OPTICAL BIOPSY: COMPLEMENTING HISTOLOGY WITH NONLINEAR OPTICAL MICROSCOPY

A Senior Honors Thesis

by

CHRISTINA M. SHAFER

Submitted to the Office of Honors Programs
& Academic Scholarships
Texas A&M University
In partial fulfillment of the requirements of the

UNIVERSITY UNDERGRADUATE
RESEARCH FELLOWS

April 2006

Major: Biomedical Engineering

OPTICAL BIOPSY: COMPLEMENTING HISTOLOGY WITH NONLINEAR OPTICAL MICROSCOPY

A Senior Honors Thesis

by

CHRISTINA M. SHAFER

Submitted to the Office of Honors Programs
& Academic Scholarships
Texas A&M University
In partial fulfillment for the designation of

UNIVERSITY UNDERGRADUATE
RESEARCH FELLOWS

Approved as to style and content by:

---	---
Alvin T. Yeh	Edward A. Funkhouser
(Fellows Advisor)	(Executive Director)

April 2006

Major: Biomedical Engineering

ABSTRACT

Optical Biopsy: Complementing Histology with Nonlinear Optical Microscopy (April 2006)

Christina M. Shafer
Department of Biomedical Engineering
Texas A&M University

Fellows Advisor: Professor Alvin T. Yeh
Department of Biomedical Engineering

Histology, though widely used for cellular imaging, suffers from its destructive nature; cells are not viewed in their natural living environment. Nonlinear optical microscopy (NLOM) has been widely used to view cells *in situ* without a need for sample processing. NLOM uses a high-powered ultrafast laser to generate nonlinear optical signals within living tissue. These signals are used to render two- and three-dimensional images. One improvement to be made on this imaging system is the capability to produce spectral images rather than merely intensity images. Spectral information provides insight to the sample's chemical environment; this insight may lead to increased efficiency in disease diagnosis, since pathological development begins at the

microscopic level. Additional detectors added to the system will allow wavelength discrimination and the creation of spectral images. To achieve this type of system, additional hardware must be built and software must be written to allow simultaneous data acquisition from 32 detectors. The initial task competed involved the scanning mechanism; a program was created to control motorized optical scanning mirrors. The next task required a circuit board to be built to interface the detectors with the computer. A sub-program was then designed to save tissue response matrices (images) as binary files. The files were able to be opened in MATLAB® and converted into scaled intensity images. The final program created used information from the circuit board (i.e. from each detector) to create the response matrix to be saved as a binary file. All hardware and software was then integrated into the current imaging system to be tested with two detectors. At this point, the new system cannot adequately render images. Future work will involve software and hardware correction until images are correctly formed. At this point, the tissue response to two different ranges of light wavelengths will able to be viewed separately, and the matrices can be added to create a grayscale image. At this point, work on this project will involve expanding the hardware and software to incorporate 30 additional detectors. The spectral detection system will allow image

segmentation of biological components, and chemical markers associated with cellular abnormalities as well as different genetic markers can be viewed on a microscopic level. Therefore, this system is of great value to medicine and science.

ACKNOWLEDGEMENTS

This research was supported, in part, by funds from the Honors Programs Office and Academic Scholarships. Additional support was provided by the National Science Foundation and the National Institutes of Health.

TABLE OF CONTENTS

TABLE OF CONTENTS CONTINUED

LIST OF TABLES AND FIGURES

INTRODUCTION[1]

In the past, clinics and the labs have relied upon histo(path)ology in order to observe tissue on a cellular level. The histological process involves excising tissue of interest from the patient and processing it for viewing purposes (in two dimensions only). Since this process is time consuming, difficult to control, and quite invasive, it has become increasingly important to find an alternate method of viewing cells and tissues. One method, referred to as optical biopsy, involves microscopy and will enable the performance of time-dependent studies on living tissue in the laboratory and, clinically, could greatly reduce patient discomfort. New improvements to be made to optical biopsy focus on development of acquiring color images of live tissues. Spectral data provides chemical information about the scanned subject, and since a cell's response depends on its chemical environment, having spectral data is crucial to the determination of cellular response to certain stimuli. These observations will also lead to more efficient disease diagnosis, since pathological development begins at the microscopic level. Moreover, the spectral detection system will allow image segmentation of

[1] This thesis follows the style and format of the *Journal of Biomedical Optics.*

biological components, and the use of ultrashort broadband laser pulses characteristic to nonlinear optical microscopy (NLOM) will enable multiplexing experiments to be performed. Likewise, simultaneous excitation of multiple fluorophores will allow multiple molecular interactions to be viewed with one scan. Thus, the development of a spectral detection system based on NLOM will be a very important breakthrough in biomedical research.

PREVIOUS RESEARCH ON IMAGING

Research similar yet not identical to the idea of a spectral, three-dimensional non-linear optical microscopy has been conducted in the past. Some conventional modes of the non-invasive imaging of live tissues are the x-ray, the computerized tomography (CT scan), and magnetic resonance imaging (MRI). Though these methods are quite successful, they lack the ability to capture images on a microscopic level. As far as disease diagnosis is concerned, early detection is often crucial to patient survival; most pathologies begin development on a cellular (microscopic) level.

Recently, there has been much development in terms of non-invasive microscopic imaging (microscopy). Perhaps the oldest method (seen as early as 1962) is second-harmonic generation (SHG). SHG is a second-order nonlinear optical process that has constraints confining the signal to regions of the specimen that lack a center of symmetry.[1] The phenomenon of SHG requires intense laser light passing through a highly polarizable material with a noncentrosymmetric molecular structure and the second-harmonic light protruding from the material is half the wavelength of that entering. The SHG process thus changes two near-infared incident photons within the

material into one emerging visible photon at twice the energy.[2] Near-infared laser excitation minimizes scattering and absorption of the source; this optimizing imaging of tissues at greater depths.[3] Thus, SHG is successful in obtaining non-damaging (no stain required), non-invasive images in three-dimensions but currently lacks spectral information.

Similar to SHG, third-harmonic generation (THG) microscopy has recently been studied as a probe to characterize transparent specimens. Third-harmonic light is generated at the focal point of a tightly focused short-pulse laser beam. Like SHG, THG relies on inhomogeneities near the focal point (such as interfaces between two media); thus the symmetry along the optical axis is broken and a measurable amount of the third-harmonic is generated. THG can be used for specimens as thick as 3mm, but it only provides structural information. Both SHG and THG are often combined with two-photon excited fluorescence imaging so that a single laser source (such as titanium sapphire) may be used.[4]

Another form of microscopy recently explored is two-photon fluorescence (TPF) spectroscopy. This type of imaging has mostly been used to study certain nucleotides and flavoproteins to quantify cell metabolism. Though studying the fluorescence of

nucleotides and flavoproteins minimizes absorption of excitation by intrinsic chromophores, light scattering, and variations in mitochondrial density, it is desired for spectral information such as that obtained by this two-photon fluorescence spectroscopy imaging to enable study of overall chemical composition rather than merely that of cell metabolism.[5] Like SHG, current TPF involves use of up to two detectors (a maximum of two colors may be collected from the specimen at a time). TPF is often combined with SHG or another similar process called third-harmonic generation so that a single laser source (such as titanium sapphire) may be used.[4]

Likewise, optical coherence tomography (OCT) has been used much in clinical practice due to its noninvasive nature dating back to 1991.[6] OCT is similar to ultrasound but measures light rather than acoustic waves. In OCT, cross-sectional images are generated by scanning a tissue sample with an optical beam and measuring the echo time delay and intensity of backscattered light. This light is quantified by correlating it with light that has traveled a known reference path.[7] The light data obtained is then displayed at a logarithmic grayscale image (no spectral information provided).[8] Perhaps the key element for this type of imaging is an inferometer (radiation from a broadband light source is split into two parts: one directed onto the surface of the specimen and the

other launched to a reference mirror). Reflected beams from these two paths are them recombined to interfere, and this interference signal is detected by a photodiode (non-zero only when the two path lengths match within the light coherence length, thus providing very high sensitivity and spatial resolution).[9] OCT has been used extensively in dermatological diagnosis[9] and perhaps more so in ophthalmic imaging.[8]

The imaging modality of differential interference contrast is commonly used for studies on intracellular vescicle trafficking and cellular substrate interaction. After passing through a prism, the differences in amplitude of visible light result in enhanced contrast in areas close to the cell membrane and bound substrate. An image may then be reconstructed in conjunction with phase or reflection contrast microscopy with a somewhat low resolution of 0.3 μm.[10]

Another mode of imaging called confocal laser scanning microscopy has most recently been used for quality control of engineering tissues and nanofibers. In this type of imaging, light from a point laser is introduced into the sample and light of the same wavelength reflected back into the microscope objective is detected. The light thus is scattered at interfaces or medium transitions (from media to collagen fiber or cell membrane). This process is most successful with the use of reflective dyes or metallic

decoration of fixed samples.[10] The first account of time-lapse confocal reflection microscopy (CRM) was documented in 2000 by Brightman, et al. Specifically, CRM was used to determine and compare the 3D structure and assembly attributes of reconstituted matrices prepared with purified collagen or interstitial matrix components.[11] It proved to be a very sensitive technique, but this study was performed *in vitro* rather than *in vivo* and again only obtained intensity images.

Moreover, multiphoton microscopy (MPM) requires the quasi-simultaneous absorption of multiple photons (two or three) to fulfill the energy requirement for dye excitation.[12] However, since multiphoton excitation occurs only in a very small intratissue focal volume (on the order of about one femtoliter), with the use of a fast galvoscanner and piezodriven objective positioner, the position of the multiphoton excitation volume can be changed in three different directions in order to scan deeply into the tissue.[13] The actual image obtained consists of a matrix of fluorescence intensity measurements made by digitizing the detector signal as the laser scans the specimen.[14] Thus, again only intensity images are able to be obtained.

Specifically, one study briefly discussed by Peter Friedl in Summer 2004 deals with dynamic imaging of the immune system using various imaging techniques. Stimulated

emission depletion (STED) has been used to improve resolution in optical imaging in which a second laser pulse eliminates the outer edge of the first laser beam. This narrowing causes an improvement from radial and axial resolution of 200 nm and 300 nm, respectively, to below 100 nm in each direction. Fluorescence-resonance energy transfer (FRET) involves the transfer of photons from one fluorphore to a neighboring fluorescent molecule, selectively detecting molecular proximity (1-10 nm). Thus, FRET indicates when molecular partners engage with each other or dissociate.[15] Again, though these techniques improve conventional microscopy (STED) and can detect interactions between cells, again no spectral information may be obtained.

The most recent developments toward spectral detection have been accomplished by the companies Leica, Zeiss, and Nikon. Leica has developed an electronically controlled, freely definable Acousto-Optical Beam Splitter (AOBS). This beam splitter allows simultaneous imaging of up to four fluorescence proteins due to enhanced transmittance. Moreover, input ports for additional lasers, external detectors for non-descanned imaging, and an output port for coupling external devices (such as more detectors or spectrometers) have been added.[16] Zeiss has developed a META detector and method for metatracking that combines fast switching of incident laserlines with

simultaneous switching of the spectral readout from the detector. Thus, different spectral ranges of interest may be acquired by using electronic switches; this process is not entirely simultaneous (only one color or range of wavelengths may be obtained at a time).[17] Nikon's C1 Plus confocal microscope system allows four detector channels as well, but, unlike Zeiss, information from these four channels may be obtained simultaneously.[18] Though these instrument companies have developed methods for spectral imaging, only four wavelengths may be viewed, and much more detailed information about the specimen may be acquired if many more detector channels are incorporated.

PROBLEM

Although many features of optical biopsy have been previously explored and developed, one imaging method incorporating all desired attributes is yet to be developed. Thus, this research qualifies as being original and has not been previously attempted. Currently, in the lab of Professor Alvin T. Yeh at Texas A&M University, the main focus is on SHG and TPF. Commercially-available software is used to control the system and render images. However, the software limits this rendering to grayscale images, which provide no spectral information about the tissue specimen. The main focus for this research is to add hardware to the current lab setup and generate new software to enable the creation of spectral images.

Lab Setup

Currently, in the Tissue Microscopy Lab, the imaging system (seen in Figure 1) involves a laser beam focused to a point of approximately 1 μm^3 by a microscope objective, and its position is determined by the swivel angle of two galvanized mirrors connected to a computer. The current laser scanning pattern is seen in Figure 2. The actual microscope as seen in the lab is presented in Figure 3.

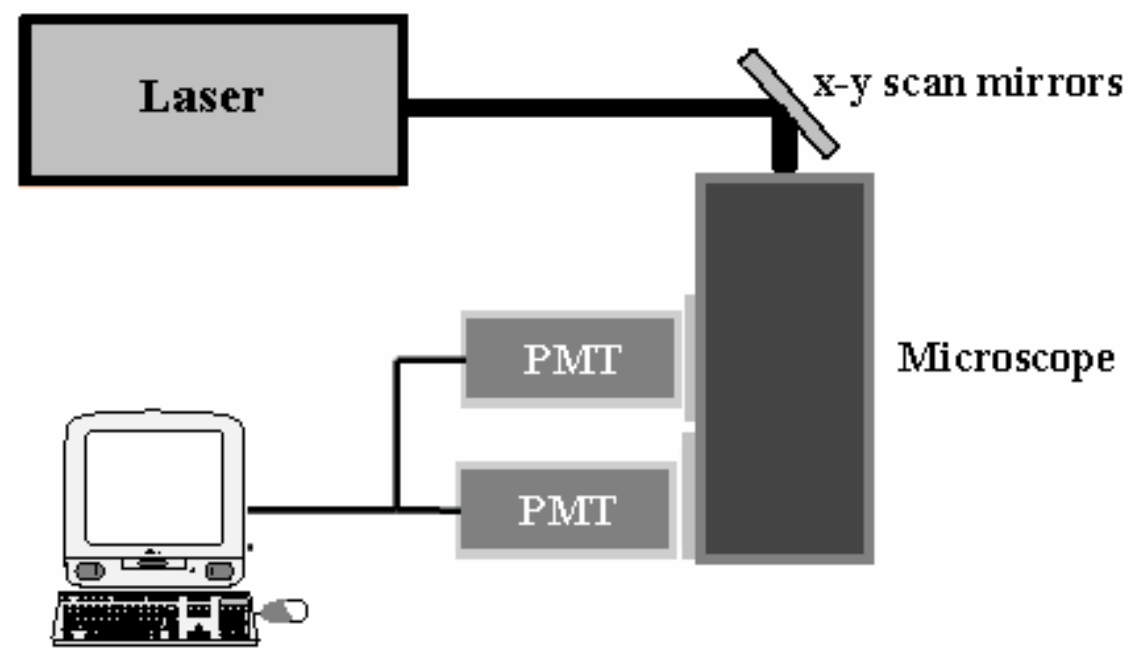


Figure 1: Lab equipment schematic

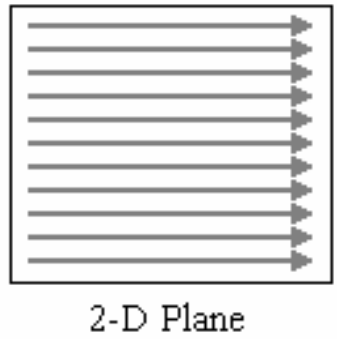


Figure 2: Current laser scanning pattern

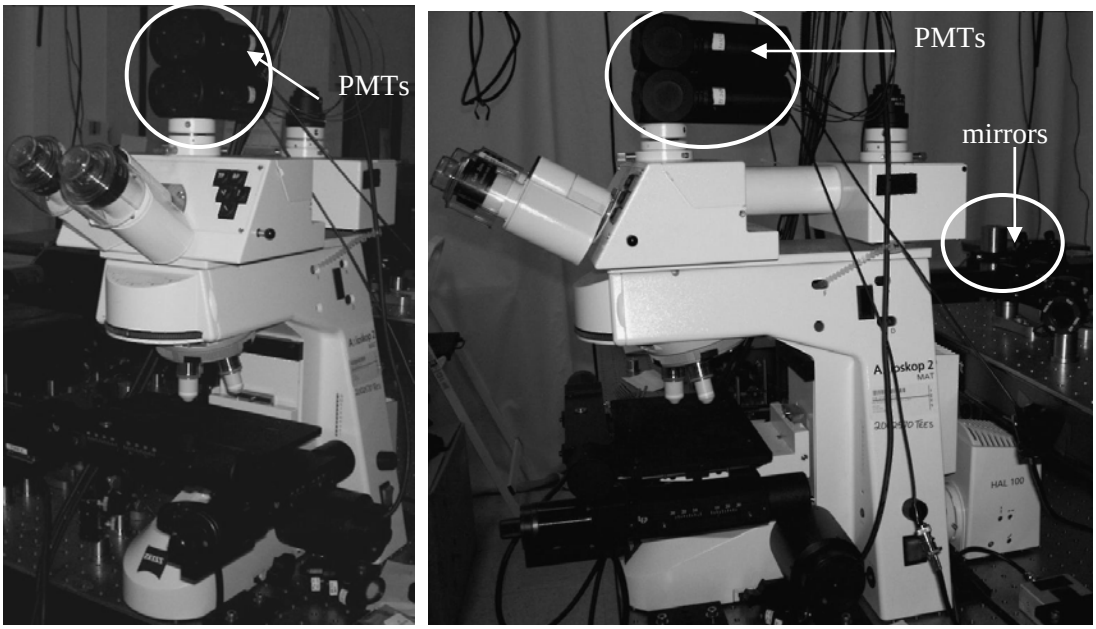


Figure 3: Front (left) and side (right) views of lab microscope

As the laser beam scans the object on the two-dimensional plane (assumed to be live tissue), photons are emitted by the specimen. These photons are detected by two photo-

multiplier tubes (PMTs) set on top of the microscope (Figure 1). A discriminator is used after each PMT that will emit a TTL pulse for each photon detected. Binary counter chips are used to count the number of TTL pulses for each image pixel. A digital-analog converter takes the resulting TTL-pulse count and converts this binary count into a voltage value. Therefore, at each pixel (a 256x256 pixel image will have 65,536 scan divisions), a voltage value will be recorded; the voltage value at matrix element (i,j) corresponds to the photon count. A color map may be applied to voltage values to render a false color image. In other words, the lowest and highest values in the voltage matrix will be set to two color extremes (black and white, for example), and matrix values between these extremes will be scaled accordingly to form a grayscale image. The image frame rate was programmed to be 1 Hz. The residence time per pixel is 1s / 65,536 or 15.259 μs per pixel.

Setup Limitations

Commercially-available software exists that controls the mirror swivel angles as well as information processing for up to two detectors. The main focus for this research is to ultimately add 30 detectors and modify each to sense a certain range of visible light wavelengths. With this spectral selectivity feature, various cell components may be viewed. Currently, the only method of viewing spectral wavelengths selectively is to render an image using a band-pass filter. For each wavelength chosen, another scan must be performed. For example, the image in Figure 4(a) shows a cell with its surrounding collagen matrix. Figure 4(b) shows the cell alone; this image was obtained by rendering a second image using a green (520 nm) filter. Likewise, Figure 4(c) showing the collagen matrix was obtained by taking a third image using a blue (400 nm) filter.[7]

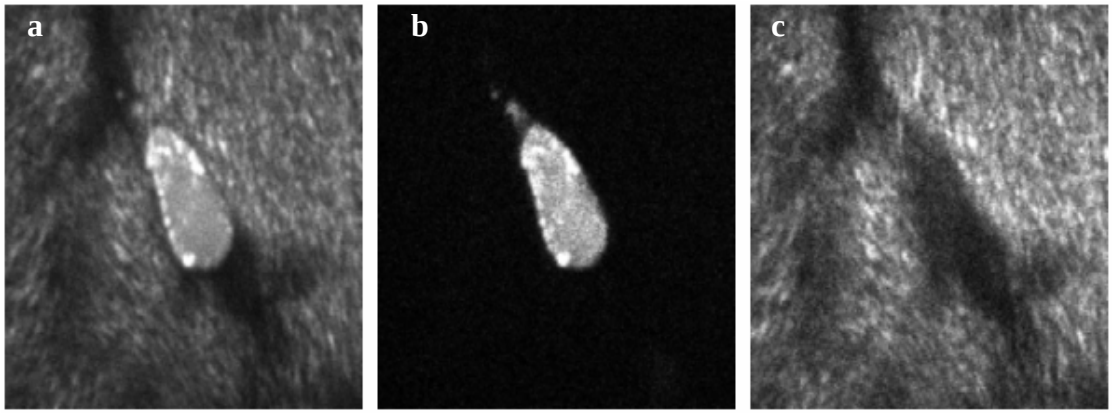


Figure 4: Effects of image filtering: (a) cell and collagen matrix without filtering; (b) cell alone imaged with green (520 nm) filter; (c)collagen matrix imaged with blue (400 nm) filter

Adding 30 detectors to the current setup will allow all of these views with the ease of only one scan rather than one scan per wavelength. This addition of detectors will require software creation and hardware modification.

METHODOLOGY AND RESULTS

For the sake of practicality, the creation process for this spectral detection system was divided into two separate modules: scanning and detection. All programming, interfacing, and testing of hardware and software was completed in the Tissue Microscopy Lab in the Wisenbaker Engineering Research Center of Texas A&M University.

Scanning: Basic Methods

The first aim completed towards the final goal of the project was the movement of the optical scanning mirrors. The initial task to be accomplished was the creation of multiple functions that would serve as appropriate mirror motion patterns. The mirror set used was Cambridge Technology's Model 6220 Galvanometer Optical Scanner (set of two mirrors – Figure 3). Each mirror is essentially a galvanometer in that it receives a voltage signal and converts that signal into a swivel angle or mirror position. Therefore, in order to scan an image in the pattern seen in Figure 1(b), one mirror needs to move the laser beam across the image while another mirror moves the beam down the image. If one "sweep" of a mirror is defined as a line across the image (each arrow in Figure 1b), the mirror responsible for the pixel rows will need a periodic function. The mirror needed to dictate the particular row of the scan will need only one sweep per scan.

Various periodic functions were possible for each mirror, including sinusoidal, raster, and triangular functions. Possible functions were designed first on paper and analyzed for simplicity and efficiency in order to select the best option. Since it is most convenient in most cases to deal with linear or quasi-linear functions, a triangular

function was chosen to be the periodic function of choice for the fast mirror (Mirror 1). A raster function was chosen for the slower-moving mirror (Mirror 2).

NI LabWindows/CVI was used to create and send functions to the scanning mirrors. LabWindows/CVI takes the programming language C and adds a graphical user interface (GUI) and many previously-defined functions. All programming was performed in C which allows for much greater user control and customization than that of a graphical program such as NI LabVIEW™. LabWindows/CVI was therefore used initially to create a program that would generate the aforementioned mirror waveforms and display them on the GUI. At first, the program included only user-defined parameters. This allowed the programmer to debug the program and discover how changes in the different parameters affected each created waveform. Once debugging was complete, basic calculations were performed to determine the ideal frequency and phase of each waveform. These values were then tested by being manually entered into the GUI. When the program correctly displayed the two mirror waveforms concurrently on one graph, the calculated frequency and phase of each waveform was set in the source code rather than being left as user-specified controls. The only controls left on the GUI were for the user to set the signal voltage range for each galvanometer.

The next step was the integration of the aforementioned program with the National Instruments PCI-6259 data acquisition (DAQ) card and SCB-68 68-pin shielded connector block. The card was properly installed and appropriate self-tests were run to ensure proper device communication. Once the card was properly tested, one connector block was connected to the DAQ with the included SHC68-68-EPM shielded cable. Two connector blocks will be needed when the detection portion of the project is attempted. However, for current purposes, only one connector block was connected to allow two analog outputs. The SCB-68 circuit diagram included with the device was used to select the analog output (AO) and ground pins. To test the outputs and eventually connect them to the galvanometers, wires were connected to the appropriate pins summarized in Table 1. Once connected, the wires were appropriately soldered to a male BNC connector (one connector was used for each analog output channel for a total of two connectors). The setup and connections are shown in Figure 5

.

Pin Number	Pin Name	Description
22	DAC0OUT	voltage output channel for AO channel 0
21	DAC1OUT	voltage output channel for AO channel 1
55	AOGND	ground reference signal for both AO channels

Table 1: Description of SCB-68 pins used for output signals

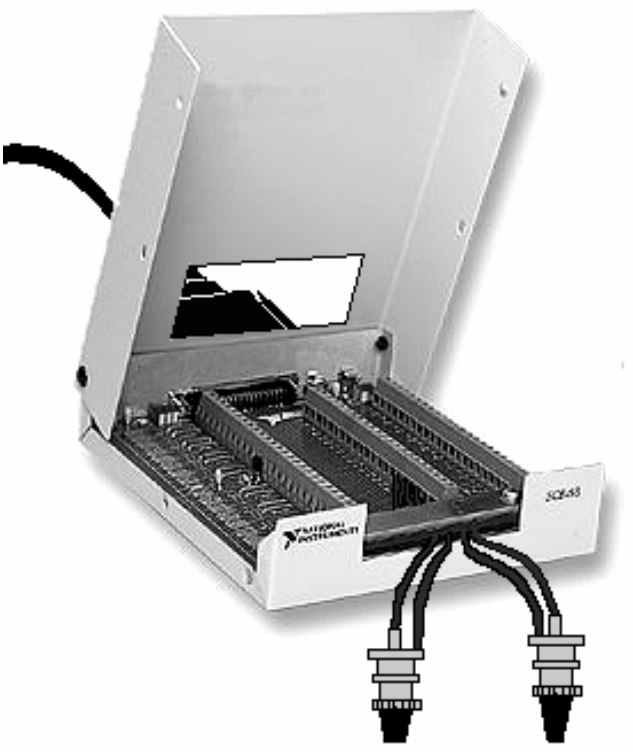


Figure 5: Connector block setup

Once the device was properly wired, the program created had to be modified so that the waveforms would be sent to the DAQ card rather than the GUI. Two waveforms

were created and stored in arrays; both arrays were combined and sent to a function responsible for taking an array and writing these values to the DAQ card as analog waveforms. A digital- analog converter (DAC) was thus intrinsic.

The source code and GUI were modified so that the program functioned more like the original commercial software for a two-detector system. The user was allowed to specify the voltage range, image size (for a 256x256- or 512x512-pixel image), signal output speed, scan mode (continuous or finite number of scans during each program run), and number of scans for the finite scanning mode. Moreover, the user can stop the scan sequence with a toggle switch on the GUI (once depressed, the scanning sequence will stop after the current scan is complete). The program also displays the estimated time (in seconds) for each scan to complete. To evaluate the program and to test whether it behaved as desired, the male BNC connector soldered to the output wires (Figure 4) was connected to a Tektronix TDS 3052B oscilloscope. Once waveforms similar to those seen in Figure 6 were observed, and all of the user controls functioned as planned, the program written was considered complete.

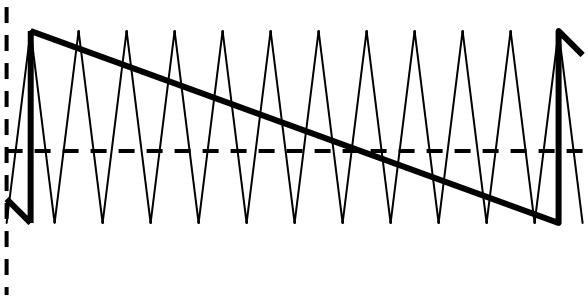


Figure 6: General mirror pattern waveforms (triangular – regular, and raster – bold);

note: frequencies not drawn to scale

Scanning: Results and Calculations

To make each waveform suitable for image scanning, a number of parameters needed to be determined. In deciding the most appropriate frequency and phase shift for each mirror function, a number of criteria were followed. As mentioned previously, an image acquisition rate of 1 Hz was programmed. For an image of 256 x 256 pixels, Mirror 1 would thus need to move back and forth 128 times to create 256 total rows across the image (256 pixels in the vertical direction); the triangle wave used for Mirror 1's position would thus need to have a frequency of 128 Hz. While Mirror 1 must move at 128 Hz, the raster waveform for Mirror 2 had a frequency of 1 Hz.

Though the time-domain frequencies were easily determined, converting into parameters for LabWindows/CVI functions required further calculations. The pre-defined triangle wave function in LabWindows/CVI, TriangleWave( ), was generated with Equation 1:

$$x_i = A * tri(\phi_0 + f * 360.0 * i), \quad \textbf{(1)}$$

where *tri( )* is defined by the waveform in Figure 7, x_i is the current array element, A is the amplitude of the triangle, ϕ_0 is the phase shift of the waveform, f is the frequency

factor, and i is the index of the current array element. LabWindow™ takes user-specified values for ϕ_0, A, and f to create an array of basic triangles; the net result of the array is a triangle waveform. A triangle waveform in LabWindows/CVI is thus created from a sum of scaled, shifted triangles.

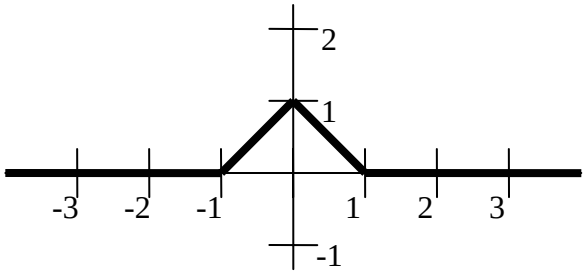


Figure 7: LabWindows/CVI triangle function (bold)

From Equation (1), the magnitude of the shift from triangle to triangle (the distance between adjacent triangles in the waveform, f*360.0) is equivalent to the fundamental period of the waveform, or

$$T_0 = 360\, f\ , \qquad \textbf{(2)}$$

where T_0 is the fundamental period of the triangle waveform. A desired Mirror 1 waveform frequency of 128 Hz would thus imply a fundamental period of 0.0078125 s. Using Equation (2), the correct f value was calculated to be $2.170*10^{-5}$. To obtain the

frequency for Mirror 2's raster waveform created with the SawtoothWave( ) function, the previously-calculated *f* for Mirror 1 was taken and divided by 256 because the triangle waveform moves 256 times faster than the raster waveform. This process yielded a value of $8.477*10^{-8}$. Once these parameters were incorporated into the program, the waves were viewed on a graphical control added to the GUI and verified for accuracy.

The phase for each waveform needed to be chosen such that the beginning of the fall of the raster scan lined up with the beginning of a rise or fall of the triangular scan waveform (Figure 6). The raster waveform would cycle 360^o in one scan because of its 1 Hz frequency. In the 360^o cycle of the raster waveform, the triangle waveform goes through 128 cycles. Therefore, each cycle of the triangle wave corresponds to 2.8125^o of the raster wave; to line up with the triangle wave, the raster wave had to be 90^o + 2.8125^o out of phase with the triangle wave. The reason for the 90^o lies in the fact that the default position (0^o phase shift) is for the midpoint between the peak and the valley of the waveform to lie at the y-axis. For the peak to be at the y-axis, a 90^o phase shift was necessary. The phase shift values chosen were -90.0^o for the triangle wave (so that the y-axis position corresponded to the valley of a triangle) and -185.625^o for the raster

waveform. The 5.625^{o} ensures that the raster will line up with the triangle wave because 5.625 is a multiple of 2.8125. This value also leaves some space in the beginning of the function plot; the large rise in voltage can be used later in the project as a trigger for the scan to begin.

The next parameter of concern in LabWindows/CVI for both raster and triangle waveforms was the number of elements for each array. When this value was entered into the GUI as a user-control, it was modified using methods to determine its ideal value for the purpose of this project. First, with the triangle wave, an arbitrary value of 1/360 was chosen for f so that the factor in Equation (1), f*360.0*i became i. With the aforementioned f value set, it was discovered that one full period of the waveform occurs when the number of elements is 360. Using the triangular waveform (f = 2.170*10-5 or $\frac{1}{46080}$), a full period of the waveform will occur when the number of array elements is 46080. Since one scan of a 256 x 256 pixel image will require 128 periods of the waveform, the number of elements should be 128*46080 or 5,898,240. To ensure that the waveforms are of appropriate length on the GUI, the actual number of array elements for each waveform was set to 6,000,000. Likewise, if the user chooses to create a 512 x 512 pixel image, the number of array elements for each waveform would be 12,000,000.

Basic conditional logic statements were set up to handle both cases (256 x 256 and 512 x 512 images). Therefore, the number of array elements for each size option was set in the source code rather than on the GUI. The complete LabWindows/CVI source code created for mirror motion and image generation can be seen in Appendix A.

Due to DAC constraints on the DAQ card, the highest sampling rate was found to be 2,000,000 samples per second. Since a 256 x 256 pixel image had 6,000,000 array elements, the sampling speed limit means that a 256 x 256 pixel image will take at least 3 seconds to obtain. Likewise, the creation of a 512 x 512 image will take at least 6 seconds. Moreover, there is an initial time delay that occurs because the computer must perform many loops with 6,000,000 – 12,000,000 iterations and deal with 6,000,000 – 12,000,000 element arrays. This hypothesis was tested by creating a very simple program that added two 12,000,000-element arrays. There was a somewhat significant delay (5-10 seconds) in program completion; when the number of elements in the arrays was reduced to 12,000, the program ran with almost no delay.

To overcome the time constraints of LabWindows/CVI, the same waveforms mentioned previously were created in NI LabVIEW™ with the Basic Function Generator pre-built virtual instrument (VI). A raster and a triangular waveform were

created with frequencies of 1 Hz and 128 Hz, respectively; these waveforms were sent through the DAQ card and viewed on the oscilloscope (Figure 8). The LabVIEW™ block diagram for this mirror motion program is seen in Appendix B.

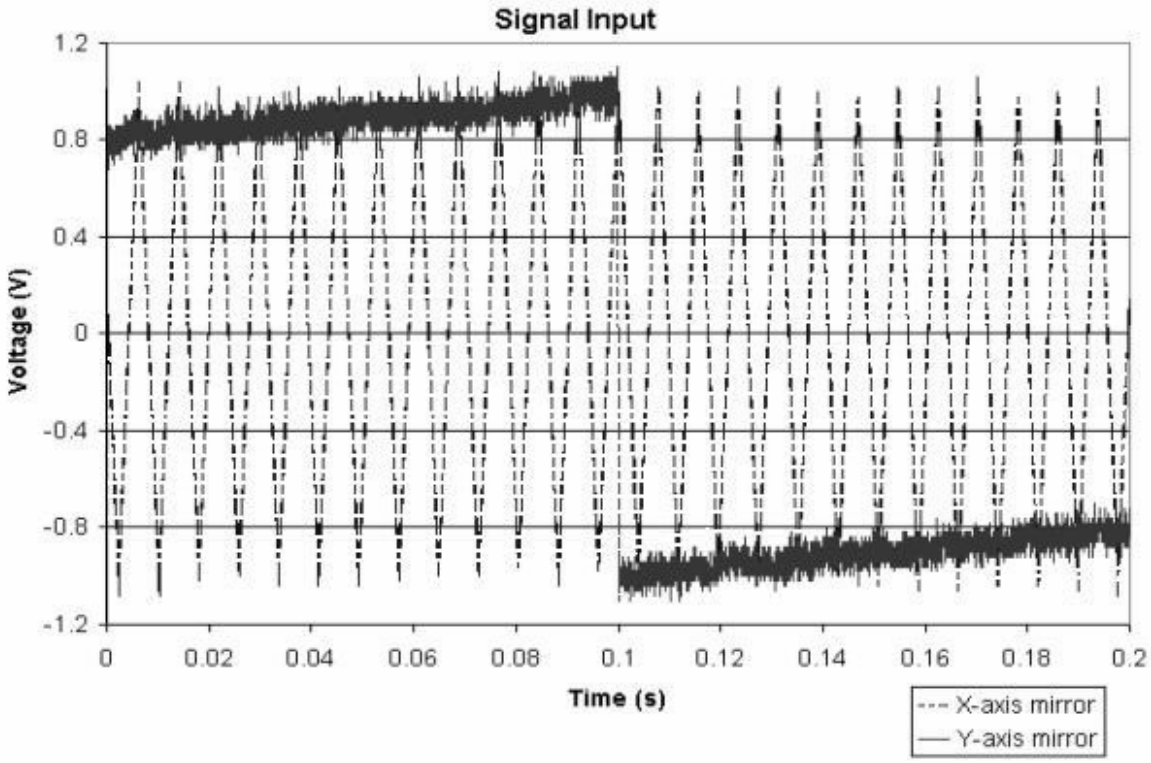


Figure 8: LabVIEW™ mirror motion waveforms

After verifying that the output waveforms behaved as expected, the two output BNC cables were disconnected from the oscilloscope and connected to the galvanometers. At

first, a very small output voltage amplitude was chosen (0.5 V) to place minimal stress upon the mirrors. Once very small motions were observed in the larger mirror (the smaller mirror was moving too quickly to observe with the naked eye), the voltage amplitude was increased to 1 V so that mirror motion was more easily observed. No major faults were found with the LabVIEW™ waveforms. LabVIEW™ proved to be easier to follow among multiple users as well as easier to program (the language is graphical rather than C-based). To verify correct mirror motion, the output of each mirror was connected via BNC-BNC connectors to the oscilloscope. The output waveforms in Figure 9 verify that the mirrors moved as expected. However, as seen when comparing Figures 8 and 9, a response of about 80% was observed in the mirror output waves. This small loss of voltage magnitude is probably due to internal impedance found within the galvanometers. The resistance of the BNC cables in this case is considered negligible.

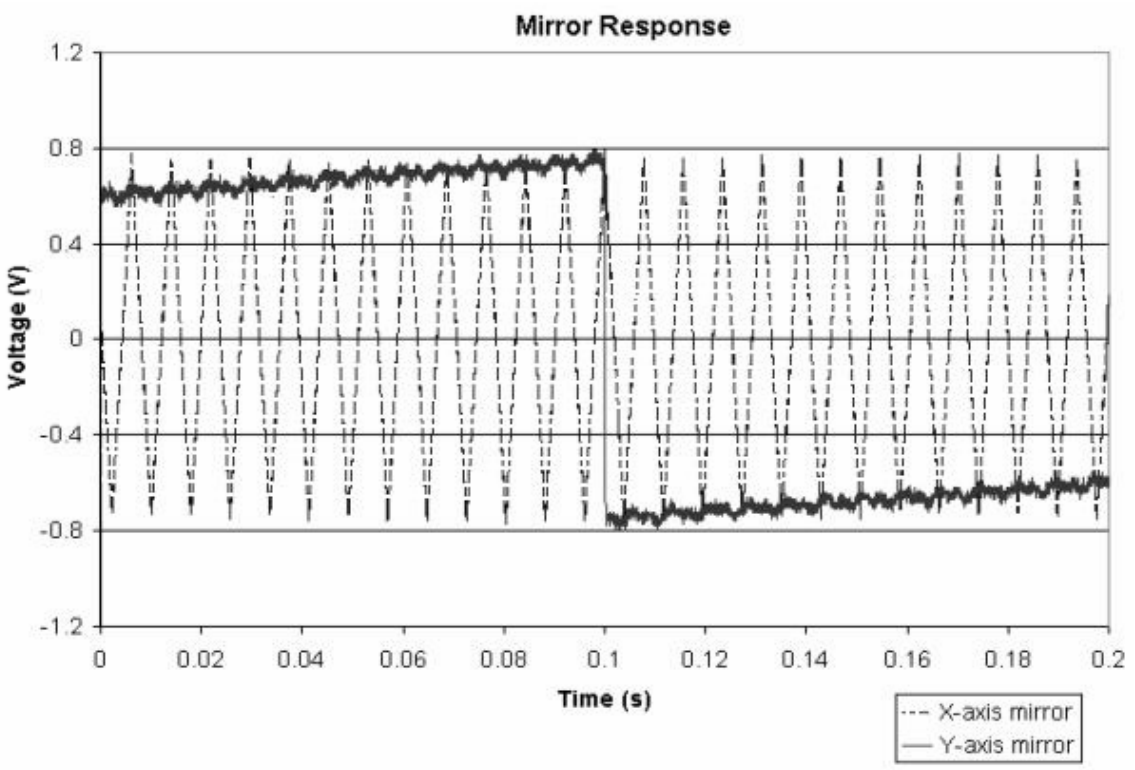


Figure 9: Actual mirror response to LabVIEW™ waveforms

Detection Hardware

For the detection portion of the spectral detection system, programming was again performed with NI LabVIEW™. The initial task completed toward spectral detection was the creation of data acquisition circuit boards. During detection, photons are represented by TTL pulses emitted by a discriminator. Four Texas Instruments SN74ALS161B binary counter chips were used to count the TTL pulses with 16-bit resolution. The TTL pulse output of the discriminator was sent into the least-significant bit of the counter chip setup. Thus, with each TTL pulse, the total count (the counts of all four counters added together) incremented by one. Since each counter contained four bits, maximum number of photons able to be counted at any given point in the tissue (or any pixel in the image) was $2^{16} - 1$, or 65535. To divide the image into pixels, the counters had to be reset at the aforementioned pixel resonance time, or once every 15 μs, so a 65,536 Hz square wave was generated with the waveform generator tool in LabVIEW™ and put into the clock input terminal of each counter chip. Therefore, as the laser beam scans the tissue on a two-dimensional plane, each photon detected by the PMTs is converted into a TTL pulse by a discriminator. Each pulse is then counted by

the counter chips; after 15 μs, the count is brought back to zero, and the detection process begins again at a slightly different laser position.

The total count after each 15-μs time interval then had to be sent into the computer so that image rendering could be performed. A Burr-Brown DAC712 16-bit digital-analog converter (DAC) was used to complete this task. All bit outputs from each counter (16 bits total) were connected to the 16 binary input terminals of the DAC. The circuit board setup containing the discriminator, binary counter chips, and DAC is seen in Figure 10. The actual circuit board built for data acquisition is seen in Figure 11.

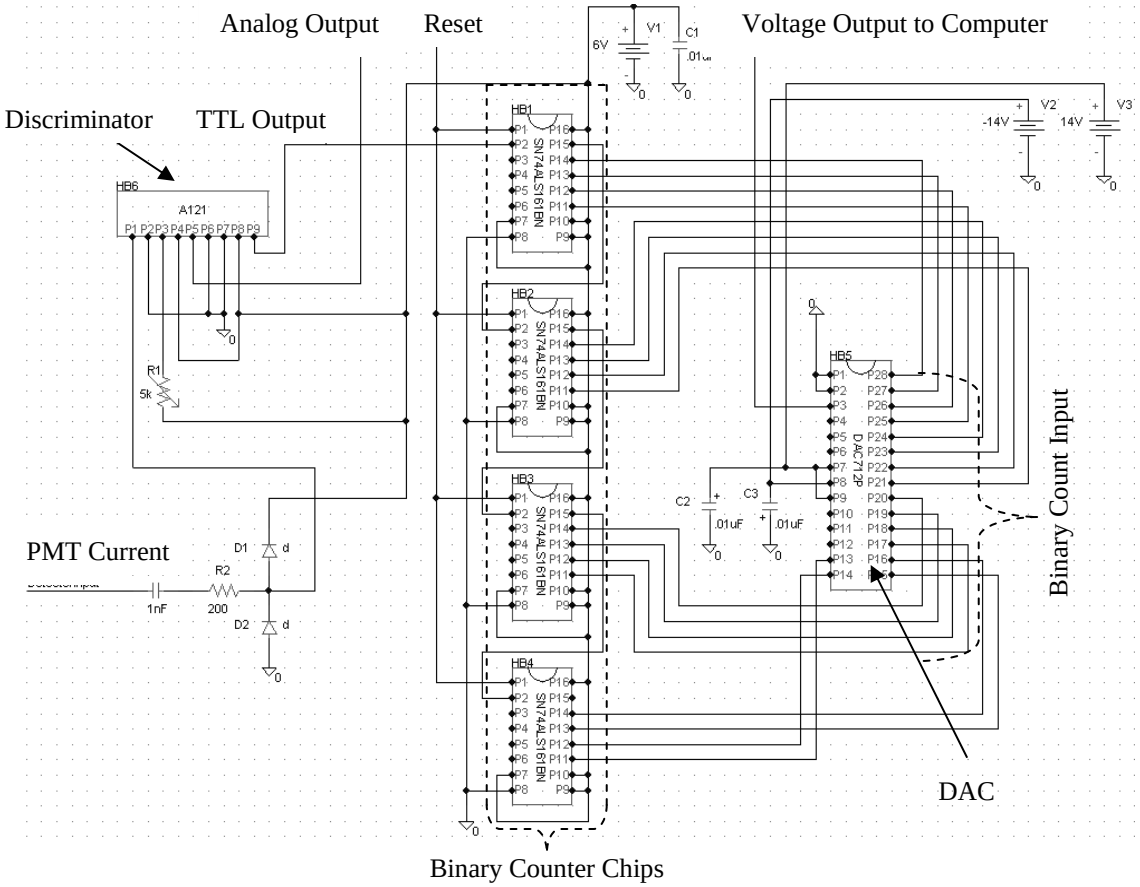


Figure 10: Detection circuit schematic

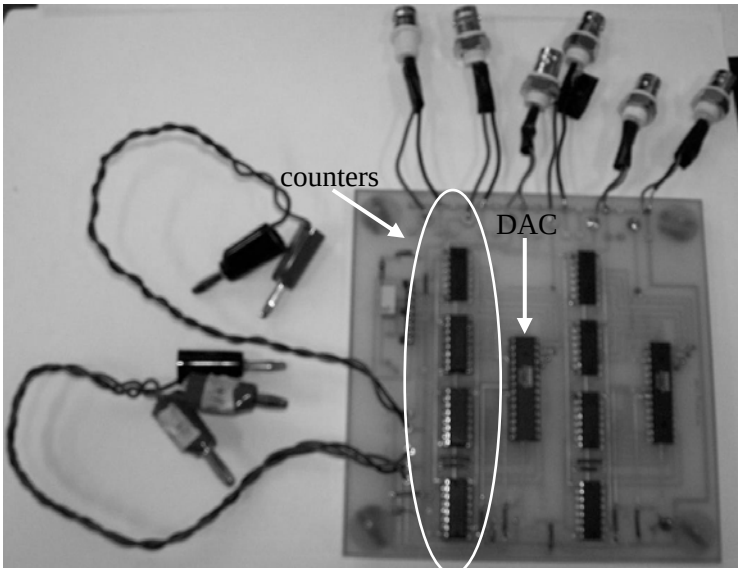


Figure 11: Actual detection circuit board (for two detectors)

The DAC takes the binary value seen at its input terminals, converts the value to a voltage, and outputs the voltages as a continuous waveform. Thus, as the count changes, the amplitude of the DAC voltage waveform will also change.

Once one circuit board was built that could count photons from two detectors, LabVIEW™ was used to create a program to use the photon counts at each "point" in the tissue to create a scaled intensity image. The first step toward the design of this program was to use the DAQ Assistant pre-built sub-VI to read the voltage waveform coming from the DAC. LabVIEW™'s DAQ Assistant sub-VI allows the user to specify the physical connector block channel from which to read data. The data range must also

be specified as well as the data acquisition mode and internal clock settings (number of samples to read and sampling rate). The data range for the DAC is -10 V to +10 V, so these values were specified in the program. The sampling mode in this case was chosen to be continuous, and modifying the clock settings ultimately changed the rate at which the program generated an image; the clock settings were changed multiple times throughout the design process such that adequate debugging could take place.

For testing purposes, an external 15-MHz TTL pulse waveform was connected to the clock terminal (CLK) of the least-significant counter. For the sake of simplicity, a basic up-counting circuit was used with the binary counters. A basic diagram for this circuit with labeled terminals as seen in the Texas Instruments specification sheet for the binary counters is seen in Figure 12.

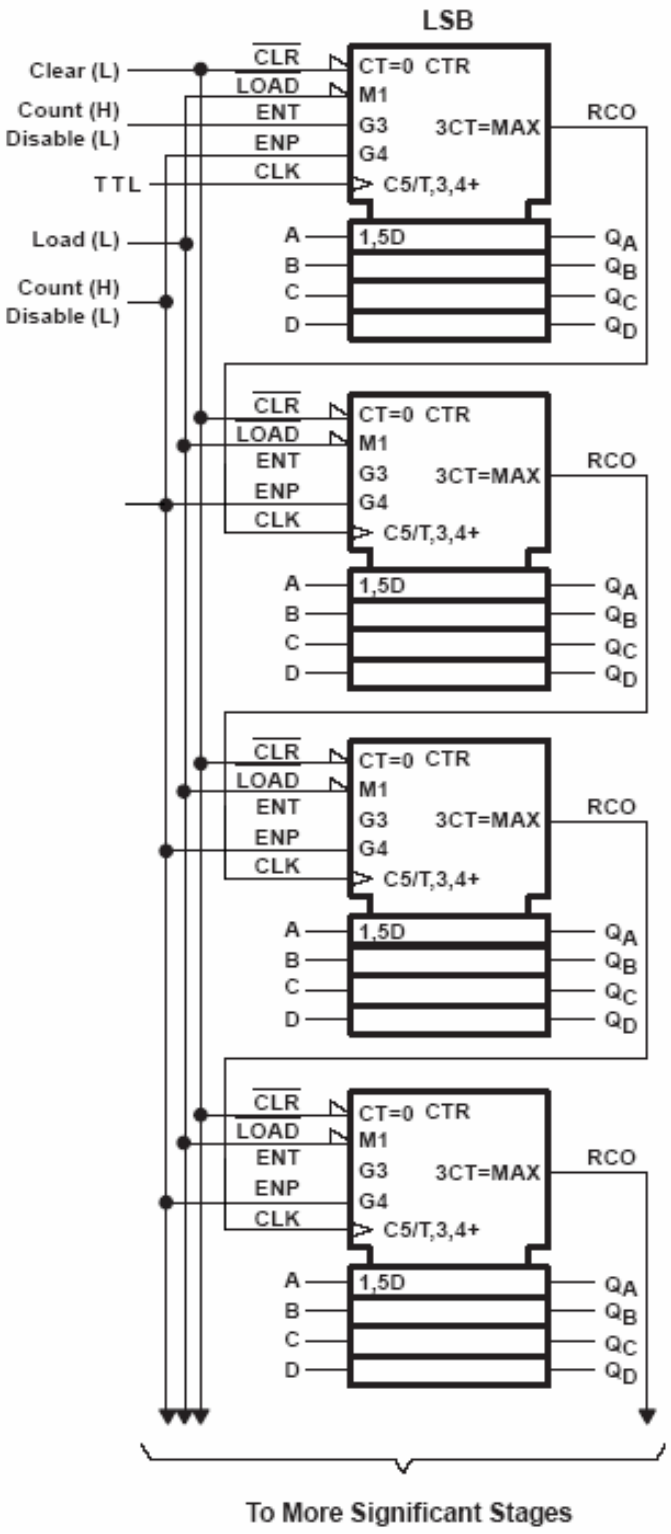


Figure 12: Basic count-up circuit for binary counters

The carry-out bit (RCO) from the least-significant counter (LSB) was connected to the clock terminal of the next counter; this connection pattern was repeated up through the most-significant counter; all counter output terminals (Q_A – Q_D) were connected to the input terminals of the DAC. Nothing was connected to the reset terminals (CLR) of the counters. Thus, the counters would count the TTL pulses until the maximum possible count was achieved (in this case 65,536), and at that point, all counters would reset to zero. The DAC would start at zero (with the counters), increase its voltage output to 10 V at a count of 2^{15} (or 32,768), jump to -10 V at a count of 32,769, and then return to 0 V for the next resetting of the counters. Since the computer was set to continuously read the voltage waveform from the DAC, and the DAC was set to receive a count that increased to a maximum and then returned to zero, the DAC the expected waveform to be seen on the LabVIEW™ front panel with the program running was a sawtooth wave. As seen in Figure 13, the resulting waveform from the DAC behaved mostly as expected. The voltage values from the DAC could easily have been converted into binary counts by dividing the voltage signal by 0.000305; this factor was found by dividing the maximum voltage range (20 V) by the maximum binary count (2^{16} or 65,536).

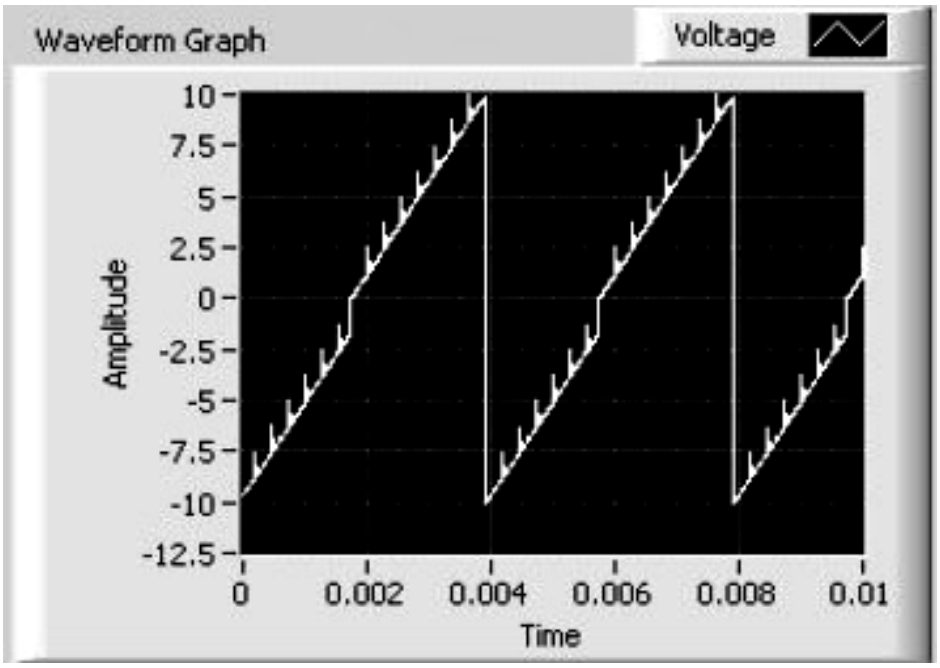


Figure 13: Voltage waveform for count-up circuit test (with 15-MHz TTL pulse input)

One unexpected effect observed in Figure 13 is the narrow "peaks" that arise throughout the voltage waveform. Ideally, the rising slops of the sawtooth wave produced during constant counting and resetting should be smooth and continuous. This problem was overcome by trying an alternative configuration for the binary counters; since they are often known for being more efficient and reliable, a carry look-ahead circuit was used. The basic configuration for this circuit obtained from the Texas Instruments specification sheet is seen in Figure 14.

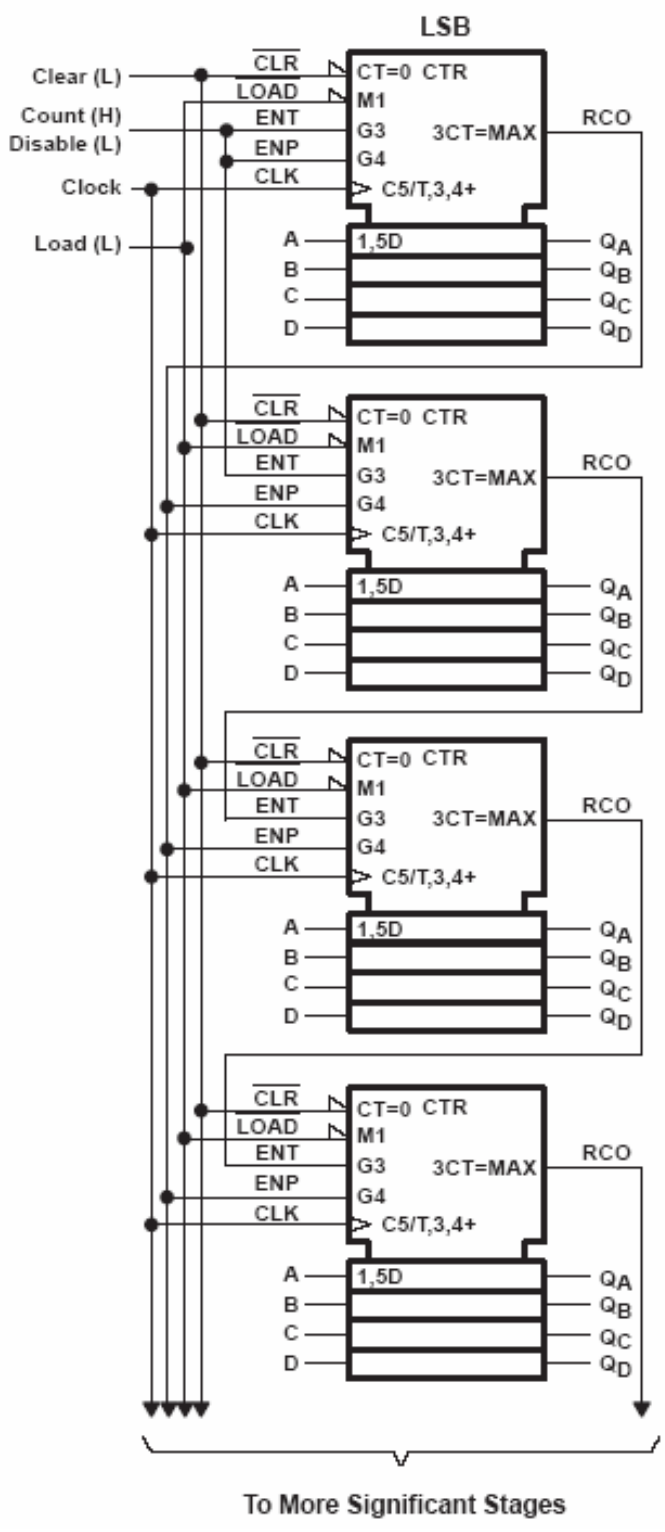


Figure 14: Carry look-ahead circuit schematic

For the carry look-ahead circuit, the RCO from the LSB counter was connected to the ENP terminals of the more significant counters. The RCO from the next counter was connected to the ENT terminal of the following counter, and the same connection pattern was repeated up to the most significant counter. For the LSB counter, the ENT and ENP terminals were connected together and then were connected to the ENT terminal of the next counter. All of these connections may be easily visualized in Figure 14. This circuit was outlined in the specification sheet for the counters.

When the same testing process was repeated for the new counter configuration, the resulting waveform was a smooth sawtooth wave; the waveform as seen on the LabVIEW™ front panel is shown in Figure 15.

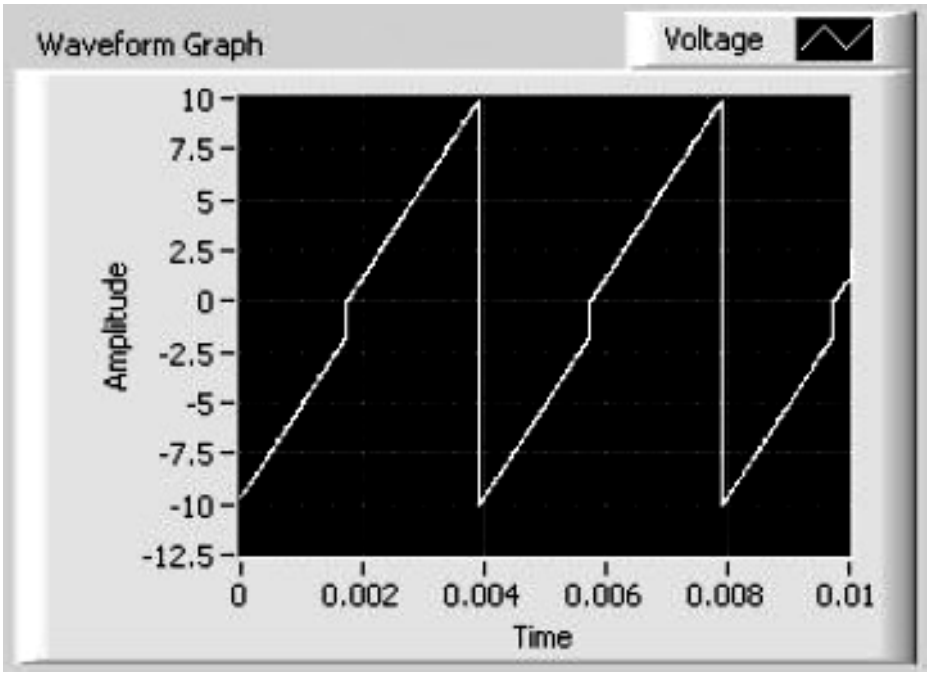


Figure 15: Voltage waveform for look-ahead circuit test

Since the waveform in Figure 15 appears as desired, the carry look-ahead counter configuration was used for all other circuit boards.

Detection Software: Saving Binary Files

The data collected from the tissue would be placed into a 256 x 256 matrix that would be converted into a 256 x 256 pixel intensity image. Since one desired feature for this spectral detection system is the ability to stack many 2-D images to form a 3-D image as well as view the tissue response to various light wavelengths, matrices created in LabVIEW™ needed to be saved as files that could be opened and converted into an image in MATLAB®. To perform this task, another program was designed to save each 256 x 256 matrix as a binary-format file able to be opened in MATLAB®. Binary format is preferred because of its space-efficiency. To perform the file-saving task, first a 256 x 256 matrix of numbers able to be set from the front panel was created using two nested for-loops. The LabVIEW™ sub-vi "Open_Create_Replace file" was used first to create a file with its name specified by the user on the front panel and then open that file. The reference location from the opened file was then sent to the LabVIEW™ sub-VI "save .MAT." This function saves a 2-D array of numbers to a binary MATLAB® file. The aforementioned 256 x 256 matrix was also sent to the "save .MAT" input terminal. The reference location of the file was then connected to the location input terminal of the

"rem .MAT" sub-vi. This VI will add a user-specified string remark to the beginning of the file; once opened in MATLAB®, the written remark may be seen at the top of the file. Currently, the remark input has been left blank. Finally, the reference location of the file was connected to the input terminal of the "Close File" VI; this VI, as its name suggests, closes the open file.

Since creating an entire 3-D image will require saving many 2-D images at a time, features were added to the program that would enable the user to specify the number of scans to have a certain file name. The file directory was specified, but the user is allowed to type in the desired name for the file sequence. For example, if a person is to take an image of a rat tail tendon, he or she might choose to take five scans of the tendon. So, on the front panel, he or she would type "tendon" or another descriptor into the appropriate text field, and the first scan would be saved as "tendon01." The final scan would likewise be saved as "tendon05." The scans would then be complete after five scans because the saving sequence was placed inside a for-loop that would iterate for the amount of times specified by the user on the front panel. The block diagram for the LabVIEW™ program used to save files in binary format can be seen in Appendix C. When the final program for photon counting and image generation is complete, the

matrix of photon counts will be connected to the input of the “save .MAT” VI in this program. The proper function of the program was verified by using MATLAB® to open and view some of the files created.

To view a 256 x 256 matrix of binary numbers as an image in MATLAB®, first the fopen(‘scan01.mat’) command was used to open the binary file (scan01.mat) created in LabVIEW™ for reading purposes. Additionally, the fopen( ) command creates a scalar MATLAB® integer valued double, called a file identifier. Next, the fread(file identifier, [A,B], ‘data type’) command was used to read the binary data from the specified file and write it into a A x B matrix of the data type specified. For instance, to store the binary values in the file scan01.mat in a 256 x 256 matrix of integers called A, the command needed would be fread(fileopen(‘scan01.mat’), [256,256], ‘int’). When a matrix was created in LabVIEW™ with all elements equal to zero, the resulting image was solid black, given the chosen grayscale color map. This image is seen in Figure 16.

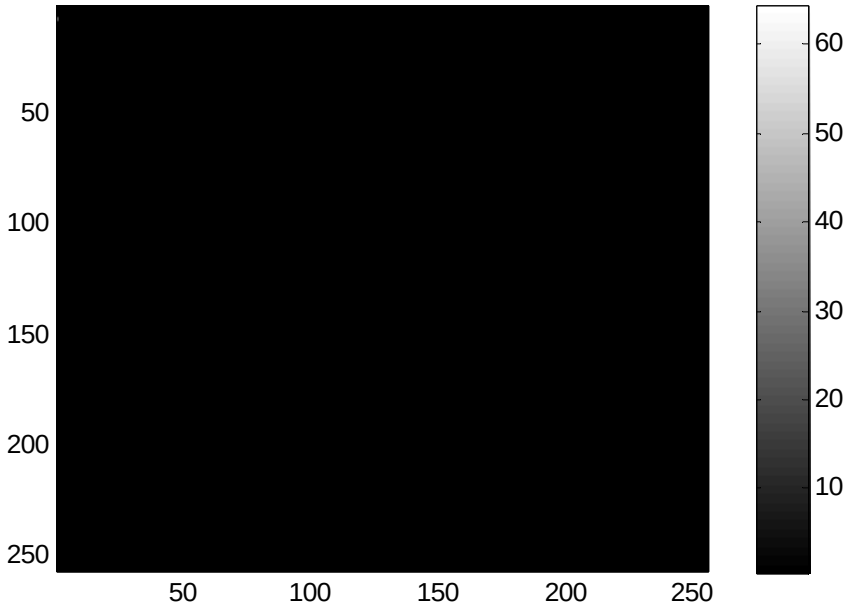


Figure 16: MATLAB® image of matrix with zeros

Since Figure 16 is, as expected, a 256 x 256 pixel image of all black pixels that represent a 256 x 256 matrix of all zeros, this program was considered a success. The LabVIEW™ block diagram for the program is seen in Appendix C.

Detection Software: Data Acquisition and Image Generation

Once the circuit board was tested and a program was created to save the image matrices as binary files, software was designed in LabVIEW™ to acquire photon counts from the circuit board and use these counts to render images. For data acquisition, a program in LabVIEW™ had to be designed to record the maximum count seen for every 15.259 μs into a 256 x 256 matrix. This matrix would then be converted into a scaled intensity image and displayed on the LabVIEW™ front panel.

The first step toward the creation of this software was to create 256 x 256 matrices of randomly-generated values that could be displayed as 256 x 256 pixel images on the front panel. The Replace Array Subset function in LabVIEW™ was used to create an array within a for-loop. For loop iteration, a different random value would be stored at each index of the array. The loop was set to iterate 256 times, and the iteration number was set equal to the array index. Thus, at the end of the for-loop iterations, the array would have 256 different random elements in it. Another for-loop created outside the aforementioned for-loop was then used to place each complete array of 256 elements into a matrix. This loop was also set to iterate 256 times, so at the end of the outside

loop's iterations, a 256 x 256 matrix of random values was generated. The Intensity Graph function of LabVIEW™ takes a 2-D array of values and creates a scaled intensity plot of those values, with each value representing one pixel in the plot. After all iterations of the aforementioned loops are complete, the 256 x 256 matrix of values was converted into an image with lighter pixels in the image corresponding to higher values from the matrix. One example of an intensity plot obtained from the above procedure is seen in Figure 17.

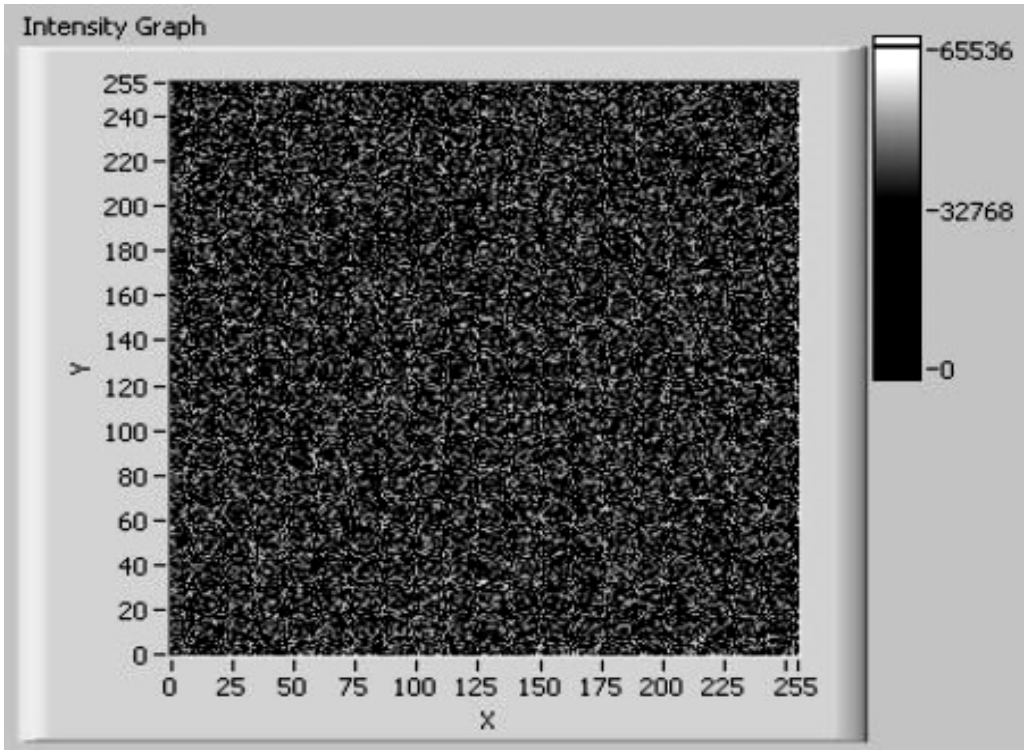


Figure 17: Image created with random matrix values

At this point, the program used for scanning and image generation was modified so that, rather than having random images created and a count signal that continuously maximizes and then resets, the system would count photons from tissue and use these photon counts to create an image. Basically, during each 15.259 μs time interval (for each pixel in the image), the photons emitted from the tissue at that specific point would gradually increase. Thus, it is expected that the maximum photon count at each position in the tissue would be achieved just before the counters reset and the laser moves to a new position. Therefore, the expected waveform to be seen from the DAC output is very similar in shape to that seen in Figure 15. However, since the photon count at each position within the tissue should be different from the surrounding areas, the magnitude of the waveform should vary somewhat from peak to peak. The values chosen in the DAQ Assistant used to read the voltage waveform from the DAC depend on the desired time for each full scan. Equation 3 can be used to calculate the Rate parameter using the number of samples to read for each line of the image and the amount of time needed to generate each line:

$$Rate[Hz] = \frac{SamplesToRead}{LineGenerationTime} = \frac{SamplesPerLine}{\left(\frac{ImageTime}{lines}s\right)}, \quad \textbf{(3)}$$

where *ImageTime* for a one-second scan would be 1. Likewise, if the desired time for a scan to be performed is four seconds, *ImageTime* would be 4. The *SamplesToRead* parameter represents the number of samples to read per line (*SamplesPerLine*). The *lines* parameter represents the number of lines in each image (i.e. the number of rows for each image matrix). For example, a 256 x 256 pixel image would require a value of 256 for *SamplesPerLine* and another value of 256 for *lines*.

Initially, the peak detector sub-VI in LabVIEW™ was used to find the peak value for every group of samples. The number of elements in each group was determined by finding the size of the data array with the LabVIEW™ Array Size sub-VI and then dividing that number by the desired size of the image array, or 65,536. The peak detector VI, "NI_AALPro.lvlib:Peak Detector," finds the location, amplitude, and second derivative of peaks or valleys within each group of elements in the input array (in this case, the voltage signal from the DAC); the number of elements in each group is specified by the user at the "width" terminal. Due to noise found within the DAC signal, the peak detector did not always find the peak at the correct location. Peaks did not always seem to be found just before the counters reset and the voltage returned to 0. The

same VI was then used with the valley detection option chosen. When a valley was detected, its location was sent from the peak detector VI into a for-loop. The loop contained the Index Array sub-VI that takes the original array (the DAC voltage signal converted from dynamic data to an array) and returns the element at the location specified by the number at the index terminal of the sub-VI. The location of the valley (the array index) was sent through the for-loop, one was subtracted from it, and the final answer (valley location – 1) was sent to the index terminal of the Index Array sub-VI. Thus, the sub-VI will return the element in the location directly before the valley. Each element returned from the for-loop throughout the entire DAC voltage array was assumed to be the maximum value for every 15-μs time interval. These values were divided by 0.000305 to obtain the actual photon count and then placed in a 256 x 256 matrix using the Build Array Subset sub-VI. The resulting matrix was then converted into an intensity image and displayed on the front panel with the Intensity Graph sub-VI. The resulting LabVIEW™ block diagram for image generation and data acquisition using the peak detector sub-VI can be seen in Appendix D.

Ultimately, after each matrix, or image, is created in LabVIEW™, the aforementioned program used to save the images as binary files will be incorporated into

the image generation program. The matrix used to form the image on the LabVIEW™ front panel in the image generation program will also be connected to the input of the "save .MAT" sub-VI.

Testing of Hardware and Software

To test the circuit board and LabVIEW™ software for mirror motion, data acquisition, and image generation, the current lab setup seen in Figure 1 was modified to include the new hardware and software. However, before the connections were changed to incorporate the new hardware and software, an image of a mouse tail tendon was generated with the current image acquisition system as a basis for comparison (i.e. a control image). Then, the output of the discriminator attached to each PMT was connected to the least significant bit of the sets of binary counters on the circuit board. The DAC output terminals from the circuit board were then connected to the appropriate terminals of the connector block (Figure 5), where the DAC signal would be read by the LabVIEW™ software. Other terminals of the connector block were connected to the scanning mirror input so that the LabVIEW™ software could control the motorized mirrors as mentioned previously. The setup for testing including the circuit board, connector block, oscilloscope, and computer is seen in Figure 18.

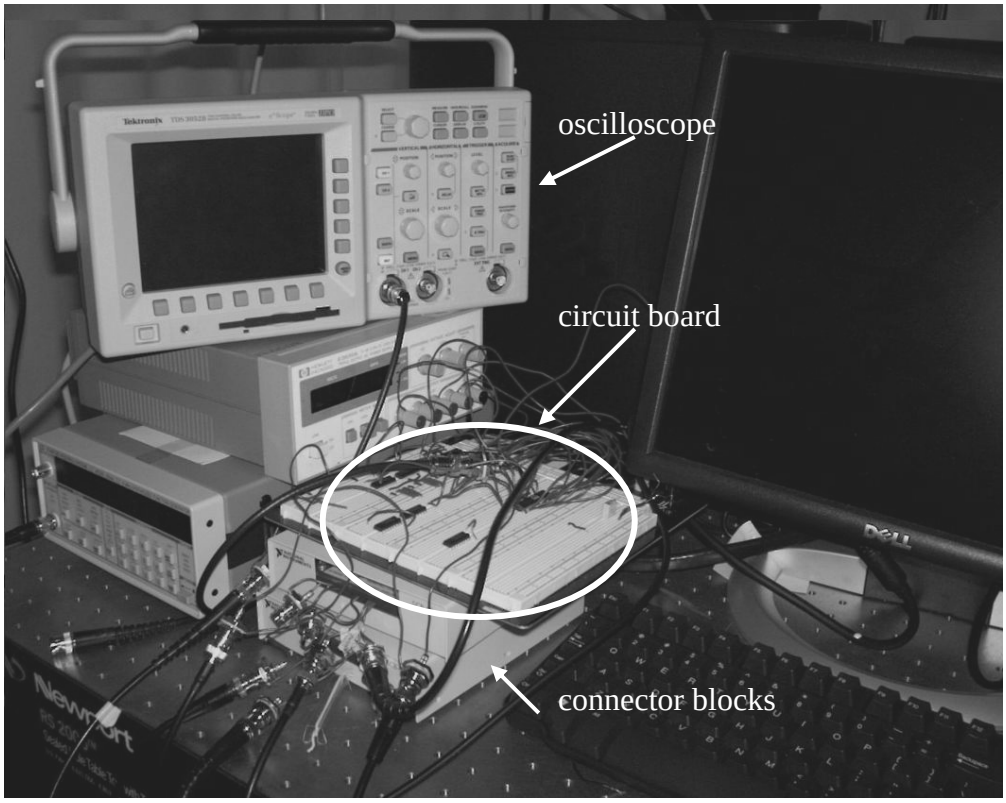


Figure 18: Setup for hardware and software testing

Limitations

Images were rendered and displayed on the LabVIEW™ front panel with the new software and hardware without any LabVIEW™ syntax errors. The control image of the mouse tail tendon is seen in Figure 19, while the image generated using the LabVIEW™ software is seen in Figure 20.

Figure 19: Control image of mouse tail tendon

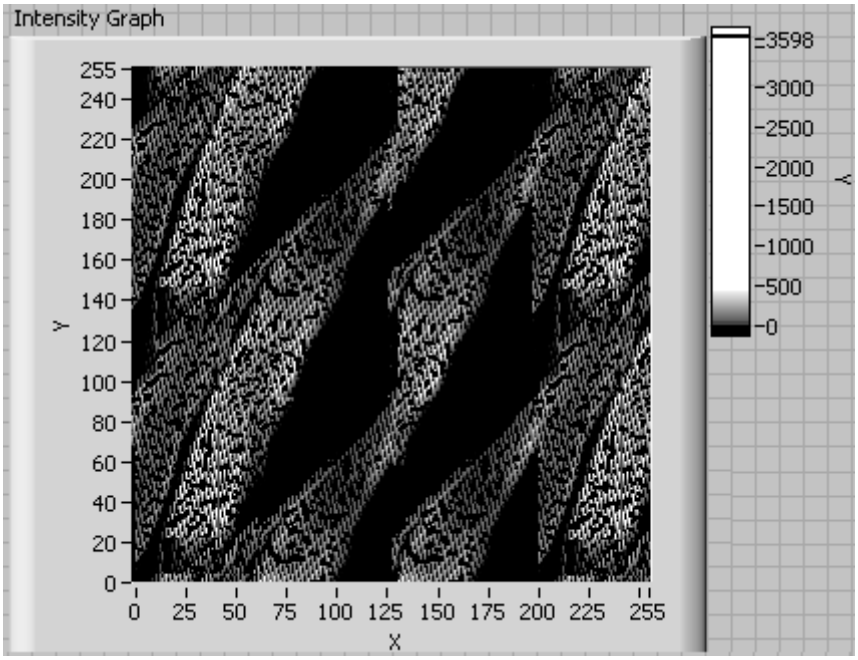


Figure 20: First image generated with software

As seen in Figure 20, much of the striated pattern seen in Figure 19 had been correctly duplicated. However, many discontinuities were seen throughout the image, and the image also appears to be separated into different parts that have been incorrectly organized. Rather than one continuous image of the mouse tail tendon, it appeared as though the image had been separated into about eight different parts that had been rearranged.

To begin to solve these issues, different components of the imaging system were analyzed individually to more easily isolate the main cause of the problems. The

oscilloscope was used to view input and output signals of the system one at a time, and signal parameters such as the amplitude, shape, and frequency were verified. First, the scanning was debugged by viewing the mirror response waveforms as the system generated images. It was noted that there were some fluctuations in the waveform amplitudes that were not present when the scanners were previously tested without the data acquisition and image rendering software running. A sensible explanation for this phenomenon was that the mirror wave generation portion of the program had difficulty sending out continuous data when other CPU-intensive processes were ongoing. The amplitude fluctuations in the mirror motion waveforms could be causing some confusion in the software about the actual laser position in the tissue at a given instant; therefore, the pixel intensities in the image may not accurately resemble the photon counts from the tissue. One way to overcome the issue of CPU processes interfering with the mirror waveforms was to use LabVIEW^{TM} functions to store the mirror waveforms, wait a small amount of time, and then send the waveforms out of the computer; this task division will ensure that a waveform will be stored and ready to be sent to the mirrors at all times throughout scanning. One method of pre-storing waveforms was to use a LabVIEW^{TM} buffering program rather than the waveform generation sub-VIs alone.

Specifically, the pre-built Waveform Buffer Generation sub-VI was used for the waveforms to be sent out from the computer (i.e. the two mirror motion waveforms and the 65.536 kHz square wave for resetting the binary counters. The Waveform Buffer Generation sub-VI has terminals for the user to designate the waveform type, the desired frequency and amplitude, and the square wave duty cycle. In this case, the 65kHz square wave was generated by the sub-VI, and the sub-VI then returned the waveform along with the actual, coerced, sample clock rate. This rate was then used as an input to the Basic Function Generator sub-VI to generate the mirror motion waveforms. The resulting waveforms (triangle, raster, and square) were then stored in an array. The task timing was then manipulated by adding a while-loop that waits a multiple of 100 milliseconds to begin the actual task of generating the waveforms. This waveform generation method waits a short time to enable buffer generation, and during continuous runs of the program, waveform and buffer generation for future waveforms occur simultaneously. The program was run with the data acquisition and image generation program, and the mirror output waveforms were again checked for amplitude discrepancies. The waveforms appeared very similar to those in Figure 8; the amplitudes remained constant throughout the trial. From this point, the new mirror

motion program including buffer generation was used for testing. The LabVIEW™ block diagram for the updated mirror motion VI is seen in Appendix E.

After the scanning mechanism was assessed and the system was tested again, the image appeared to be the same as that seen in Figure 20. Next, the detection portion of the program was debugged. The initial steps taken in the detection analysis was to study the images generated as they appeared on the LabVIEW™ front panel, rather than immediately changing the program code. As in the current image acquisition system, the images generated with the new imaging system should update after each scan. As expected, with a fixed tissue sample being imaged, the images generated should remain roughly the same with each scan if the laser focus depth is kept constant. However, even with a constant focus depth, the components in each new image seemed to shift relative to that of the preceding image. These phenomena meant that the mirrors were not synchronized to the data acquisition portion of the program. In other words, the mirrors did not begin scanning at the time that the program began to record photon counts.

To solve the synchronization problem, the mirror waveforms need to be saved as files in the mirror generation sub-VI shown on the block diagram for the data acquisition and image generation VI; once the mirror waveforms are saved, the file can be opened,

and the waveforms will be sent to the mirrors at the beginning of data acquisition within the LabVIEW™ program. Also, to avoid timing issues in which the mirror waveform file may be opened before it is created, rather than having the mirror motion program incorporated as a sub-VI, the entire mirror motion block diagram must be manually copied and pasted directly into the data acquisition block diagram. If the mirror motion block diagram is placed in the same loop as the data acquisition algorithm, the mirror motion and data acquisition will be synchronized.

The data acquisition and image rendering VI was then modified such that the peak detector was no longer necessary. Rather than detecting the valleys in the data and then storing the previous array element, for every given number of samples in the data array, one sample would be sent to the final image array. This approach assumes that the maximum value for every 15 μs was evenly spaced from other maxima in the array. To correct this, only elements that are a multiple of eight will be stored in the final image array. This algorithm will need further testing to determine the actual spacing (in terms of array elements) for each maximum.

To enact this change to the data acquisition and image generation VI, the data array from the DAQ Assistant was sent through a LabVIEW™ formula node. C-based code

was created to perform certain operations on variables defined at the borders. The code basically used a for-loop to go through the entire data array and store every element that was a multiple of the number specified. For example, the formula node seen in Figure 21 creates an array for matrix variable Y with 256 elements. The variable X is the incoming data array from the DAQ Assistant, and the variable "size" represents the size of the data array. The for-loop basically takes every 64th element from the data array and stores it in matrix Y. The loop terminates once its index reaches the value specified by "size."

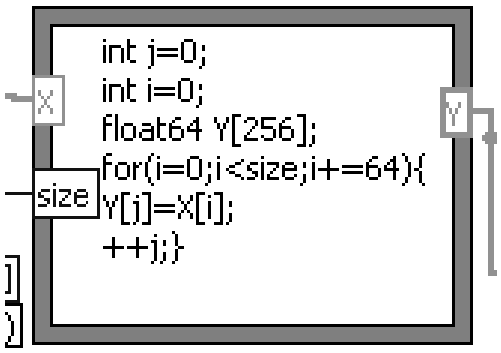


Figure 21: Example of LabVIEW^{TM} formula node

After the matrix Y is created in the formula node, another for-loop is used in LabVIEW^{TM} to divide each element in matrix Y by the aforementioned factor 0.000305; this operation converts the DAC voltage values to actual photon counts. As in the previously-built VI for image generation, each matrix of photon counts is displayed on

the LabVIEW™ front panel with the Intensity Graph sub-VI. The modified data acquisition and image generation VI is seen in Appendix F.

Again, the system was tested with the improved image generation software. This time, as the program ran, the value used to store the maximum data value for every 15 μs was changed slightly, with the initial value determined by the aforementioned method of dividing the data array size by 65,536. The image generated is seen in Figure 22.

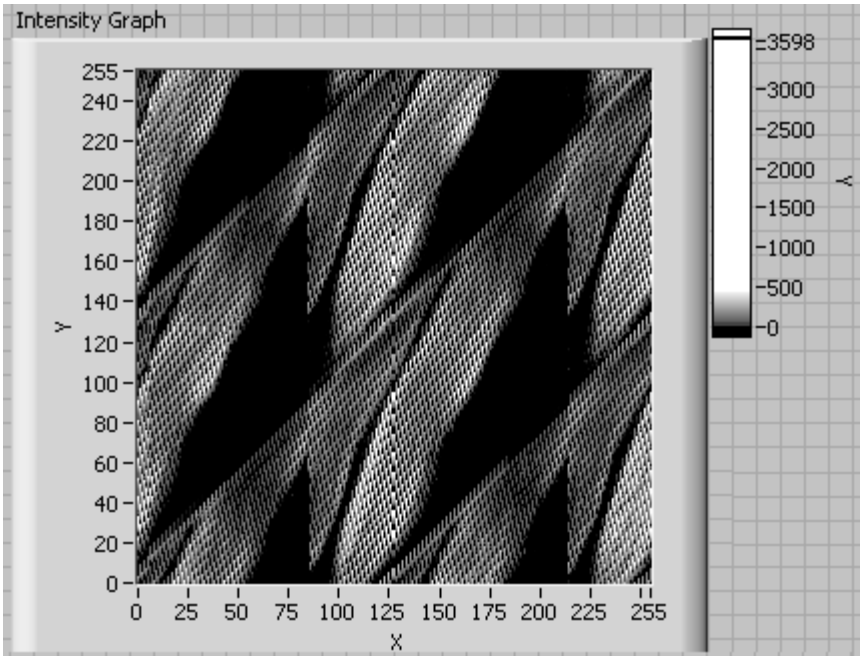


Figure 22: Image generated with modified software

Even when this method was carried out, the image still appeared disorganized. Another limitation was discontinuities observed within the image, though the discontinuities were less pronounced than those seen in Figure 20. The image with a limited viewing window of 10 pixels by 50 pixels is seen in Figure 23.

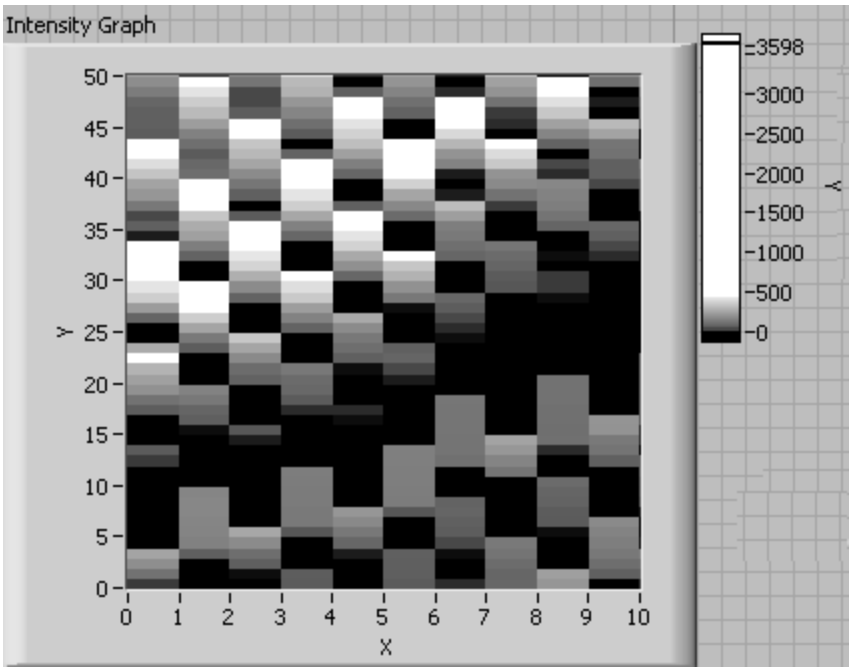


Figure 23: Generated image with limited viewing window

As seen in Figure 23, these discontinuities appeared as black pixels adjacent to colored pixels. When a similar viewing window was used with the control image, the

pixels appear much more continuous. The control image with a smaller viewing window is seen in Figure 24.

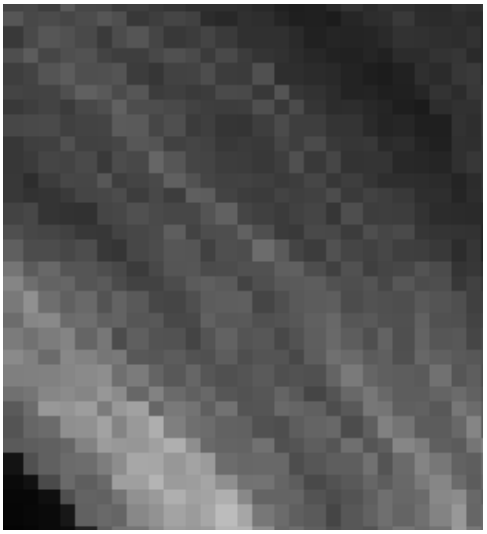


Figure 24: Control image with limited viewing window

Discontinuities arose from incorrect formation of the image matrix in the data acquisition and image generation VI. The data acquisition portion of the program appeared to be “skipping” some of the peaks in the DAC signal. Thus, another method must be developed to store the highest photon counts into a matrix.

To solve this image generation problem, longer scan times need to be used to ensure that the counters have enough time to reset. If the maximum voltage does not occur

every 15 μs, the algorithm cannot be used, and an alternative method for determining the photon counts for the image matrices will need to be developed.

FUTURE WORK AND CONCLUSIONS

A complete spectral detection system remains unfinished at this point, even though much progress has been made toward its completion. Overall, the software for the movement of the scanning mirrors was complete, and one circuit board for two detectors was correctly built and made to interface the LabVIEW™ software with the discriminators to enable photon counting. Future work on this project will include debugging and rectifying the data acquisition and image generation software. Once the new LabVIEW™ software and circuit board hardware create an image comparable to the control image created with the lab's current hardware and software, the new setup will considered successful with two detectors. An additional 15 circuit boards will be needed for the spectral detection system. Additional features in software may be needed to account for possible cross-talk between detectors.

Once the system is expanded to incorporate 32 detectors, a set of 256 x 256 x 32 matrices will be stored in binary files. Each set will be opened in MATLAB® and converted into a 3-D grayscale image by stacking all of the 2-D images for each detector. Therefore, 32 different 3-D grayscale images will be created for each scan.

The spectral response of tissue may be viewed separately, or all 32 channels may be integrated to form a grayscale image. Post-image processing may be used to view certain spectral components of interest.

In conclusion, the new spectral detection system will introduce many new features to conventional nonlinear optical microscopy. The introduction of spectroscopic information from live tissue will provide insight into its underlying chemistry. Furthermore, spectral detection allows image segmentation of biological components. For example, the collagen matrix of tissue may be selectively imaged by displaying channels corresponding to SHG. Many different biological components may be simultaneously viewed within each scan.

Because of its unusual features, this system will have a wide range of applications in both science and medicine. Simultaneous excitation of multiple fluorophores will allow multiple interactions to be viewed within a single scan. Likewise, in medicine, chemical markers characteristic to cancers and other cellular aberrations may be viewed microscopically. This microscopic imaging capability may lead to earlier diagnoses, and timing becomes extremely crucial with fast-spreading abnormalities such as cancer. Moreover, temporal gene expression profiling may be accomplished by injection of

different genetic markers into the DNA of a parent cell. With the spectral detection system, a number of genetic markers may be viewed simultaneously within a single scan.

The aforementioned examples are but a few applications of this spectral detection system. Further modifications of the spectral detector, such as the addition of an optical fiber or an endoscope, will enable interrogation of specimens, such as small animals, not amendable to a microscope stage. Once the spectral detection system is perfected, NLOM capabilities can be expanded to include longitudinal studies of *in vivo* systems.

REFERENCES

1. A. O. Brightman, B. P. Rajwa, J. E. Sturgis, M. E. McCallister, J. P. Robinson, and S. L. Voytik-Harbin, "Time-Lapse Confocal Reflection Microscopy of Collagen Fibrillogenesis and Extracellular Matrix Assembly In Vitro," *Biopolymers*. 54, 222-234 (2000).

2. P.J. Campagnola, A. C. Millard, M. Terasaki, P. E. Hoppe, C. J. Malone, and W. A. Mohler, "Three-Dimensional High-Resolution Second-Harmonic Generation Imaging of Endogenous Structural Proteins in Biological Tissues," *Biophysical Journal*. 81, 493-508 (2002).

3. P. J. Campagnola and L. M. Loew, "Second-harmonic imaging microscopy for visualizing biomolecular arrays in cells, tissues, and organisms," *Nature Biotechnology*. 21(11), 1356-1360 (2003).

4. M. E. Dickinson, E. Simbuerger, B. Zimmermann, C. W. Waters, and S. E. Fraser, "Multiphoton excitation spectra in biological samples," *Journal of Biomedical Optics*. 8(3), 329-338 (2003).

5. S. E. Fraser, "Crystal gazing in optical microscopy," *Nature Biotechnology*. 21(11), 1272-1273 (2003).

6. P. Friedl, "Dynamic imaging of cellular interactions with extracellular matrix," *Histochemistry and Cell Biology*. 122, 183-190 (2004).

7. P. Friedl, "Immunological techniques: Dynamic imaging on the immune system (Editorial overview)," *Current Opinion in Immunology*. 16, 389-393 (2004).

8. J. G. Fujimoto, C. Pitris, S. A. Boppart, and M. E. Brezinski, "Optical Coherence Tomography: An Emerging Technology for Biomedical Imaging and Optical Biopsy," *Neoplasia*. 2(1-2), 9-25 (2000).

9. J. G. Fujimoto, “Optical coherence tomography for ultrahigh resolution *in vivo* imaging,” *Nature Biotechnology*. 21(11), 1361-1367 (2003).

10. T. Fukuchi, K. Takahashi, M. Uyama, and M. Matsumura, “Comparative Study of Experimental Choroidal Neovascularization by Optical Coherence Tomography and Histopathology,” *Japanese Journal of Ophthalmology*. 45, 252-258 (2001).

11. N. D. Gladkova, G. A. Petrova, N. K. Nikulin, S. G. Radenska-Lopovok, L B. Snopova, Y. P. Chumakov, V. A. Nasonova, V. M. Gelikonov, G. V. Gelikonov, R. V. Kuranov, A. M. Sergeev, and F. I. Feldchtein, “In vivo optical coherence tomography imaging of human skin: norm and pathology,” *Skin Research and Technology*. 6, 6-16 (2000).

12. S. Huang, A. A. Heikal, and W. W. Webb, “Two-Photon Fluorescence Spectroscopy and Microscopy of NAD(P)H and Flavoprotein,” *Biophysical Journal*. 82, 2811-2825 (2002).

13. K. König and I. Reimann, “High-resolution multiphoton tomography of human skin with subcellular spatial resolution and picosecond time resolution,” *Journal of Biomedical Optics*. 8(3), 432-439 (2003).

14. A. T. Yeh, N. Nassif, A. Zoumi, and B. J. Tromberg, “Selective corneal imaging using combined second-harmonic generation and two-photon excited fluorescence,” *Optics Letters*. 27(23), 2082-2084 (2002).

15. D. Yelin, D. Oron, E. Korkotian, M. Segal, and Y. Silberberg, “Third-harmonic microscopy with a titanium-sapphire laser,” *Applied Physics B: Lasers and Optics*. 74(Supplement), S97-S101 (2002).

16. W. R. Zipfel, R. M. Williams, and W. W. Webb, “Nonlinear magic: multiphoton microscopy in the biosciences,” *Nature Biotechnology*. 21(11), 1369-1377 (2003).

APPENDIX A: LABWINDOWS/CVI PROGRAM FOR MIRROR MOTION AND IMAGE GENERATION

```
/* Description:
*    This program demonstrates how to output multiple continuous periodic
*    waveforms of different frequencies using an internal sample clock.
*
* Instructions for Running:
*    1. Select the Physical Channel/Channels to correspond to where your signal
*       is output on the DAQ device. For multiple channels, select browse then select
*       the desired output channels.
*    2. Enter the Minimum and Maximum Voltage Ranges.
*    3. Enter the desired rate for the generation. The onboard sample
*       clock will operate at this rate.
*
* Steps:
*    1. Create a task.
*    2. Create an Analog Output Voltage channel/s.
*    3. Define the update Rate for the Voltage generation. Additionally,
*       define the sample mode to be continuous.
*    4. Write the waveform to the output buffer.
*    5. Call the Start function.
*    6. Loop continuously until the user presses the Stop button. Check
*       for errors every 100 ms using the IsTaskDone function.
*    7. Call the Clear Task function to clear the Task.
*    8. Display an error if any.
*
* I/O Connections Overview:
*    Make sure your signal output terminal matches the Physical Channel
*    I/O Control. For further connection information, refer to your
*    hardware reference manual.
*
* Recommended use:
*    Call Configure, Write and Start functions.
```

```
*    Call IsDone function in a loop.
*    Call Stop function at the end.
*
      Note:  comments indicated in bold
*******************************************************************/

#include <windows.h>
#include <analysis.h>
#include <cvirte.h>
#include "userint_mod.h"
#include <stdlib.h>
#include <math.h>
#include <stdio.h>
#include "toolbox.h"
#include <formatio.h>
#include "NIDAQmx.h"
#include "DAQmxIOctrl.h"
#include "MultFreqContGen-IntClk.h"
#include "MultFreqContGen-IntClk_Fn.h"
#define NUMCHANS 2


static int panelHandle;         //reference to panel location (user interface, UI)
static int gRunning;
int bitmap;

int check,mode;                 //mode describes the sample acquisition (continuous or
                                finite)
// check allows the user to stop scanning after current scan is complete (toggle
switch on UI)
```

```
int main (int argc, char *argv[])
{
	if( InitCVIRTE(0,argv,0)==0 )  /*needed if linking in external compiler;
harmless otherwise*/
		return -1;  /* out of memory */
//loads specified panel
	if( (panelHandle=LoadPanel(0,"MultFreqContGen-IntClk.uir",PANEL))<0 )
				return -1;

//creates a new DAQmx IO control for selecting physical analog output channels.
	NIDAQmx_NewPhysChanAOCtrl (panelHandle, PANEL_CHANNEL, 1);


	DisplayPanel(panelHandle);  //displays panel stored in specified location
	RunUserInterface();		//runs the UI (user allowed to select values and
					control the UI)
	DiscardPanel(panelHandle);  //removes anything previously stored in the
					panel
	return 0;
}

int CVICALLBACK PanelCallback(int panel, int event, void *callbackData, int
eventData1, int eventData2)
{
	if( event==EVENT_CLOSE ) {
		gRunning = 0;
		QuitUserInterface(0);   //destroys event when program terminates
	}
	return 0;
}

int CVICALLBACK StartCallback(int panel, int control, int event, void *callbackData,
int eventData1, int eventData2)
{
```

```
int outwfm[2];          //definition of waveform characteristics (one for each
                        wave)
double outamp[2];
double outfreq[2];
double offset[2];

int typeofwaveform=0, typeofwaveform2=1;        //defines waveform types
                                                (see switch commands later
                                                in program)
int       error=0,pixels;       //defines error check variable and number of
                                pixels  (set in UI)
short int pixelfactor;  //defines number of full scans (max of 1 for 512x512
                        and 2 for 256x256, set in UI)
int scans;
TaskHandle  taskHandle=0;

char      chan[256];            //needed for color of waveforms in plot
double    min,max,min2,max2;  //waveform parameters set in UI
double    rate;                 //rate of waveform generation set in UI

uInt32    numChannels, TotalsampsPerCycle=0, arrayspot = 0;
float64   *data=NULL,*temp=NULL;

static double line[12000000];         //needed to create vertical offset of each
                                      wave
char      errBuff[12000000]={'\0'};
bool32    done=0;
double    phase=0.0;                  //sets initial phase value at 0
int       j,k,i,place=0;              //integers used for loop increments

double xposition[256][256];
double xposition2[512][512];
```

```
//used for plotting scaled intensity graph
        int xpixels[262144],loCol,hiCol;
        int hiColor=VAL_BLACK;
        ColorMapEntry  colors[256];


        if( event==EVENT_COMMIT ) {

                DeleteGraphPlot(panel,PANEL_GRAPH,-1,VAL_DELAYED_DRAW);

                GetCtrlVal (panel, PANEL_MIN, &min);     //GetCtrlVal gets values
                                                         entered by the user from the UI
                GetCtrlVal (panel, PANEL_MAX, &max);
                GetCtrlVal (panel, PANEL_MIN2, &max2);
                GetCtrlVal (panel, PANEL_MAX2, &min2);

                GetCtrlVal(panel,PANEL_CHANNEL,chan);

                GetCtrlVal(panel,PANEL_RATE,&rate);

                GetCtrlVal(panel,PANEL_PIXELS,&pixels);
                GetCtrlVal(panel,PANEL_MODE,&mode);
                GetCtrlVal(panel,PANEL_CHECK,&check);

                GetCtrlVal(panel,PANEL_LOCOL,&loCol);
                GetCtrlVal(panel,PANEL_HICOL,&hiCol);

                if (pixels==256) {              //sets correct frequency
                                                parameters for specified image size
                        outfreq[0] = 0.0000217;
                        outfreq[1] = 0.00000008477;
                        pixelfactor=1;

                }
```

```
        else    {               //sets lower frequency for slow waveform and
                                doubles number of scans for a 512x512 image

                outfreq[0] = 0.0000217;
                outfreq[1] = 0.000000042385;
                pixelfactor=2;

        }


    SetCtrlAttribute(panelHandle,PANEL_TIME,ATTR_CTRL_VAL,pixelfactor*6
000000/r
            ate);

                outwfm[0] = typeofwaveform;         //specifies waveform types
                outwfm[1] = typeofwaveform2;

                outamp[0] = (max-min)/2;            //sets correct amplitudes given
                                                    user-specified voltage ranges
                outamp[1] = (max2-min2)/2;


                offset[0]=min+outamp[0];            //sets correct vertical offset given
                                                    user-specified voltage ranges
                offset[1]=min2+outamp[1];

                colors[0].color= loCol;             //sets high and low colors for
                                            scaled intensity plot (chosen on the UI)
                colors[1].color= hiCol;
```

```
                //double xposition[256*pixelfactor][256*pixelfactor];

                Clear1D(line,12000000);              //clears array used to create
                                                     appropriate vertical offset
                        //creates arrays used for images
                         for (i=0;i<256*pixelfactor;i++){
                                        for (j=0;j<256*pixelfactor;j++){
                                                xposition[i][j]=0;
                                                xposition2[i][j]=0;
                                        }

                         TotalsampsPerCycle = pixelfactor*12000000;  //checks for error
                                                                     given the specified
                                                                     parameters

                         if( (data=malloc(TotalsampsPerCycle*sizeof(float64)))==NULL )
{
                                        MessagePopup("Error","Not enough memory");
                                        goto Error;
                                        //functions check for sufficient memory
                         }
                         if( (temp=malloc(TotalsampsPerCycle*sizeof(float64)))==NULL
) {
                                        MessagePopup("Error","Not enough memory");
                                        goto Error;
                         }

//for loop and switch statement define the two waveforms used for mirror motion

        for(j=0;j<2;j++){
            switch( outwfm[j] ) {
                case 0:     phase=-90.0;        //phases ensure that peaks of both
                                                waveforms line up
```

```
TriangleWave(5898240*pixelfactor,outamp[0],outfreq[0],&phase,temp);
                Ramp(5898240*pixelfactor,offset[0],offset[0],line);
                for(i=0;i<5898240*pixelfactor;i++)
                        temp[i]=temp[i]+line[i];
        break;

        case 1:         phase=-90.0;

TriangleWave(5898240*pixelfactor,outamp[1],outfreq[1],&phase,temp);
                Ramp(5898240*pixelfactor,offset[1],offset[1],line);
                for(i=0;i<5898240*pixelfactor;i++)
                        temp[i]=temp[i]+line[i];
        break;
                }
        for(k=0;k<5898240*pixelfactor;k++){      //stores both waveforms in
                                                 a large array for
                                                 simultaneous output
                data[arrayspot+k] = temp[k];
        }
        arrayspot = arrayspot + 5898240*pixelfactor;   //sets appropriate array
                                                        index for second loop
                                                        iteration
}

if (mode==1) {          //continuous scanning
        while (check==1) {
        //configures internal clock for data acquisition
                DAQmxErrChk (Configure_ContGenPerWfmIntClk(chan,rate,
                        &numChannels,&taskHandle,pixelfactor));
        //creates a grayscale image from a 256x256 matrix of values
        representing the voltages (positions) of the fast mirror
        //currently the matrix contains mirror voltages normalized by
        maximum fast mirror motion value from UI
```

```
//position matrix
        if (pixels==256) {      //256x256 pixel image
                place=0;
                for (i=0;i<256;i++){
                        for (j=0;j<256;j++){
                                xposition[i][j] = data[place*90/pixelfactor];
                                place++;
                                }
                }
        }

        else {                  //512x512 pixel image
                place=0;
                for (i=0;i<512;i++){
                        for (j=0;j<512;j++){
                                xposition2[i][j] = data[place*90/pixelfactor];
                                place++;
                                }
                }
        }

//deletes previous values in UI graph and plot
        DeleteGraphPlot(panel,PANEL_GRAPH,-
1,VAL_IMMEDIATE_DRAW);
        DeleteImage(panel,PANEL_PICTURE);

//commands used to create array of pixels from fast mirror voltages
(normalized for grayscale image)

        place=0;
        for (j=0;j<256*256*pixelfactor*pixelfactor;j++){
```

```
            xpixels[j]=MakeColor(255*(data[place*90/pixelfactor]-
            min)/(max-min),255*(data[place*90/pixelfactor]-min)/(max-
            min),255*(data[place*90/pixelfactor]-min)/(max-min));
            place++;
        }

//creates bitmap from colors defined above
        NewBitmap(-1,32,256*pixelfactor,256*pixelfactor,NULL,xpixels,NULL,
            &bitmap);
        SetCtrlBitmap(panel,PANEL_PICTURE,0,bitmap);  //displays bitmap
                                                        created

//creates intensity plots of fast mirror waveform (for later use, values will be
changed to photon counts)
        if (pixels==256)                //256x256 pixel image
            PlotScaledIntensity(panel,PANEL_GRAPH, xposition,256,256,
                VAL_INTEGER, .0,0,1.0,0,colors,hiColor,2,1,1);
        else                            //512x512 pixel image
            PlotScaledIntensity(panel,PANEL_GRAPH,xposition2,512,512,
                VAL_DOUBLE,1.0,0,1.0,0,colors,hiColor,2,1,0);

//sends position voltages of fast mirror to a text file for opening in
MATLAB®
        ArrayToFile("c:\\Christy\\LabWindows Programs\\mirrors\\position
            voltages.txt",xposition,VAL_DOUBLE,256*256*pixelfactor*pixe
            lfactor,256*pixelfactor,VAL_DATA_MULTIPLEXED,VAL_GR
            OUPS_AS_COLUMNS,VAL_SEP_BY_TAB,0,VAL_ASCII,VA
            L_TRUNCATE);

        SetCtrlAttribute(panel,PANEL_START,ATTR_DIMMED,1);
            //sets appropriate panel attributes for waveform display
        ProcessDrawEvents();
        gRunning = 1;
```

```
            DAQmxErrChk
      (Write_ContGenPerWfmIntClk(taskHandle,data,pixelfactor));
      //function writes waveform data to DAQ device
            DAQmxErrChk (Start_ContGenPerWfmIntClk(taskHandle));
             //function sets internal clock for DAQ



            while( gRunning ) {                    //checks for task completion
                  Sleep(100);
                  DAQmxErrChk
(IsDone_ContGenPerWfmIntClk(taskHandle,&done));
                  if( done )
                        gRunning = 0;
                  else
                        ProcessSystemEvents();
            }
            //clears the task for the next loop iteration
            DAQmxErrChk (Stop_ContGenPerWfmIntClk(taskHandle));
            DiscardBitmap(bitmap);       //clears bitmap from memory

            GetCtrlVal(panel,PANEL_CHECK,&check);     //checks if user has set
                                                      toggle  switch to "stop"
            }
            //if user has chosen "stop", loop is exited
      }

      else {          //finite scans

            GetCtrlVal(panel,PANEL_SCANS,&scans); //obtains the desired
                                                      number of scans to
                                                      perform
            for (i=0;i<scans;i++) {
                  while (check==1){
```

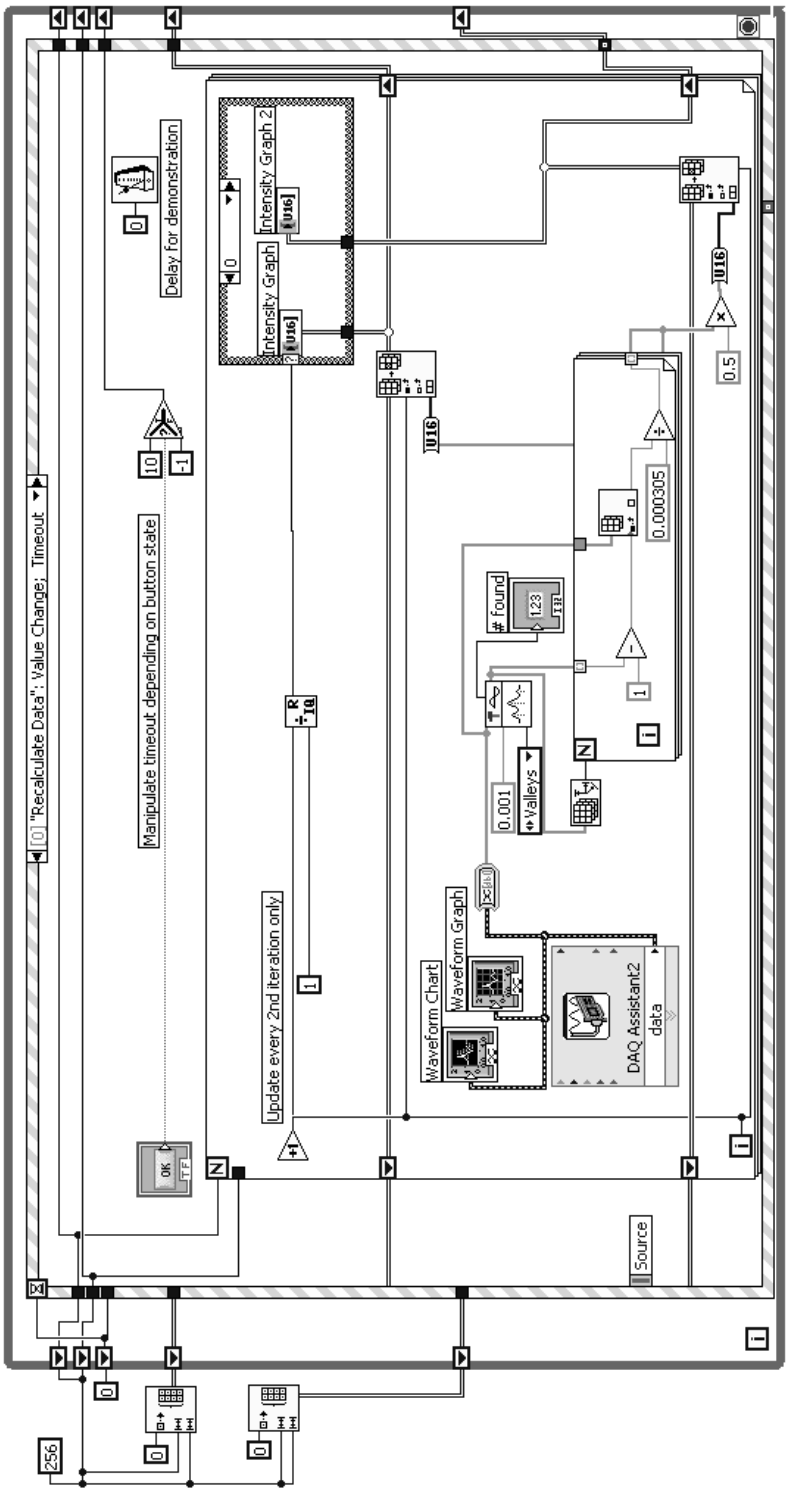
```
                    //position matrix
                        place=0;
                        for (i=0;i<256*pixelfactor;i++){
                            for (j=0;j<256*pixelfactor;j++) {
                                xposition[i][j] =
                    data[place*90/pixelfactor];    //creates intensity values
                                                   for plot
                            place++;
                        }
                    }
        //plots intensity values
            PlotScaledIntensity(panel,PANEL_GRAPH,xposition,256*pixelfactor,25
            6*pixelfactor,VAL_DOUBLE,1,0,1,0,colors,hiColor,2,1,0);
        //deletes intensity plot
            DeleteGraphPlot(panel,PANEL_GRAPH,-
1,VAL_IMMEDIATE_DRAW);
        //deletes bitmap image
            DeleteImage(panel,PANEL_PICTURE);


            place=0;        //used for bitmap image creation
            for (j=0;j<256*256*pixelfactor*pixelfactor;j++){

                xpixels[j]=MakeColor(255*(data[place*90/pixelfactor]-
                min)/(max-min),255*(data[place*90/pixelfactor]-min)/(max-
                min),255*(data[place*90/pixelfactor]-min)/(max-min));
                place++;
            }
        //creates bitmap of previously defined colors
            NewBitmap(-1,32,256*pixelfactor,256*pixelfactor,NULL,xpixels,
                NULL,&bitmap);
        //sends bitmap to UI
            SetCtrlBitmap(panel,PANEL_PICTURE,0,bitmap);
```

```
        //sends mirror position values to text file (for opening in MATLAB®)
            ArrayToFile("c:\\Christy\\LabWindows Programs\\mirrors\\position
                voltages.txt",xposition,VAL_DOUBLE,256*256*pixelfactor*pixelf
                actor,256*pixelfactor,VAL_DATA_MULTIPLEXED,VAL_GROU
                PS_AS_COLUMNS,VAL_SEP_BY_TAB,0,VAL_ASCII,VAL_TR
                UNCATE);

        //sets appropriate panel attributes for waveform display
        SetCtrlAttribute(panel,PANEL_START,ATTR_DIMMED,1);

        //configures internal waveform clock for DAQ
            DAQmxErrChk
                (Configure_ContGenPerWfmIntClk(chan,rate,&numChannels,
                &taskHandle,pixelfactor));
        //function writes waveform data to DAQ device
            DAQmxErrChk
(Write_ContGenPerWfmIntClk(taskHandle,data,pixelfactor));
        //functions starts internal clock
            DAQmxErrChk (Start_ContGenPerWfmIntClk(taskHandle));

            ProcessDrawEvents();
            gRunning = 1;

            while( gRunning ) {                 //checks for task completion
                Sleep(100);
                DAQmxErrChk
(IsDone_ContGenPerWfmIntClk(taskHandle,&done));
                if( done )
                    gRunning = 0;
                else
                    ProcessSystemEvents();

                }
                check=0;
```

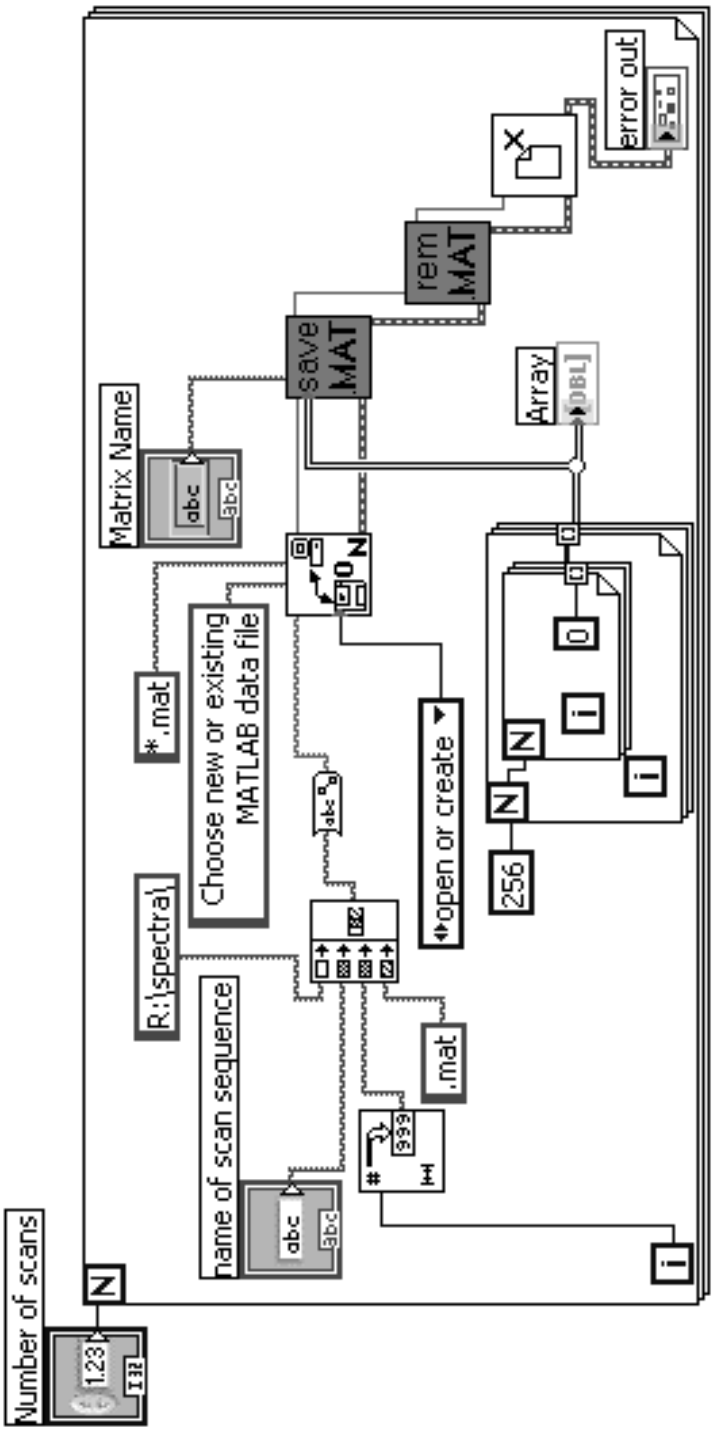
```
            }
        //clears task for next loop iteration
            DAQmxErrChk (Stop_ContGenPerWfmIntClk(taskHandle));

        //clears bitmap from memory
            DiscardBitmap(bitmap);
         //checks for control switch value
            GetCtrlVal(panel,PANEL_CHECK,&check);

            }
        }
                    //exits loop when switch set to "stop"
}

Error:
        if( DAQmxFailed(error) )    //checks for errors and outputs the correct error
                                    description
            DAQmxGetExtendedErrorInfo(errBuff,12000000);
        if( taskHandle!=0 ) {
            SetCtrlAttribute(panel,PANEL_START,ATTR_DIMMED,0);
        }
        if( data )
            free(data);
        if( DAQmxFailed(error) )
            MessagePopup("DAQmx Error",errBuff);
        return 0;
        }

return 0;
}

int CVICALLBACK StopCallback(int panel, int control, int event, void *callbackData,
int eventData1, int eventData2)
```

```
{
	if( event==EVENT_COMMIT )
		gRunning = 0;
	return 0;
}

int CVICALLBACK QuitCallback (int panel, int control, int event,
		void *callbackData, int eventData1, int eventData2)
{
	switch (event)
		{
		case EVENT_COMMIT:
			QuitUserInterface (0);		//closes the UI

			break;
		}
	return 0;
}
```

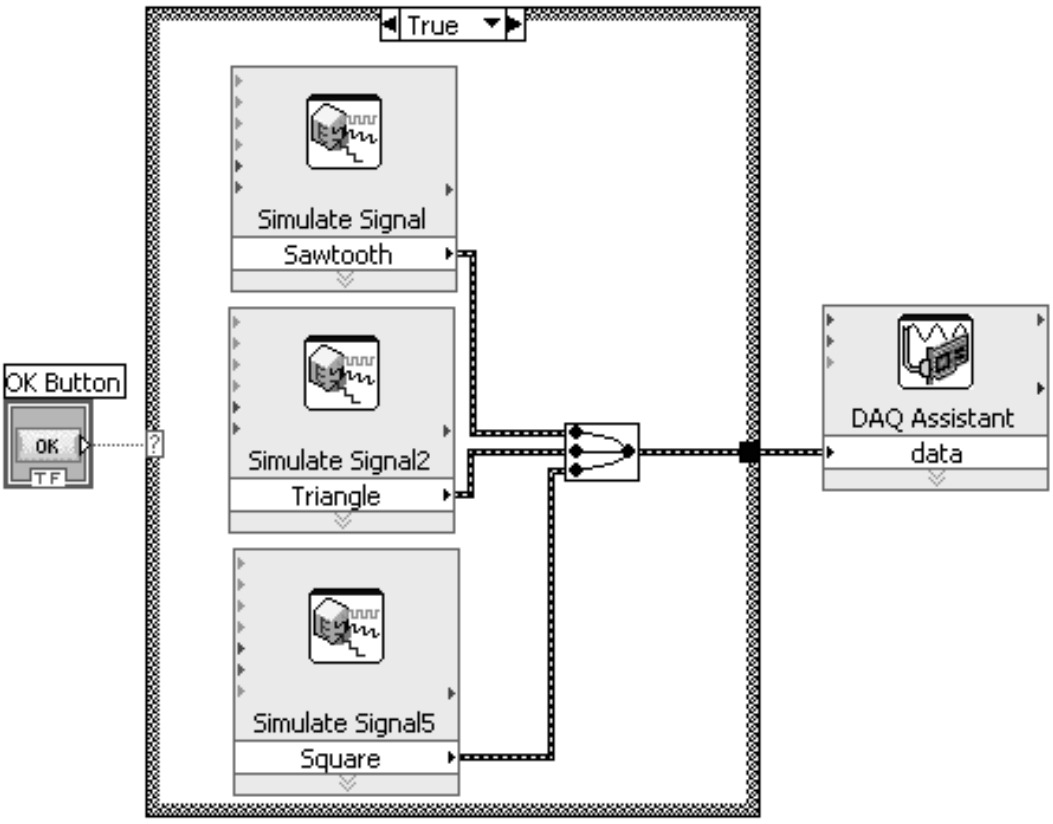
APPENDIX B: FIRST LABVIEW™ BLOCK DIAGRAM FOR DATA ACQUISITION AND IMAGE GENERATION



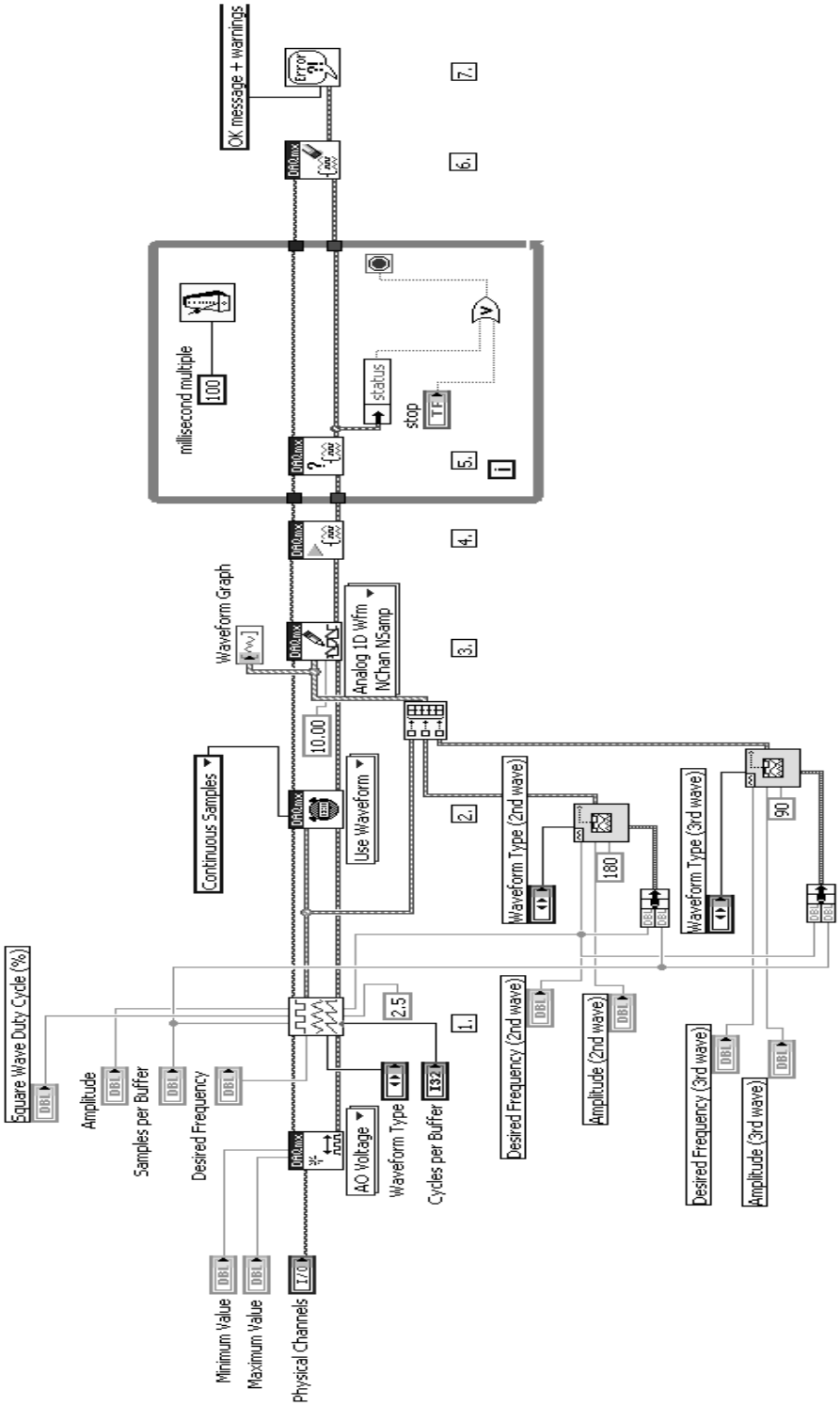
APPENDIX C: LABVIEW™ BLOCK DIAGRAM FOR SAVING BINARY FILES TO BE OPENED IN MATLAB®



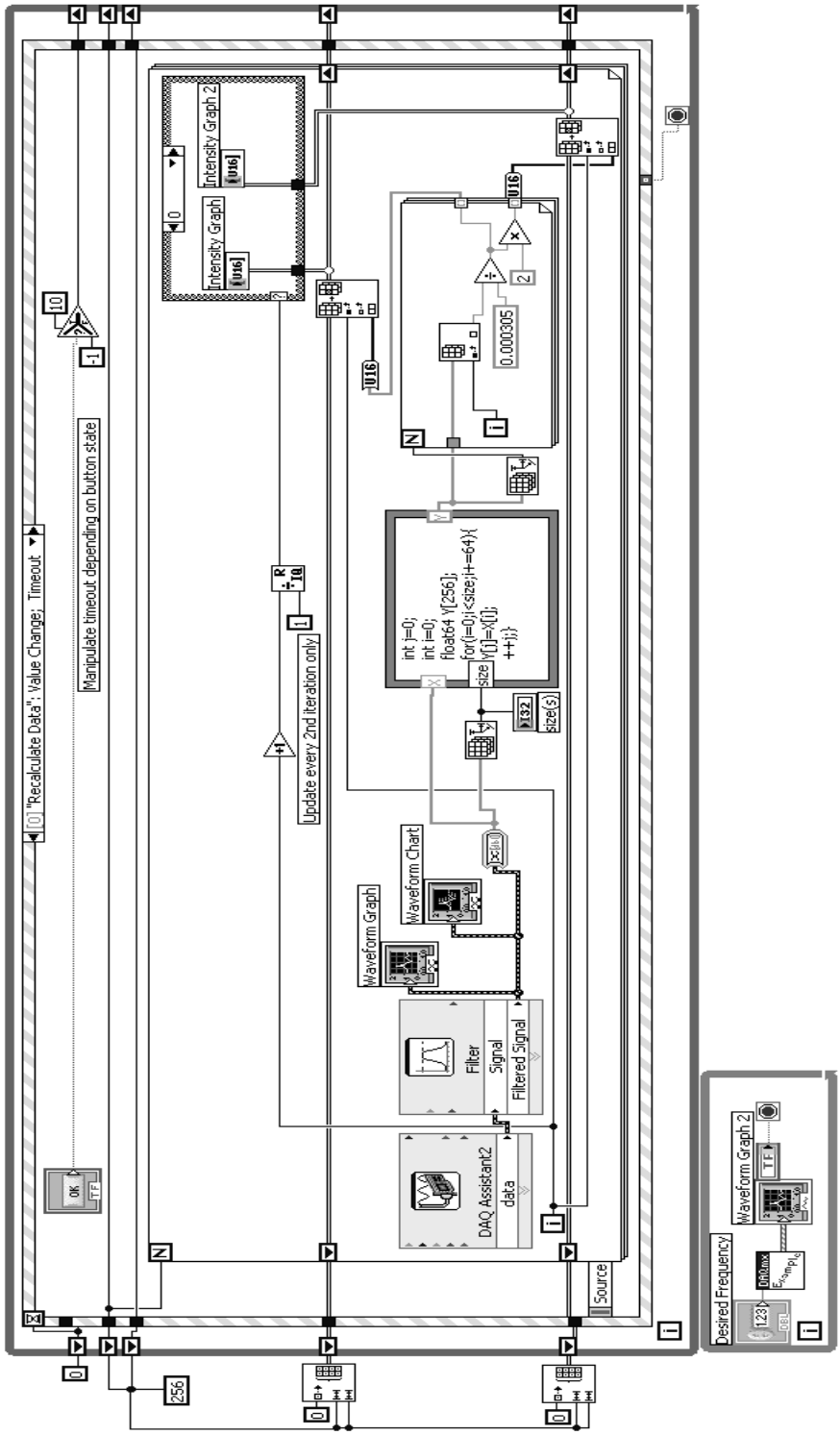
APPENDIX D: FIRST MIRROR MOTION BLOCK DIAGRAM (NO BUFFER)



APPENDIX E: MODIFIED MIRROR MOTION BLOCK DIAGRAM (WITH BUFFER)



APPENDIX F: MODIFIED DATA ACQUISITION AND IMAGE GENERATION BLOCK DIAGRAM



Curriculum Vita

Christina Mae Shafer

Education:

Texas A&M University (College Station, Texas) Fall 2002 - Spring 2006

Degree / Date Expected: Bachelor of Science / May 2006

Major / Minor: Biomedical Engineering / Electrical Engineering

Cumulative GPA: 4.0 / 4.0

Research:

Undergraduate Researcher: Tissue Microscopy Lab Summer 2005 - Present

Dr. Alvin Yeh, Texas A&M University (979-845-5468) College Station, Texas

- Researched 5-40 hours per week with graduate students on a spectral imaging project
- Created imaging software in NI LabView, programmed in LabWindows (C-based), created hardware circuit boards, and developed an interface between non-linear optical microscopy hardware and LabView software to enable 3-D spectral images of live tissue

Work and Volunteer Experience:

Student Worker: Electrical Engineering Department Summer 2005 - Present

Dr. B. Don Russell, Texas A&M University (979-845-7912) College Station, Texas

- Worked 10-20 hours per week with other undergraduate students
- Tested smoke and carbon-monoxide detectors under different fire conditions, compared photoelectric and ionization smoke detectors using statistical analysis, determined conditions for extension cord overload, and developed efficient gas detection methods

Volunteer: Biomedical Engineering Lab Summer 2004

LeTourneau University (903-233-3907) Longview, Texas

- Researched 4-5 hours per weekday with LeTourneau biomedical and electrical engineering students
- Conducted an experiment to observe the adaptation of subjects to different methods of controlling a robotic arm with a joystick, placed electrodes on major muscle groups, and processed data from a leg dominance experiment using Motion Analysis software

Volunteer: Animal Rescue Summer 2005 - Present

Woodstock Animal Foundation of Texas (877-251-2480) College Station, Texas

- Helped find permanent homes for abandoned animals in the Bryan/College Station area
- Assisted in the delivering of food to animal foster homes, visited the homes of prospective animal adopters, and helped set up for the annual rummage sale

Extracurricular activities:

Texas A&M University Symphonic Band Fall 2002 - Spring 2005

College Station, Texas

- Performed multiple school concerts, participated in European Tour (Summer 2004), and performed at Texas Music Educators Association Convention (February 2003)

Accomplishments:	Honors Undergraduate Research Fellows Participant	2005 - Present
	President's Endowed Scholarship Recipient	2002 - Present 2005
	Texas A&M Undergraduate Summer Research Grant Recipient	Summer 2005
	Tau Beta Pi Officer	Spring 2005
	Who's Who Among American College Students	2003
	Symphonic Band Principal Chair	2003 - 2005
Professional memberships:	Tau Beta Pi Honors Engineering Fraternity, Texas Delta	2004 - Present
	Golden Key International Honour Society	2003 - Present
	Biomedical Engineering Society	2002 - 2003
Publications:	Senior Honors Thesis: Optical Biopsy: Complementing Histology with Nonlinear Optical Microscopy	May 2006
Presentations:	Texas A&M Honors Symposium	April 2006
	Texas A&M Student Research Week Poster Session	March 2006
	Houston Conference on Biomedical Engineering Research	February 2006
	Texas A&M Honors Undergraduate Research Fellows	October 2005, February 2006
	Texas A&M Undergraduate Summer Research Grant Poster Session	August 2005