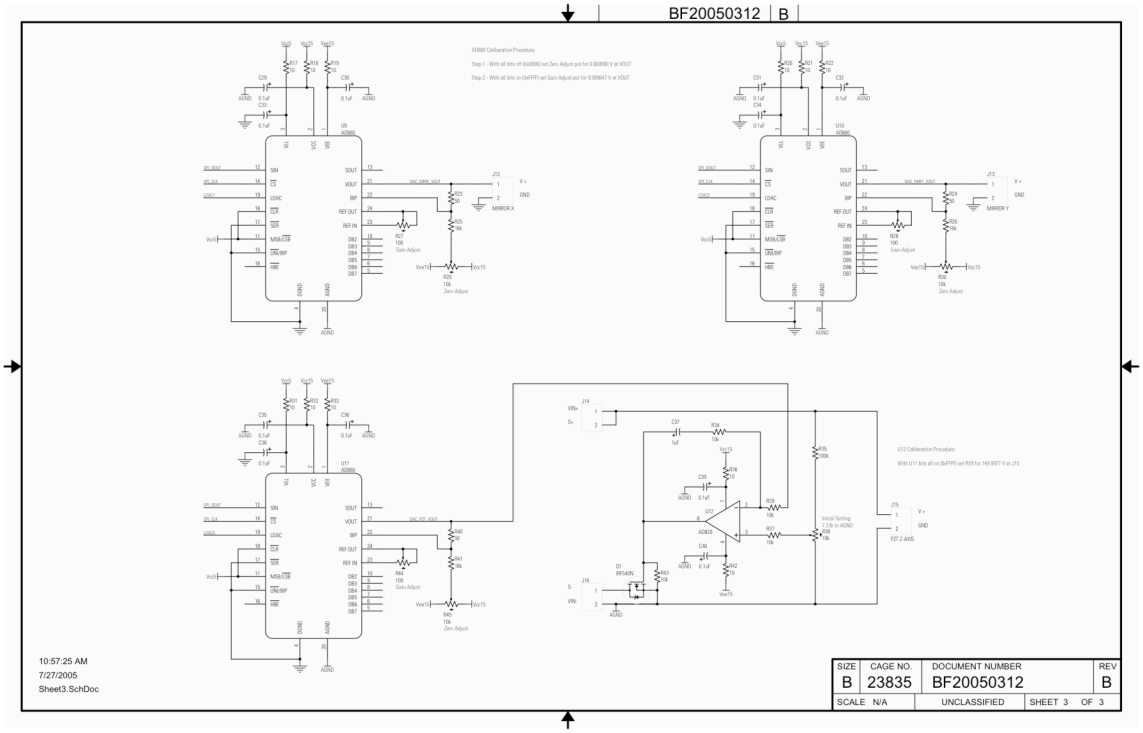



**Figure 22 - Main PCB Schematic #3**

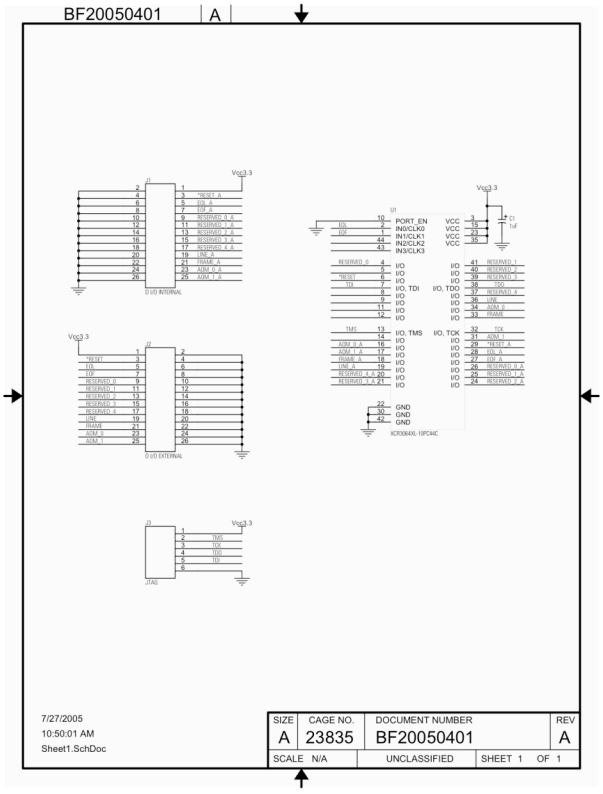



**Figure 23 - CPLD Board Schematic**

# APPENDIX C – PREVIOUS SYSTEM

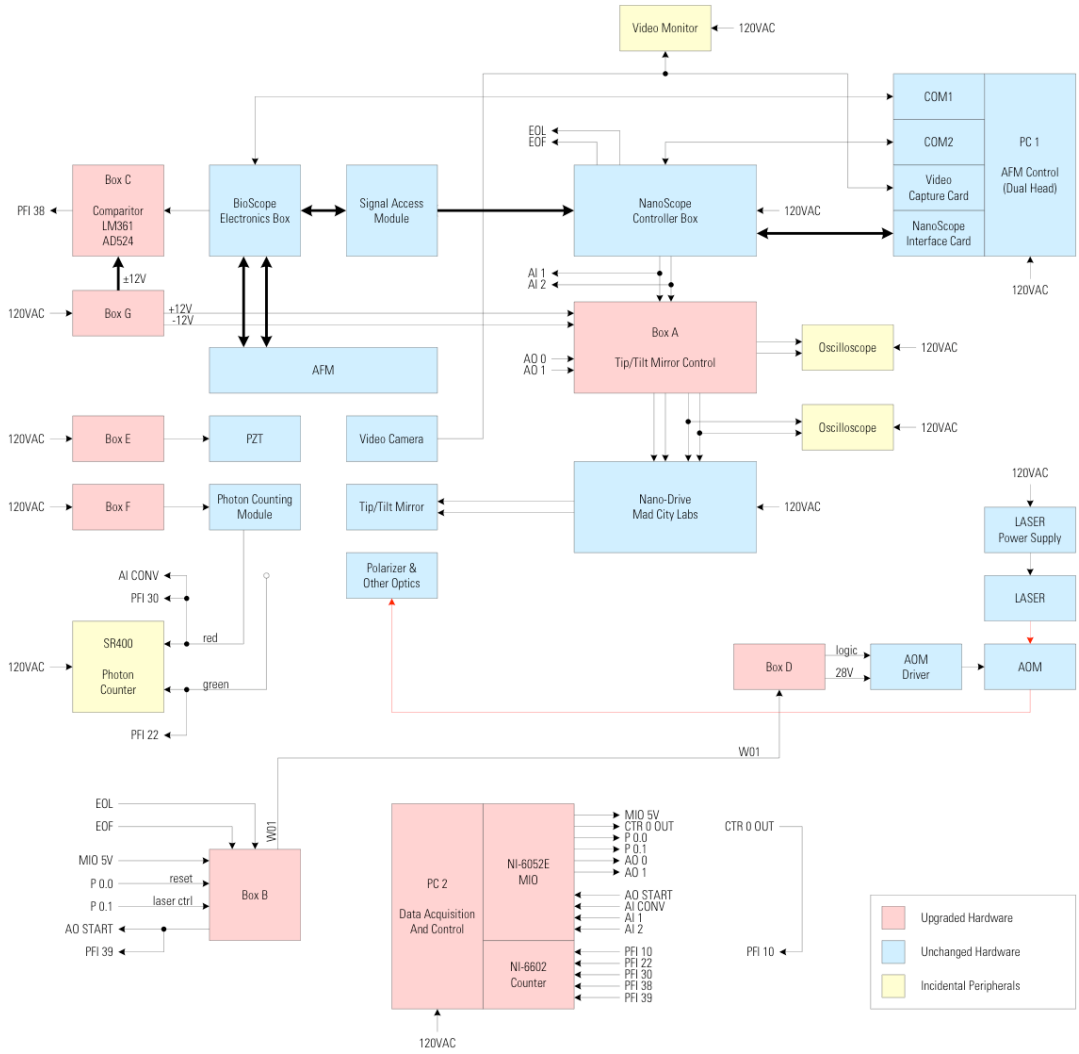



**Figure 24 - Previous System Block Diagram**

**Image Processing Software**

The image processing software was written in Matlab by Eyal Shafran with minor modification by me (version 2.2). The code analyzes the data and creates 4 images: Photon, AFM amplitude, Lock-in magnitude and lock-in phase. There are a few subroutines that are called by the main function:

**Read_txt_File** – Reads the text config file. The following parameters are extracted from the config file: number of lines, scan size, scans rate and aspect ratio.

**Larry_load_data** – Loads the photon data, afm data or both for a specific line.

**Larry_clean_afm_data** – Takes the afm data and filters the data with a +- 2kHz band pass filter around the tips frequency. After filtering, the zero crossing are computed and returned to the main program (Essentially this subroutine converts the analog afm data -> timestamps).

**Tap_Hist** – A c code that finds the phase of each photon.

**Vector_Lock_in** – Each photon is converted into a unit vector in phase space. For each pixel the vectors are summed. The resulting vector is separated into 2 fields: magnitude and phase.

Before you can use the code the c code needs to be compiled. If you are using a mac download the XTool from the operating system installation disk. Once the XTool is installed type: "mex directory /Tap_Hist.c" in the matlab command window (change directory to the actual directory that the file is in). You might need to choose a compiler. Choose the Lcc-win32 C compiler. If the file can't be compiled try typing mex –setup and choose the compiler you want to use. If the file can't be compiled with any of the compilers available give me a call.

There are a few different options to run the code. All of them assume that the config text files have the same template (if this is not true the code might not work).

**Analysis_v1** – A dialog box will be opened. The user needs to select the config txt file. Number of lines is taken from the txt file. The bin number is always equal to the number of lines.

**Analysis_v1(base)** - The base name is specified. Also looks for the config file. The number of lines is taken from the text file and the number of bins will be the same as the number of lines.

**Analysis_v1(base,bin_num)** - The base name is specified. The number of bins is specified in bin_num (can be any integer) and is not necessarily equal to the number of lines.

**Analysis_v1.m**

```
function y = Analysis_v1(varargin)
% ANALYSIS_V1 - Plots photon,afm amplitude, lock-in magnitude and lock-in
% phase images.
% Analysis_v1() - The user needs to select the config txt file. Number of lines is taken from the txt file.
% The bin number is always equal to the number of lines.
% Analysis_v1(base) - The base name is specified. Also looks for the config file.
% Analysis_v1(base,bin_num) - The base name is specified. The number of bins is specified in bin_num (can be any integer) and is not necessarlyequal to the number of lines.

switch nargin
    case 0
        [FileName,base] = uigetfile('*.txt','Choose header file');
        Param = Read_txt_File([base FileName]); % Reads txt config file.
        Lines_num = Param.NumberOfLines;
        M_time = 1/Param.ScanRate/2;
        bin_num = Lines_num;
        k = strfind(FileName, '_config')-1;
        base = [base FileName(1:k)];
    case 1
        base = varargin{1};
        Param = Read_txt_File([base '_config.txt']);
        Lines_num = Param.NumberOfLines;
        M_time = 1/Param.ScanRate/2;
        bin_num = Lines_num;
    case 2
        base = varargin{1};
        Param = Read_txt_File([base '_config.txt']);
        Lines_num = Param.NumberOfLines;
```

```
        M_time = 1/Param.ScanRate/2;
        bin_num = varargin{2};
end;

% Initialize values

Im_amplitude = zeros(Lines_num,bin_num);
Im_Photon = zeros(Lines_num,bin_num);
Im_Magnitude = zeros(Lines_num,bin_num);
Im_Phase = zeros(Lines_num,bin_num);
clock = 1000000;

%Starting main loop

for Line=0:Lines_num-1

    Data = Larry_load_data(base,Line,'all');% Loads the afm and photon data from file
    if(~isempty(Data))
        timestamp = Larry_clean_afm_data(Data.afm); % Cleans the noisy afm data and
returns the afm timestamps
        time = [0:length(Data.afm)]/clock; % Creating a time vector for the afm
        time = time(:);
        bins = 0:M_time/(bin_num):M_time;
        c_photon = histc(Data.data/20000000,bins); % finding the number of photons per
pixel.
        c_photon(end)=[];
        c_Tap = histc(time,bins); % finding the number of taps per pixel.
        c_Tap(end)=[];
        counter_p = 1;
        counter_t = 1;
        index1 = cumsum(c_photon);
        index2 = cumsum(c_Tap);
        amplitude = zeros(bin_num,1);
        Magnitude = zeros(bin_num,1);
        Phase = zeros(bin_num,1);
        Rel_Tap = Tap_Hist(Data.data/20000000, timestamp);% c code that computes the
phase for each photon
        for i=1:bin_num
            Tap = Data.afm(counter_t:index2(i));
            counter_t = index2(i)+1;
            % Doing a fft
            L = length(Tap);
            NFFT = 2^nextpow2(L);
            FT = fft(Tap,NFFT)/L;
            power = abs(FT);
            f = clock*linspace(0,1,NFFT);
```

```
        k = find(f>20000 & f < 100000);
        mc = max(power(k)); % finding the amplitude at the frequency of the tip.
        amplitude(i)= (mc);
        if(~isempty(Rel_Tap(counter_p:index1(i)))) % If there is at least 1 photon
            [Magnitude(i),Phase(i)] = Vector_Lock_in(Rel_Tap(counter_p:index1(i)),M_time/bin_num); % Run vectorial lock-in
        else
            Magnitude(i)=NaN;
            Phase(i)=NaN;
        end;
        counter_p = index1(i)+1;
    end;
    Im_Magnitude(Line+1,:) = Magnitude;
    Im_amplitude(Line+1,:) = amplitude;
    Im_Phase(Line+1,:) = Phase;
    Im_Photon(Line+1,:) = c_photon;
  else
    Im_Magnitude(Line+1,1:bin_num) = NaN;
    Im_amplitude(Line+1,1:bin_num) = NaN;
    Im_Phase(Line+1,1:bin_num) = NaN;
    Im_Photon(Line+1,1:bin_num) = NaN;
  end;

  Line
end;
y.x = linspace(0,1,bin_num)*Param.ScanSize; % creating x vecto
y.y = linspace(0,1,Lines_num)*Param.ScanSize; % creating y vector
y.Magnitude = Im_Magnitude;
y.amplitude = Im_amplitude;
y.Phase = Im_Phase;
y.photon = Im_Photon;
if(Param.ratio==1 && bin_num==Lines_num) % if aspect ratio is 1 and bins=lines the image is scaled properly.
  subplot(2,2,1);imagesc(y.y,y.x,y.photon);axis image; title('Photon');colorbar;
  subplot(2,2,2);imagesc(y.y,y.x,y.amplitude);axis image; title('Amplitude'); colorbar;
  subplot(2,2,3);imagesc(y.y,y.x,y.Magnitude);axis image; title('Lock-in magnitude'); colorbar;
  Phase_hsv = zeros(size(y.Phase,1),size(y.Phase,2),3);
  Phase_hsv(:,:,1) = (y.Phase-min(y.Phase(:)))/max((y.Phase(:)-min(y.Phase(:))));
  Phase_hsv(:,:,2) = (y.Magnitude-min(y.Magnitude(:)))/max((y.Magnitude(:)-min(y.Magnitude(:))));
  Phase_hsv(:,:,3) = 1.0;
                                                    Phase_rgb = hsv2rgb(Phase_hsv);
   subplot(2,2,4);imagesc(y.y,y.x,Phase_rgb);axis image;  title('Lock-in phase'); colorbar;
  Overlay = zeros(size(y.Phase,1),size(y.Phase,2),3);
```

```
   Overlay(:,:,1)          =          (y.amplitude-min(y.amplitude(:)))/max((y.amplitude(:)-
min(y.amplitude(:))));
   Overlay(:,:,2) = (y.photon-min(y.photon(:)))/max((y.photon(:)-min(y.photon(:))));
   figure;
                imagesc(y.y,y.x,Overlay);axis image; title('AFM = Red ; Photon = Green');

else
   subplot(2,2,1);imagesc(y.photon); title('Photon')
   subplot(2,2,2);imagesc(y.amplitude); title('Amplitude')
   subplot(2,2,3);imagesc(y.Magnitude); title('Lock-in magnitude')
   subplot(2,2,4);imagesc(y.Phase); title('Lock-in phase');
end;
```

**Larry_afm_images.m**

```
function y = Larry_afm_images(base,Lines_num, bin_num, M_time)

for Line=0:Lines_num-1

   Data = Larry_load_data(base,Line,'all');
   clock = 1000000;
   time = [0:length(Data.afm)]/clock;
   time = time(:);
   bins = 0:M_time/(bin_num):M_time;
   c_photon = histc(Data.data/20000000,bins);
   c_photon(end)=[];
   c_Tap = histc(time,bins);
   c_Tap(end)=[];
   counter_p = 1;
   counter_t = 1;
   index1 = cumsum(c_photon);
   index2 = cumsum(c_Tap);
   for i=1:bin_num
      P = Data.data(counter_p:index1(i))/20000000;
      counter_p = index1(i)+1;
      Tap = Data.afm(counter_t:index2(i));
      t = time(counter_t:index2(i));
      counter_t = index2(i)+1;
      L = length(Tap);
      NFFT = 2^nextpow2(L);
      FT = fft(Tap,NFFT)/L;
      power = abs(FT);
      f = clock*linspace(0,1,NFFT);
%     f = clock*(0:(N-1))/N;
      k = find(f>20000 & f < 100000);
      [mc,mi] = max(power(k));
      frequency(i) = f(mi+k(1)-1);
```

```
        height(i)= power(1)/length(Tap);
        amplitude(i)= (mc);%/length(Tap)*2;
    end;
    Im_frequency(Line+1,:) = frequency;
    Im_amplitude(Line+1,:) = amplitude;
    Im_height(Line+1,:) = height;
    Im_Photon(Line+1,:) = c_photon;
    Line
end;

y.frequency = Im_frequency;
y.amplitude = Im_amplitude;
y.height = Im_height;
y.photon = Im_Photon;
subplot(2,2,1);imagesc(y.photon);
subplot(2,2,2);imagesc(y.frequency);
subplot(2,2,3);imagesc(y.amplitude);
subplot(2,2,4);imagesc(y.height);
```

**Larry_Analysis.m**

```
function Data = Larry_Analysis(bin_num)
% Initialize values
[FileName,base] = uigetfile('*.txt','Choose header file');
Param = Read_txt_File([base FileName]);
Lines_num = Param.NumberOfLines;
time = 1/Param.ScanRate;
Amplitude_image = zeros(Lines_num,bin_num);
Photon_image = zeros(Lines_num,bin_num);
Magnitude = zeros(Lines_num,bin_num);
Phase = zeros(Lines_num,bin_num);
time = time/2;
k = strfind(FileName, '_config')-1;
base = [base FileName(1:k)];
clock = 1000000;

for Line=0:Lines_num-1 % Run code for all the lines

    Data = Larry_load_data(base,Line,'all'); % Loads Photon & AFM data for a given
Line.

    if(~isempty(Data)) % Checking if line is not empty. If Data is empty go to else.
        Data.data = Data.data/20000000; % Divide photon data by card clock to get time.
        [timestamp,Data.afm] = Larry_clean_afm_data(Data.afm); % Cleans the noisy afm
data
```

```
        bins = 0:time/(bin_num):time;
        afm_time = [0:length(Data.afm)]/clock;
        afm_time = afm_time(:);
        c_photon = histc(Data.data,bins); % finds the number of photons per bin.
        c_photon(end)=[];
        c_Tap = histc(afm_time,bins);
        c_Tap(end)=[];
        counter_p = 1;
        counter_t = 1;
        index1 = cumsum(c_photon);
        index2 = cumsum(c_Tap);
        for i=1:bin_num
            P = Data.data(counter_p:index1(i));
            Tap = Data.afm(counter_t:index2(i));
            counter_t = index2(i)+1;
            FT = fft(Tap);
            power = abs(FT);
            N = length(FT);
            f = clock*(0:(N-1))/N;
            k = find(f>20000 & f < 100000);
            [mc,mi] = max(power(k));
            amplitude(i)= (mc)/length(Tap)*2;
            if(~isempty(P)) % If there is at least 1 photon
                counter_p = index1(i)+1;
                Rel_Tap = Tap_Hist(P, timestamp); % Convert photon timetag to phase
                [Data.Lock_in(i),Data.Phase(i)] = Vector_Lock_in(Rel_Tap,time/bin_num); %
Run vectorial lock-in
            else
                Data.Lock_in(i)=NaN;
                Data.Phase(i)=NaN;
            end;
        end;
        Amplitude_image(Line+1,:) = amplitude;
        Photon_image(Line+1,:) = c_photon;
        Magnitude(Line+1,:) = Data.Lock_in;
        Phase(Line+1,:) = Data.Phase;
    else
        Amplitude_image(Line+1,1:bin_num) = NaN;
        Photon_image(Line+1,1:bin_num) = NaN;
        Magnitude(Line+1,1:bin_num) = NaN;
        Phase(Line+1,1:bin_num) = NaN;
    end;
    Line
end;
keyboard;
Data.Amplitude_image = Amplitude_image;
```

```
Data.Magnitude = Magnitude;
Data.Phase = Phase;
Data.Photon_image = Photon_image;
x = 0:Param.ScanSize/(bin_num-1):Param.ScanSize;
y = 0:Param.ScanSize/(Lines_num-1):Param.ScanSize;
figure;
subplot(2,2,1);imagesc(x,y,Photon_image);axis image;title('Photon Image');
subplot(2,2,2);imagesc(x,y,Magnitude);axis image;title('Lock-in Magnitude')
subplot(2,2,3);imagesc(x,y,Phase);axis image;title('Lock-in phase');
subplot(2,2,4);imagesc(x,y,Amplitude_image);axis image;title('Amplitude image');
```

**Larry_clean_afm_data.m**

```
function [timestamp,afm] = Larry_clean_afm_data(afm)
% LARRY_CLEAN_AFM_DATA(afm) - cleans the afm data and returns timastamps

clock = 1000000;
time = [0:length(afm)-1]/clock;
time = time(:);
FT = fft(afm);
power = abs(FT(1:length(FT)/2));
N = length(FT);
f = clock*(1:N/2)/(N/2)*1/2;
k = find(f>10000 & f < 100000);
[mc,mi] = max(power(k)); % Finds the tips frequency from the power spectrum.
frequency = f(mi+k(1)-1); % Finds the frequency at the peak of the power spectrum.
FT(f< frequency-2000 | f > frequency+2000)=0; % band pass filter with +-2kHz on each
side of the frequency.
afm = real(ifft(FT));
%afm = smooth(afm,5);
timestamp = zero_crossing(time, afm,1/clock); % Finds the zero crossings.
ave_frequency = 1/mean(diff(timestamp));
if(abs(frequency-ave_frequency)/frequency > 0.01)
   [frequency ave_frequency]
   disp('Average frequency for the calculated timestamps is not close to the AFM
frequency from fft ');
end
```

**Larry_fft_compare.m**

```
function  Larry_fft_compare(Photon,afm)

clock = 1000000;
time = 1/clock:1/clock:length(afm)/clock;
time = time(:);
FT = fft(afm);
```

```
power = zeros(length(FT)/2,1);
power = abs(FT(1:length(FT)/2));
N = length(FT);
f = clock*(1:N/2)/(N/2)*1/2;
b = 0:1/clock:time(end);
F = zeros(length(b),1);
F = histc(Photon,b);
F(end)=[];
FT2 = zeros(length(F),1);
FT2 =fft(F);
%FT(end)=[];
power2 = zeros(length(FT2)/2,1);
power2 = abs(FT2(1:length(FT2)/2));
N = length(FT2);
figure; loglog(f,power/max(power),f,power2/max(power));
figure; plot(f,power/max(power),f,power2/max(power));
```

**Larry_load_data.m**

```
function y = Larry_load_data(base,line,data_type)
% LARRY_LOAD_DATA(base,line,data_type) - Loads measured line from data. base is
the
% path of the file, line is the line that is being loaded, and data_type is
% one of 3 options - 'photon','afm','all'.


try
   if(strcmp(data_type,'photon'))
      fname = [base '-' num2str(line, '%03d') '-redphotons.raw'];
      file = fopen(fname, 'r', 'ieee-be');
      [y.data, y.count] = fread(file, 'uint32');
      fclose(file);
   elseif(strcmp(data_type,'afm'))
      fname = [base '-' num2str(line, '%03d') '-afmtipheight.raw'];
      file = fopen(fname, 'r', 'ieee-be.l64');
      [y.afm, y.afm_count] = fread(file, 'float64');
      fclose(file);
   elseif(strcmp(data_type,'all'))
      fname = [base '-' num2str(line, '%03d') '-redphotons.raw'];
      file = fopen(fname, 'r', 'ieee-be');
      [y.data, y.count] = fread(file, 'uint32');
      fclose(file);
      fname = [base '-' num2str(line, '%03d') '-afmtipheight.raw'];
      file = fopen(fname, 'r', 'ieee-be.l64');
      [y.afm, y.afm_count] = fread(file, 'float64');
      fclose(file);
```

```
    else
        error('Can not recognize data type. Choose one of the following: photon, afm or all')
    end;
catch
    y = [];
    disp('Line is empty ');
end;
```

**Larry_optical_image.m**

```
%% QUICK OPTICAL IMAGE

for Line=0:511
    Data = Larry_load_data(base,Line,'photon');
    bins = 0:0.5/511:0.5;
    c_photon = histc(Data.data/20000000,bins);
    Photon_image(Line+1,:) = c_photon;
end;
figure;imagesc(Photon_image); axis image
```

Read_txt_File.m

```
function y = Read_txt_File(path)

fid = fopen(path);
File = fread(fid,inf,'*char')';
fclose(fid);
k = strfind(File, 'NumberOfLines=')+14;
y.NumberOfLines = str2double(File(k:k+4));
k = strfind(File, 'ScanRate=')+9;
y.ScanRate = str2double(File(k:k+4));
k = strfind(File, 'ScanSize=')+9;
y.ScanSize = str2double(File(k:k+4));
k = strfind(File, 'aspectratio=')+12;
y.ratio = str2double(File(k:k+2));
```

**Tap_Hist_matlab.m**

```
function Theta = Tap_Hist_matlab(Photons, Phase)
m_Photons = length(Photons);
m_Phase = length(Phase);
i=1;k=1;
while (Photons(i)< Phase(1) )
    i = i+1;
    if(i>m_Photons)
```

```
      Theta(1)=-1;
      return
   end;
end;
for j=i:m_Photons
   while (Photons(j)>=Phase(k) && k<m_Phase)
      k = k+1;
   end;
   try
      diff = Phase(k)-Phase(k-1);
      if(diff>=0)
         P = Photons(j)-Phase(k-1);
         d = P*2*pi/diff;
         Theta(j) = d;
      else
         Theta(j)=-1;
      end;
   catch
   end;
end;
```

**Tap_Hist.c**

```
//Phase_Hist_v6(Photon(k), Ave_Phase, N); Matlab code to run the dll
#include "mex.h"

void Phase_divider(double *Photons, double *Phase, mwSize m_Photons,mwSize
m_Phase, double *Theta)
{

   int i=0,k=0;
   double diff, P,d, PI = 3.141592653589793;
   for(; *(Photons+i)<*(Phase);i++)
      *(Theta+i) = -1; // Taking care of a photon arriving before the first Phase marker

   for (; i <= (m_Photons-1); i++)
   {
      for (; *(Photons+i)>=*(Phase+k) && k<=(m_Phase-1); k++);//Advance Photon
until it is bigger then the Tap
      diff = (*(Phase+k) - *(Phase+k-1));
      if (diff >= 0)
      {
         P = *(Photons+i)-*(Phase+k-1);
         d = (P/diff)*2*PI;
         *(Theta+i) = d;
      }
```

```
        else
            *(Theta+i) = -1;
    }

}



void mexFunction( int nlhs, mxArray *plhs[],
                  int nrhs, const mxArray *prhs[])
{
    double *Photons, *Phase, *z ;
    mwSize m_Photons, m_Phase;

    Photons = mxGetPr(prhs[0]);
    Phase = mxGetPr(prhs[1]);

    m_Photons = mxGetM(prhs[0]);
    m_Phase = mxGetM(prhs[1]);
    if(m_Photons!=0 && m_Phase!=0){
        plhs[0] = mxCreateDoubleMatrix(1,m_Photons, mxREAL);
        z = mxGetPr(plhs[0]);
        Phase_divider(Photons,Phase,m_Photons, m_Phase,  z);
    }
    else{
        plhs[0] = mxCreateDoubleMatrix(1,1, mxREAL);
        z = mxGetPr(plhs[0]);
        *z = -2;
    }


}
```

**Vector_Lock_in.m**

```
function [R,phi] = Vector_Lock_in(Rel_Tap,Time,varargin)
% Vector_Lock_in(Rel_Tap) is a lock-in code. The magnitude and phase are
% calculated by a vetorial sum on all phases(Rel_Tap)
if(nargin==3)
    beta= varargin{1};
    mu = beta(1);
    sigma = beta(2);
    Rel_Tap(find(Rel_Tap<mu-pi))=Rel_Tap(find(Rel_Tap<mu-pi))+2*pi;
    Rel_Tap(find(Rel_Tap>mu+pi))=Rel_Tap(find(Rel_Tap>mu+pi))-2*pi;
```

```
   Rel_Tap = Rel_Tap(find((abs(Rel_Tap-mu)<sigma) | (Rel_Tap-mu<-pi+sigma) | (Rel_Tap-mu>pi-sigma))); %[-sigma,sigma]
   Time = Time*2*sigma/pi;
elseif(nargin>3)
   error('Too many input arguements')
end
R = sqrt(sum(cos(Rel_Tap)).^2+sum(sin(Rel_Tap)).^2)/Time;
if(sum(cos(Rel_Tap))> 0 )
   phi = atan(sum(sin(Rel_Tap))/sum(cos(Rel_Tap)));
else
   phi = atan(sum(sin(Rel_Tap))/sum(cos(Rel_Tap)))+pi;
end;
```

**zero_crossing.m**

```
function y = zero_crossing(time, signal,deltaT)
% ZERO_CROSSING(time, signal, deltaT) - finds the zero crossing of an
% oscillating signal. signal and time are the actual signal and the time
% vector of that signal. deltaT is the time difference between to time
% points.

time = time(:);
signal = signal(:);
d = -1*signal(1:end-1)./(diff(signal));
%ZC = time(2:end)+d*deltaT./exp(deltaT*(0.5-d));
ZC = time(2:end)+d*deltaT;
index1 = find(signal(1:end-1).*signal(2:end)< 0 );
if(index1(end)==length(signal))
   index1(end) = [];
end
index1 = index1(find(signal(index1)-signal(index1+1)>0));
y = ZC(index1);
```

## References

[1] Closed-Look Bioscope with Nanoscope IV controller, Veeco Metrology, 112 Robin Hill Road, Santa Barbara, CA 93117, USA.