UNIVERSITY OF SOUTHAMPTON

FACULTY OF NATURAL AND ENVIRONMENTAL SCIENCES

Chemistry

Development of a Quantitative *In-Situ* UV-Vis Scanning Reflectance Spectroelectrochemical Probe for the Investigation of Microwell Electrodes

by

David Joce

Thesis for the degree of Master of Philosophy

February 2014

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF NATURAL AND ENVIRONMENTAL SCIENCES

Electrochemistry

Thesis for the degree of Master of Philosophy

DEVELOPMENT OF A QUANTITATIVE IN-SITU UV-VIS SCANNING REFLECTANCE SPECTROELECTROCHEMICAL PROBE FOR THE INVESTIGATION OF MICROWELL ELECTRODES

David John Joce

A study of the performance of fibre optic reflectance probes for the investigation of spectroelectrochemistry at Pt and Ag disk electrodes and Pt and Ag microwell electrodes is detailed. Specific attention is paid to the relationship between spectral and electrochemical measurements. As a test system, electrochemical oxidation and reduction of the redox system $[Ir(Cl)_6]^{3-}$ is reported for chronoamperometric and cyclic voltammetric experiments. Good quantitative agreement between a spectroelectrochemical theory and the experimental data obtained is observed, if the time period of the experiment is maintained below ~ 10 s. At longer times, limitations associated with natural convection and edge effects are illustrated.

A dynamic imaging technique is developed with the aid of a white resin supported working electrode (WSWE). This is shown to enable the tracking of the diffusion field of the electrode highlighting the edge and natural convection effects associated with the diffusion field generated. The ability for this dynamic imaging technique to produce quantitatively analysable results is investigated.

A single-fibre reflectance probe is assessed for its ability to investigate spectroelectrochemical changes occurring in microwell electrodes that are used for lab-on-chip devices.

Contents

DECLARATION OF AUTHORSHIP

I, David Joce
declare that the thesis entitled

Development of a Quantitative *In-Situ* UV-Vis Scanning Reflectance Spectroelectrochemical Probe for the Investigation of Microwell Electrodes

and the work presented in the thesis are both my own, and have been generated by me as the result of my own original research. I confirm that:

- this work was done wholly or mainly while in candidature for a research degree at this University;
- where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
- where I have consulted the published work of others, this is always clearly attributed;
- where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
- I have acknowledged all main sources of help;
- where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
- none of this work has been published before submission,

Signed: ………………………………………………………………………..

Date:……………………………………………………………………………….

Acknowledgements

I would like to thank the following:

- My supervisor, Dr Peter Birkin, for the support he has given me during the completion of this work. I would also like to thank all the members of his research group that have helped me throughout; Dr Douglas Offin, Dr Chris Vian, Dr Jekaterina Kuleshova, Tom Foley, Jenny Barber, and Frances Burrell for helping with the silver chloride experiments.
- My advisor, Dr Guy Denuault, particularly for providing helpful simulations.
- My industrial supervisor, Dr Jason Hyde, for his great advice.
- Dr Noel Healy, from the Optoelectronics Research Centre, for fabricating the single-fibre reflectance probe.
- The University of Southampton and Oxford Nanopore Technologies for their funding and making this research possible.

List of Symbols

A	absorbance	–
c	concentration of a species in solution	M
D	diffusion coefficient	$cm^2\ s^{-1}$
F	Faraday constant	$96485\ C\ mol^{-1}$
i	current	A
i_{lim}	limiting current	A
I	or light intensity	–
k	rate constant for a chemical reaction	s^{-1}
L	depth of microwell	cm
l	path length	cm
m	number of moles involved in electrode reaction	mol
n	number of electrons involved in electrode reaction	–
Q	charge	C
r	radius of electrode	cm
S	signal strength	–
t	time from commencement of an experiment	s
V	volume	cm^3
z	double the distance between probe tip and electrode	cm
δ	distance between probe tip and electrode	mm
ε	*molar extinction coefficient*	$M^{-1}\ cm^{-1}$
λ	wavelength	nm

θ	angle relative to the normal of a disk electrode	°
υ	potential scan rate	V s^{-1}

Chapter 1: Introduction

1.1 Background

The field of electrochemistry encompasses a variety of analytical techniques, devices and technologies, including, but not limited to; batteries, sensors, electro-plating, catalysis, etc. Traditionally, analytical electrochemistry provides an excellent method for determining concentrations, reaction potentials, electrode kinetics, and reaction mechanisms. These measurements are possible through the analysis of electrochemical variables such as the current passed, the charge consumed, and the potential of an electrode submerged in an appropriate electrolyte. However, sometimes these measurements alone are not sufficient to provide a complete understanding of a given redox reaction, such as the identity of any intermediates or products formed.

Spectroelectrochemistry is a methodology that combines two different types of analytical techniques; electrochemistry and spectroscopy, to provide new information on the system considered which has been shown to be highly relevant to the study of organic, inorganic, and biological products produced or consumed by redox chemistry under electrochemical control [1].

All of the techniques that fall under the spectroelectrochemistry banner can be split into two different groups; *in-situ* and *ex-situ*. *In-situ* techniques involve making spectroscopic measurements of the electrode surface (or electrolyte) while simultaneously measuring the current or charge passing through the electrode when it is held under electrochemical control. *Ex-situ* techniques involve inspection of the electrode (once it has been removed from the electrochemical cell) after it has undergone some sort of electrochemical process.

The majority of the work reported here consists of studies using *in-situ* spectroscopy; specifically *in-situ* UV/Vis reflectance spectroscopy. This technique consists of placing a fibre-optic reflectance probe directly above an electrode – with both the electrode and probe submerged in an electrolyte within the electrochemical cell. The probe provides a means of shining ultraviolet or visible light on to the electrode surface and a means of detecting the light after it has reflected off of that electrode surface. A commercial reflectance probe will typically consist of a tightly packed bunch of optical fibres split into two groups; one that transmits light from a light source to the tip of the probe, and another that transmits the reflected light to a spectrometer.

In principle, the electrochemical component of the technique is used to change the oxidation state of a chemical species (either causing a direct colour change that can be measured, or an indirect colour change through a secondary chemical reaction) through the exchange of electrons at an electrode, while the current response, charge and potential are measured. The spectroscopic component of the technique measures absorbance changes (within the UV/Vis range of the electromagnetic spectrum) occurring at the electrode, or in the solvent, as a consequence of the electrochemical process. The two sets of data are recorded simultaneously and can complement each other to provide a more detailed analysis of the chemical species produced by the electrolysis.

This body of work deals with adapting this technique for the investigation of microwell electrodes and study in detail the quantitative relationships between the electrochemical and spectroscopic measurements. To better understand the concepts that are involved with UV-Vis reflectance spectroscopy, it is worth considering some of the precursor techniques from which it was derived from; chiefly specular reflectance spectro-electrochemistry and transmission spectroscopy [2].

1.2 Specular Reflectance Spectroscopy

When specular reflectance spectroscopy first appeared, it was used as a method of interpreting reaction mechanisms in the production of films at the surface of an electrode [3]. The specular reflectance technique consists of shining a light at an angle to a smooth reflective working electrode and measuring the signal of light that is reflected by the electrode, as shown in Figure 1.1. As a thin film or adsorbed layer is electrochemically induced at the surface of the electrode, its reflectivity changes and this is shown in the spectral measurements.

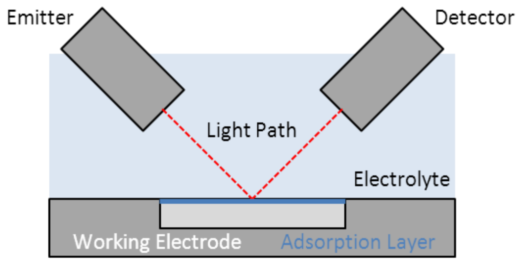


Figure 1.1: Schematic showing a typical setup for a reflectance spectroscopy experiment.

The technique was originally developed to monitor the formation of adsorbates and thin films on electrodes [4]. Over time, the technique was continued to be used to study adsorption [5] [6] and reaction intermediates [7]

1.3 UV-Vis Transmission Spectroscopy

UV-Vis Transmission Spectroscopy is a technique that consists of shining a light through an optically transparent working electrode (OTE) that is exposed to an electrolyte on one side [2]. The light source is orientated so that the path of light is perpendicular to the surface of the OTE. As redox species are generated at the interface between the OTE and the electrolyte, the light they absorb can be measured by placing a detector at the opposite end of the electrode to the light source.

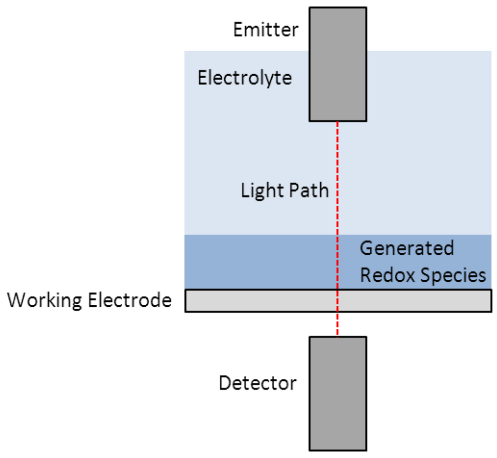


Figure 1.2: Schematic showing a typical setup for a transmission spectroscopy experiment.

The OTEs can either be made from conducting glass (glass coated with a thin layer of tin oxide [8] or a fine mesh of thin wires that allows light to pass through [9]. The technique proved valuable for analysis of both organic and inorganic redox products [9], [10], [11], [12].

The advantage of using these OTEs was that they allowed a direct qualitative comparison between spectroscopic and electrochemical measurements [11]. Steckhan and Kuwana were able to show that the absorbance measurements they recorded were proportional to the charge that had passed through the electrochemical cell [11].

1.4 Quantitative Theory

Strojek, Kuwana, and Feldberg were the first to publish work that quantitatively linked measurements produced by transmission spectroscopy with measurements produced by coulometry [13].

Strojek *et al.* considered the following electrochemical redox reaction:

$$A \rightleftharpoons B + ne^-$$

Faraday's Law was then used to relate the charge (Q) produced (or consumed) by the reaction with the amount of product produced at the surface of the electrode in moles (m), the amount of electrons produced by converting one molecule of reactant to product (n), and Faraday's Constant (F) [14]:

$$Q = nmF \qquad (1.1)$$

For an ideal redox system that occurs at a macro-electrode disk, it is assumed that the mass transfer is limited by diffusion alone and that the diffusion field is limited to one-dimension perpendicular to the electrode. Under these conditions a classic Nernst diffusion layer is expected to form [15]. The Nernst diffusion layer is modelled as a layer of redox product that has a constant concentration gradient. The boundary conditions are that the initial concentration at the surface of the electrode is equal to the bulk concentration, and that at a distance (δ) from the electrode, the concentration of products is zero. Concentration of redox material within this Nernst diffusion layer only varies as a function of distance from the electrode.

Consider a light emitter (e.g. laser light source) and detector that are placed either side of an optically transparent electrode, and an electrolyte that interacts with the electrode on one side. When a potential step is applied to the working electrode a Nernst diffusion layer forms, as is shown in Figure 1.3. All of the light that is detected by the detector has to pass through this Nernst diffusion layer.

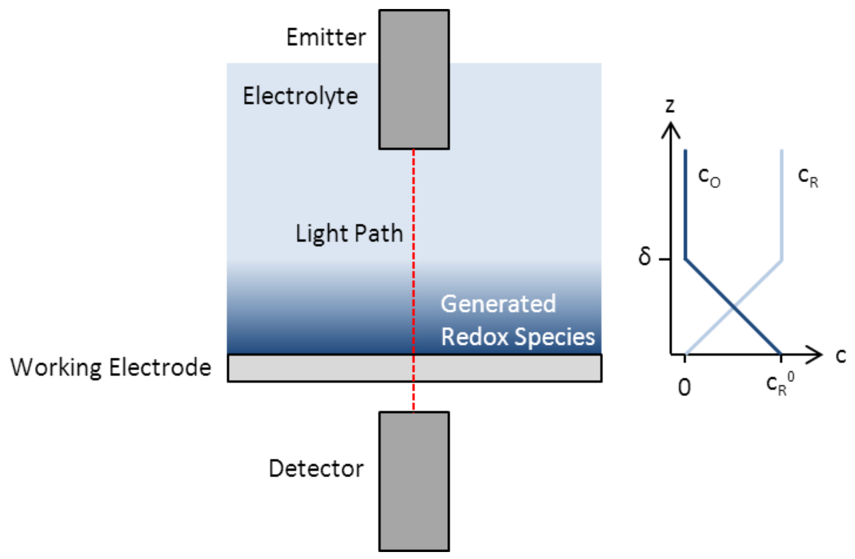


Figure 1.3: Schematic of a typical transmission spectroscopy experiment showing a beam of light passing through a Nernst diffusion layer and an optically transparent working electrode before reaching a detector. The concentration gradient of the Nernst diffusion layer is shown on the right.

If the product of the redox system absorbs more light at an arbitrary wavelength (λ) than the reactant, then Beer's Law [16] describes the magnitude of that absorbance (A_λ) as a product of the difference in molar extinction coefficients of the reactant (ε_A) and product (ε_B), the path length that the light takes when passing through the electrolyte (l), and the average concentration of product within the volume of electrolyte inspected by the probe (c). In this case, l can be taken as the distance between the emitter and the detector.

$$A_\lambda = (\varepsilon_B - \varepsilon_A)cl \qquad (1.2)$$

This equation assumes that the concentration of redox product is constant along the entire path length that the light travels along.

The value of this constant concentration can be found by remodelling the Nernst diffusion layer as a homogeneous layer that fills a volume extending from the emitter to the detector (*l*) with a cross-sectional area the same as the working electrode (πr^2).

$$\frac{Q}{nF} = m = c \times V = c \times (\pi r^2 l)$$

$$\therefore c = \frac{Q}{nF(\pi r^2 l)} \qquad (1.3)$$

Therefore, the charge produced from the electrochemical reaction should produce an absorbance value that obeys the formula:

$$A_\lambda = \frac{2Q(\varepsilon_B - \varepsilon_A)}{nF(\pi r^2)} \qquad (1.4)$$

Other than the Faradaic charge produced by the redox system, all other parameters will be constant throughout any experiment that uses standard electrochemical techniques, such as applying a potential step to the working electrode, or cyclic voltammetry. For a potential step driving a redox reaction that is controlled by diffusion alone, the current response should obey the Cottrell Equation [16]:

$$i = \frac{nF(\pi r^2)D^{0.5}c_R}{\pi^{0.5}t^{0.5}} \qquad (1.5)$$

Integrating the Cottrell Equation gives the electrochemical charge as a function of time:

$$Q = \frac{2nF(\pi r^2)D^{0.5}c_R}{\pi^{0.5}} \times t^{0.5} \qquad (1.6)$$

Substituting Equation 1.6 into Equation 1.4 gives:

$$A_\lambda = \frac{2Q(\varepsilon_B - \varepsilon_A)D^{0.5}c_R}{\pi^{0.5}} \times t^{0.5} \qquad (1.7)$$

Therefore, for a potential step, there should be a linear relationship between absorbance and the square-root of time.

By analysing how absorbance and electrochemical charge vary with time, the mechanisms and kinetics of redox species produced can be evaluated [14], [18], [19].

Transmission spectroscopy has also been adopted to study the electrolysis occurring in small volume cells of 30 µl to 50 µl [9]. Optically transparent thin layer electrode (OTTLE) cells can vary in design [12], [20], [21], but a typical one would involve placing a mini-grid electrode in between two flat glass slides and utilising a spacer to make sure a small amount of electrolyte can surround the electrode. Spectra can either be measured by emitting light in line with the normal of the electrode (short path length) or by emitting light across the electrode (long path length). The advantage of using these cells is that they can undergo complete electrolysis in a short time span and have been used to confirm the values of *n* and the formal equilibrium potential of different redox couples [22], [23].

However, one of the disadvantages with working with optically transparent electrodes is that it limits the types of materials that could be used as a working electrode to those that are electrically conductive and are optically transparent. Alternatively, conventional materials can be used as a mini-grid electrode, but these produce absorbance measurements that are difficult to quantitatively analyse, due to their complex geometry. As well as this, OTTLE cells suffer from high ohmic losses, which lead to distorted voltammograms [24].

Nevertheless, if the conditions of the experiment are carefully selected, it is possible to gather useful chemical information on the processes which occur in response to electrochemical stimuli. Consider the electrolysis 5,6 diamino uracil. In this system the starting material is converted to a metastable intermediate which undergoes subsequent hydrolysis (see Figure 1.4).

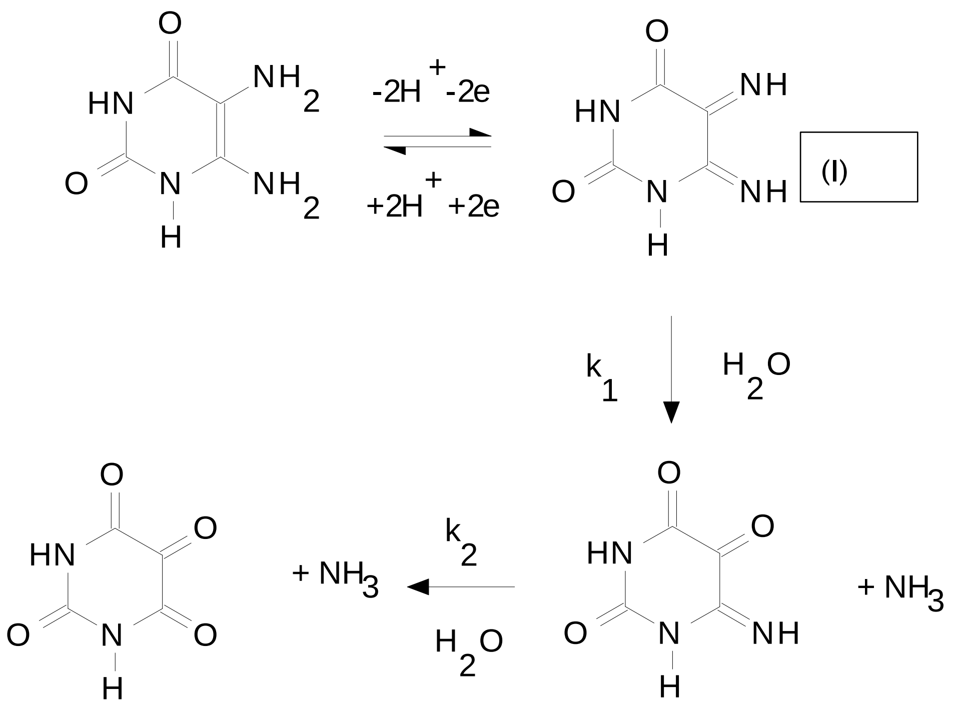


Figure 1.4: Reaction scheme showing the electrochemical oxidation of 5,6 diamino uracil to the intermediate species (I).

In this case the intermediate species (I) absorbs at ~ 330 nm and can be observed with a UV-Vis spectrometer as it is produced in the OTTLE cell. The kinetic of the hydrolysis process (rate constant k_1) can then be followed after the electrolysis period (typically 10-100 s) is complete. Figure 1.4 shows the production of species (I) and its subsequent chemical reaction.

While the use of transmission experiments is appealing, it does impose some limitations on the materials and cell design (particularly the electrode material and the working electrode construction) employed. In order to avoid these limitations reflection techniques which can be deployed on a variety of different substrates and architectures have been developed.

1.5 Near Normal Incidence UV-Vis Reflectance Spectroscopy (NNI-UVRS)

While useful tools for electrochemical analysis, OTE based techniques are disadvantaged by the limited electrode materials that can be used. OTTLE based techniques can overcome this advantage, but the sensitivity of an experiment may be limited by the available diffusion layer thickness.

Pyun and Park were one of the first to overcome the issues of having to work with optically transparent electrodes by using a flat working electrode disk as a reflective surface together with a bifurcated reflectance probe [25], [26], [27], [28].

A working electrode disk is submerged in electrolyte and is used to generate a redox product that absorbs light at a different wavelength to the redox reactants. The bifurcated probe is placed directly above the electrode, and shines light on to it. Some of the light that is reflected by the electrode is detected by the probe and analysed using a spectrometer. This light has passed through the theoretical Nernst diffusion layer adjacent to the working electrode twice, so absorbance values will be twice the value as the theoretical values had they been produced using an optically transparent electrode setup.

It is worth noting that when using a bifurcated probe, light will be emitted on to the electrode at an angle, θ, relative to the normal of the electrode. As such, Equation 1.4 becomes:

$$A_\lambda = \frac{2Q(\varepsilon_B - \varepsilon_A)}{nF(\pi r^2)\cos(\theta)} \qquad (1.8)$$

Shaw *et al.* demonstrated this quantitative relationship by oxidising ferrocene in MeCN [29]. They observed a linear relationship between the absorbance and coulometric sets of data. However, the spectra they observe are short lived (<5 s) and have not shown whether the linear relationship between charge and absorbance is maintained over long-time scale electrolysis.

Since, near normal incididence UV-Vis reflectance spectroscopy (NNI-UVRS) has been applied to rotating disk electrodes (RDEs) to help provide a better understanding of reactions occurring under forced convection [30], [31].

A similar UV-Vis reflectance technique has been applied to OTTLE cells as well by Schroll *et al.*[32]. They prove the concept works using a ferrocyanide redox couple. The electrolysis is carried out using a potential step, while the current and absorbance are recorded simultaneously. The absorbance response was shown to increase proportionally with the square-root of time, but only for the first 25 seconds. Beyond this threshold, the absorbance increases at a much slower rate.

This deviation was attributed to the optical configuration of the cell, rather than any changes in the electrolytic processes occurring within the cell. The fibre bundle used had a diameter of 2 mm, while the electrode disk used had a diameter of 3 mm, with a distance of 1 mm between the two. Light emitted from the fibre bundle illuminated a volume of electrolyte that extended beyond the boundary of the working electrode, and that the volume being illuminated is the same volume that is under inspection. Since a region of this electrolyte would be unchanged by the electrolysis occurring at the electrode, Beer's Law could not be analytically accurate, hence the levelling off.

This work attempts to apply this NNI-UVRS technique to microwell electrodes that are 50 µm in diameter (nearly an order of magnitude smaller than some electrodes that are typically used). An array of these microwells electrodes are used to an integral part of a DNA sequencer. It is hoped that a spectroelectrochemical investigation of these microwells during the process of DNA sequencing can provide new data that can aid in the optimisation of their design.

1.6 DNA Sequencing

The interest in being able to apply NNI-UVRS to microelectrodes is motivated by their use in lab on chip DNA sequencers [33]. These DNA sequencers are based on the concept of Coulter counters in order to record which DNA bases make up a certain DNA sequence.

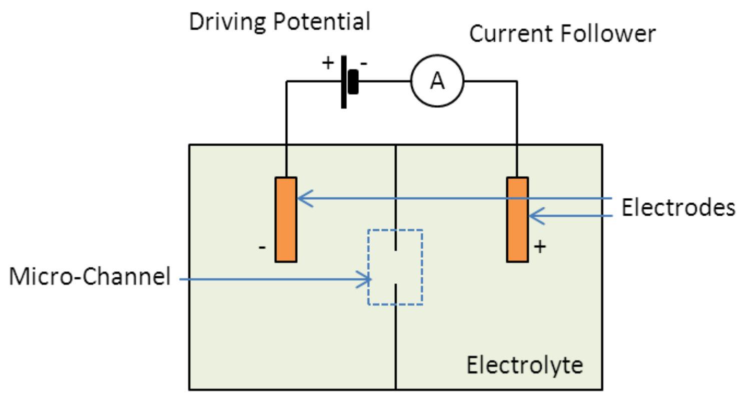


Figure 1.4: Schematic of a Coulter counter.

A Coulter counter is made up of two chambers that are connected by a micro-channel. Both chambers are filled with an electrolyte and an electrode is submerged into each chamber. These electrodes are connected to an external voltage supply and a current follower [34].

When the voltage supply exerts a driving potential, ions flow from one electrode to the other via the micro-channel, and a current flows through the cell. The magnitude of the current flowing through the cell is largely determined by the rate at which ions flow through the micro-channel. Ions flowing through a micro-channel are analogous to free electrons flowing through a resistor; in that the diameter and length of the micro-channel will have an impact on the current flowing through the cell.

If large particles relative to the ions (e.g. single-celled organisms) are introduced to the cell and driven through the micro-channel, they will change the cross section of electrolyte flowing through the channel; and therefore the cell's resistance. This change in resistance can be detected by the current follower as a change in current, and can be used to find the size of the particle inside the micro-channel.

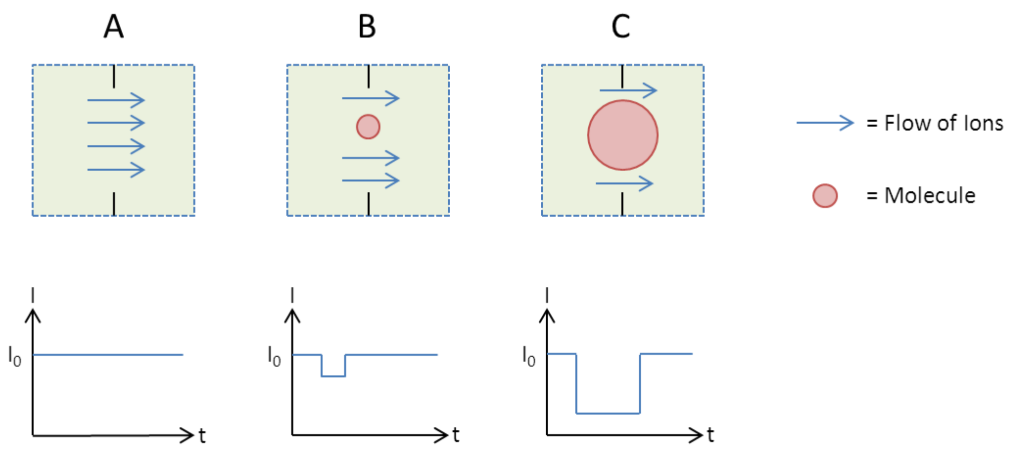


Figure 1.5: Illustration of how different particle sizes affect the current flowing through the cell over time.

The lab-on-chip devices that have motivated this work are based on the same principle as Coulter counters [33]. A microwell, with an electrode at the base, is submerged in an electrolyte. A lipid bilayer is used to seal the microwell; preventing ions from getting in or out of the microwell. The volume bounded by the microwell and lipid bilayer forms one chamber (*trans* chamber), while the volume of electrolyte outside the bilayer acts as the second chamber (*cis* chamber).

A protein nanopore is inserted into the bilayer and allows a flow of ions in or out of the microwell, and acts as the micro-channel. The most frequently used protein nanopore used in this DNA sequencing technique is alpha-hemolysin [35], [36], [37], [38]. A driving potential is applied across the channel to induce a flow of ions through the nanopore, and a current through the cell. The movement of DNA strands passing through the channel is regulated by an enzyme (e.g. DNA polymerase) [39], [40], [41].

Each of the four different DNA bases has a unique molecular size [33]; and so each has a different effect on the flow of surrounding ions, and hence the current flowing through the cell [42], [43], [44]. Figure 1.6 shows an example of current changing over time as individual DNA bases pass through a nanopore. When the nanopore is clear, the current flowing through the cell is 54 pA when a potential 0f +180 mV is applied across the nanopore. The current then changes to one of four values depending on what type of DNA base is passing through (34 pA for cytosine, 30 pA for adenine, 26 pA for thymine, and 23 pA for guanine). The order that these current changes occur indicate the DNA sequence of the strand that was fed into the *cis* chamber.

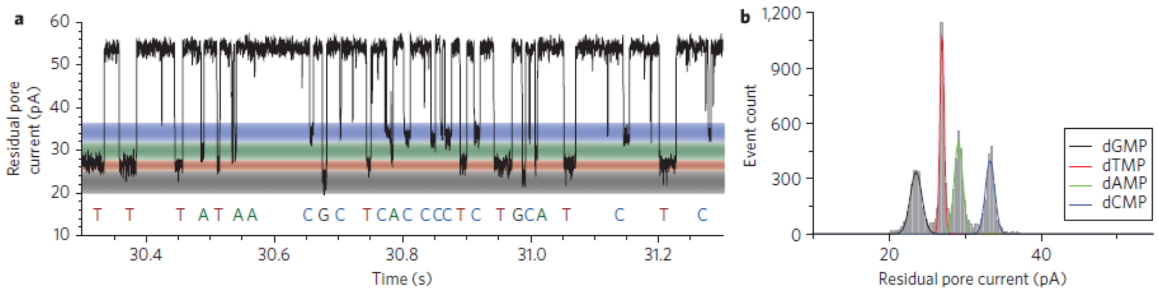


Figure 1.6: (Left) Example of current changing as different DNA bases pass through an alpha-hemolysin nanopore. (Right) Graph showing the frequency of different current changing events taking place, and highlighting which currents correspond to which bases (also include Gaussian fits). Ref [33]

By creating a lab-on-chip device that uses over a hundred microwells running in parallel, an entire DNA sequence can be mapped out in short time spans relative to the those offered by more traditional methods of DNA sequencing [33].

The lipid bilayers are optically transparent due to its extremely low thickness (4-5 nm). Therefore, a UV-Vis reflectance probe can be used to investigate the changes that may be occurring at the surface of the electrode (e.g. adsorption), or any colour changes in the electrolyte that occur between the lipid bilayer and the electrode. The technique is ideal due to its non-invasive nature. It is hoped that a spectroelectrochemical investigation of lab-on-chip devices designed by Oxford Nanopore Technologies can provide supplementary data that can reveal chemical processes that may otherwise be undetectable by using standard electrochemical analysis alone.

1.7 Aims

The aims for this thesis are as follows:

- Investigate the behaviour of light emitted by the bifurcated reflectance probe and find what regions of the electrode/diffusion layer are actually under inspection during a typical UV-Vis reflectance spectroscopy experiment.
- Investigate the UV-Vis reflectance spectroscopy technique with the bifurcated probe and quantitatively link the absorbance data with the coulometric data, while explaining the discrepancies observed by Schroll *et al.*
- Scale down the technique so that it can be applied to investigate electrochemical processes occurring in microwells and underneath lipid bilayers.

Chapter 2: Experimental

2.1 Disk Electrodes

All electrode disk were sealed in glass, unless otherwise stated. Electrodes were initially polished on abrasive paper followed by a fine polishing using 1 μm and 0.3 μm alumina slurries (used as received from Struers) in a conventional manner. Unmodified electrodes were polished using 1 μm and 0.3 μm alumina slurries on a polishing pad (Buehler, Microclothe) and were then rinsed with distilled water.

The white resin supported working electrode (WSWE) was made by casting Pt wires (0.5 mm and 1 mm diameters, 99.95% purity, Advent Research materials) in a white polyester resin (East Coast fibre glass supplies). The disk was then polished with P180 sand paper (particle size ≈ 82 μm) and rinsed with distilled water.

2.2 Chemicals

Potassium Chloride (KCl) (Fisher Scientific, > 99%), Sodium Chloride ($NaCl$) (Fisher Scientific, 99.9%), Strontium Nitrate ($Sr(NO_3)_2$) (Sigma Aldrich ACS), Sodium Sulphate (Na_2SO_4) (Fisher Scientific, Lab Reagent Grade), Trizma (Sigma, 99.9%), Potassium Chloroiridate (III) ($K_3[IrCl_6]$) (Aldrich), Ammonium Chloroiridate ($(NH_4)_3[IrCl_6]$) (Aldrich, 99.99%), Potassium Ferrocyanide ($K_4[Fe(CN)_6]\bullet 3H_2O$) (Sigma Aldrich ACS, > 98%), thymol blue (Aldrich ACS, 95%), Fluorescein (FSA Laboratory), Sulphuric Acid (H_2SO_4) (Fisher Scientific, 98%), Hydroquinone (Aldrich, >99%), Ethanol (C_2H_6O) (Fisher Scientific), Hexane (C_6H_{14}) (Fisher Scientific), Pentane (C_5H_{12}) (Fisher Scientific) were used as received.

The water used was purified to >15 MΩ cm using a Purite Select purification system. All experiments were performed at room temperature 18–22 ^{o}C under aerobic conditions.

2.3 Spectral Measurements

Throughout this work three different reflectance probes are used; two bifurcated probes ((Avantes FCR-7UV200-2-ME (plain tip) or FCR-7UV200-2-1.5x100 (narrow tip), Anglia Instruments)) and one single fibre reflectance probe (Thorlabs).

The only difference between the two bifurcated probes is the diameter of the sheathing that fits around the core of fibre-optic cables; the positions of the fibres within the core are exactly the same for both probes.

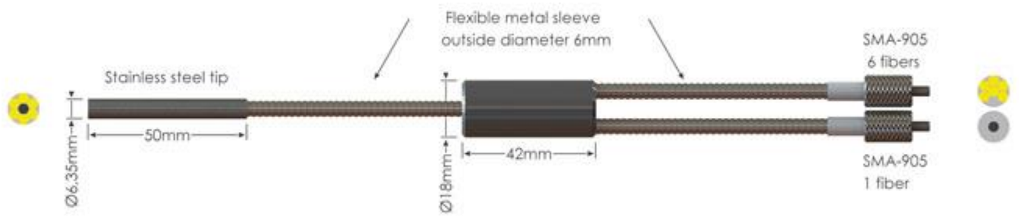


Figure 2.1: Diagram of the bifurcated probe (taken from Avantes website, http://www.avantes.com/Chemistry/Reflection-Probe-Standard/Detailed-product-flyer.html)

The single fibre probe was made using a modified fibre optic splitter. A 60 μm diameter patch cable with SMA connectors (Thorlabs) was cut in half, and each half was fused to one input fibre of the splitter. The output fibre was placed inside a rubber sheath and polished using alumina slurries on polishing pads.

The single fibre probe works by employing a fibre-optic splitter; a passive device that splits an incoming signal from an input leg into two signals that are transmitted through two output legs. A splitter is usually made by wrapping two fibre optic cores together, putting them under tension, and applying enough heat to fuse the two cores together. One fibre is cut at the joint, leaving a junction that has one leg splitting into two. While the splitter can be made so different signal ratios are transmitted down each leg (60:40, 20:80)[], the splitter used in all of the following experiments split the signal equally (50:50) across each output leg.

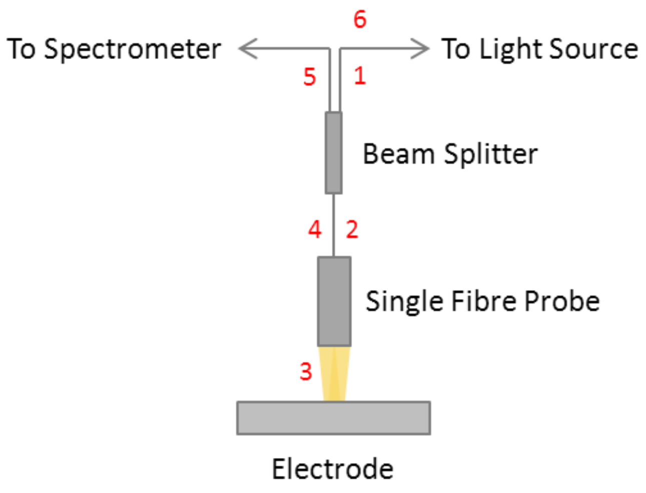


Figure 2.2: Schematic of how the beam splitter is used as a single fibre probe. Light is emitted from a light source and travels to the splitter via an optical fibre (1). At the beam splitter, the light is transmitted to the input leg, towards the probe (2). Light is emitted by the probe and reflected by the electrode back to the probe (3). This reflected light travels along the same single fibre, back to the beam splitter (4). The beam splitter splits this light so that half of the signal is transmitted to the spectrometer (5), while the other half is transmitted back towards the light source (6).

The same spectrometer was used throughout; a Thorlabs USB SP1 diode array spectrometer. The spectrometer measures spectra between 380 nm to 850 nm with a resolution of 0.15 nm. To reduce noise, recorded spectra are produced by averaging the signal measurements for three adjacent wavelengths. For example, a signal at the 499.8 nm wavelength is produced by averaging the signals recorded at the 499.76 nm, 499.61 nm, and 499.91 nm wavelengths. The resolution of any absorbance contour plots varies between 0.4 and 0.5 nm. Integration times vary between 10 and 200 ms, depending on the strength of the signal measured by the spectrometer. If integration times were less than 10 ms, averages of several measurements were used to produce spectra. When possible, settings were adjusted such that noise observed in absorbance measurements were kept below 0.01, and that recordings were taken frequently enough to capture all spectroelectrochemical events for a given experiment.

Two different light sources were used throughout. The first was a white LED (12 V, 6 mA) that was placed directly next to the light emitting fibre bundle. The second light source used was a 19.7 W LED light source (Toshiba E-core E27 19.7 Watt Par38 Led 2700k 25 Degree Beam, Home White) (see Figure 2.3) that was focused on to the light emitting fibre bundle using a lens (Fujinon TV Lens 1:1.4/25).

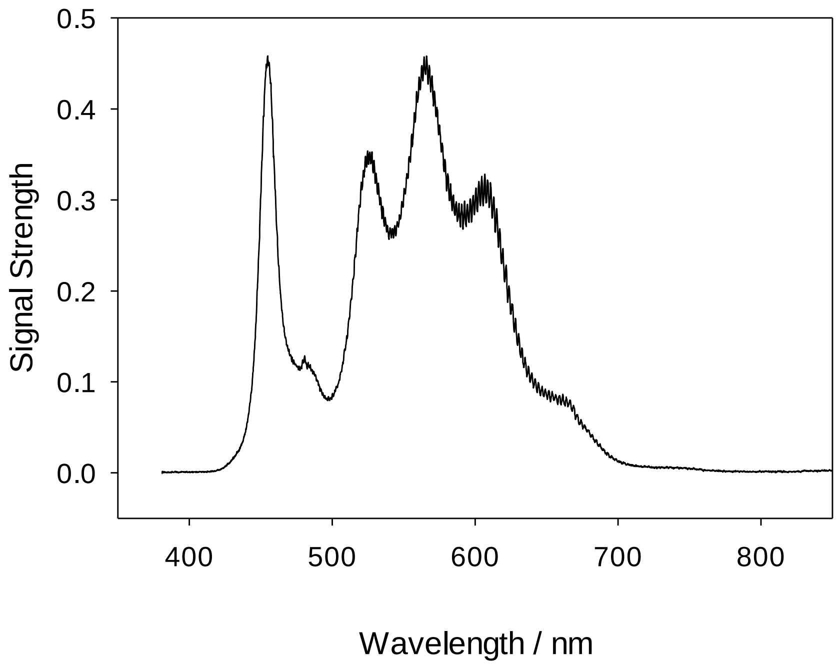


Figure 2.3: Typical spectra emitted by the Toshiba LED light source.

The reflectance probe is secured to a stage capable of moving in all directions. Two actuators (Zaber, TSB28-M) are aligned at right angles and control horizontal movement of the probe, while a third actuator (Zaber, TSB60-M) is used for vertical movement. The actuators are controlled by in-house software (written in Visual Basic 6 - see Appendix A).

2.4 Electrochemical Cell

All electrochemical cells through this work use a three electrode setup. Reference electrodes were either mercury/mercurous sulphate electrodes (MMS), or saturated calomel electrodes (SCE), and were rinsed with distilled water before being placed in the cell. The counter electrode was always a piece of platinum gauze that had been flame-cleaned before being placed in the cell.

Figure 2.4 shows a typical electrochemical cell. The glass cell has a hole in the bottom for the working electrode to fit through. A rubber O-ring is fitted around the electrode and sits inside a plastic screw cap that keeps the cell from leaking. Figure 2.5 shows a similar electrochemical cell that uses a glass cactus cell instead.

The whole configuration (cell and 3D stage) is setup inside an open Faraday cage.

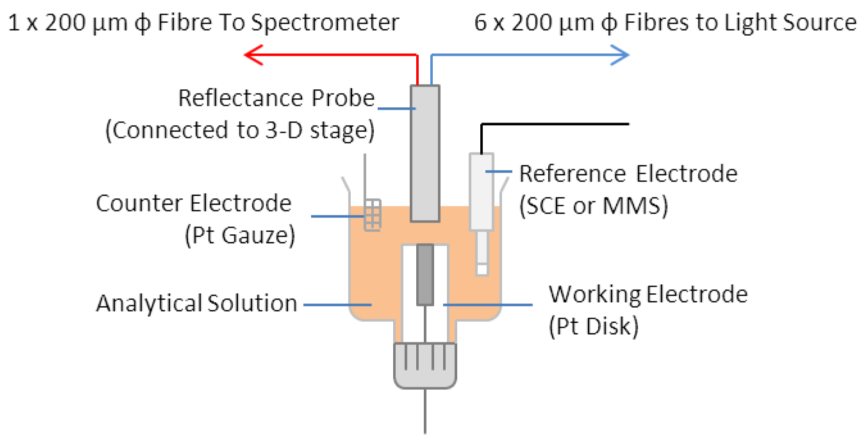

1 x 200 µm ϕ Fibre To Spectrometer
6 x 200 µm ϕ Fibres to Light Source
Reflectance Probe (Connected to 3-D stage)
Reference Electrode (SCE or MMS)
Counter Electrode (Pt Gauze)
Analytical Solution
Working Electrode (Pt Disk)

Figure 2.4: Schematic of a typical electrochemical cell used throughout this work.

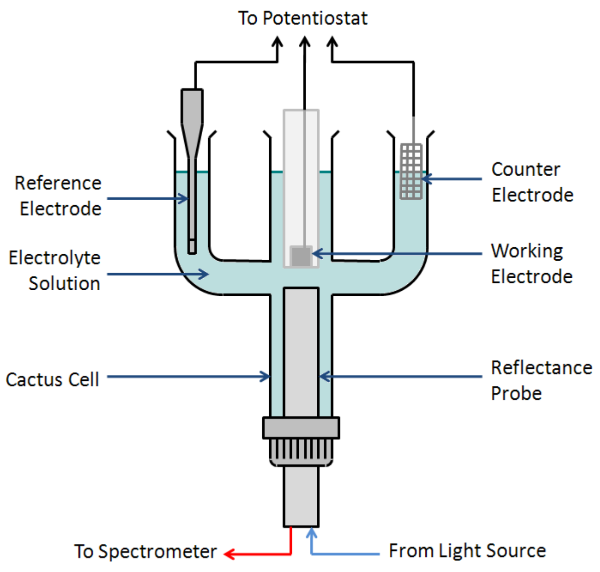


Figure 2.5: Schematic of a typical cactus cell used throughout this work.

Before each experiment, the reflectance probe was properly aligned with working electrode by conducting a series of preliminary scans. This is the systematic measurement of signal at different points above the electrode to produce a pseudo-image of the working electrode, like the plot shown in Figure 2.6. This image is used to find an approximate position for the centre of the electrode. Once above the centre of the electrode, the reflectance probe was moved along the vertical axis while simultaneously measuring the signal, in order to find the optimal distance at which the most signal was received. The distance between the probe and the electrode was typically found to be between 1 mm and 2 mm, as shown in Figure 2.7.

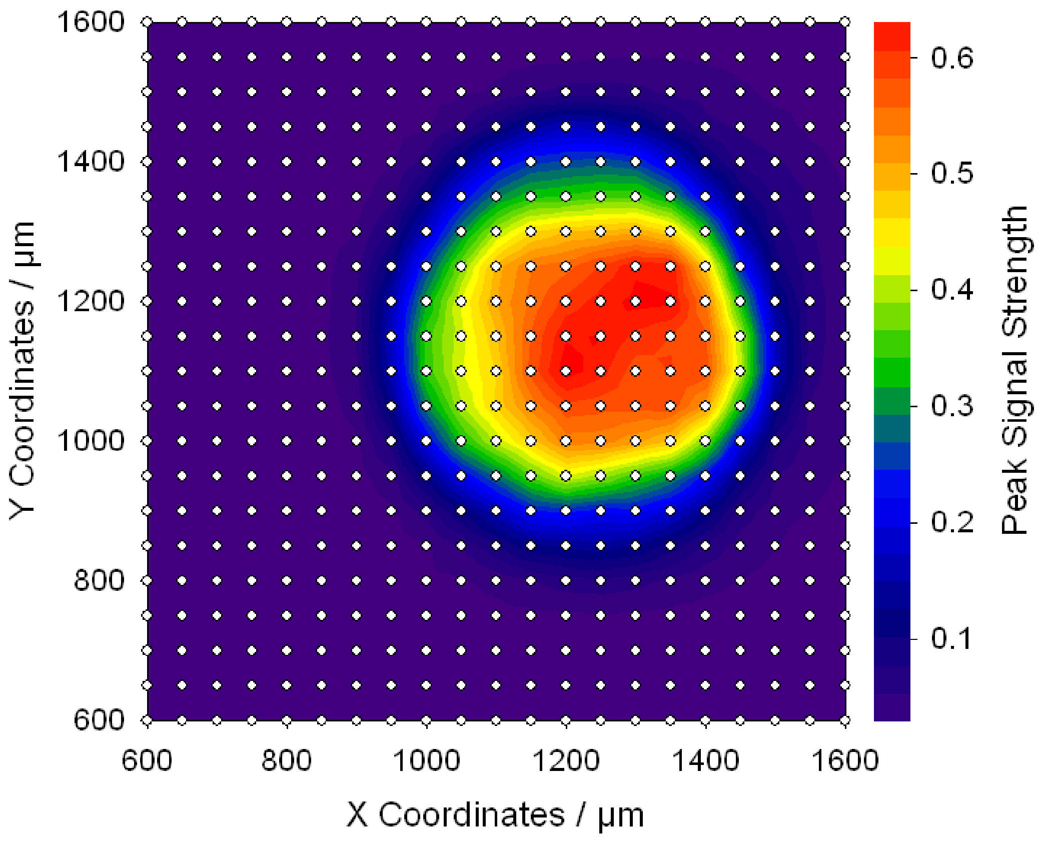


Figure 2.6: Contour plot of peak signal strength as a function of position across a 500 µm diameter disk electrode. White dots indicate each position that the probe was positioned to for acquiring a spectrum. At each point, the peak of the spectrum recorded is plotted. The dots make up a 20 x 20 grid, with each dot separated by a distance of 50 µm. The probe is kept at a distance of 1.5 mm above the electrode at all times. The scan produces a pseudo image of the electrode that is used to locate the centre of the electrode – in this case (1260, 1150).

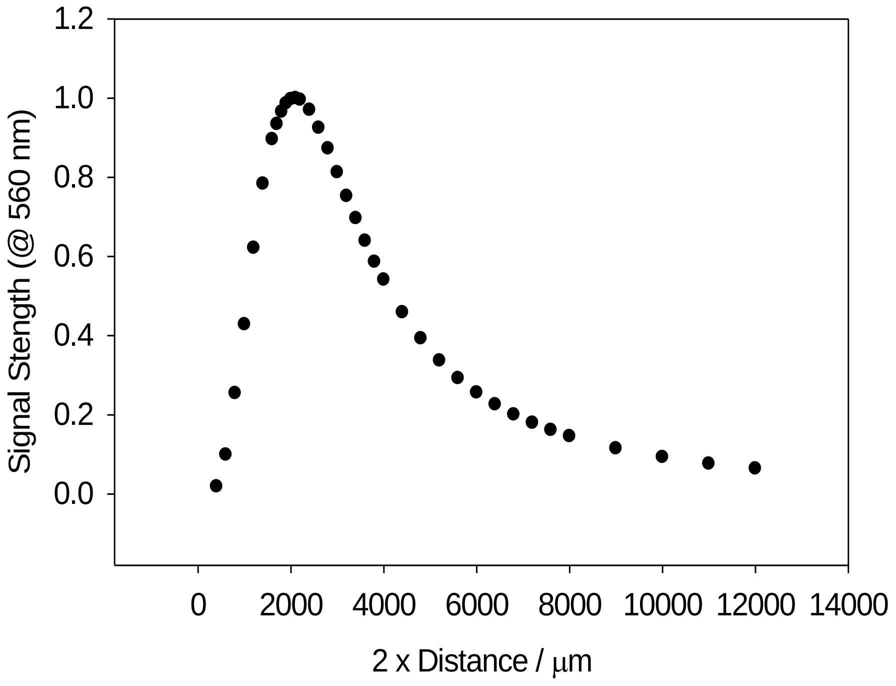


Figure 2.7: Plot of recorded signal strength as a function of double the distance between the probe and the reflective surface (or z – as defined by the Gaussian model). The distance is doubled since the light has to travel to the electrode and back again. The signal strength is normalized to give a maximum value of 1. The peak signal occurs when the probe is > 1 mm from the electrode.

Once the probe was positioned above the centre of the electrode, the light source was switched off and a ‘dark’ spectrum was recorded. This was a recording to make sure background light being recorded is kept to a minimum. Afterwards, the light source was switched on and a ‘reference’ spectrum was recorded. This was the reference spectrum used to generate subsequent absorbance spectra once the experiment began.

2.5 Visual Basic Codes

Simultaneous measurement of absorbance, current, and potential was controlled by several pieces of software written in Visual Basic (see Appendix A).

A1: Potential Sweep and Absorbance Acquisition – used for the simultaneous measurement of current, voltage, and signal strength and was used for all potential sweep experiments in Chapter 4

A2: Potential Step and Absorbance Acquisition – used for the simultaneous measurement of current and signal strength and was used for all potential step experiments in Chapter 4

A3: Scanning Software – used for measuring the signal across an array of x and y coordinates across a given electrode. Was used throughout this work for all the scans that did not require and electrochemistry (e.g. Figure 2.6)

A4: Scanning and Potential Step Acquisition – used for scans that applied a potential step to the electrochemical cell in between each x-y position. Was used for majority of experiments that used the WSWE (Chapter 5), as well as the $Fe(CN)_6$ redox experiment in Chapter 6.

2.6 Microwell Arrays

Microwell arrays were provided by Oxford Nanopore Technologies and featured 128 microwells arranged in a grid made up of 4 rows and 32 columns, with a distance of 250 μm separating each well. Each microwell had a diameter of 50 μm and a depth of 20 μm. Each electrode is connected to a pin on an edge connector (found on one side of the array) via a conductive strip that is visible from above the array. Each microwell is insulated by a reflective polymer. Each column has a number above it (~ 0.1 mm in size), indicating which microwell lied at the top of that column of microwells. The column consisting of microwells 1 to 4 would have a '1' at the top of that column, and the column consisting of microwells 5 to 8 would have a '5', and so on up to 125.

Two types of arrays were used; ones with silver microwell electrodes, and another with platinum microwell electrodes.

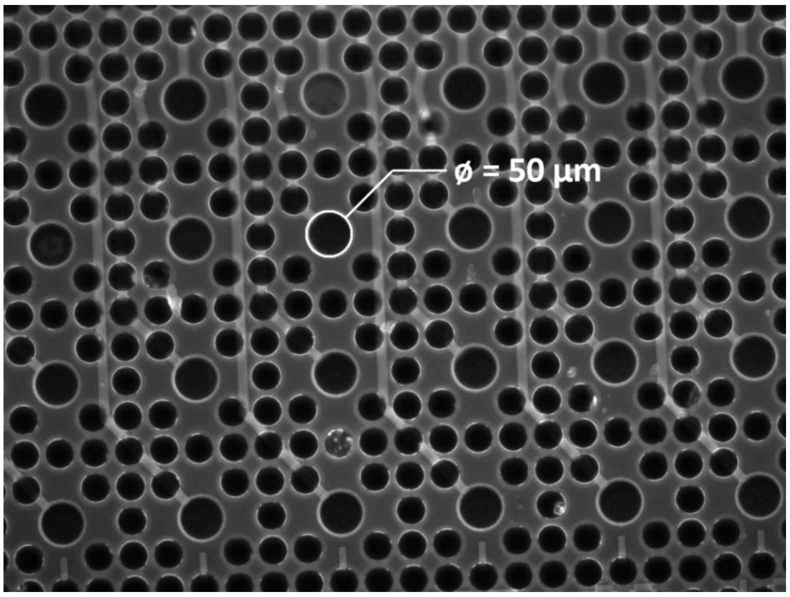


Figure 2.7: An example of a silver electrode microwell array. One of the microwells is highlighted. The smaller circles are smaller 30 μm wells that are believed to help bilayers form across the larger microwells. The light grey lines connected to each microwell are the conductive tracts that lead to a series of output pins (not pictured).

For cleaning, the arrays were rinsed with ethanol, and dried with nitrogen gas (BOC).

2.7 Lipid Bilayers

Pristane oil (Sigma, > 98%) was used as received and stored in the fridge at a temperature of 5°C. The pre-treatment solvent used for producing lipid bilayers was made by mixing one part Pristane to nine parts hexane. This solvent was also stored in the fridge when not in use.

The lipid bilayers were made using Diphytanoyl Phosphatidylcholine (DPhPC) lipids (Avanti Ploar Lipids). The lipids were initially dissolved in C_5H_{12}, at a concentration of 11.8 mM. Once dissolved, the solution was then heated in an oven at a temperature of 50 °C, to allow the C_5H_{12} to evaporate. Afterwards, the lipids were dissolved in an aqueous solution of 0.4 M KCl and 25 mM Trizma buffer at a concentration of 3.6 mg ml^{-1}.

This solution was kept in a series of vials and stored in the freezer. When needed, a vial of lipid solution was defrosted at room temperature. Once defrosted, the solution was kept in an ultrasonic bath for 10 minutes to break up any long lipid chains, and was then ready to use to make bilayers.

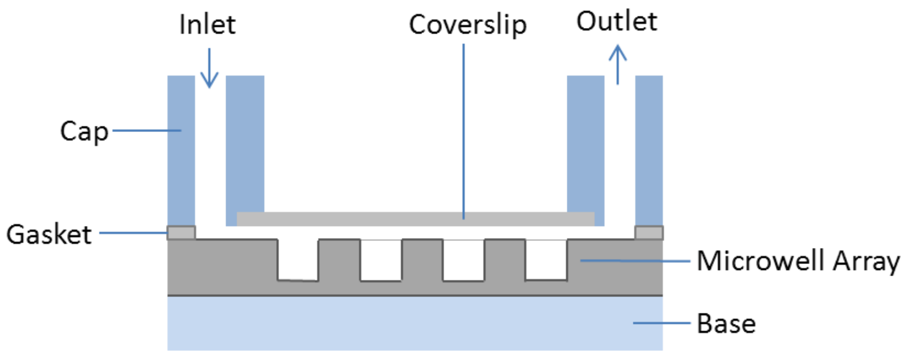


Figure 2.9: Cross-section of the cell used to create bilayers on the microwell arrays. The microwell array is sandwiched between a base below, and a gasket, a glass coverslip, and a cap on top. This configuration is secured together with plastic screws that go through holes in the cap, gasket, and array; and screw into the base. The coverslip is held in place by the force applied by these screws. Fluid can only flow in and out through inlets within the cap.

Figure 2.9 shows the cell used to create bilayers. The cell is made by placing a gasket on top of the array, followed by a plastic cap. The cap provides the inlet and outlet and includes a glass coverslip that ensures that fluid can only enter or leave the cell through these two ports. The gasket helps create an air tight seal, while also acting as the walls of the cell. The cap, gasket and array are held together using plastic screws that screw into a plastic base, which sits beneath the chip.

To make the lipid bilayers, 100 μl of a pre-treatment solvent is slowly passed through the cell (5 to 10 μl s^{-1}) using a handheld syringe. Once the solvent has flowed across the whole microwell array, the syringe is used to suck the mixture back out of the cell. The cell is then left to dry in a ventilated fume hood for 5 minutes. The desired result is each microwell lined with a thin layer of pristane, which acts as an interface between the lipid bilayer and the wall of the microwell.

The next stage is to pass 200 μl of a lipid solution containing dPhPC, Trizma buffer and KCl, slowly through the cell (5 to 10 μl s^{-1}). Once the fluid flows past the whole microwell array, the syringe is used to suck the mixture back out of the cell. The result of this process is each microwell filled with a small volume of lipid solution. Due to the hydrophobic nature of the tails of the phospholipids [45], the interface between the lipid solution and the air above the microwell is packed with aligned lipids as shown in Figure 2.10.

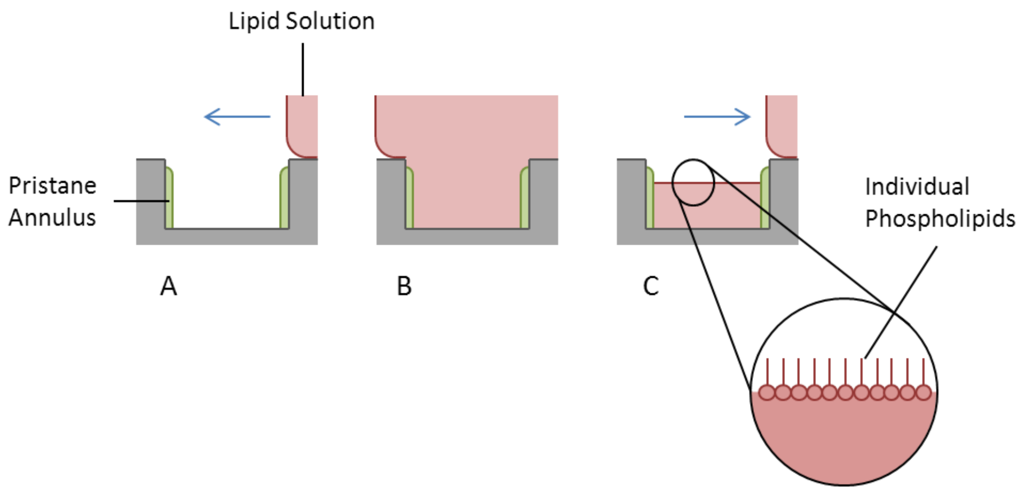


Figure 2.10: Illustration of the early stages of forming bilayers. First the lipid solution is slowly flowed across the array (A), filling up each microwell (B). The lipid solution is sucked back out of the cell, ideally resulting in microwells filled with lipid solution (C).

This process is repeated twice to make sure all the wells are filled with lipid solution. The surface of the lipid solution will also have the tails of the lipids pointing outwards.

The lipid solution is then passed over the array a third time, as shown in Figure 2.11. As the liquid-air interface of the incoming lipid solution passes the liquid-air interface of the lipid solution already in each microwell, the phospholipids aggregate to form an arrangement with the hydrophobic tails buried in the middle, and hydrophilic heads exposed to the water, i.e. a lipid bilayer.

The cell is then left for ten minutes to allow the lipid bilayers to stabilise.

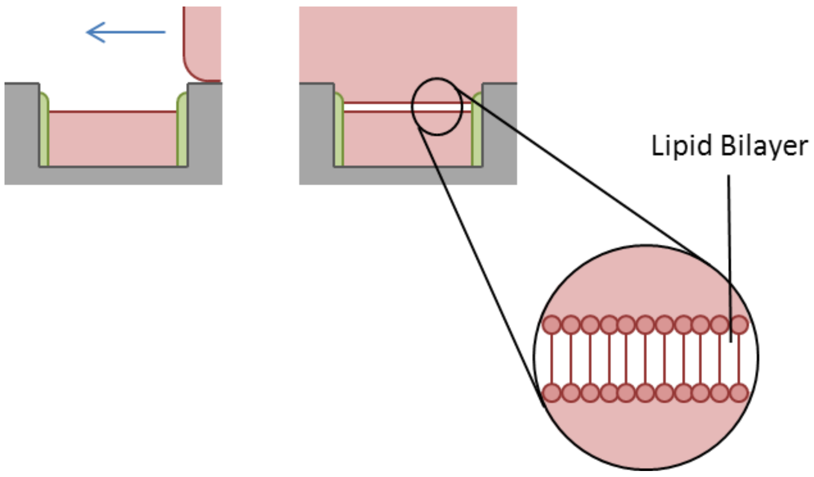


Figure 2.11: Illustration of completed bilayers. The lipid solution is passed over a microwell already containing a lipid solution. The air-liquid interfaces of both bodies of solution are packed with phospholipids with tails pointed outwards. When they come together, the tails aggregate together, trapping a thin packet of air; and forming a lipid bilayer.

Finally, an aqueous buffer solution of 0.4 M KCl and 25 mM Trizma buffer is slowly syringed through the cell, flushing out the unwanted lipid solution that is outside of each microwell. Once all the lipid solution has visibly appeared to have been flushed out of the cell, the volume within the cap and above the coverslip is filled with buffer solution. The screws securing the components of the cell together are loosened; just enough so that the coverslip can be pulled out from between the cap and the gasket.

Once the coverslip is removed, more buffer solution is added to the cell to stop the bilayers from drying out. The desired end result is each microwell containing a lipid, sealed with a lipid bilayer. Outside of these microwells is a colourless buffer solution.

Chapter 3: Understanding the Bifurcated Probe

3.1 Introduction

Bifurcated probes are used in a variety of sensing applications and are used to measure variables such as distance [46], pressure [47], and general optical properties [48]. When the probe is used for the purposes of UV-Vis reflectance spectroscopy, it is important to have a complete understanding of the geometry of both the surface area of the electrode and the volume of the electrolyte that is interrogated by the probe. By knowing these, accurate conclusions about what is happening at (or near) the electrode can be made using the spectroscopic data that the probe provides.

Due to their high versatility, previous efforts have been made to model the behaviour of bifurcated probes [47], [48], [49]. These studies have been able to accurately model the geometries that the probe interrogates, and how the signal received by the probe varies with distance between probe and electrode However, they do not directly model the commercial probes that are used not just for the spectroelectrochemical experiments presented in this work, nor have they been produced with the intended application of reflectance spectroscopy.

However, these same efforts can be adapted to model the behaviour of these commercial probes when they are specifically used for the purposes of UV-Vis reflectance spectroscopy.

3.2 Basic Modelling of the Probe

The bifurcated probe operates by shining light on to a reflective surface and capturing a portion of the light that is reflected back. It does this by using two sets of optical fibres; six outer fibres transmitting light from a source (e.g. a lamp) to the tip of the probe, and one central fibre transmitting reflected light from the surface in question to a spectrometer.

A simple schematic of a typical probe is shown in Figure 3.1. Previous studies have made the behaviour of the bifurcated probe easier to model, by assuming that the reflective surface under inspection is an ideal reflector and reflects the light at a symmetrical angle to the normal [47], [48], [49]. This allows the probe to be visualised as several light-emitting fibres shining on to a 'virtual' detecting fibre beneath, while the electrode itself is largely ignored for now. The distance between the light-emitting fibres and the detecting fibre is the equivalent of twice the distance between the probe and the electrode. A schematic of this visualisation is shown in Figure 3.2.

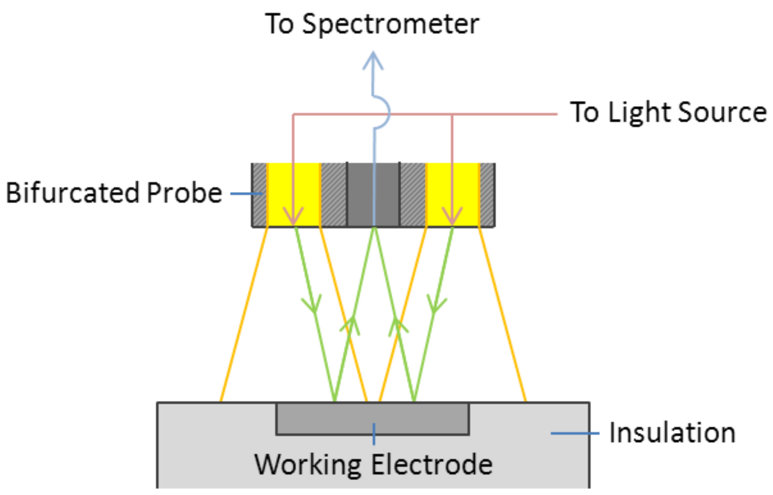


Figure 3.1: Schematic showing a simplified cross-section of a bifurcated reflectance probe that has been configured to emit light from outer fibres (pink arrows), and detect light using a central fibre (blue arrow). Green arrows show a simple path that light must take to travel from the emitting fibres to the detecting fibre using the reflectivity of the working electrode below. The yellow lines outline a typical region between the probe and electrode that would be illuminated by the probe.

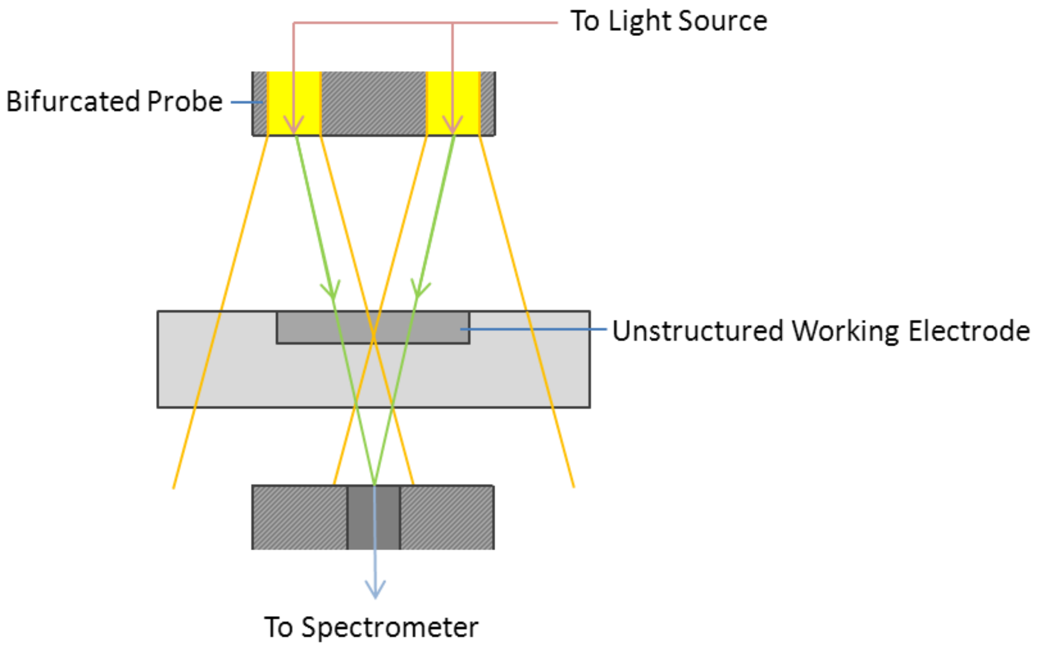


Figure 3.2: Adjusted visualisation of the workings of the bifurcated model, showing the detecting fibre, which is connected to the spectrometer, below a working electrode. Arrows indicate a straight path that the light takes from the emitting fibre to the target fibre.

Previously published work by Schroll *et al.* [32] on UV-Vis reflectance spectroscopy of diffuse redox products at an electrode have observed a relationship between absorbance and charge measurements that deviate from the expected linear relationship, as theorised in Chapter 1 (Equation 1.8). The relationship would start linear, but after a certain time threshold, absorbance would increase at a slower rate with respect to charge, Q, than observed for lower values of Q. Schroll *et al.* reason that this deviation from linearity was occurring because the probe was illuminating an area greater than that of the working electrode. The volume of solution, being interrogated by the probe, extended beyond the diffusion layer of redox products adjacent to the electrode. Schroll *et al.* used a fibre bundle with a diameter of 2 mm and a glassy carbon electrode of 3 mm.

Figure 3.2 demonstrates that any light that reaches the detection probe must travel inwards, towards the central axis of the probe (and electrode). Even though the probe may be illuminating regions beyond the electrode, those regions are not necessarily being interrogated. If the probe is configured to have light emitting fibres lying outside a receiving fibre, and the diameter of the fibre bundle is smaller than that of the working electrode, then all the light that reaches the detection fibre must reflect off of the working electrode; and as such the region under interrogation will always lie within the diffusion layer. Therefore, there has to be another cause for the deviation from the linear relationship between the absorbance and coulometric measurements that has yet to be considered.

This example reinforces the importance to have an in-depth understanding of how the probe behaves; otherwise incorrect conclusion can easily be made.

3.3 UV-Vis Reflectance and Microwells

Figure 3.3 shows a simplified model of the bifurcated probe, where the light emitting fibres are modelled as point sources of light. Light is assumed to move in a straight line from the source, passing through the working electrode, to the target fibre that is connected to the spectrometer.

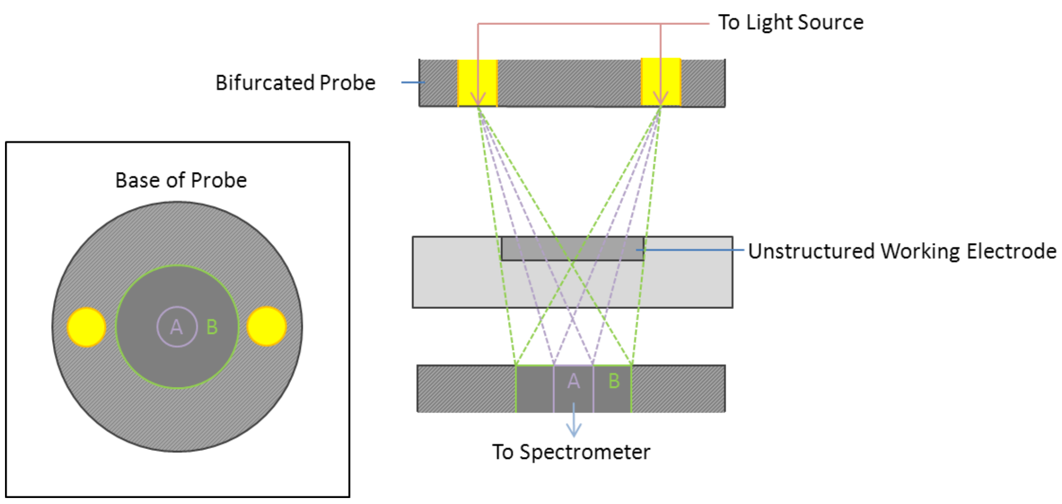


Figure 3.3: Main: Schematic of simplified behaviour of the bifurcated when considering two different diameters for the detecting fibre in the centre, A (purple outline) and B (green outline). The dashed lines show a simplified profile of light that reaches each respective fibre from each of the light sources. For the sake of simplicity, each light emitting fibre is imagined as a point source, and that the light emitted from these sources travel in straight lines. Inset: Top-Down view of the base of the probe.

The schematic demonstrates that when the target fibre has a wide diameter, like in the case of B, the light that has passed through the majority of the electrode. When the fibre has a smaller diameter, like in the case of A, the light that has reached the detecting fibre has passed smaller region of the electrode. By using thinner fibres for the probe, a smaller region of inspection will be achieved. However, it is worth noting that judging by each profile of light at the electrode, areas that are being interrogated lie either side of the centre of the electrode. If the probe was to be placed directly over a microelectrode (which would have a smaller diameter than those of the fibres in the probe), there is a likelihood that it would lay within the gap of the areas being interrogated.

One of the aims of this work is to further investigate the limits of bifurcated probes to inspect small electrode geometries and to find a solution that allows UV-Vis reflectance spectroscopy to be used to inspect the absorbance changes that are caused by redox chemistry occurring at these smaller electrodes.

This chapter will attempt to provide an accurate geometry of the surface areas of electrodes and (adjacent solution volumes) that are interrogated by the reflectance probe in order to better understand what is contributing to the spectral measurements that are recorded, and the limitations of electrode diameters the probe can be reasonably expected to interrogate.

3.4 The Gaussian Beam

Previous analyses that have modelled bifurcated probes are based around the theory that the intensity of light emitted from an optical fibre has a Gaussian profile [47], [48], [49].

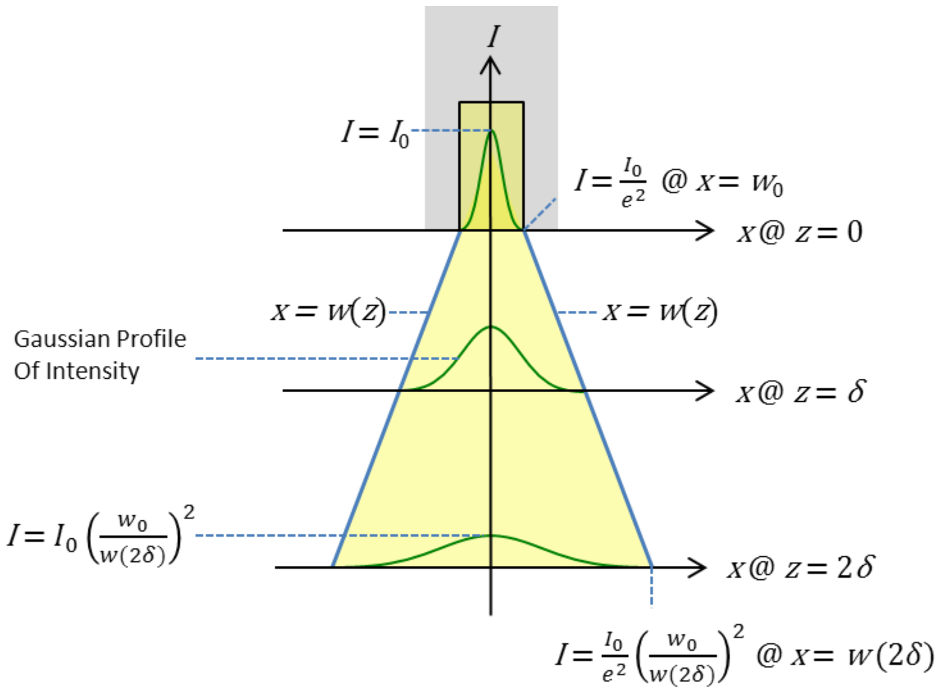

I
$I = I_0$
$I = \frac{I_0}{e^2}$ @ $x = w_0$
x@ $z = 0$
$x = w(z)$
$x = w(z)$
Gaussian Profile Of Intensity
x@ $z = \delta$
$I = I_0 \left(\frac{w_0}{w(2\delta)}\right)^2$
x@ $z = 2\delta$
$I = \frac{I_0}{e^2} \left(\frac{w_0}{w(2\delta)}\right)^2$ @ $x = w(2\delta)$

Figure 3.4: Figure showing how the intensity of light is distributed at certain distances from the tip of a fibre optic cable. The diagram shows that the profile of light at the tip of the fibre has a narrow peak, and that this peak becomes broader and less intense further from the tip of the fibre. Shown are the values of the peak intensity and intensity at x = w(z) for the Gaussian profiles found at z = 0 and z= 2δ. Note: Diagram is not to scale.

This intensity profile can be expressed as a function of the distance along the longitudinal axis, z, and the radial distance from this axis, r:

$$I(r,z) = I_0 \left(\frac{w_0}{w(z)}\right)^2 \exp\left(\frac{-2r^2}{w^2(z)}\right) \qquad (3.1)$$

In this equation, $w(z)$ is defined as the radius of the beam where the intensity is $1/e^2$ of the value along the beam's axis:

$$I(w(z), z) = \left(\frac{1}{e^2}\right) I(0, z) \qquad (3.2)$$

In Equation 3.1, w_0 is simply to $w(0)$. This is the minimum value that $w(z)$ can take and is found at the tip of the optical fibre from which light is emitted.

3.5 A Theoretical Model

Using the Gaussian model, the distribution of light intensity across the 'virtual' detector fibre can be found for different values of z. The target fibre is considered to lie on a plane at a distance z μm away from the probe, and this plane consists of an x and y axis, as shown in Figure 3.5.

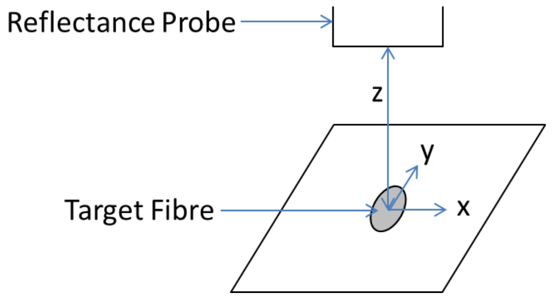


Figure 3.5: Schematic of the x-y plane in which the target fibre is situated at the origin. This plane lies at a distance z from the reflectance probe.

The bifurcated probe used throughout this work is shown as a schematic representation in Figure 3.6. Each fibre has a diameter of 200 μm, and the distance between the centres of each fibre is 215 μm. Each is equally separated around the central fibre. Table 3.1 gives the x and y coordinates of the centre of each fibre on the x-y plane, with the central (target) fibre defined at the origin.

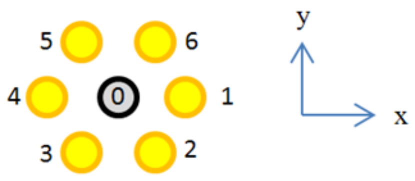


Figure 3.6: Simple schematic showing the arrangement of fibres in the bifurcated probe. Exact coordinates of the centre for each fibre are given in Table 3.1. Note fibres 1-6 are the light emitters while fibre 0 is the receiver.

Fibre No.	**X Coordinate / µm**	**Y Coordinate / µm**
0	000.0	000.0
1	215.0	000.0
2	107.5	-186.2
3	-107.5	-186.2
4	-215.0	000.0
5	-107.5	186.2
6	107.5	186.2

Table 3.1: Positions of each fibre relative to the central fibre in terms of an x and y coordinate.

The profile of light intensity across the x-y plane when the probe shines light on to it from a distance, z, can be found by adding the Gaussian profiles produced by each light emitting fibre:

$$I(x,y) = \frac{k}{w(z)^2} \sum_{n=1}^{6} \left[\exp\left(-\frac{1}{w(z)^2} \times \sqrt{(x - x_n)^2 + (y - y_n)^2} \right) \right] \qquad (3.4)$$

Where k is an arbitrary constant that can be varied when normalising data, x_n is the x coordinate of the centre of light-emitting fibre, n (e.g. $x_1 = 215$ µm), and y_n is the corresponding y coordinate (e.g. $y_1 = 0$ µm).

Figure 3.7 shows the profile of light intensity as a function of x and y for different values of $w(z)$. The intensity values have been normalised so that the maximum light intensity has a value of 1. For the smaller values of $w(z)$, a ring of high intensity light can be seen around the target fibre, with only small a small amount of light actually reaching the target itself. As $w(z)$ increases, the ring dissipates and the light starts to become more intense at the centre of the target fibre. As $w(z)$ gets even larger, the light starts to become less intense across the target fibre as light it spread across a larger area of the x-y plane.

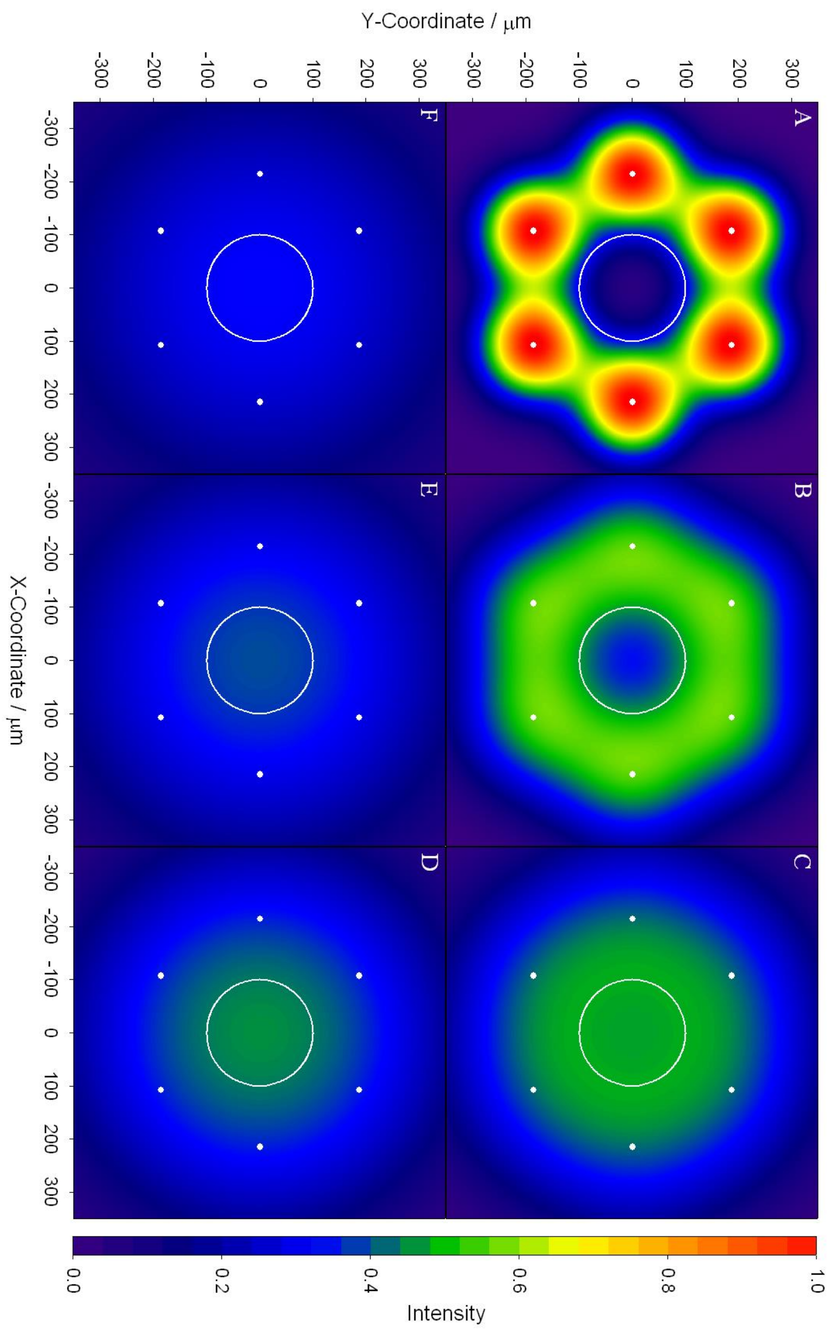


Figure 3.7: Distribution of light across the x-y plane as a function of the x and y coordinates for different values of w(z) (A = 100 μm, B = 150 μm, C = 200 μm, D = 250 μm, E = 300 μm, F = 350 μm). The white line in each plot indicates the position of the outline of the target fibre. White dots in each plot indicate the coordinates of the centre of each light emitting fibre for the probe. Contour plots are constructed from evenly spaced data points; with a distance of 5 μm seperating each point in the x and y direction.

The signal that is measured by the bifurcated probe should be the sum of all the light that reaches the central target fibre. This signal, S, can be found as a function of $w(z)$ by finding the sum of the intensity profile that is bounded by the target fibre – like the outlines shown in Figure 3.7:

$$S = \sum_{\sqrt{x^2+y^2} \leq 100} I(x,y) \qquad (3.5)$$

Where $I(x,y)$ is the same intensity profile as defined in Equation 3.4. Figure 3.8 takes three examples of the contour plots shown in Figure 3.7, but now only considers the light intensity profile that hits the target fibre. Finding the signal measured by the probe as a function of $w(z)$ is simply the sum of all the data points that make up these plots for different values of $w(z)$. These values can then be normalised accordingly.

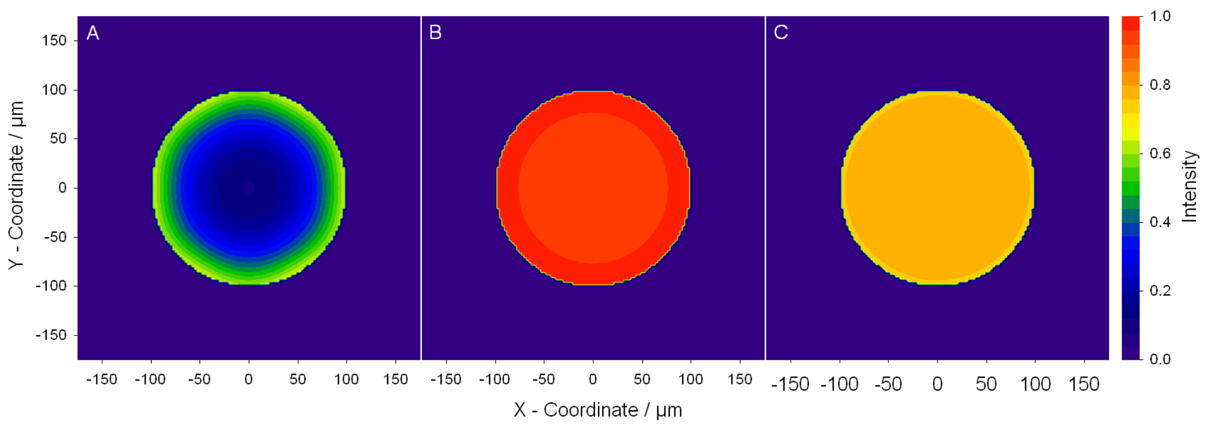


Figure 3.8: Contour plots of light intensity as a function of x and y coordinates for different values of w(z); A = 100 µm, B = 200 µm, and C = 300 µm. Only the light intensity profile within the boundary defined by $x^2 + y^2 < 100^2$ has been plotted. Light intensity outside this region is taken as zero. The theoretical signal detected for each of plots is a function of the sum of the light intensity profile for each contour plot. The signals detected in cases A,B and C have the ratio 0.682, 1 and 0.826, respectively. Contour plots are constructed from evenly spaced data points; with a distance of 2.5 µm seperating each point in the x and y direction.

Figure 3.9 shows how this signal, S, varies with $w(z)$; and has been normalized so the peak signal is equal to 1. The plot shows that when $w(z)$ is smaller than 200 µm, then the probe is too close to the reflective surface to allow for the light to reach the target fibre. Beyond $w(z)$ = 400 µm, the distance between the probe and the reflective surface is large enough to allow the light to reach the target fibre, but the light starts to become too sparsely spread to provide a strong signal. Between the two, at $w(z)$ = 286 µm, the optimal signal is found.

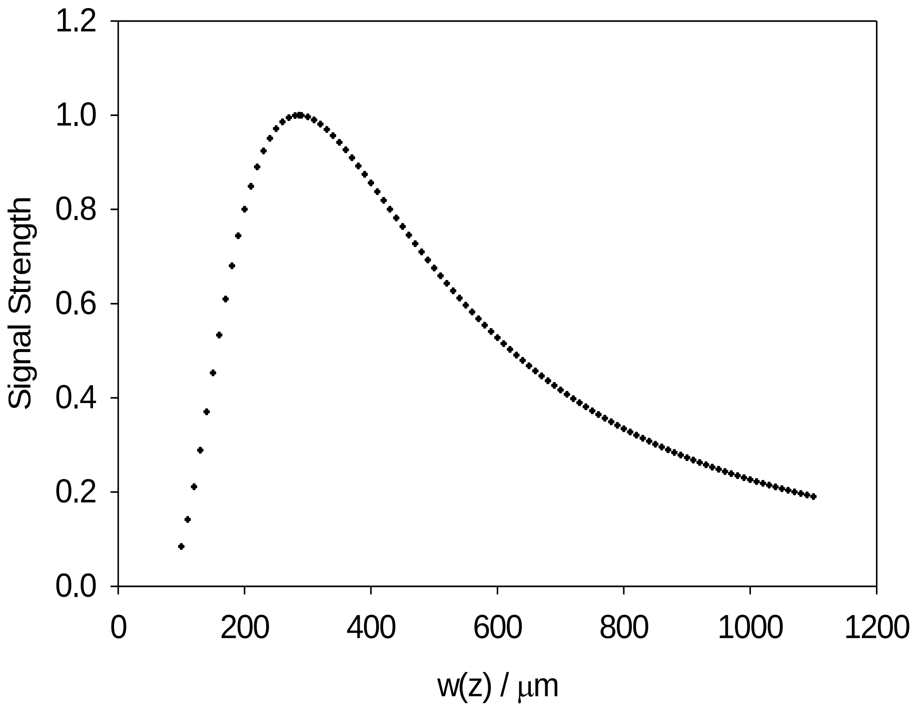


Figure 3.9: Plot of signal strength as a function of w(z) using the theoretical Gaussian model. The signal strength is normalized to give a maximum value of 1.

This model has provided a theoretical relationship between the signal received by the probe as a function of $w(z)$. Real signal data can be used together with this model to find a relationship between $w(z)$ and z.

3.6 Comparisons of Theoretical Model with Real-Life Behaviour

In order to accurately assess the relationship between $w(z)$ and z, the theoretical model of the bifurcated reflectance probe has to be compared with the real-life probe that is used in the experiments. The real bifurcated probe was placed above a smooth flat metal surface, with the six fibres connected to a white light source, and the central fibre connected to a spectrometer. The probe was connected to a stage that can move in 3 dimensions relative to the metal surface, and moved to different distances away from the metal plate while the signal was measured. Figure 3.10 shows how the signal varies with twice the distance between the probe and the plate (distance is doubled since z is described in the theoretical model as the distance between the light emitting fibres and the target fibre). The signal has been normalized to give a peak value of 1. Similar traits can be seen in the graph to the ones that were observed in Figure 3.9. The signal increases quickly as the distance is increased at the start, and exponentially decays when the distance is increased beyond the threshold where the probe records the optimal signal value.

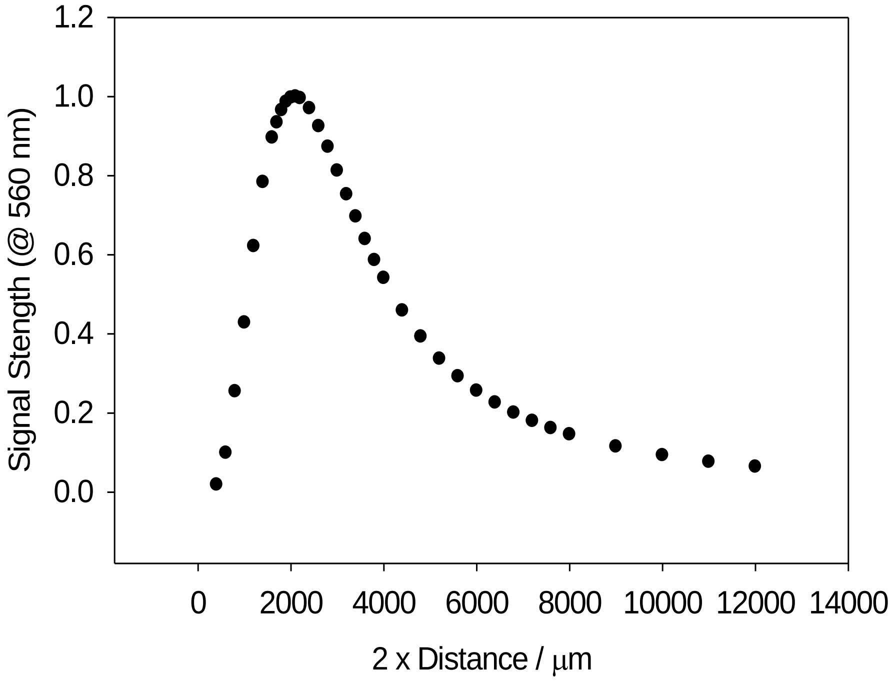


Figure 3.10: Plot of recorded signal strength as a function of double the distance between the probe and the reflective surface (or z – as defined by the Gaussian model). The signal strength is normalized to give a maximum value of 1.

Since the signals given in Figures 3.9 and 3.10 have both been normalized to have peak signals of 1, like-for-like signal values can be directly compared to give a relationship of how the real value of $w(z)$ varies with z. Figure 3.11 shows a graph of this relationship, showing a near linear relationship between z and $w(z)$.

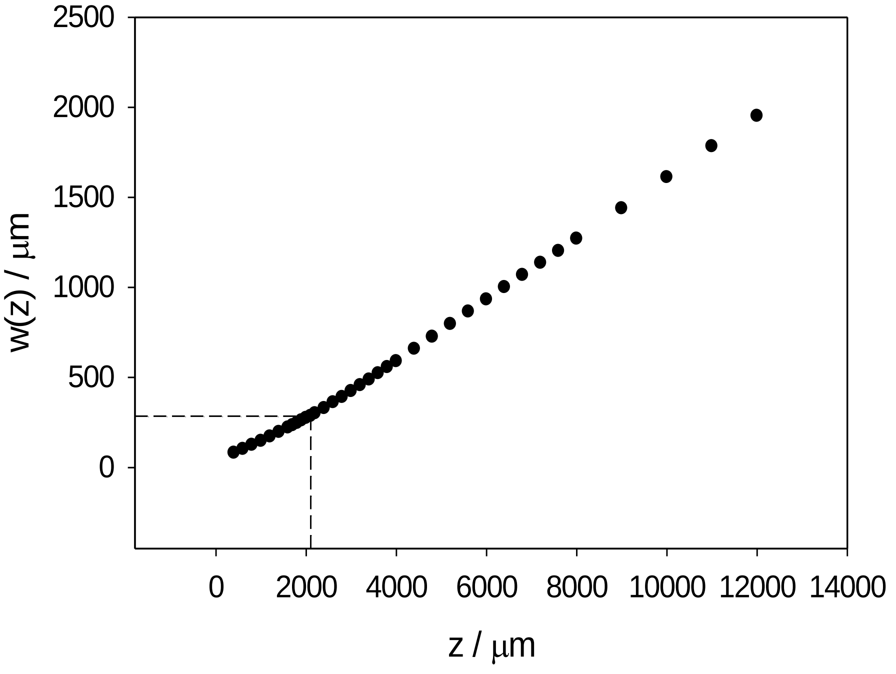


Figure 3.11: Relationship between *z* and *w*(*z*) derived using data from Figures 3.9 and 3.10. Reference lines are taken from the value of *z* and *w*(*z*) that produce the highest signal strength.

A closer inspection on the half dozen points closest to the origin reveals that w_0 may be approximately 40 µm by curve fitting the points to a quadratic curve and finding where it intercepts the *w(z)* axis. A quadratic curve is chosen due to its excellent ability to accurately fit all the data (R^2 = 0.9999), as opposed to a linear plot (R^2 = 0.9956).

Overall, Figure 3.11 gives a good estimation of how much the beam diverges as it moves further and further from the tip of the probe.

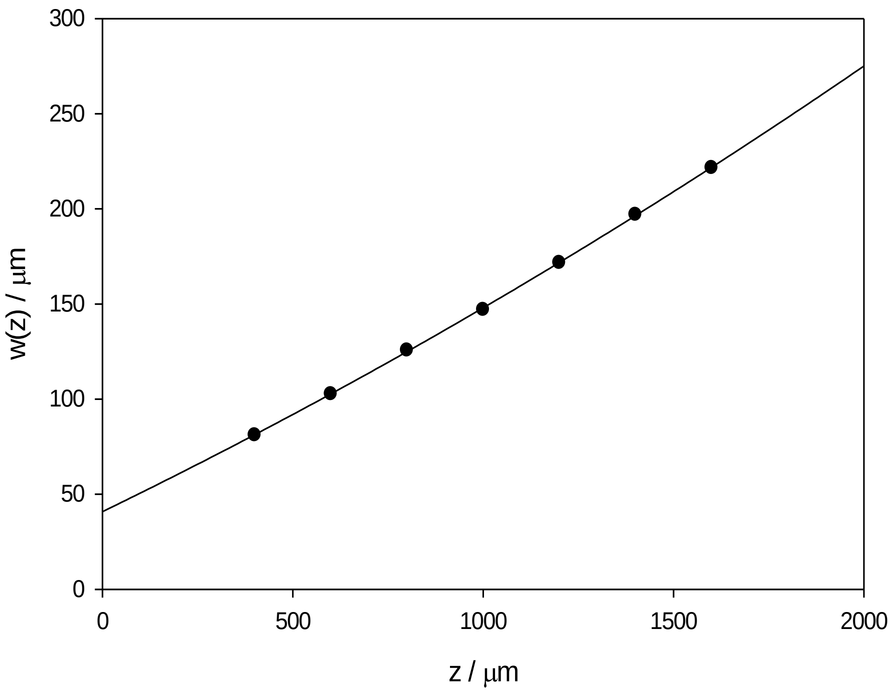


Figure 3.12: Plot of the 7 points closest to the origin in Figure 8. The data has been fitted with a quadratic curve (R^2 = 0.9999), and gives a value of w(z) = 40.8 μm at z = 0.

3.7 Regions of the Electrode under Inspection

Since the relationship between *z* and *w(z)* is known, a scenario can be considered where the probe is placed at a distance above an electrode where the maximum signal is being recorded. When a beam of light from the probe reaches the target fibre, the value of *w(z)* for that beam will be equal to approximately 285 µm. Half way between the light-emitting fibres and the target fibre lies the electrode, which the light is reflecting off of. Using Figure 3.12, *w(z)* at this point should be equal to 153 µm. Using this value of *w(z)*, the profile of light intensity across the surface of the electrode can be found, and more specifically, the profile of light intensity that is hitting the target fibre.

Figure 3.13 shows the light intensity profile across the reflective surface being interrogated. Again, the intensity has been normalized to give maximum values of 1. This light forms a ring across the reflective surface, with the centre of it being directly underneath the centre of the probe.

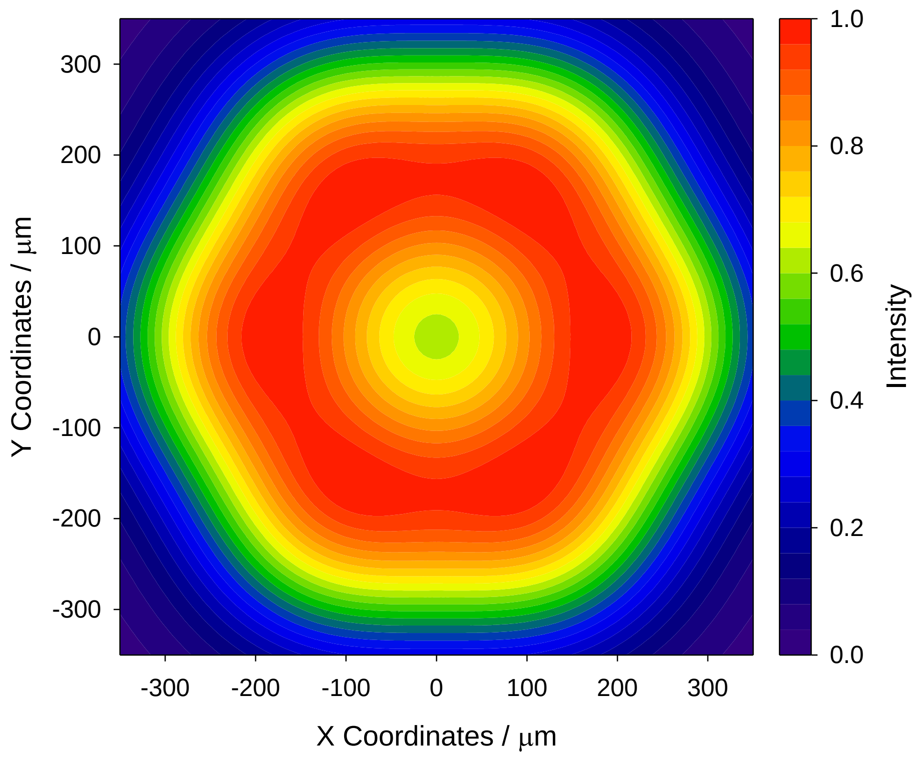


Figure 3.13: Theoretical distribution of light across a reflective surface as a function of x and y coordinates. *w(z)* is taken as 153 µm. Values of intensity are normalized so the maximum value is 1.

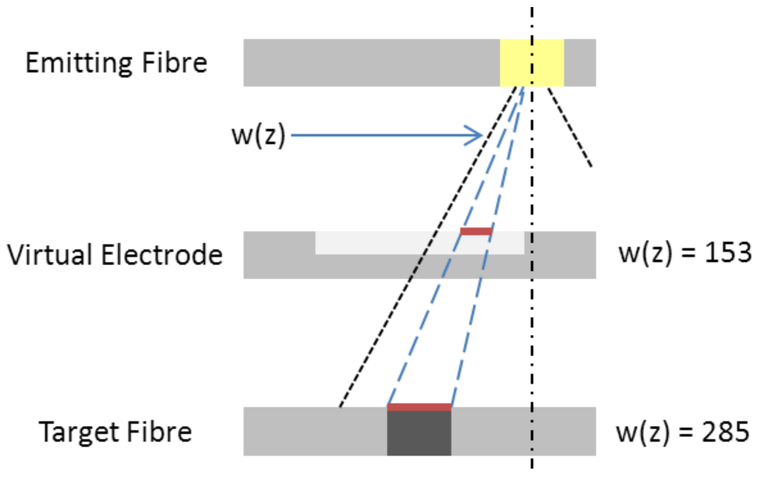


Figure 3.14: Diagram demonstrating how light that reaches the target fibre is distributed over the reflective surface. Light that reaches the target fibre is bounded by the blue dashed lines.

Figure 3.14 shows how the light that will reach the target fibre can be found by considering the light emitted from one fibre, e.g. Fibre 1. To begin with, the central axis of the beam emitting fibre is taken as the z-axis. At the point along the z-axis where $w(z) = 285$ μm, the centre of the detection fibre lies approximately 215 μm away (as defined in Table 1) and that fibre has a radius of 100 μm. The signal measured is the sum of the light intensity profile that lies within this boundary.

Assuming no light energy is lost, it is reasonable to think that this boundary must be scaled down along the z-axis by a factor, *k*, such that the signal always remains the same. At the point along the z-axis where w(z) is equal to 153 μm, k has to equal 0.54. The boundary is a circle with a radius of 54.1 μm, whose centre lies 116.4 μm from the z-axis. Figure 3.15 illustrate the light intensity profile for both these instances, and the total intensity in both graphs is the same.

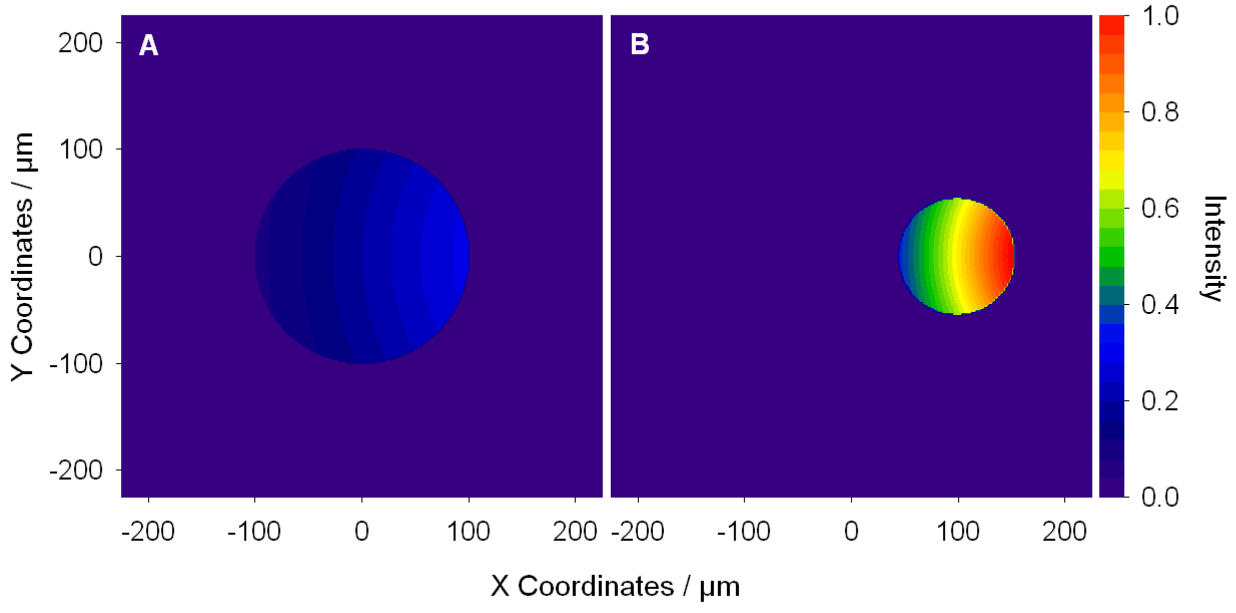


Figure 3.15: Contour plots of light intensity as a function of x and y coordinates for different values of w(z); A = 285 μm, B = 153 μm. The light intensity plot in Plot A is bounded by a circle with a 200 μm and its centre at (0, 0). The light intensity plot in Plot B is bounded by a circle such that the sum of the light intensity profile in each plot is the same.

Applying this process to each fibre gives the region that inspected on the reflective surface and is shown in Figure 3.16.

The figure shows that the area of the electrode under interrogation by the probe is a ring-like shape about the central axis of the reflectance probe. It shows that there are smaller regions along the outside of the ring (about 150 μm from the central axis of the probe) that have a relatively high influence on the signal, while the regions closer to the centre have a smaller impact on the signal. The area that this probe inspects is approximately 0.053 mm^2.

This area of interrogation will not change significantly as the probe is moved further away from the electrode. The graph in Figure 3.11 shows that the relationship between $w(z)$ and z tends towards a linear one as z increases. Therefore it is expected that $w(z)$ at the electrode will tend towards 50% of $w(z)$ at the detecting fibre as the distance between the electrode and the probe is increased (instead of the 53% that was predicted when $w(z)$ is 285 μm.

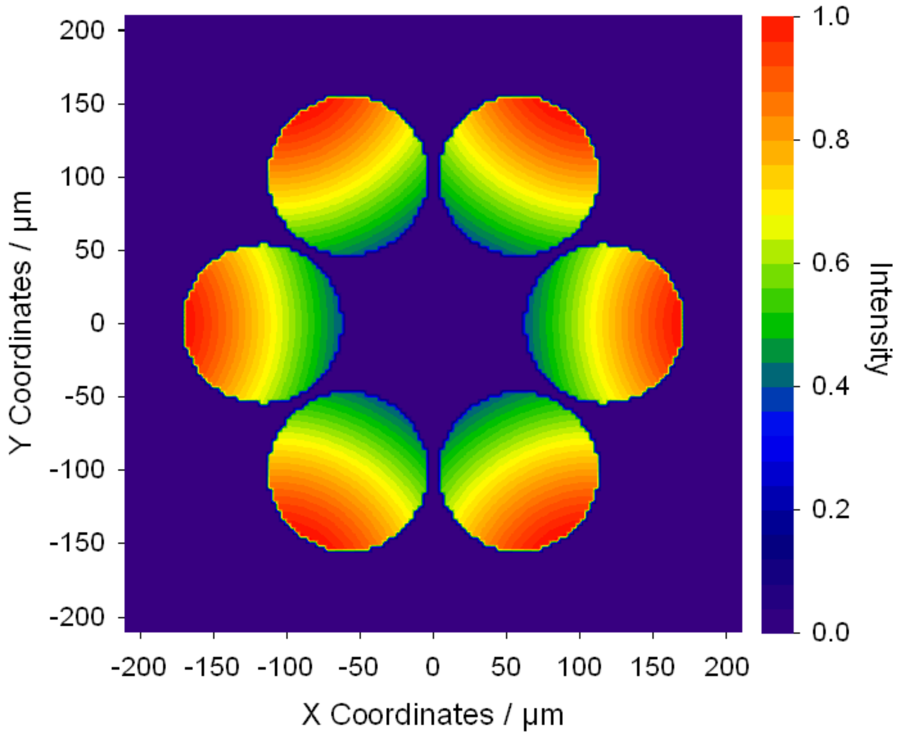


Figure 3.16: Intensity of light that reaches the target probe (and contributes to the signal recorded by the spectrometer) as a function of x and y coordinates. All values are normalized to give a maximum intensity of 1. Contour plots are constructed from evenly spaced data points; with a distance of 2.5 μm seperating each point in the x and y direction.

In order to reduce this area, a different bifurcate probe could be selected that makes used of only two fibres instead of seven. On fibre that transmits light, and another that receives the reflected signal. The area could be reduced further still by using thinner fibres.

However, this would result in a very low signal (~ 84% lower due to few light emitting fibres and a reduced by another factor depending on how thin the fibres are). The lower signal would make spectral measurements vulnerable to noise.

3.8 Example Scans

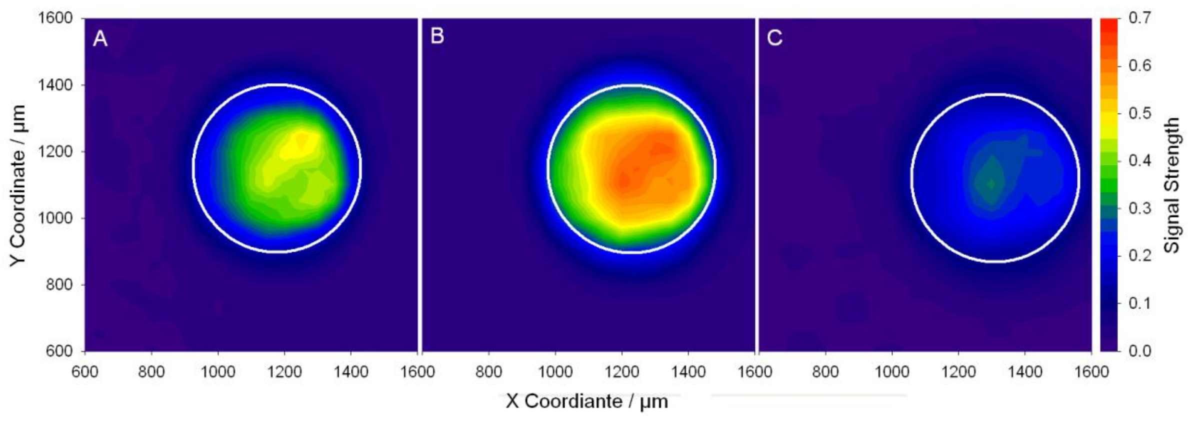


Figure 3.17: Scans of a 500 µm diameter platinum disk electrode sealed in glass, using the bifurcated probe. Plots use varying distances between the probe and the electrode; A = 1 mm, B = 1.5 mm, and C = 3 mm. Data points are separated by distances of 50 µm along the x and y axes. White circles have a 500 µm diameter and indicate the approximate outline of the electrode. No fluid media lies between the probe and the electrode – only air.

Figure 3.18 shows three scans made using the bifurcated probe and a 500 µm diameter disk electrode. Each scan is taken with a different distance between the probe and the electrode. The plots show that as the distance is increased, the electrode appears to drift slightly in the positive x direction. This is caused by either a minor misalignment of the probe, which is not pointing directly downwards, or the electrode is not perfectly flat.

However, the plots show signal patterns as predicted by the model; mainly that the probe is detecting a signal even when probe is moved beyond the boundary of the electrode. For example, plot B shows a wide blue ring outside the white line indicating where the boundary of the electrode approximately lies. The probe is detecting a signal even when it is more than 100 µm away from the electrode, as expected according to Figure 3.15.

3.9 Conclusions

The approximate region of the electrode that is under inspection extends a region with a maximum diameter of 300 μm. Therefore a working electrode with a diameter of 500 μm can be used; knowing that any redox products produced will form within the region of interrogation when the probe is above the centre of the electrode.

However, one of the more unintuitive findings is that there is a region directly beneath the probe that is not inspected. This region approximately equates to a circle with a diameter of 100 μm. The immediate implication of this finding is that if there was a desire to conduct a UV-Vis reflectance investigation on a microelectrode, the probe would have to be placed in a position that was slightly shifted 100 μm from directly above that electrode.

However, by doing this, a large percentage of the area that the probe is interrogating would be covering relatively unreflective glass insulation. Therefore, the signal picked up by the probe would be incredibly small and susceptible to noise. A 50 μm diameter electrode would only fill 0.6% of the area that is interrogated by the probe – making the bifurcated probe incredibly unsuitable to investigate spectral changes occurring at microelectrodes. If an attempt was made to inspect a microelectrode, it is most likely that the scan would produce 6 separate lobes; similar to the plot of Figure 3.16. However, noise would most likely be too dominant and no clear pattern would be observable.

Chapter 4: Quantitative Spectroelectrochemical Analysis of Diffusion Based Redox Reactions

4.1 Introduction

This chapter investigates the quantitative relationship between the electrochemical charge passing through the electrode and the absorbance measurements recorded above the surface of the electrode [50]. This has been attempted before [29], and more recently Shroll *et al.* [32], attempted *in-situ* UV-Vis reflectance spectroscopy (using a bifurcated reflectance probe) on a reversible redox couple ($[Fe(CN)_6]^{3-}$/$[Fe(CN)_6]^{4-}$) and quantitatively link the absorbance data with electrochemical data. A potential was applied to a disk electrode where by the redox species was oxidised, and absorbance changes occurring at the electrode were measured. The work concluded that at the start of this oxidation process, electrochemical and absorbance measurements followed a linear relationship as dictated by the equation:

$$A_\lambda = \frac{2Q(\varepsilon_B - \varepsilon_A)}{nF(\pi r^2)} \qquad (4.1)$$

where all symbols have their usual meaning. However, after a certain time threshold, the data starts disobeying this relationship and the absorbance is observed to increase at a slower rate than the electrical charge that has passed through the electrochemical cell. As previously explained in Chapter 3, Shroll *et al.* concluded that the diffusion field of the absorbing redox products was due to the optical configuration of the cell, but this has been shown to be incorrect.

The work in this chapter attempts to reproduce the effects observed, and find out what may be causing this divergence from the linear relationship as predicted by Equation 4.1. The work in this chapter will also serve as a platform to build upon when it comes to accurately measuring spectral changes occurring in microwells like the ones in the arrays used by Oxford Nanotechnologies.

4.2 Iridium System

For spectro-electrochemical experiments, an $[Ir(Cl)_6]^{3-}$/ $[Ir(Cl)_6]^{2-}$ redox couple was used, since it is well-characterised [51] and the oxidation process produces a strong colour change. At the 489 nm wavelength, $[Ir(Cl)_6]^{2-}$ has a molar extinction coefficient of 3200 M^{-1} cm^{-1}, while at the same wavelength $[Ir(Cl)_6]^{3-}$ only has a molar extinction coefficient of about 10 to 20 M^{-1} cm^{-1} [52]. Therefore, any electrochemical oxidation should produce an easily detectible absorbance change at the working electrode over a short time-scale.

$[Ir(Cl)_6]^{2-}$		**$[Ir(Cl)_6]^{3-}$**	
λ_n / nm	**ε_n / M^{-1} cm^{-1}**	**λ_n / nm**	**ε_n / M^{-1} cm^{-1}**
575	460	615	7.5
489	3200	560	10
431	2540	415	76
414	2480	356	64

Table 4.1: Extinction coefficient at different wavelengths for $[Ir(Cl)]^{2-}$ and $[Ir(Cl)]^{3-}$ species [52], [53]

A series of experiments used a cactus cell filled with an electrolyte of 5 mM $K_3[Ir(Cl)_6]$ and 1 M NaCl and used a three electrode system. The working electrode used was a platinum disk (varying diameter disk electrodes are used throughout, see the appropriate figure legend for exact details), while a saturated calomel electrode was used as a reference electrode, and a piece of platinum gauze was used as a counter electrode.

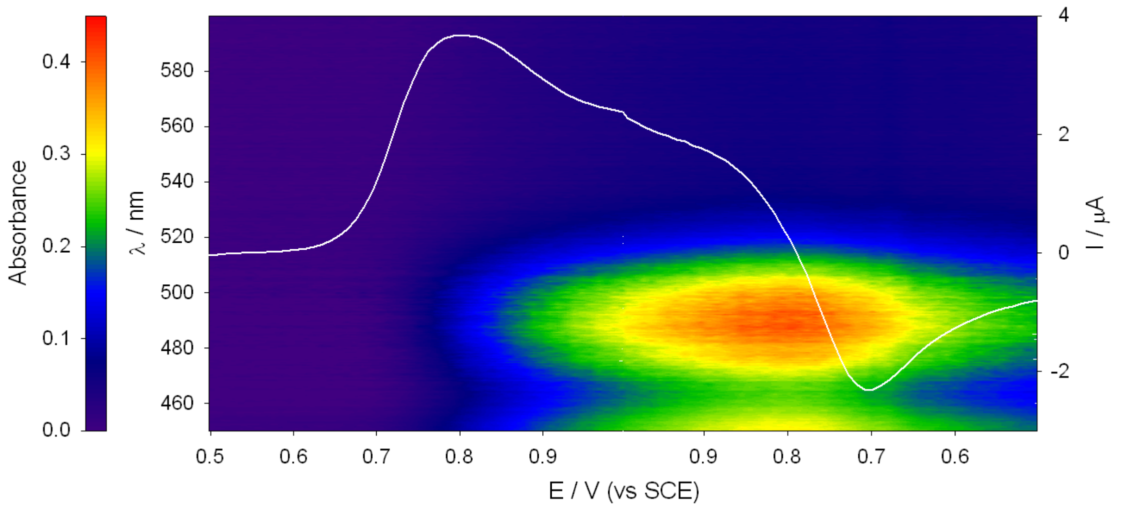


Figure 4.1: Contour plot of absorbance and line plot of current fore a potential sweep on a 1 mm Pt disk electrode in 5 mM $K_3[Ir(Cl)_6]$ and 1 M NaCl at 20 mV s^{-1}. The solution was bubbled with argon for 20 minutes, and a temperature of 21 °C. The reflectance probe is 2 mm above the centre of the electrode.

Figure 4.1 shows the absorbance and current response for a potential sweep applied to a 1 mm diameter Pt electrode in an electrochemical cell containing an aqueous solution of $[Ir(Cl)_6]^{3-}$ species. As oxidation of the $[Ir(Cl)_6]^{3-}$ species occurs, an increase in current is accompanied by a rise in absorbance. Note, the peak in absorbance at any given voltage is found at the 490 nm wavelength.

The peak of the absorbance plot occurs when the current passes reaches 0 μA on the return sweep. This is as expected since this will be the point where the charge that has passed through the cell is at its highest. Beyond this point, a negative reduction current starts to pass through and the $[Ir(Cl)_6]^{2-}$ starts to undergo reduction back to the $[Ir(Cl)_6]^{3-}$ species.

Figure 4.2 shows individual absorbance spectra measured at different points along the potential sweep; with the time taken between recording each of these spectra being close to identical. The graph shows after the potential reaches 0.9 V on the forward sweep, absorbance increases at a slower rate, which would be expected since the peak current has already been observed by this time.

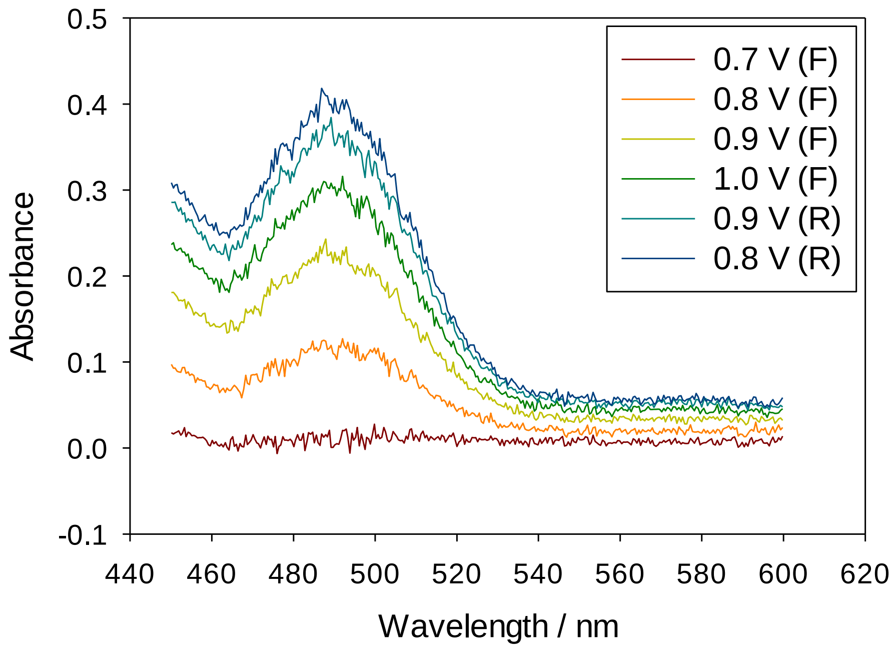


Figure 4.2: Plot of absorbance measured by the probe as a function across the range of wavelengths between 450 nm and 600 nm, for different potentials of the same forward (F) and reverse (R) potential sweeps shown in Figure 4.1. The integration time was 0.1 s, and the average of 2 measurements were taken.

4.3 Absorbance Variation with Probe Distance

Equation 4.1 implies that the distance between the probe and the electrode should not have any effect on the absorbance values that are measured, providing that the area that is being interrogated lies within the boundary of the disk electrode. In Chapter 3, it was shown that moving the probe further from the electrode reduces the incoming signal. Since absorbance is calculated as a ratio of the signal before and after an electrochemical event, the absolute value of the signal before the event should be irrelevant.

To help verify the accuracy of this behaviour, the potential sweep setup was used, but with varying distances between the reflectance probe and the working electrode disc.

Figure 4.3 shows the current and absorbance plots for three different potential sweeps; each one using a different distance between the electrode and the probe. The distances used were 1 mm, 3 mm, and 5 mm. All three current plots seem identical, and the general shape of each absorbance plot seems the same too, with absorbance peaks occurring in all the same places. All other experimental variables were kept the same.

The most distinctive difference between each contour plot is their smoothness. As the probe moves further away from the distance where the optimum signal is received, the absorbance data measured becomes increasingly noisy. The spectrometer itself is susceptible to noise, so as the probe is moved further from the electrode (and the signal values recorded are lower) this noise becomes more apparent.

Figure 4.4 shows the base signal measured for each experiment in Figure 4.3 by which all absorbance measurements are calculated. The figure shows that the signal being received by the probe when it is 3 mm away from the electrode is almost an order of magnitude weaker when the probe is 1 mm away. The signal is almost another order of magnitude smaller when the probe is moved further away by an extra 2 mm. At the distance, oxidation products are only going to cause a small reduction in signal strength compared to the level of noise in the spectral measurements. As a result, absorbance measurements become dominated by noise.

Overall, the data shows that the magnitude and position of absorbance peaks are unaffected by the distance between the probe and the electrode. The only significant effect that the distance between the probe and the electrode causes is the quality of the data that is gathered. As the probe is moved further from the electrode, the strength of the signal gets weaker, and becomes noisier. However, the overall behaviour of what the absorbance measurements are doing with respect what is happening electrochemically remains the same.

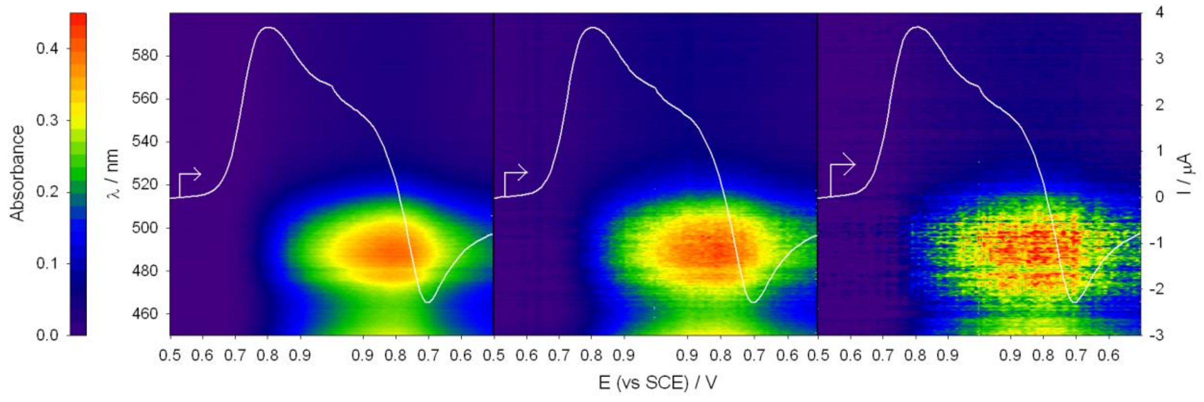


Figure 4.3: Three plots of current (line) and absorbance (contour) as a function of potential applied to a 1 mm Pt disk electrode in 5 mM $K_3[Ir(Cl)_6]$ and 1 M NaCl at 20 mV s^{-1}. The reflectance probe is held at a different distance from the electrode in each case; 1 mm (left), 3 mm (centre), and 5 mm (right).

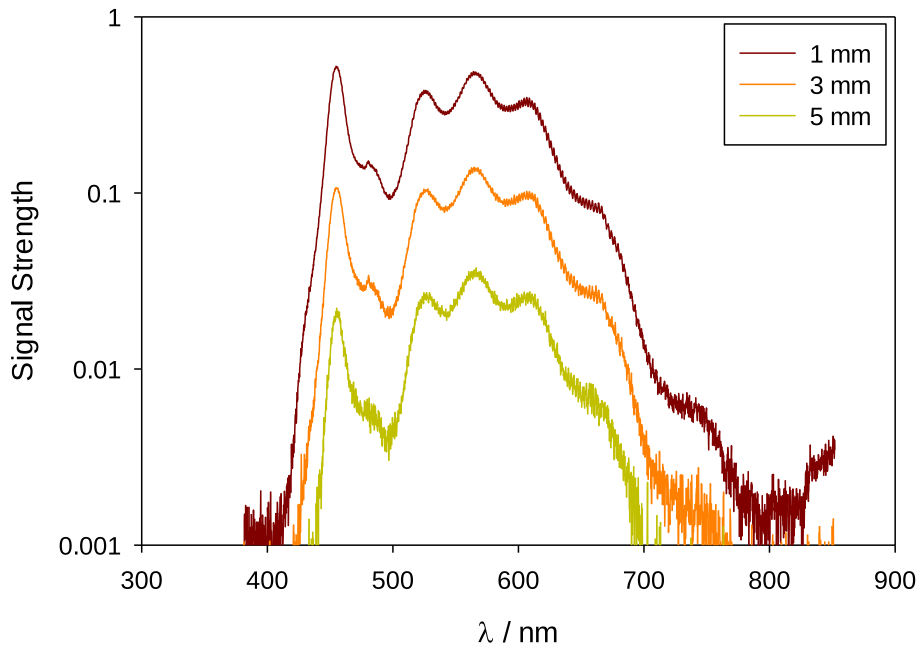


Figure 4.4: Plots of signal strength as a function of wavelength when the probe is at three different distances away from the probe; 1 mm, 3 mm, and 5 mm.

4.4 The Linear Relationship between Charge and Absorbance

To demonstrate the linear relationship between charge and absorbance described in Equation 4.1, the same electrochemical cell as used in Section 4.3 was employed, but instead used an 8 mm diameter platinum disk electrode. The bifurcated probe placed about 1.6 mm above the centre of the electrode disk, where the maximum signal was received.

For this experiment, a potential step was applied to the cell, such that the rate of oxidation of the $[Ir(Cl)_6]^{3-}$ species would be diffusion controlled. The potential was stepped up from 0.5 V to 1.0 V (*vs.* SCE), and held at that potential for 10 s and absorbance and current were recorded. Afterwards, recording was stopped while the potential was stepped back down to + 0.5 V to allow the oxidised species to be reduced back again; returning the electrochemical cell back to its initial conditions. The potential step was repeated and data recorded another four times.

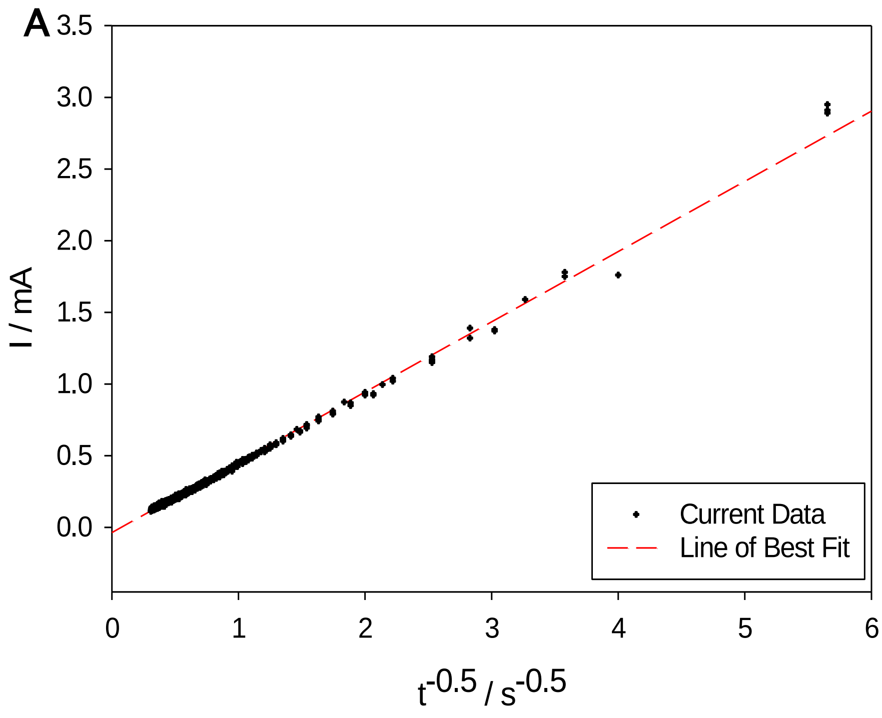
Figure 4.5 shows two sets of data for this experiment. Graph A shows the current response with respect to the inverse square root of time (i.e. a Cottrell plot) from when the pulse was applied. Graph B shows the absorbance response recorded at the 489 nm wavelength as a function of the electrochemical charge that has passed through the cell. This charge was calculated using the chronoamperometric data together with Simpson's Rule. Linear regression has been applied to both graphs, and lines-of-best-fit are shown in red.

The current response shows strong agreement with the Cottrell Equation [16], which states that the current should increase proportionally with the inverse of the square root of time from which the potential step is applied.

$$i = \frac{nFD^{0.5}c_R(\pi r^2)}{\pi^{0.5}} \times t^{-0.5} \qquad (4.2)$$

Using this equation, taking *n* as 1 and Faraday's constant as 96485 C mol^{-1} and the diffusion coefficient of $[Ir(Cl)_6]^{2-}$ was found to be 1.28×10^{-5} cm^2 s^{-1} at a temperature of 22 °C; considerably larger than the expected value taken from the literature of 7 x 10^{-6} cm^2 s^{-1} [54], [55].

The absorbance response shows a strong linear relationship with the electrochemical charge that has passed through the cell, as described in Equation 4.1. Using the afore mentioned values of n and F, and r as 4 mm, the a molar extinction coefficient was found to be 3000 M^{-1} cm^{-1} – a close value to that described in the Table 4.1. It is worth noting that this is not an absolute value of the molar extinction coefficient of the $[Ir(Cl)_6]^{2-}$ species, but a relative value with respect to the $[Ir(Cl)_6]^{3-}$ species.



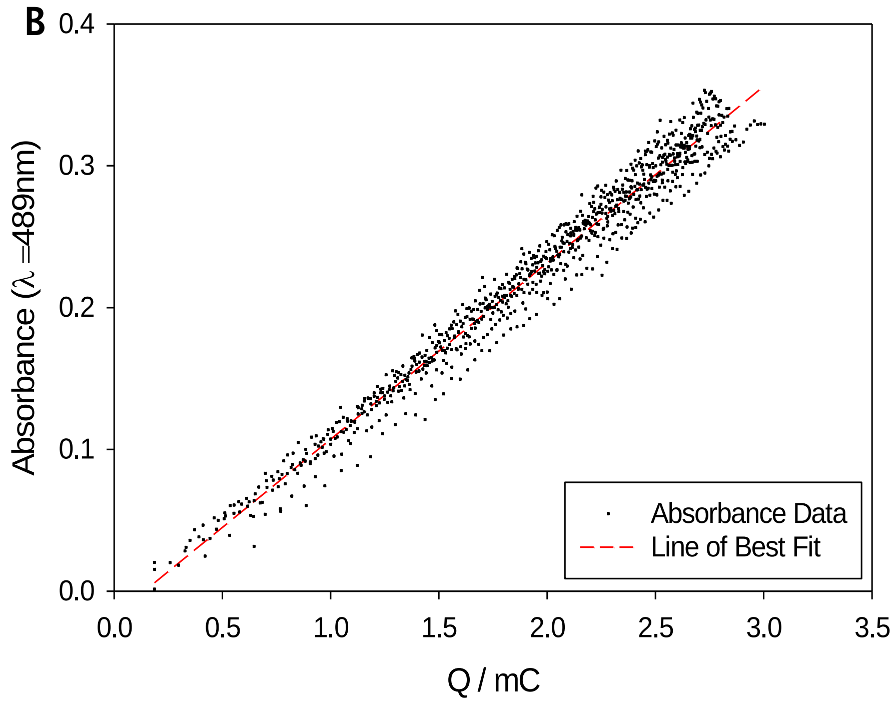


Figure 4.5 : (A) Current responses of 8 mm diameter Pt working electrode disk in aqueous solution of 5 mM $K_3[Ir(Cl)_6]$ and 1 M NaCl when potential step from 0.5 V to 1.0 V (*vs.* SCE) is applied (R^2 = 0.995). Data from 5 separate runs is presented. (B) Absorbance measurements taken from the same experiment, as a function of charge passed through the working electrode (R^2 = 0.973).

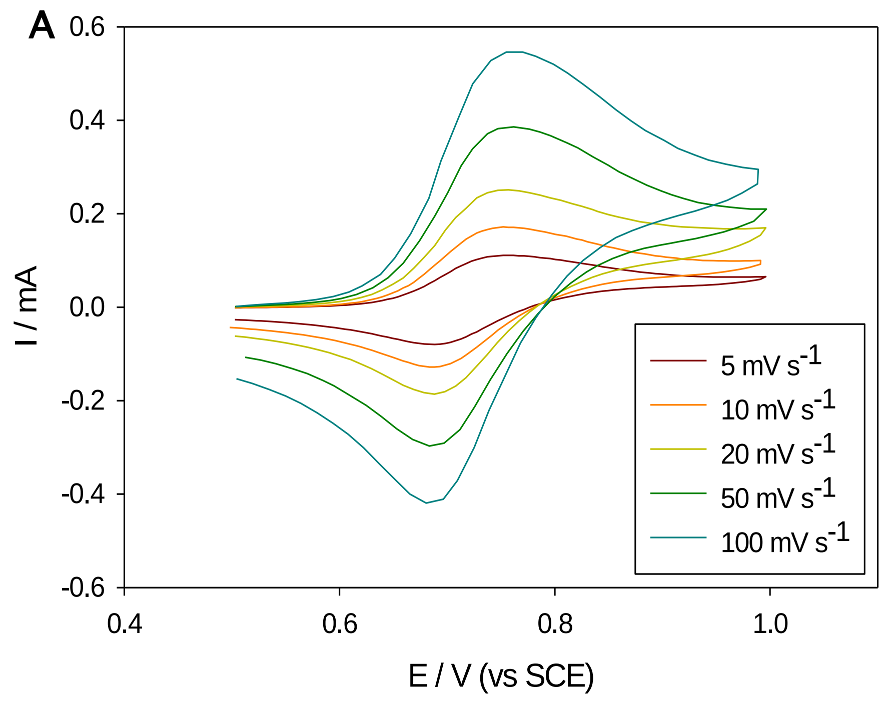
Next, a series of potential sweeps were applied to the same electrochemical cell. The potential was swept from 0.5 V to 1.0 V at sweep rates varying between 5 mV s^{-1} and 100 mV s^{-1}. Figure 4.6 shows two sets of electrochemical data. Graph A shows the voltammograms for each sweep rate, and Graph B shows the peak currents as a function of square-root of the sweep rate.

The shape of the cyclic voltammograms are typical of a reversible one electron redox system, with a ΔE_p of 80 mV – far from an ideal value of 60 mV at room temperature. Graph B can be used together with the Randles-Sevcik Equation to give a diffusion coefficient of 7.1 $\times$ 10^{-6} cm^2 s^{-1} for the $[Ir(Cl)_6]^{2-}$ species [56]:

$$i_p = 2.69 \times 10^5 \times n^{\frac{3}{2}} A D^{\frac{1}{2}} C v^{\frac{1}{2}} \qquad (4.3)$$

To check the agreement between the electrochemical and spectral data, the current responses were used together with the trapezoidal rule [57] to find the electrochemical charge passing through the cell over time. Using Equation 4.1 and the molar extinction coefficient previously found ($\varepsilon_B - \varepsilon_A$ = 3000 M^{-1} cm^{-1}), these charge values were converted into absorbance values that predict how the absorbance should change during the potential sweep. Figure 4.7 shows how these predicted values compared with the absorbance values measured during the cyclic voltammetry for each potential sweep. (Each sweep starts close to the point (0.5, 0) on each graph.)

The plots generally demonstrate a strong similarity between the predicted and measured values of absorbance, with the faster sweep rates showing better agreement. Some deviation starts to emerge at the slower sweep rates, and is most evident on reverse sweeps of the 5 mV s^{-1} and 10 mV s^{-1} voltammograms. This would suggest that time is a key factor in causing the deviation from the linear relationship suggested by Equation 4.1.



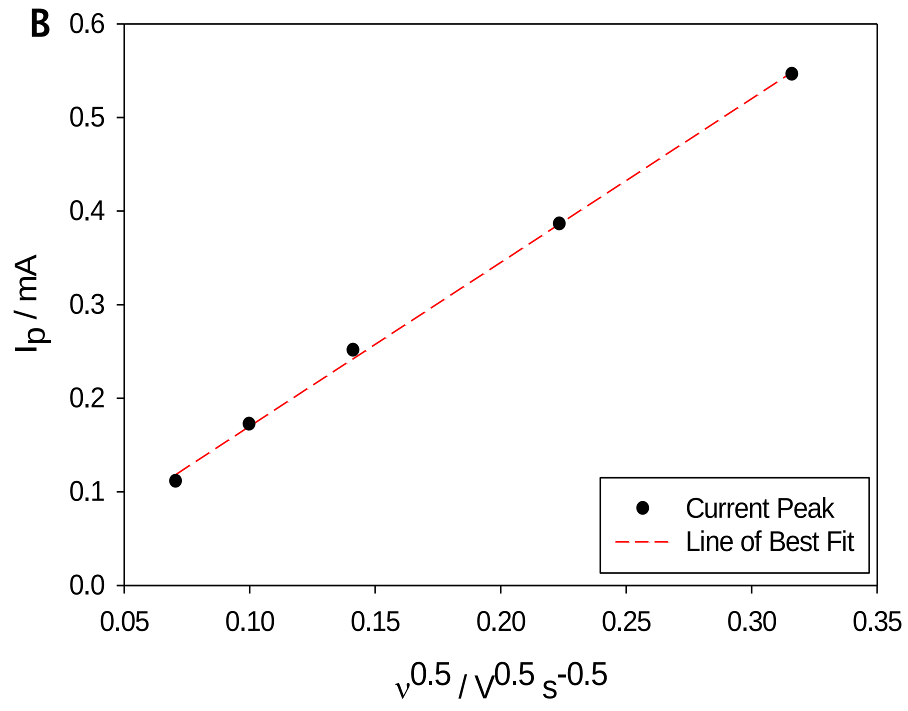


Figure 4.6: (A) Cyclic voltammograms of 8 mm diameter Pt working electrode disk in aqueous solution of 5 mM $K_3[Ir(Cl)_6]$ and 1 M NaCl. (B) Plot of anodic peak currents as a function of sweep rate (R^2 = 0.999).

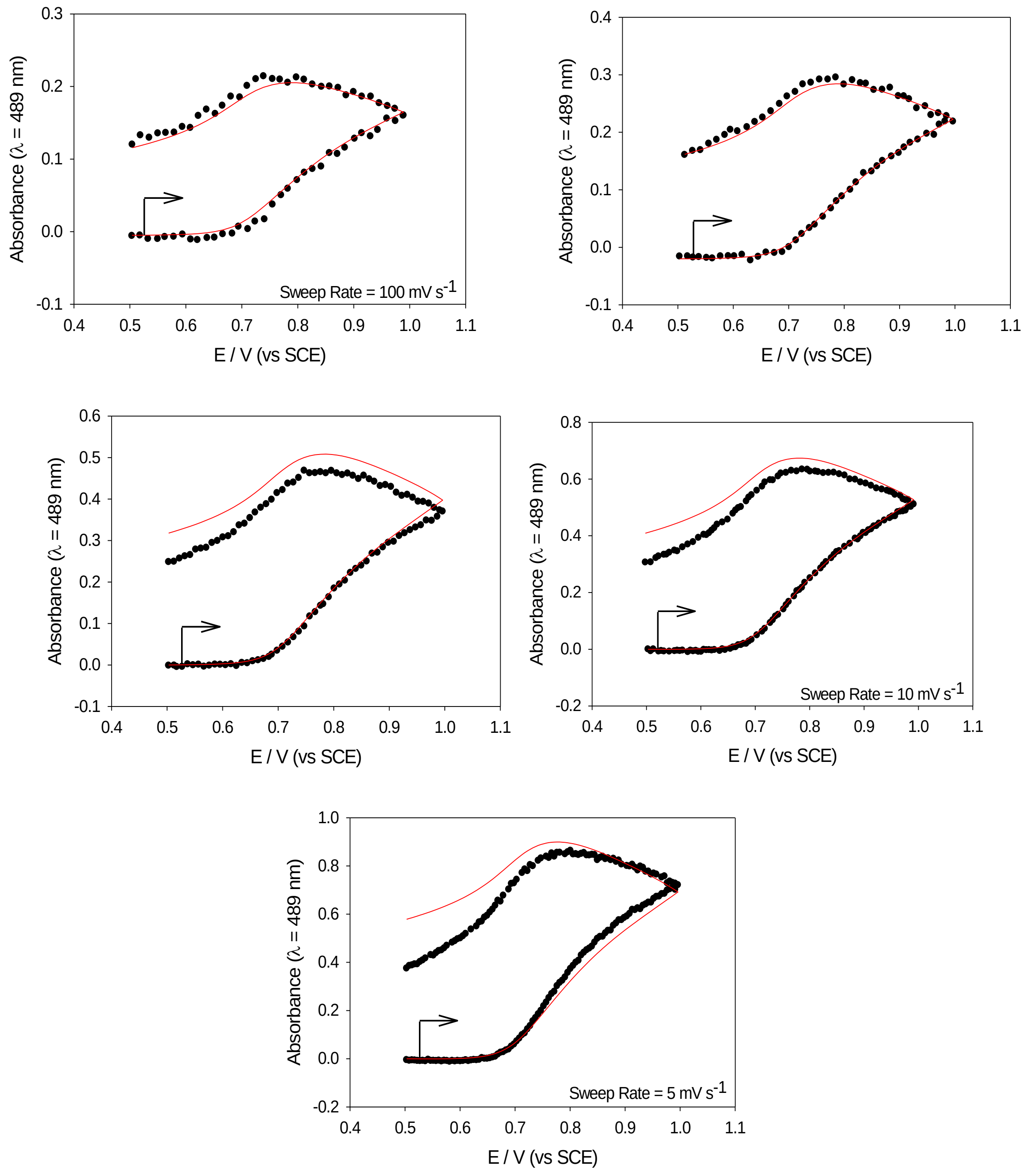


Figure 4.7: Predicted (red) and measured (black) absorbance for cyclic voltammetry of varying sweep rates applied to 8 mm diameter Pt working electrode in aqueous solution of 5 mM $K_3[Ir(Cl)_6]$ and 1 M NaCl. Arrows indicate where the cyclic voltammetry begins and in which direction.

The same cyclic voltammetry technique was repeated, but with a smaller 0.5 mm diameter Pt electrode. Figure 4.8 shows how the predicted absorbance and measured absorbance vary with potential. The differences between the two are far larger than those observed when using the larger 8 mm diameter electrode, which suggests that using a smaller diameter electrode may also be a factor in the deviation from behaviour predicted by Equation 4.1.

A closer examination of the potential sweeps conducted at slower sweep rates are shown in Figure 4.9. The potential axis has been split into two phases for the forward and reverse sweep, and is now equivalent to a time axis. Figure 4.9 shows how the deviation between predicted and measured absorbance varies with time (rather than potential), as well as the current response. Note that the deviation was calculated by subtracting measured absorbance from the predicted absorbance.

Figure 4.9 shows that the deviation starts to increase when the voltammogram reaches the peak anodic current, and the rate of oxidation is diffusion controlled. The deviation seems to increase linearly with time, and starts to flatten out as the voltammogram reaches the peak cathodic current.

A similar effect is seen when a potential step is applied on the smaller 0.5 mm diameter Pt working electrode. As before, the potential was stepped up from 0.5 V to 1.0 V (*vs.* SCE), and held at that potential while absorbance and current were recorded. Afterwards, recording was stopped while the potential was reduced back down to allow oxidised material to reduce back again; returning the electrochemical cell back to its initial conditions. The potential step was repeated and data recorded another four times.

Figure 4.10 shows the absorbance measurements as a function of electrochemical charge (calculated by numerical integration of the current time history using the trapezoidal rule). Linear regression has been applied to the data points within the region $0\ \mu C < Q < 8\ \mu C$ and the line-of-best-fit is shown in red.

The graph shows that as oxidation continues to occur at the working electrode, the linear relationship between absorbance and charge becomes distorted and less absorbance was observed compared to that expected. This, and the voltametric data presented, indicates that the material produced from the electrochemical oxidation of the redox species (specifically the $IrCl_6^{2-}$ species) was lost from the system and not interrogated by the spectroelectrochemical measurement.

This implies that the simple absorbance model used to derive Equation 4.1 is inadequate if the diameter of the working electrode under investigation is too small and the time-span of the experiment extends past a certain threshold. To solve the problem, adjustments may need to be applied to the diffusion model used in analysing this *in-situ* spectro-electrochemical technique.

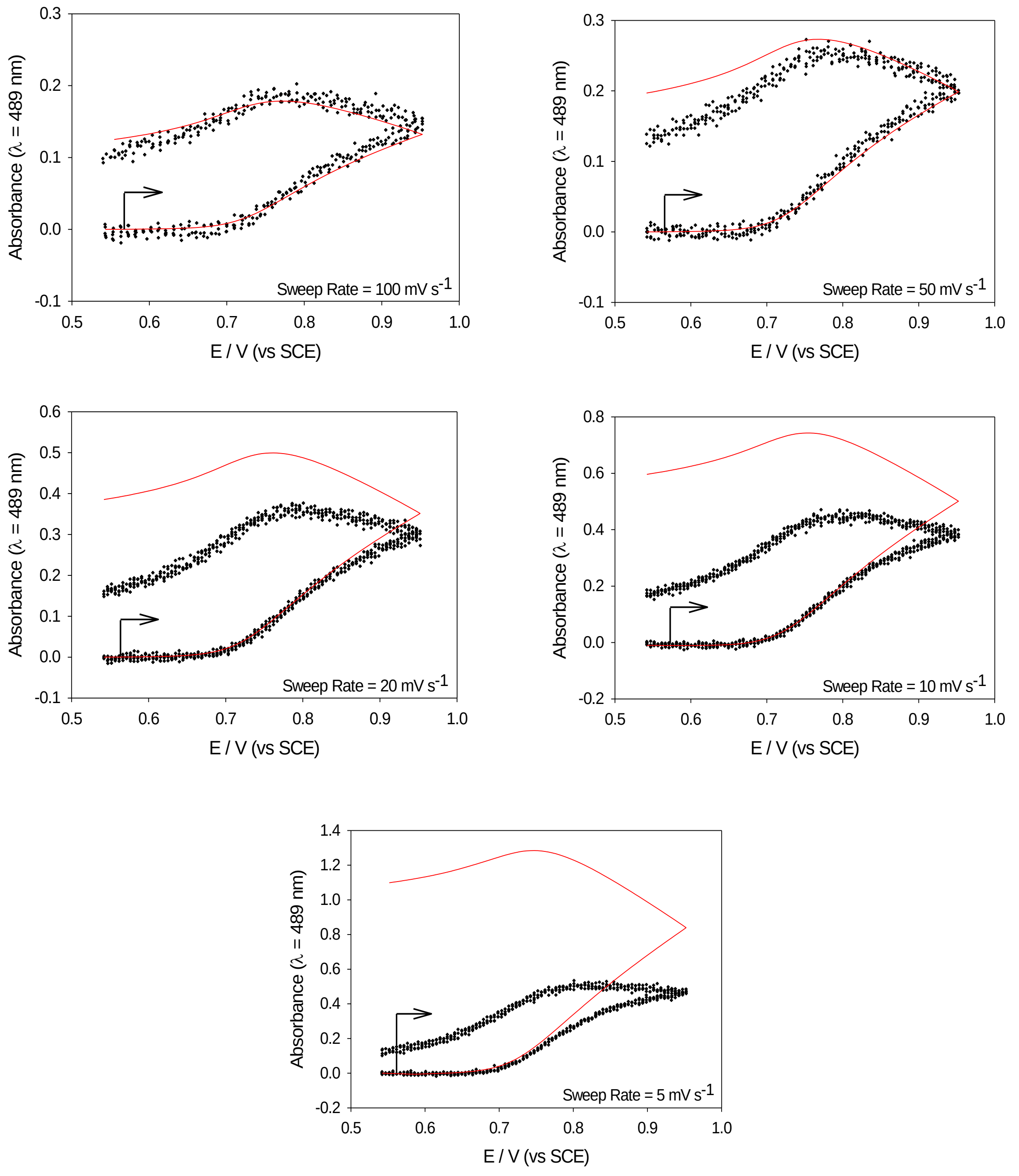


Figure 4.8: Predicted (red) and measured (black) absorbance for cyclic voltammetry of varying sweep rates applied to 0.5 mm diameter Pt working electrode in aqueous solution of 5 mM $K_3[Ir(Cl)_6]$ and 1 M NaCl. Arrows indicate where the cyclic voltammetry begins and in which direction.

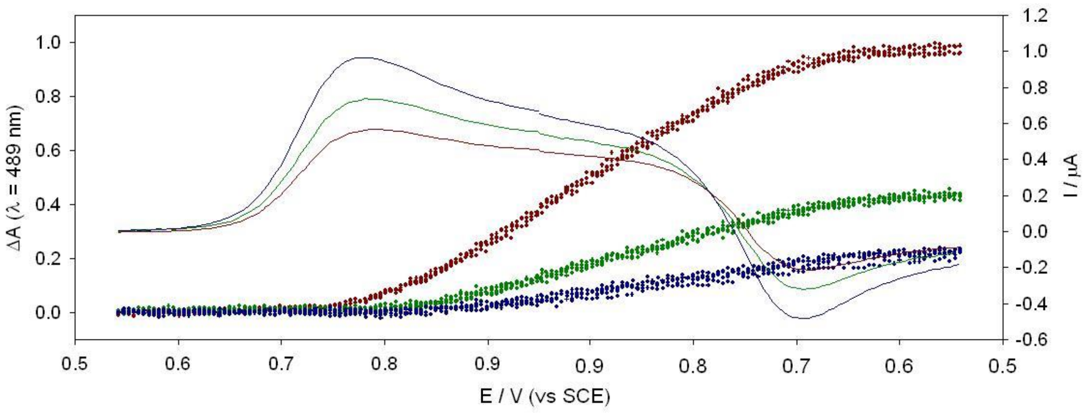


Figure 4.9: Current responses (lines – right axis) and difference in predicted and measured absorbance (dots – left axis) for potential sweeps plotted as a function of potential for sweep rates of 20 mV s^{-1} (blue), 10 mV s^{-1} (green) and 5 mV s^{-1} (red). The difference is measured as predicted minus measured absorbance.

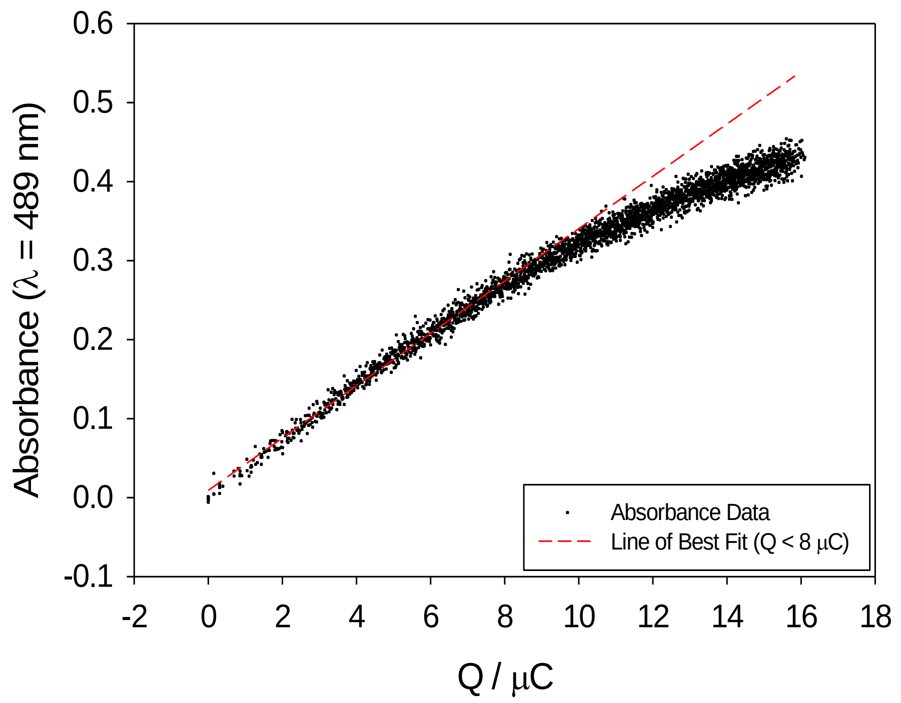


Figure 4.10: Plot of absorbance as a function of charge passed through a 0.5 mm diameter Pt working electrode in an aqueous solution of 5 mM $K_3[Ir(Cl)_6]$ and 1 M NaCl, produced by a potential step from 0.5 V to 1.0 V (*vs.* SCE).

There may be two possible causes as to why these deviations occur. The first is that natural convection within the electrolyte of the cell is shifting the redox products at the electrode away from the volume of electrolyte that the reflectance probe was inspecting. This convection effect would shift more material over longer time period, offering an explanation as to why the deviation increases over time periods of the experiments encountered for the slow sweep rates or the extended potential steps employed.

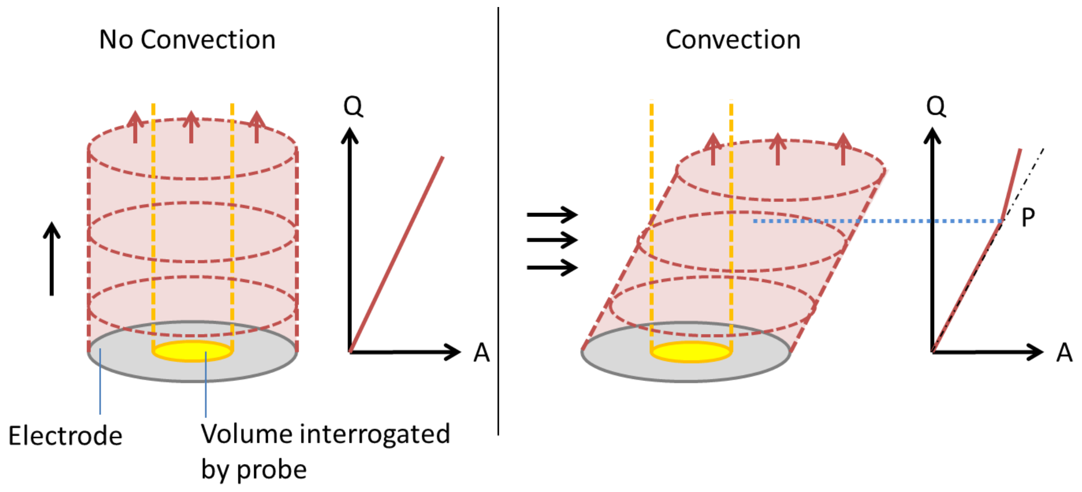


Figure 4.11: Illustration demonstrating the effect of convection in the cell on the relationship between absorbance and charge measurements. At point P, convection has shifted enough absorbing material away from the volume of electrolyte that is under inspection by the probe such that absorbance measurements are lower than they would be otherwise.

The second cause is that the initial assumption that redox material produced at the electrode is diffusing along one dimension away from the working electrode disk is not enough to accurately describe the diffusion behaviour of the redox products. A more accurate model may take into consideration that there will be radial diffusion occurring as well as the one dimensional linear diffusion away from the electrode. This cause will be investigated first since it only requires adjustment of key assumption used to derive Equation 4.1, whereas convection will require a more experimentally intensive investigation.

4.5 Consideration of Edge Effects:

The oxidation current generated at a disk electrode has so far been assumed to be diffusing linearly in one direction at the normal to the electrode, which will hold true at the centre of the electrode. However, oxidised material will be diffusing across three dimensions, particularly at the edge of the electrode. Figure 4.12 illustrates that at the edge of the electrode, oxidised material is able to diffuse outwards up to a 90° angle to the normal of the electrode surface. The current recorded during these experiments is produced by the whole electrode; and can be simplified and split into two basic components; the current caused by redox material diffusing linearly along one dimension perpendicular to the disk electrode, and redox material diffusing radially at the edges of the electrode.

The reflectance probe lies directly above the electrode disk and will only be able to detect the redox products that diffuse linearly away from the electrode. Redox products that diffuse radially at the edge of the disk electrode will lie above the glass insulation, which is poor at reflecting light (relative to the electrode), and will be near impossible to detect. It was initially assumed that all the oxidised material produced at the electrode was distributed within a volume partly defined by the boundary of the electrode, but in actuality this volume may be larger and the absorbing material less concentrated than initially thought.

The divergence between the absorbance values measured by the probe and absorbance values predicted by the electrochemical measurements may be occurring because the absorbance values are determined by the linear diffusion component, while the electrochemical data that has been used to calculate absorbance predictions comes from a current response that is made up of both linear and radial diffusion components.

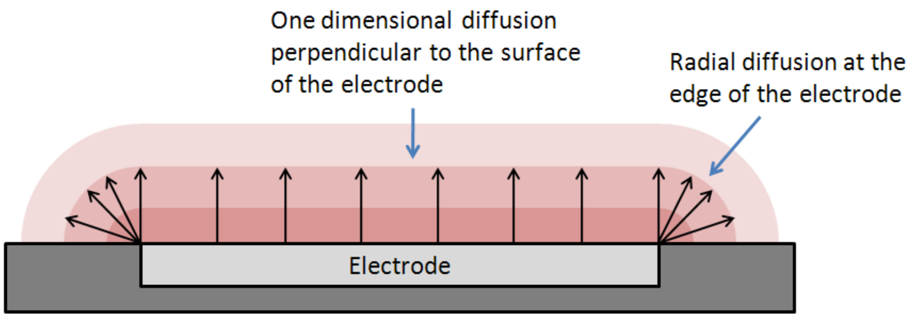

One dimensional diffusion perpendicular to the surface of the electrode
Radial diffusion at the edge of the electrode
Electrode

Figure 4.12: Diagram showing a combination of one-dimensional diffusion occurring along the surface of the electrode, together with radial diffusion occurring at the edges of the electrode.

A model needs to be found that can describe the current response as separate linear and radial components, so that the linear component can be isolated and integrated with respect to time and used to generate a new absorbance prediction. Several theoretical equations of varying convenience and accuracy are available, which describe the diffusion current at a disk electrode when a potential step is applied. Of these, Equation 4.4 has been established as one of the most accurate [58], [59]:

$$\frac{I(t)}{\pi nFc_bDr} = 1 + \frac{r}{\sqrt{\pi Dt}} + \left(\frac{4}{\pi} - 1\right)e^{\left\{\frac{-0.39115r}{\sqrt{Dt}}\right\}} \qquad (4.4)$$

While the second term on the right hand side of Equation 4.4 can be used to describe the linear component of the diffusion current, the other two terms can be used to describe the radial diffusion component. Rearranging the equation gives a linear component, I_{li}, of:

$$I_{li} = \frac{\sqrt{\pi Dt}}{r}\left[\frac{I(t)}{\pi nFc_bDr} - 1 - \left(\frac{4}{\pi} - 1\right)e^{\left\{\frac{-0.39115r}{\sqrt{Dt}}\right\}}\right] \qquad (4.5)$$

Equation 4.5 needs to be integrated with respect to time in order to find the charge produced by the linear diffusion component and applied to Equation 4.1 to produce a new absorbance prediction based on the electrochemical data. The solution of this integral is accurate, but lacks convenience. Instead, the Shoup-Szabo Equation is simplified into two terms; one term for linear and another for radial diffusion [60]:

$$I(t) = nF\,\pi r^2 c_b\sqrt{\frac{D}{\pi t}} + \alpha nFDcr\sqrt{\pi} \qquad (4.6)$$

The constant, α, can be found experimentally by applying a potential step to the disk electrode. Flanagan and Marcoux found an experimental value of α to be 2.12 ±0.11 [60].

Rearranging and integrating Equation 4.6 to find the charge produced by the linear component of diffusion, Qli, gives:

$$Q_{li} = \left[\int I(t)\,dt\right] - \left(\alpha nFDcr\sqrt{\pi}\right)t \qquad (4.7)$$

The last term of Equation 4.7 represents the absorbing material produced at the edge of the electrode, but remains undetected by the probe. This term increases linearly with time, a trait observed in the Figure 4.9. This term is proportional to the radius of the electrode, while charge is proportional to the area of the electrode; so decreasing the radius of the electrode will increase the size of this term with respect to all other terms in this equation; as expected since the deviation increased when a smaller electrode was used.

Equation 4.8 can be substituted into Equation 4.1 to give:

$$A_\lambda = \frac{2(\varepsilon_B - \varepsilon_A)}{nF(\pi r^2)} \times \quad Q_{li} \qquad (4.8)$$

Figure 4.13 shows the current response from the same potential step experiment as in Figure 4.10. Linear regression has been applied to the data to produce a line-of-best-fit, which is shown in red. Using the gradient of this line gives, the diffusion coefficient is found to be 6.9×10^{-6} cm^2 s^{-1}, and the y-intercept of this line produces a value for α of 1.65; a value lower than expected.

Using these values and integrating the current with respect to time gives the charge produced at the working electrode by linear diffusion as described in Equation 4.7. Figure 4.14 shows the absorbance measured as a function of this charge. The graph shows excellent linear agreement between the two, unlike the previous result shown in Figure 4.10.

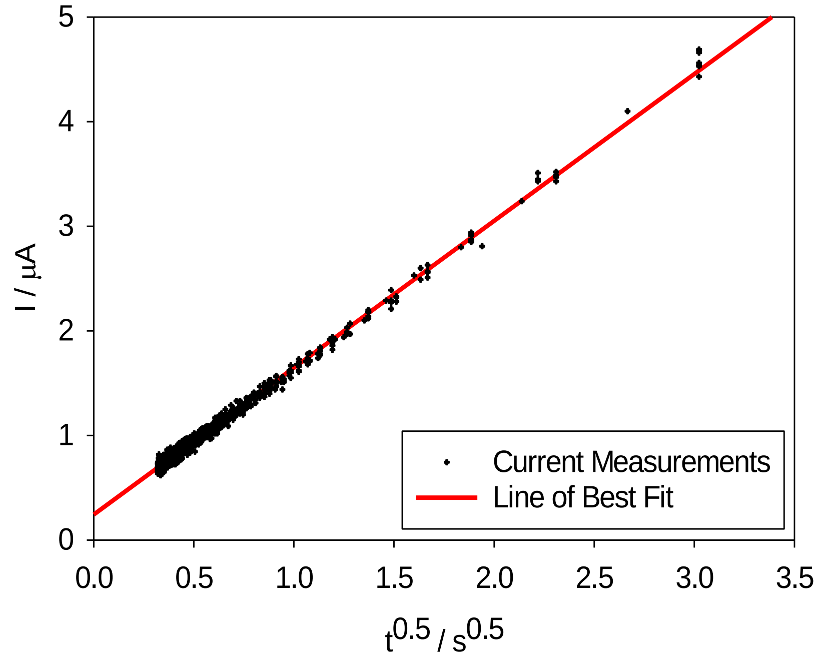


Figure 4.13: Current response from a 0.5 mm diameter Pt working electrode in an aqueous solution of 5 mM $K_3[Ir(Cl)_6]$ and 1 M NaCl, produced by a potential step from 0.5 V to 1.0 V (*vs.* SCE) (R^2 = 0.994).

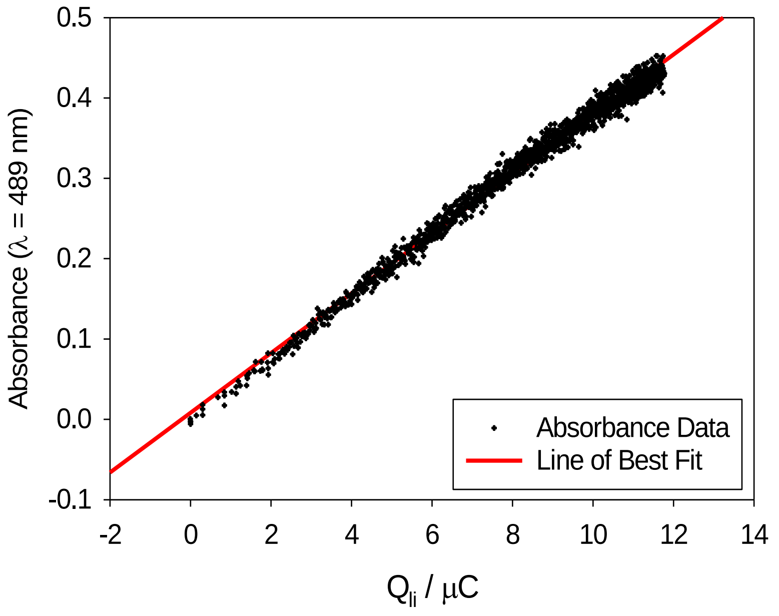


Figure 4.14: Plot of absorbance as a function of corrected linear charge (as described in Equation 10) passed through a 0.5 mm diameter Pt working electrode in an aqueous solution of 5 mM $K_3[Ir(Cl)_6]$ and 1 M NaCl, produced by a potential step from 0.5 V to 1.0 V (*vs.* SCE) (R^2 = 0.99).

Figure 4.14 strongly suggests that the previously observed deviation from linearity seen in Figure 4.10 is caused by edge effects introducing a negative bias to absorbance measurements recorder above the centre of the electrode. If absorbance data is intended for quantitative analysis then the time scale of the experiment has to be taken in consideration. The longer a potential step/sweep is applied for, the more likely edge effects are going to inflict a negative bias on absorbance data.

4.6 Conclusions

This chapter has been able to reproduce the phenomenon observed by Schroll et al. whereby absorbance measurements do not increase linearly with the electrochemical charge generated by the cell, as predicted by Faraday's Law and Beer's Law. The deviation from this linear behaviour was found to increase when oxidation occurred over longer time periods, and when a smaller working electrode was used.

This deviation has been attributed to the edge effect; where oxidised species diffuse radially from the electrode, instead of in one direction along the normal of the electrode. The current response produced by a potential step applied to the electrode is considered to be the combination of two components; a linear component that varies with the inverse square root of time, and a radial component that is constant with time. The charge calculated by this linear component of current is found to have a linear relationship with the absorbance measured at the centre of the electrode. The work has highlighted the importance of considering the time-scale of NNI-UVRS experiments if the absorbance data is intended to be used for quantitative analysis; as edge effects can cause a negative bias to the absorbance data.

Chapter 5: White Resin Supported Working Electrode

5.1 Introduction

The inability to quantifiably link absorbance measurements with the electrochemical charge passing through the cell has been suggested, as in Chapter 4, to be caused by two factors. The first is radial diffusion occurring at the edge of the electrode, and the second is uncontrollable convection in the electrolyte of the cell.

In order to investigate this effect further, a novel approach has been developed; utilising a disk electrode that is insulated in a reflective white resin instead of glass. All the disk electrodes used so far have been sealed in glass; and for *in-situ* UV-vis reflectance spectroscopy, this is problematic since the glass insulation reflects very low amounts of light back to the fibre-optic probe. Any absorbance data gathered is too noisy to be effectively analysed. This makes it impossible to find out how much of any redox products are diffusing at the edge of the electrode, and confirm what effect this is having on the absorbance measurements across of the working electrode. Using a reflective insulation instead, should allow the reflectance probe to measure any absorbance changes that may be occurring outside the boundary of the electrode.

An indicator responsive to changes in local pH will be used to follow local effects induced by electrochemical control of the system in question (specifically the electrochemistry of Pt in aqueous environments). For this approach, a set of experiments employing a solution contained thymol blue were performed. This indicator was chosen as it exhibits dramatic colour changes [61], [62] around pH 1.2-2.8 (red/yellow) and 8-9.6 (yellow/blue) making it ideal to explore swings in pH over the Pt surface induced by proton consumption as a result of electrochemical stimuli. Local pH changes are induced by modest cycling rates [63], [64], [65] (as opposed to more extreme examples [66]).

5.2 The White Resin Supported Electrode

The insulation of the disk electrode is fabricated using a mould made from a styrene monomer (Crystic Monomer C) and a catalyst hardener (methyl ethyl ketone peroxide); both obtained from East Coast Fibreglass Supplies and were used as received.

To make sure it behaved like a standard glass-insulated disk electrode would, the electrode was placed in an electrochemical cell containing an aqueous solution of 1 M H_2SO_4 with a MMS reference electrode and a Pt gauze counter electrode. Figure 5.1 shows a cyclic voltammogram of this cell taken at a sweep rate of 0.5 V s^{-1}. The voltammogram shows features as expected from a Pt electrode in acid [66]. On the far left-hand side in between the -0.6 V and -0.4 V are two hydrogen adsorption and desorption peaks, with the peaks being symmetrical. At -0.3 V, a double layer capacitance region is observed.

This cyclic voltammogram was recorded after the electrode had been polished with P180 sand paper (particle size ≈ 82 μm). It was found that as the electrode was polished with finer paper, the shape of cyclic voltammogram became distorted. As a result, all further electrochemical experiments were conducted using this electrode polished with P180 sandpaper, rather than any alumina slurries.

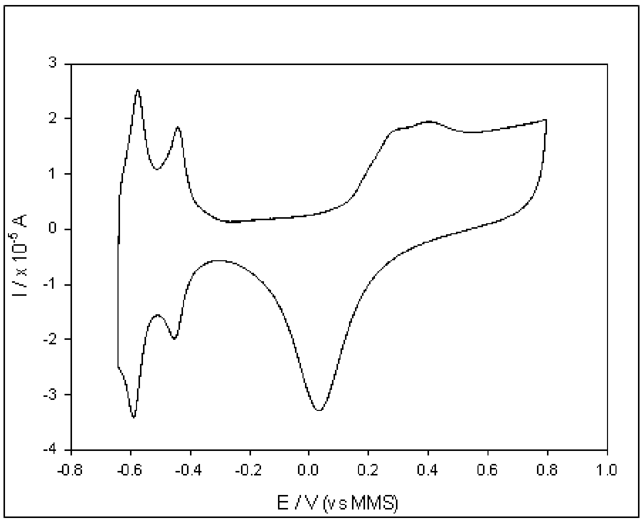


Figure 5.1: Cyclic voltammogram of 1 mm diameter Pt electrode in aqueous solution of 1 M H_2SO_4. Sweep rate is equal to 500 mV s^{-1}. The solution was degased with argon for 20 minutes and was 21 °C.

Figure 5.2 shows a scan of the white resin supported working electrode (WSWE). The scan shows that the signal recorded over the insulation is about 30% of that recorded over the electrode disk. This is a vastly improved ratio when compared to the glass electrodes (like the ones shown in Figures 2.5 and Figure 3.18) that show a ratio below 1%.

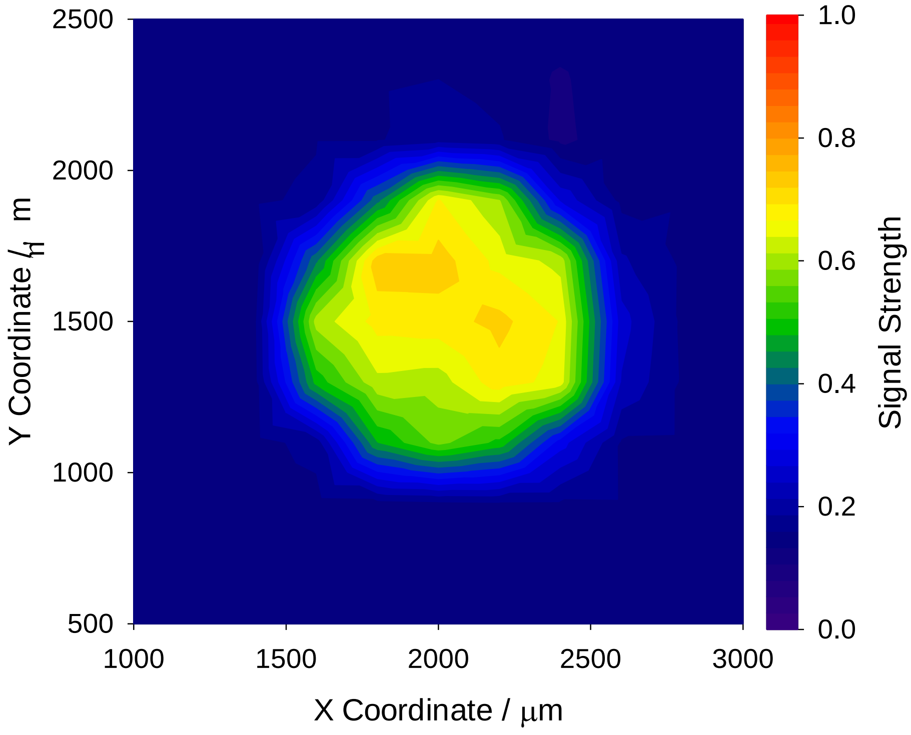


Figure 5.2: Contour plot of signal strength as a function of position across the 1 mm diameter WSWE. Data points are separated by a distance of 200 μm in both x and y directions.

5.3 Investigations of pH Changes

The first application of this new electrode is an investigation into local pH changes occurring in an electrochemical cell when a potential step is used to incur hydrogen adsorption. By recording absorbance over time across an array of points across the electrode and insulation, it should be possible to observe how these pH changes evolve over both time and space.

This experiment uses an electrochemical cell containing an aqueous solution 0.1 M Na_2SO_4 and 0.05 mM thymol blue. Thymol blue was selected because of the intense colour change that it undergoes when the pH of the solution changes from neutral to alkali [61], [62]. At neutral pH, the solution is a pale orange/yellow colour, but at an alkali pH, the solution turns to a dark blue, which is strongly absorbing at the 600 nm wavelength. A potential of -1.3 V (*vs.* MMS) is applied for 5 seconds at the electrode in order to induce an alkali pH swing. This pH swing is the result of the consumption of protons in the electrolyte in front of the electrode; either through the formation of platinum hydride or bulk proton reduction.

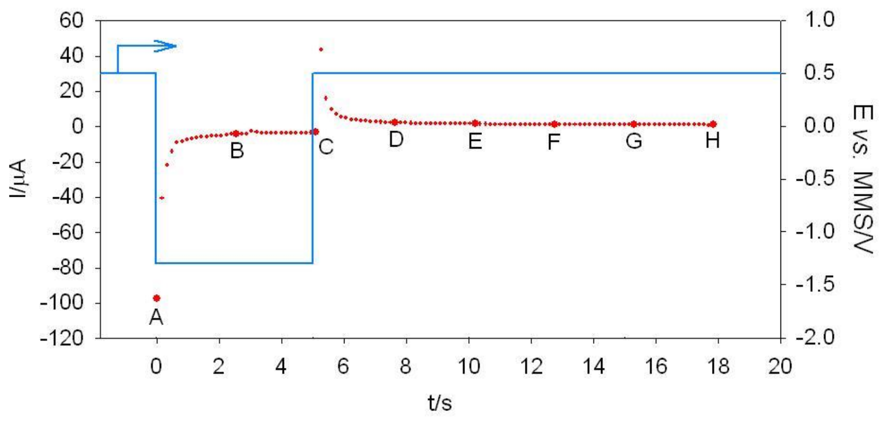
After the 5 seconds, the potential is returned to +0.5 V (*vs.* MMS); releasing protons back into the bulk solution. The cell is then left for 30 seconds to allow it to return to its initial conditions.

This process was repeated 121 times, with the bifurcated reflectance probe placed at a different point each time. When the probe was placed in position, a spectrum was taken to use as the reference, and then the potential step was applied. The same dark reading was used for each iteration of this process. After the 30 second period to allow the cell to return to initial conditions, the probe was placed to a new condition, and the process was repeated. Absorbance measurements were made throughout both the 5 second potential step, and the following 30 second period.

Figure 5.3 shows the result of this potential pulse. The top half shows the amperometric response from the electrode; a typical anodic transient as hydrogen is adsorbed into the platinum. After the 5 second pulse, a cathodic transient is observed, as protons are released back into the bulk solution.

The bottom half of Figure 5.4 shows a series of contour plots detailing the absorbance across the electrode at different times. Images A to C show the growth of a cloud of material above the electrode that is absorbing light at 600 nm wavelength. This is the deprotonated thymol blue being produced during the cathodic pulse. Images D to H show how this cloud of thymol blue dissipates when the voltage is returned to an anodic potential. The effect of convection is clearly seen as the cloud shifts from left to right.

This behaviour is inherently reproducible since each image is composed from a set of 121 different runs of the same experiment. Overall, the result shows this *in-situ* reflectance scanning technique could provide a useful tool for measuring spectroscopic changes occurring over an electrode, and how those changes evolve over time and space.



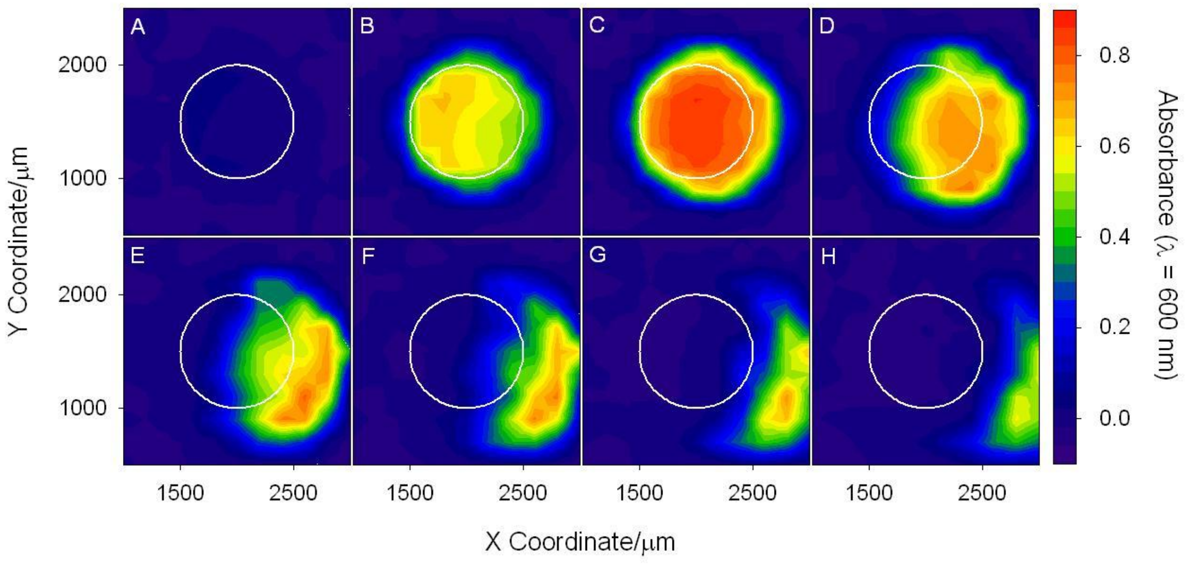


Figure 5.3: (Potential sequence (—) and corresponding current time history (●) of a single pulse performed at a 1 mm diameter Pt disk electrode sealed in resin (WSWE). The annotation (A-H) corresponds to the positions in time of the images shown in (b). **(b)** shows the images constructed from the double potential step experiments performed at each position above the surface of the Pt electrode and the support. The legend refers to the adsorption data at 600 nm at each respective position over the area interrogated. The time for each frame is indicated in (a). the white circle correspondes to the position of the 1 mm diameter Pt electrode. The 0.1 M Na2SO4 electrolyte contained 0.05 mM thymol blue. The solution was aerobic and had a temperature in the 18-22 oC range.

5.4 Towards Quantitative Analysis

Using this *in-situ* scanning technique, it should be possible to measure all the absorbing material produced by the electrode, rather than material produced at the centre of electrode (as was the case in Chapter 4). A spectroelectrochemical experiment using this technique should produce absorbance and charge values that obey a linear relationship. Chapter 4 concluded that one reason that a linear relationship between the electrochemical charge passing through the electrode and the absorbance measure above the centre was not observed because absorbing material was being produced at the edge of the electrode. This absorbing material was undetectable due to the poor reflectivity of the glass support. However, using the WSWE, it may be possible to measure the absorbance at the edge of the electrode and whether it compensates for the reduced absorbance measurements observed in Chapter 4.

The absorbance produced by the generation of redox products at the electrode can be considered as a function of X and Y coordinates across the plain of the electrode. Figure 5.4 shows a simplified illustration of how absorbance varies as a function of position across the electrode in two different cases; one where diffusion is limited to linear behaviour across the whole electrode (red), and one where radial diffusion is allowed to occur at the edges (blue). The electrode surface is bounded by x_0 and x_1.

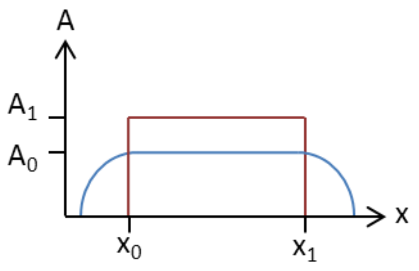


Figure 5.4: Simplified and exaggerated plot of absorbance as a function of position x for the cross-section of an electrochemical cell, where the electrode lies on the x-axis, between x_0 and x_1. Two cases are considered; one where products diffuse linearly (red), and one where products diffuse radially at the edge (blue). The areas underneath both curves are considered identical.

In both of these cases, the same amount of absorbing redox products are produced by the same amount of electrochemical charge, but are distributed across the electrode in different ways, and produce two different absorbance levels at the centre of the electrode; A_0 and A_1. It is suspected that the lower than expected absorbance values measured in Chapter 4 are because the absorbing material diffuses in a manner similar to the blue profile, while the expected values are calculated by assuming the absorbing material diffuses in a manner similar to the red profile.

Section 5.3 showed that the WSWE can be used to measure absorbing material that have diffused beyond the edge of the electrode, so it should now be possible to measure an absorbance pattern similar to the blue profile. This absorbance profile can then be redistributed within the boundaries of the electrode, to produce a single absorbance level – like the red profile in Figure 5.4. If all the absorbing material is accounted for, this corrected absorbance level should be equivalent to the absorbance level that would have been recorded had all the material produced at the electrode had diffused in one direction perpendicular to the electrode.

For example, the image on the left of Figure 5.5 shows how the absorbance varies over the electrode after a potential step has been applied for 4.9 seconds to a disk electrode, submerged in an iridium redox system (as described in Chapter 4). The image on the right shows has the same volume as the plot on the left, but is redistributed so there is only one average absorbance level within the boundary of the electrode. The height of this profile is what the absorbance would have been had all the material generated by the potential step diffused linearly away from the electrode.

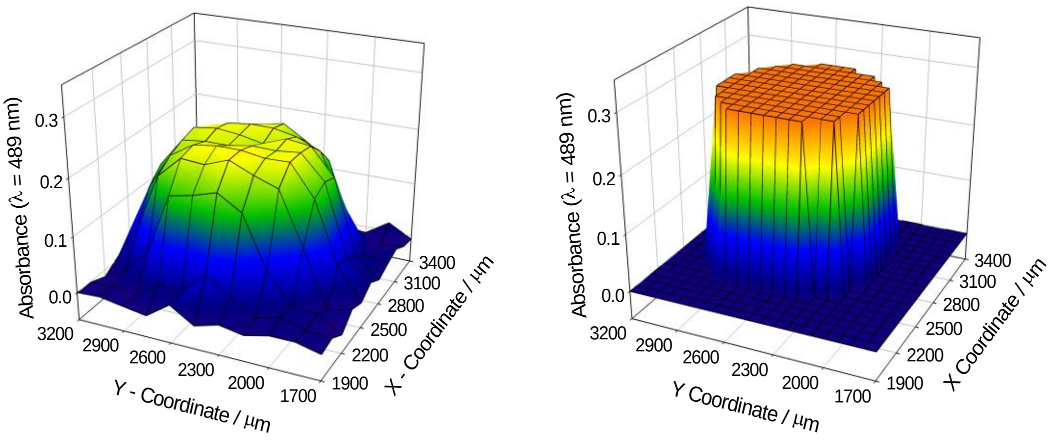


Figure 5.5: **(Left)** Plot of absorbance as a function of coordinates across a 1 mm diameter Pt WSWE in an aqueous solution of 5 mM $K_3[Ir(Cl)_6]$ and 1M NaCl,m, 4.9 seconds after a potential step from 0.5 V to 1.0 V (*vs.* SCE) was applied. **(Right)** Plot of theoretical absorbance as a function of coordinates across the same 1 mm diameter Pt electrode if all absorbing material produced by the potential step diffused directly in one direction, upwards from the electrode.

However, using this method to record absorbance introduces new problems, which all relate to the nature of how the probe works.

As shown in Chapter 3, at any given time the probe inspecting a region on the reflective surface that is about 250 μm in diameter. This implies that the probe can be placed as far as 125 μm from the edge of the electrode and will still have the ability to measure any absorbance changes occurring at the electrode. Since the probe is only partially inspecting the electrode, this absorbance measurement will be significantly smaller than it would be if the probe was fully inspecting the electrode. However, the measurements will still give the impression of absorbance activity occurring beyond the edges of the electrode, even if all absorbing material diffused linearly away from the electrode.

Therefore, any absorbance measurements (like those recorded in Figure 5.5) using this method are likely to be biased. To find out the significance of this effect, the model devised in Chapter 3 is used to produce a series of hypothetical absorbance scans across an electrode.

5.5 Absorbance Measurements Models

Consider a theoretical disk electrode with a diameter of 1 mm (see Figure 5.6). This electrode is made up of a reflective material, such that if the probe is directly over it, the signal is measured as 0.5. The electrode is insulated in a less reflective material, such that if the probe is directly over the insulating material, the signal is measured as 0.1.

The central axis of the probe (and central axis of the detection fibre) is considered as the point (0, 0). The theoretical scan is made by considering the centre of the electrode in different positions relative to this origin.

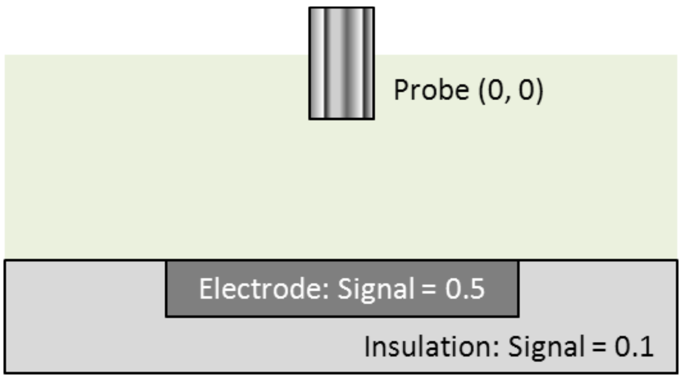


Figure 5.6: Schematic of theoretical setup used to generate theoretical absorbance scans.

To generate the signal that the probe measures at each point, the same light intensity as shown in Figure 3.16 is used. This consists of 19981 data points; with each point consisting of a set of coordinates and a corresponding light intensity value. For each individual position that the electrode moves to during a scan, each of the 19981 data points are first found whether to be lying on the electrode or the surrounding insulating material. If a data point lies on the electrode, the corresponding light intensity is multiplied by 0.5. If it lies on the insulating material, it is multiplied by 0.1. After assessing all 19981 data points, the light intensities are then accumulated to find the total sum, which is the signal measured by the probe.

Figure 5.7 gives an example of this process when the centre of the electrode is placed at (0 μm, 500 μm) relative to the centre of the probe. The contrast between the signals that the electrode surface provides, relative to the insulating material, can clearly be seen.

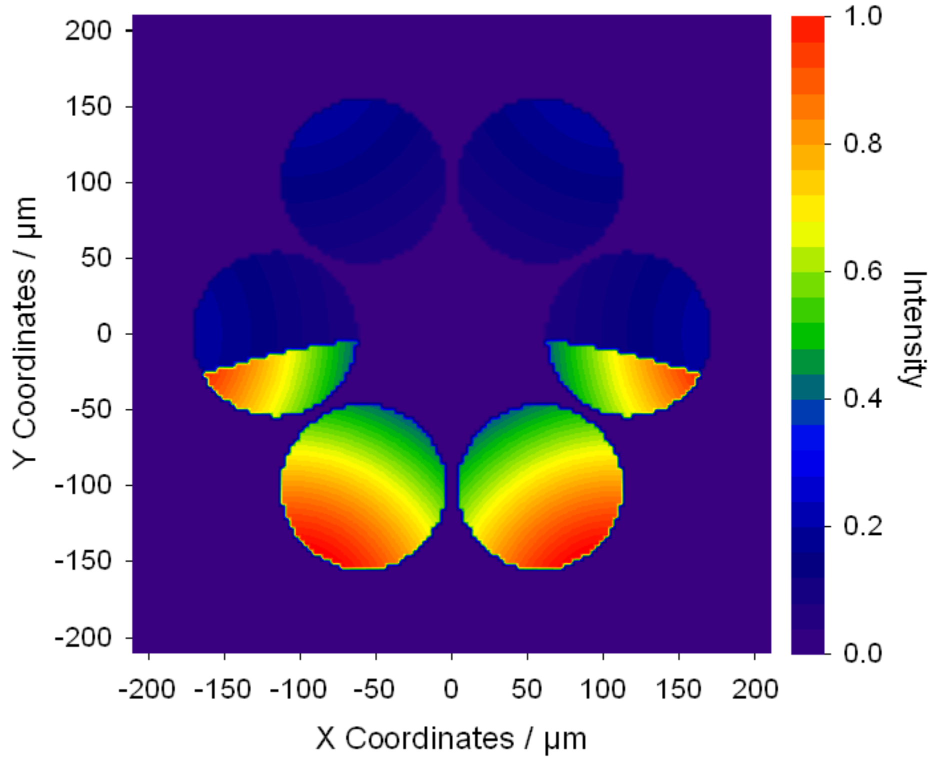


Figure 5.7: Intensity of light that reaches the target probe (and contributes to the signal recorded by the spectrometer) as a function of x and y coordinates. All values are normalized to give a maximum intensity of 1. In this case, the centre of the reflective electrode (diameter = 1 mm) is at (0, 450).

The process is repeated for all positions that the electrode needs to move through to produce the necessary scan. Each scan is made up of 121 points (arranged in 11 by 11 grid), 150 μm apart. Finally, the signals are normalized so that the maximum signal recorded is equal to 1. Figure 5.8 shows an example of a theoretical scan on the 1 mm diameter electrode using a resolution of 150 μm.

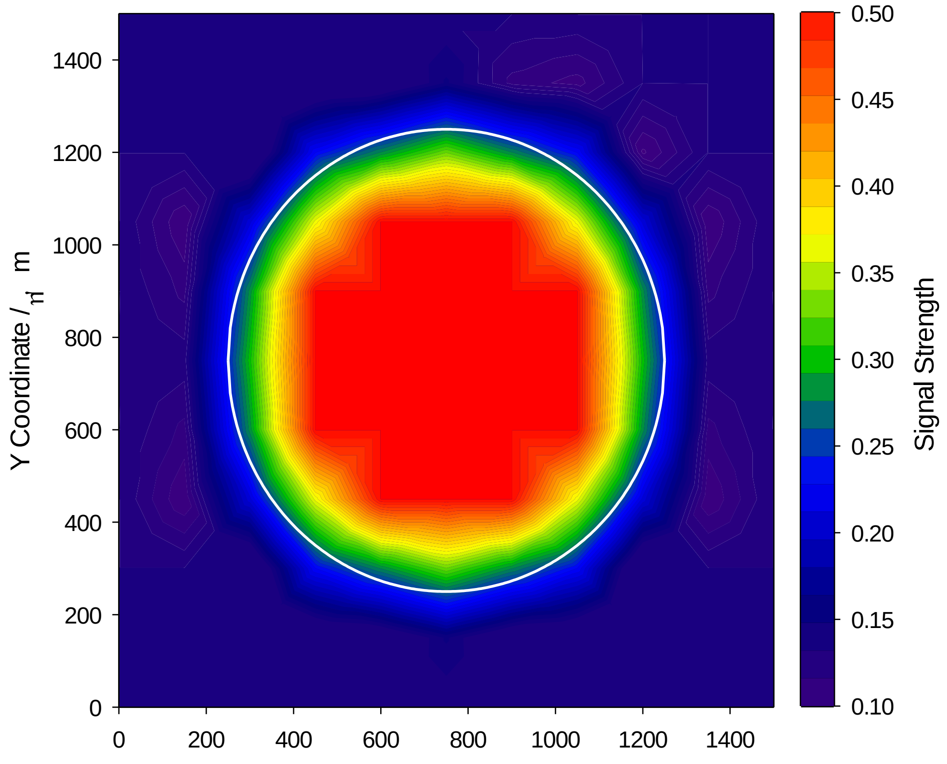


Figure 5.8: Plot of signal strength across the theoretical electrode as a function of x and y coordinates. The centre of the electrode is modelled at (750,750).

To produce hypothetical absorbance measurements across the electrode, a scenario is considered in which an amount of absorbing species is produced at the electrode, such that the signal over the electrode is now equal to 0.3, as illustrated in Figure 5.9. The absorbance profile assumes the absorbing species is limited to linear diffusion (the concentration profile of this absorbing species is irrelevant). The same process of predicting a scan is repeated using this new signal strength. The resulting absorbance scan is found by considering each coordinate used for both scans, and applying the log function to the ratio of the two signal values:

$$A_{x,y} = \log\left(\frac{S_1}{S_2}\right)$$

S_1 is the signal strength when there is no absorbing species present (see Figure 5.8) and S_2 is the signal strength when the absorbing species is present. Since the previous signal of 0.5 is used as a reference, the absorbance measured by the probe should be equal to 0.30. The model absorbance scan is shown in Figure 5.10.

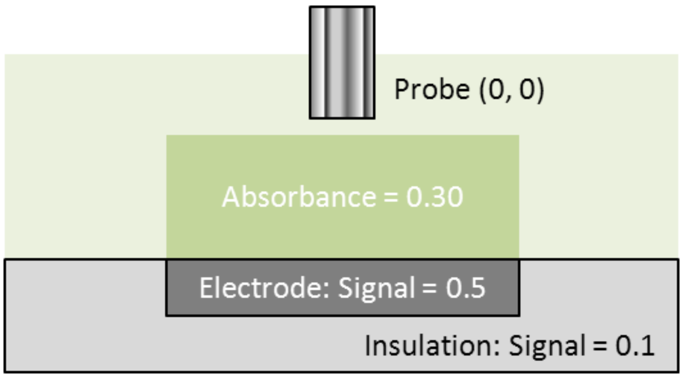


Figure 5.9: Schematic of theoretical setup used to generate theoretical absorbance scans when the absorbance is equal to 0.3.

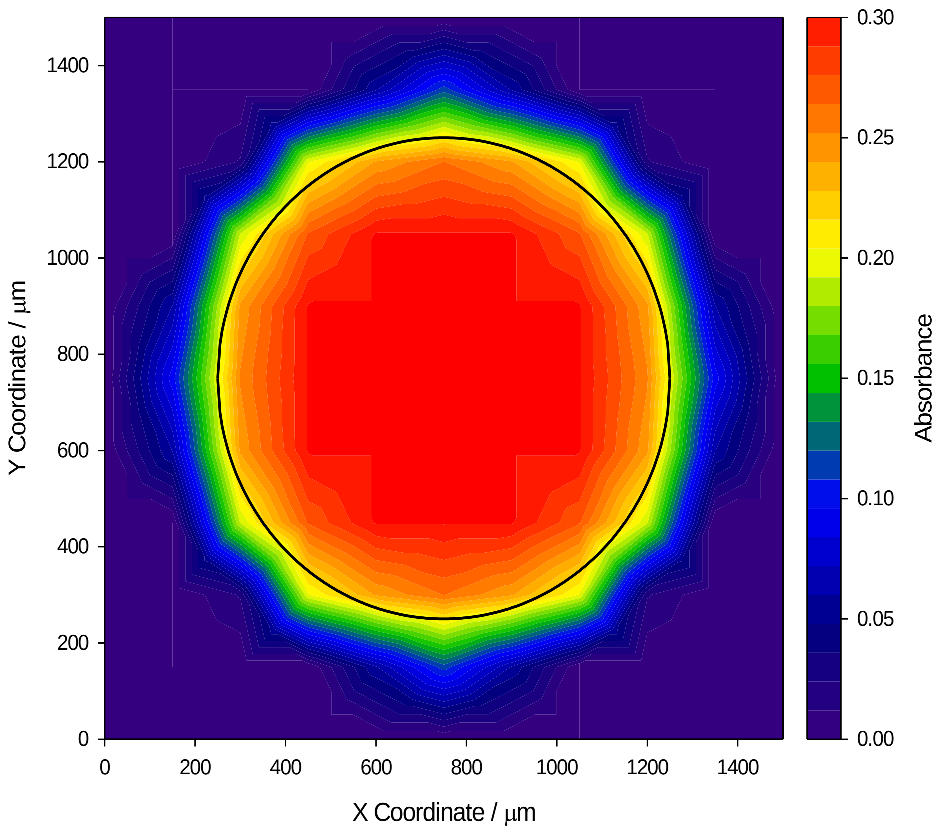


Figure 5.10: Theoretical absorbance scan produced by the scenario described in Figure 6. The edge of the 1 mm diameter electrode is outlined in black and the centre is found at (750, 750).

The Figure 5.10 shows that the absorbance signal detected covers a region over the electrode, as expected, but extends beyond the edge of the electrode, the probe measures positive absorbance changes; despite the fact that the scenario specifies there are no absorbing species there. This is caused by the fact that probe inspects a specific surface area about its central axis (as shown in Chapter 3). While the probe may be placed at a point beyond the edge of the electrode, there is a chance it will still be close enough to still measure absorbance changes occurring directly above the electrode.

Attempts to redistribute the volume underneath the surface of the absorbance plot within the boundaries of the electrode, as illustrated in Figure 5.5, gives an absorbance level of 0.375. This is 25% higher than the expected value of 0.3 and indicates that this approach is prone to error.

The model was used repeated with varying values of reflectivity for the support to find out whether this offset is consistently present. In all cases, the reflectivity of the electrode was kept at 0.5.

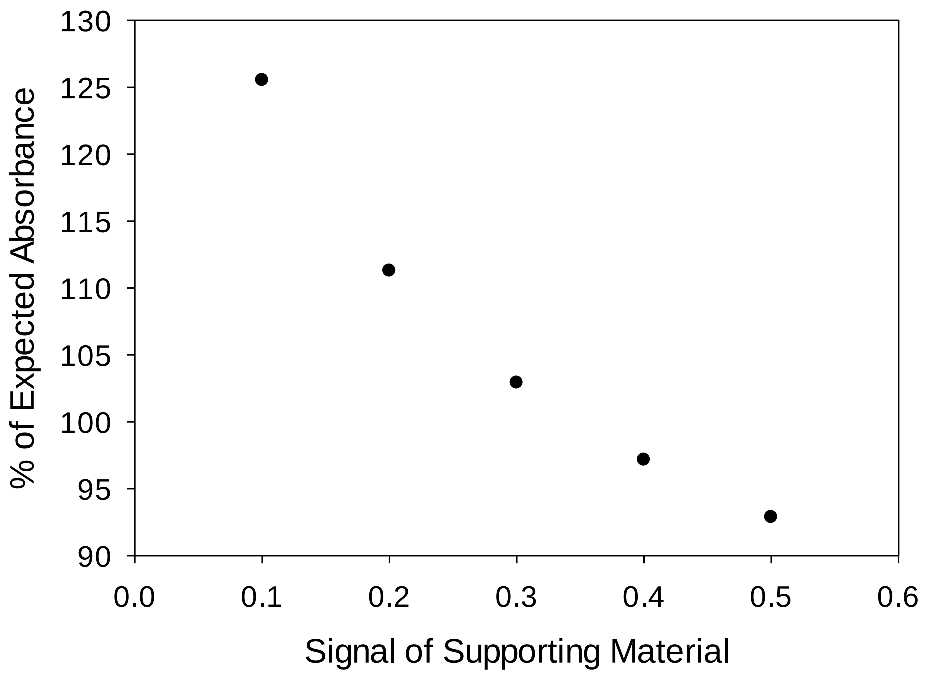


Figure 5.11: Plot showing the inaccuracies of measuring absorbance as a volume, as a function of the reflectivity of the material insulating the disc electrode. The signal produced by the electrode is 0.5 and has a diameter of 1 mm. In each model, a 150 μm distance between data points was used, and the position of the probe was defined by an 11 x 11 grid.

The results from this model show that the reflectivity of the insulation has to be a specific value (0.33) in order to give an accurate absorbance measurement using the method of correction as described. Therefore using this method to quantitatively measure absorbance would be flawed, since there would likely be a bias to the results, depending on the reflectivity of the support relative to the electrode.

However, the method may still be used in a qualitative method to investigate the relationship between absorbance and electrochemical charge, and validate whether edge effects are impacting the absorbance measurements, or whether the divergence seen in Chapter 4 was caused by convection. The results in Chapter 4 showed a deviation in absorbance over time from what was expected according to the electrochemical data. Using this model, scenarios with varying diffusion behaviours can be simulated to find whether there is still a linear relationship between the absorbance measured (across the entire electrode and the support) and the electrochemical charge that would be required to produce such an amount of absorbance.

Two cases are considered; one where the diffusion field is growing linearly away from the electrode (see Equation 4.1), and one case where the diffusion field is growing radially away from the centre of the electrode (to mimic edge effects). In both cases, scans are used to determine what the adjusted absorbance measurements would be, and these values are compared with the absorbance values expected to be measured.

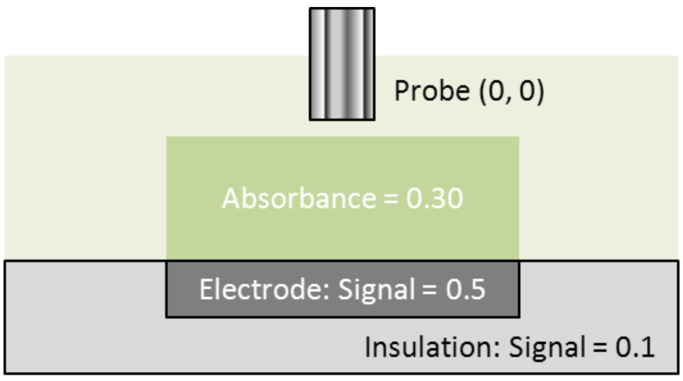


Figure 5.9: Schematic of theoretical setup used to generate theoretical absorbance scans when the absorbance is equal to 0.3.

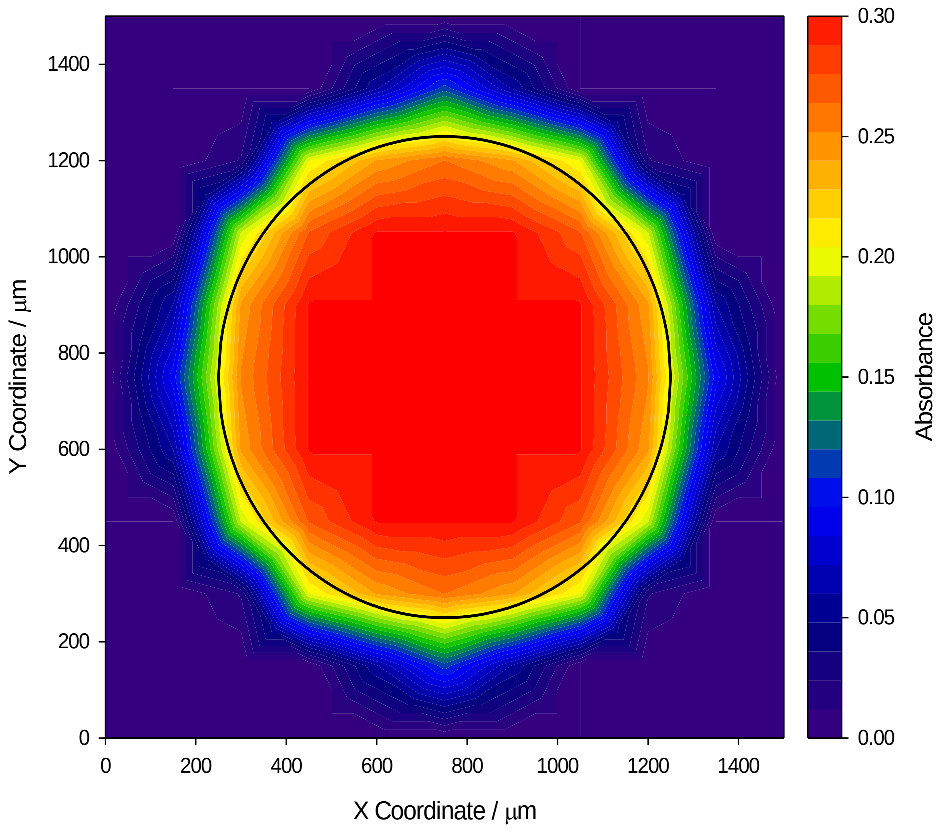


Figure 5.10: Theoretical absorbance scan produced by the scenario described in Figure 6. The edge of the 1 mm diameter electrode is outlined in black and the centre is found at (750, 750).

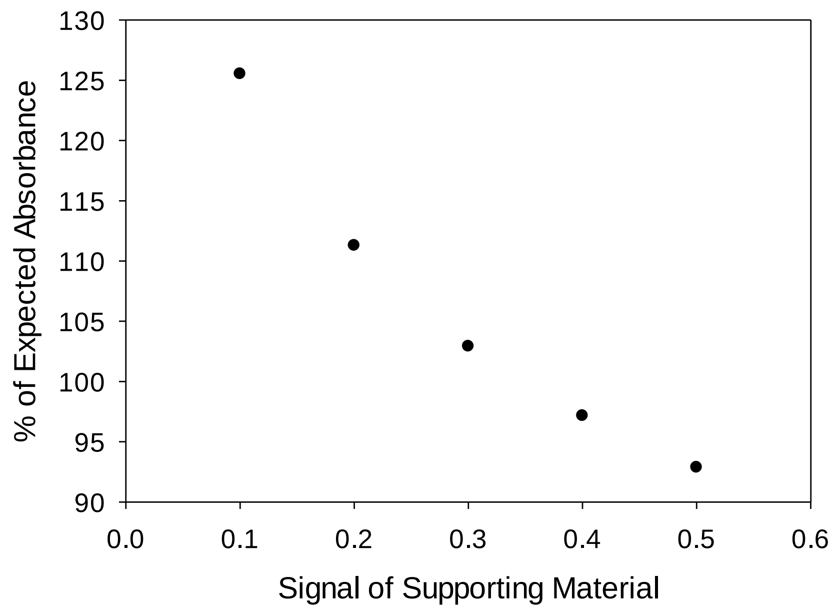


Figure 5.11: Plot showing the inaccuracies of measuring absorbance as a volume, as a function of the reflectivity of the material insulating the disc electrode. The signal produced by the electrode is 0.5 and has a diameter of 1 mm. In each model, a 150 µm distance between data points was used, and the position of the probe was defined by an 11 x 11 grid.

The results from this model show that the reflectivity of the insulation has to be a specific value (0.33 if the reflectivity of the electrode is 0.5) in order to give an accurate absorbance measurement using the method of correction as described. Therefore using this method to quantitatively measure absorbance would be flawed, since there would likely be a bias to the results, depending on the reflectivity of the support relative to the electrode.

However, the method may still be used in a qualitative method to investigate the relationship between absorbance and electrochemical charge, and validate whether edge effects are impacting the absorbance measurements, or whether the divergence seen in Chapter 4 was caused by convection.

In order to find out, two theoretical cases are considered; one where the diffusion field is growing linearly away from the electrode (see Equation 4.1), and one case where the diffusion field is only growing radially away from the centre of the electrode (to mimic edge effects). In both cases, scans are used to determine what the adjusted absorbance measurements would be, and these values are compared with the absorbance values expected to be measured.

5.5.1 Case A – Linear Diffusion:

In this scenario, the absorbance field is growing outwards in one direction away from the electrode, as illustrated in Figure 5.12. Scans are measured for different levels of absorbance.

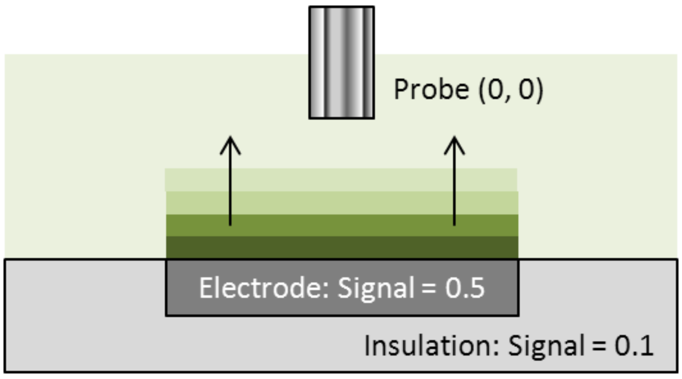


Figure 5.12: Illustration of the scenario whereby all material produced at the electrode stays within the boundary of the electrode.

A range of absorbance values are selected between 0.1 and 0.5 (a typical range absorbance of absorbance values measured during an experiment) and the model is used to determine the absorbance field across the electrode measured by the probe for each value. Each scan is then adjusted to produce an average absorbance value (see Figure 5.5) and this is compared with the expected absorbance value. The expected absorbance value in this case is simply the absorbance value over the electrode; since all the absorbing material is contained within the boundary of the electrode.

The top graph of Figure 5.13 shows how the adjusted absorbance values vary with the expected absorbance values. A linear regression has been applied to the sample and a line-of-best-fit is shown as a black line. The data shows a near linear relationship between the two. This implies that should this scanning technique be used with a real redox system (like the Ir(III)/(IV) system in Chapter 4), and the redox products are diffusing in one dimension, then there should still be a linear relationship observed between the electrochemical charge passing through the electrode and the total absorbance measured across the electrode.

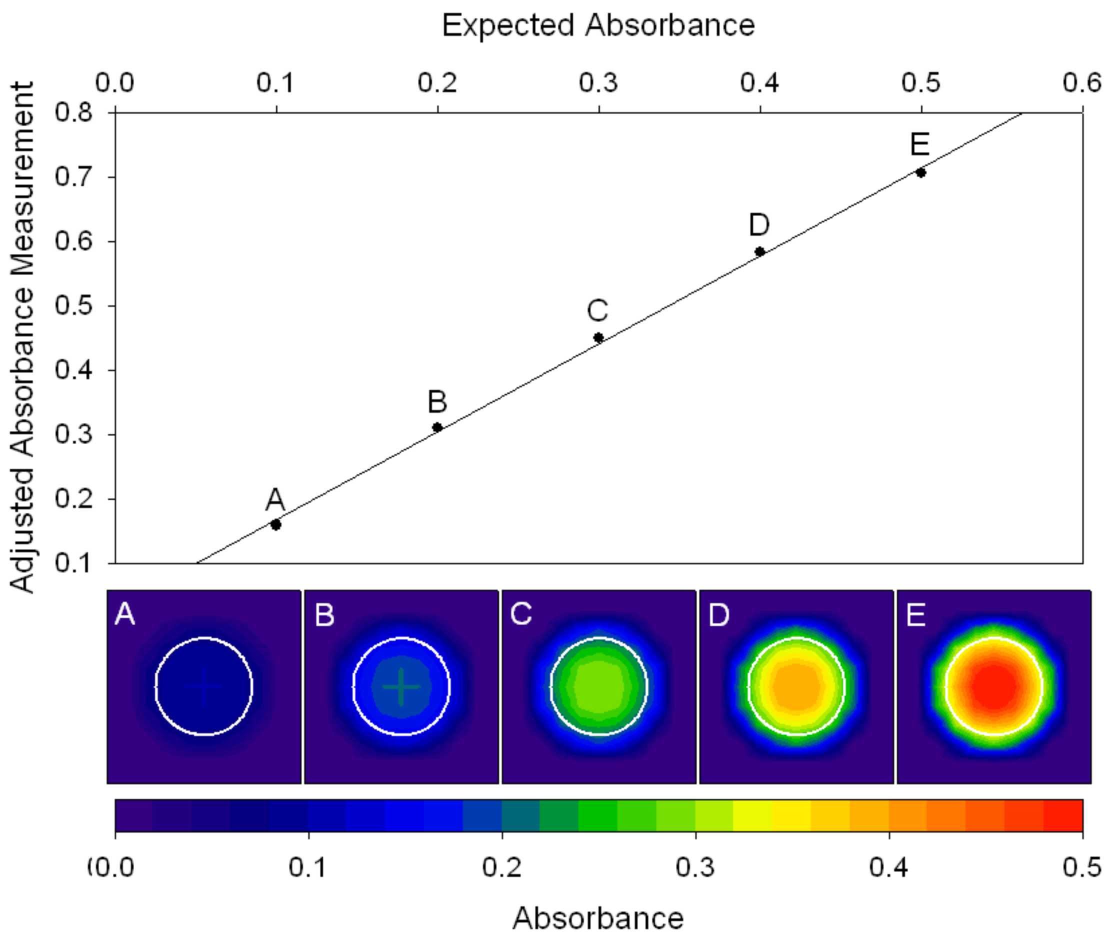


Figure 5.13 (Top) Overall absorbance measured by the probe as a function of the absorbance over the electrode. **(Bottom)** Contour plots of the absorbance fields as measured by the model probe for absorbance values of 0.1 (A), 0.2 (B), 0.3 (C), 0.4 (D), and 0.5 (E). The outline of the electrode is shown as the white circle. In the model, the signal across the electrode is taken as 0.5 and the signal across the insulating material is taken as 0.1, as described in Figure 9. The diameter of the electrode is 0.5 mm, and each contour plot covers an area measuring 1 mm by 1 mm. R^2 is equal to 0.997.

5.5.2 Case B – Radial Diffusion:

In this scenario, the absorbance field is modelled to only grow radially outwards; starting from the boundary of the electrode, as shown in Figure 5.14. The level of absorbance across the electrode is kept at a constant level, while the radius of the region it covers is incrementally increased with each scan. To begin with an absorbance level of 0.3 is present within the boundary of the electrode. In the next instance, the absorbance level is still 0.3; but has now extended by 25 μm (beyond the edge of the electrode) in all horizontal directions.

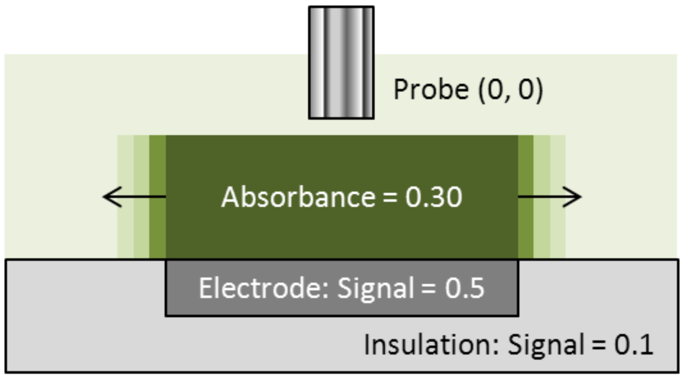


Figure 5.14: Scenario where the absorbance field diffuses radially outwards, beyond the edge of the electrode. The field maintains a uniform concentration throughout, such that it always has an absorbance of 0.3.

In order to link this absorbance growth with an electrochemical response, Faraday's Law states that the charge passed through the electrode is proportional to the moles of redox products produced. The number of moles will be equal to the product of the average concentration of redox products and the volume of electrolyte in which they occupy. The model dictates that the level of absorbance is considered constant throughout, so the average concentration of redox products and the height of the absorbance field must also remain constant (as dictated by the Beer-Lambert Law). Therefore, the only varying parameter is the radius of the region that the redox species occupy.

Therefore, charge can be expressed as being proportional to this radius of the diffusion field, r, squared:

$$Q \propto m = cV \propto r^2$$

Therefore, if Equation 4.1 holds true, then there should be a linear relationship between absorbance and r^2.

The top of Figure 5.15 shows the relationship between r^2 and the adjusted absorbance measurement (redistribution of absorbance into the boundary of the electrode). A linear regression has been applied to the data and is shown as the black line. Beneath are the scans that generated each data point. The relationship appears to be nearly linear, suggesting that if absorbing material was to diffuse radially outwards, the adjusted absorbance measurement will still maintain a linear relationship with the electrochemical charge passed through the electrode.

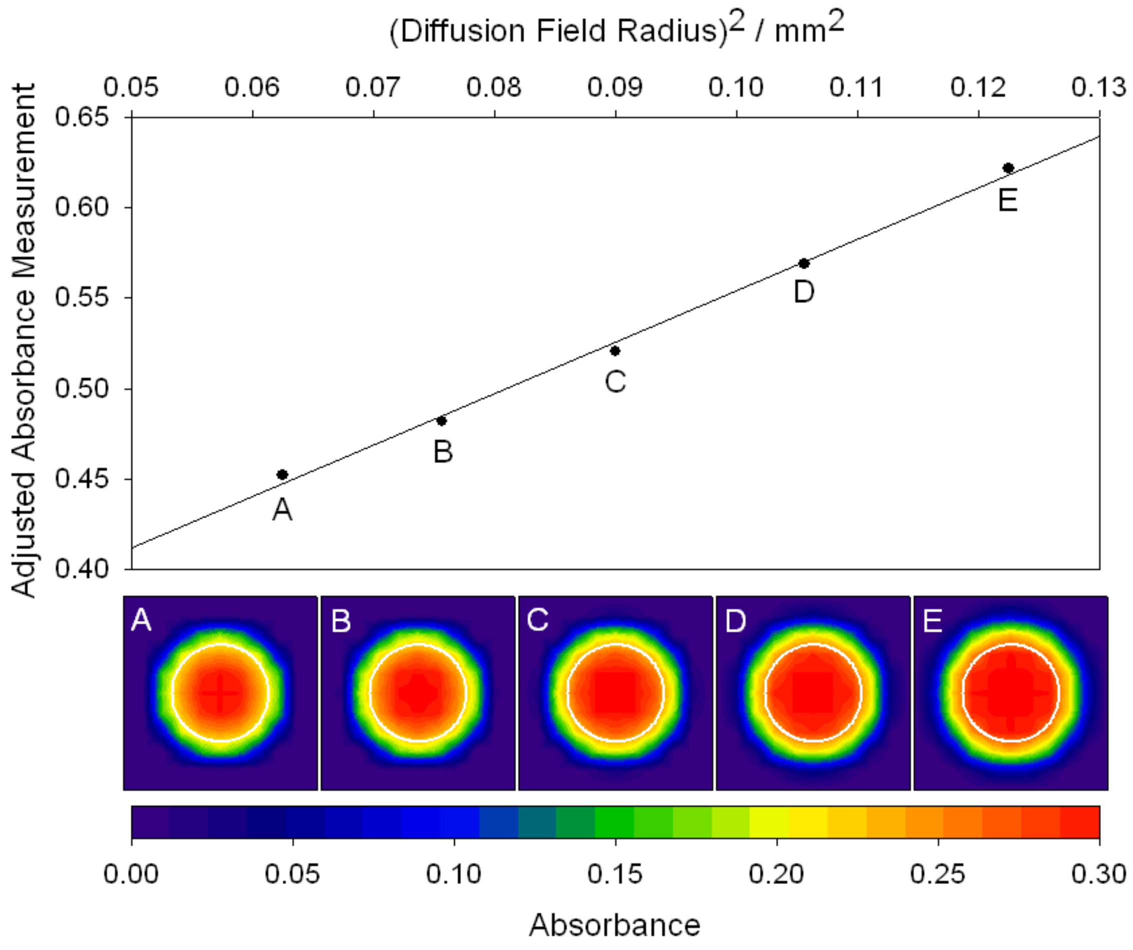


Figure 5.15: (Top) Plot showing the adjusted absorbance measurement as a function of the diffusion field radius squared. **(Bottom)** Contour plots show the absorbance plots as a function of coordinates across the electrode. The outline of the electrode is shown as the white circle. In the model, the signal across the electrode is taken as 0.5 and the signal across the insulating material is taken as 0.1, as described in Figure 9. The diameter of the electrode is 0.5 mm, and each contour plot covers an area measuring 1 mm by 1 mm. R^2 is equal to 0.996.

In both cases, near linearity is maintained during the growth of the diffusion layer. Therefore, the direction that the absorbing material diffuses towards should not affect the linearity of the relationship between the electrochemical charge passing through the electrode and the adjusted absorbance measurement across the whole electrode. If the WSWE technique was applied to a real redox system, then there should be a linear relationship between the adjusted absorbance values and the electrochemical charge passing through the electrode. As stated earlier in the section, the data would probably not be reliable enough to calculate any molar extinction coefficient values. However, the data should be able to explain whether the lower than expected absorbance values seen in Chapter 4 are due to edge effects or natural convection within the cell.

5.6 Potential Step Experiment using $[Ir(Cl)_6]^{3-}$ Redox System

A similar potential step experiment to Section 4 was carried out using a 0.5 mm Pt disk WSWE (fabricated in the same way as the 1 mm diameter WSWE was) and the $[Ir(Cl)_6]^{3-}$/ $[Ir(Cl)_6]^{2-}$ redox couple. The electrochemical cell used a SCE reference electrode, and platinum gauze for the counter electrode, and the cell was filled with an aqueous electrolyte containing 5 mM $(NH_4)_3IrCl_6$ and 0.1M $SrNO_3$. The bifurcated reflectance is placed about 1.6 mm from the surface of the electrode, where the peak signal was measured.

Initially, the potential is held at +0.5 V (*vs.* SCE). Recording absorbance and current is initiated when the potential is stepped to +1.0 V (*vs.* SCE) and the $[Ir(Cl)_6]^{3-}$ species is oxidised. The potential is held for 15 seconds, and then brought back down to +0.5 V for another 45 seconds to allow the cell to return to initial conditions. The probe is then moved to a different point above the electrode, and the process is repeated again.

Figure 5.16 shows the electrochemical and absorbance data recorded from the experiment. The electrochemical data shows a typical current transient response from the electrode. The charge consumed in each case was calculated numerically using Trapezium Rule [57].

Also shown are how both the absorbance over the centre of the electrode (purple) and the corrected absorbance value (green) of the whole electrode vary as a function of the charge passing through the cell. Linear regression has been applied to both sets of data lying between $Q = 0$ μC and $Q = 8$ μC and the lines-of-best-fit are shown. In this region, excellent linearity is observed between the electrochemistry and both sets of absorbance data; similar to the results seen in Chapter 4. However, beyond $Q > 8$ μC, the absorbance values over the centre of the electrode start to deviate from this linear relationship, as was observed previously. In this system the absorbance values can be seen to deviate below the expected values. Meanwhile, the corrected absorbance values have maintained the same linear relationship with the electrochemical data beyond this threshold.

The contour plots show the growth of the absorbance field as a function of position across the electrode and time. For reference, an approximate outline of the working electrode is given in each contour plot (white line), as a 500 μm diameter circle. The contour plots show the gradual growth of redox products over time. Importantly, the contour plots show that all of the oxidation products stay in close proximity to the electrode, and seem to be unaffected by any convection occurring within the cell – unlike the experiment conducted with thymol blue.

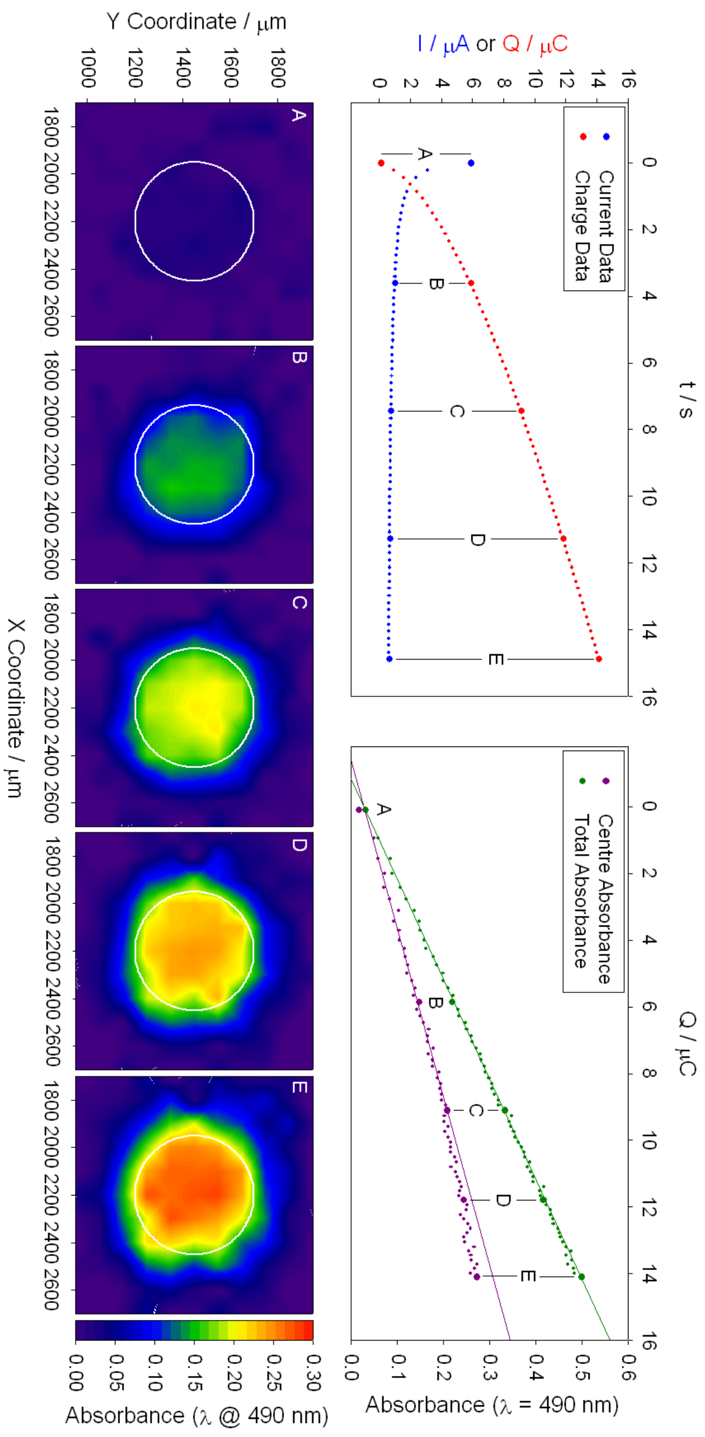


Figure 5.16: Chronometric and absorbance response from a potential step applied to the 0.5 mm diameter WSWE in 5 mM $(NH_4)_3IrCl_6$ and 0.1M $Sr(NO_3)_2$. Current data is shown in blue, while charge data is shown in red. Absorbance measurements made at the centre of the electrode are shown in purple, while absorbance measurements that take into account the whole electrode are shown in green. Lines of best fit are calculated based on data point between A and C. Absorbance plots A to E show illustrate the spread of Ir(IV) species forming at the electrode and spreading across the support.

As predicted in Figure 5.11, the adjusted absorbance measurements are consistently larger than the absorbance measurements recorded over the centre of the electrode.

The contour plots show that the oxidation material may be moving slightly due to convection in the cell, but not by enough that would affect absorbance measurements at the centre of the electrode. All the peak absorbance measurements are found to be lying within the boundary of the electrode. Along with the results from Chapter 4, this strongly suggests that the deviation between absorbance and charge previously seen at the centre of the electrode is caused by edge effects.

5.7 Conclusions

The work presented here shows that while the adjusted absorbance measurements cannot be reliably used to quantitatively assess the molar extinction coefficients of any absorbing redox materials, they can be certainly be used to assess the whether all the faradaic processes that measured are producing the corresponding absorbing products.

This method of conducting multiple potential pulses with the probe in different position across the electrode can also be used to qualitatively assess what effect convection is having on the redox products at the electrode. The first experiment using thymol blue excellently shows how convection was shifting the cloud of alkali material away from the electrode. Conversely, the second experiment was able to show that convection has a very limited effect on the oxidised $[IrCl6]^{2-}$ products.

However, this method of using adjusted absorbance values cannot be quantitatively used to compare absorbance values with the electrochemical processes occurring in the cell, to determine molar extinction coefficients.

Chapter 6: Single Fibre Reflectance Probe

6.1 Introduction:

As shown in chapter 3, the bifurcated reflectance probe investigates an area 250 µm to 300 µm in diameter. This area of inspection is too large to be able to sufficiently investigate spectral changes occur in a 50 µm diameter microwell. To demonstrate this, Figure 6.1 shows a scan of an array of 50 µm diameter microwells silver chloride electrodes at the base. The microwells are separated by a distance of 500 µm.

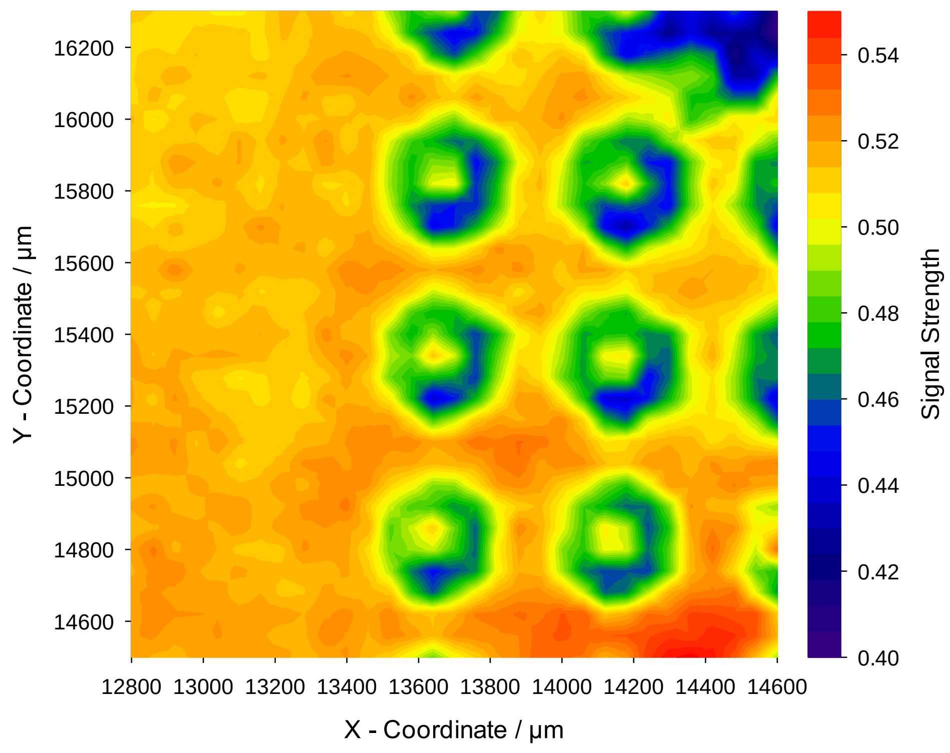


Figure 6.1: Scan of an array of silver chloride microwell electrode using the bifurcated probe. Data points are 60 µm apart. The probe is 1500 µm from the array, with only air present between the two.

The scan shows each microwell as a small 100 µm wide region of high signal situated within a 300 µm wide ring of relatively low signal. As would be expected, these features are separated by a distance of approximately 500 µm.

The smaller bright region reveals the location of the microwell, and so shows that the bifurcated probe is capable of locating these.

However, silver chloride tends to scatter light, due to it rough surface. As such, a low signal should be expected when the probe is measuring reflectance within the microwell, but a small region of high reflectance is seen. This can be explained by Figure 3.16 in Chapter 3, which shows the regions of the reflective surface that are under inspection when the bifurcated reflectance probe is placed above them. At the centre of this plot is a small 100 µm wide region that goes undetected by the probe. This implies that when the probe is directly above the microwell, the microwell itself will be undetected; and all of the signal that is measured will be made up from light reflecting off the shiny substrate of the chip.

However, if the probe is moved between 50 to 150 µm away from the microwell (along the horizontal plane), then the microwell now falls into the region of inspection, and as such scatters light and reduces the signal to the probe. This explains why there is a ring of low signal around each point where a microwell should be.

The signal of these rings is still about 0.45 (compared to the 0.5 – 0.55 signal from the substrate), and most of this signal is produced by the reflective substrate. As such, any changes occurring in the microwell would have an insignificant change on the signal reaching the probe, and absorbance measurements would be minimal.

In order to overcome this problem, measurements of spectral changes occurring within the microwells will be made using a 62.5 µm diameter single fibre reflectance probe.

The advantage of using this probe is that it should have a much smaller area of inspection compared to the bifurcated probe previously used. Hence it should be ideally suited to inspect spectral changes occurring inside small microwells.

However, there are disadvantages with using this system. When using the bifurcated probe, the light source was shining on to six 200 µm diameter fibres (an area of 0.188 mm^2), whereas in the single fibre system, the light source is shining on to one 65 µm diameter fibre (an area of 3070 µm^2). Using the same light source, the single fibre is only capable of emitting about 1–2% of the light that the bifurcated probe can emit on the electrode. Therefore, more light is required for the system; otherwise the signal measured by the probe will be incredibly vulnerable to noise.

This can be achieved by either using a brighter light source, increased integration times. However, if absorbance measurements require more time to take, then this introduces problems with studying systems that have rapid transients, since less data can be recorded.

Another problem that needs to be considered is internal reflection occurring at the glass–liquid interface at the tip of the probe, which was first observed during preliminary experiments. When light passes from the glass of the optical fibre through to the liquid electrolyte, some of the light is reflected at the interface; due to the difference in density between the two. Some of this light reaches the spectrometer and gives any signal that are recorded a positive bias. This is in turn means that absorbance measurements will have a negative bias:

$$\log\left(\frac{I_0+I_{ir}}{I_1+I_{ir}}\right) > \log\left(\frac{I_0}{I_1}\right) \quad \text{if} \quad I_0 > I_1 \qquad (6.1)$$

For the bifurcated probe, this internal reflection is not an issue; any internally reflected light is transmitted back to the light source and therefore remains undetected.

This internal reflectance can be solved by taking a different dark reading. In previous experiments with the bifurcated probe, the dark reading is subtracted from both the reference signal and all spectral measurements carried out during the experiment; as a way of cancelling out any unwanted background light that might be picked up by the probe. Light caused by internal reflection in the single fibre probe can be juxtaposed with this background light.

At the start of all experiments, the probe is positioned several millimetres away from the electrode so that no detectable light is reflected back to the probe by a surface beneath. Only the internally reflected signal is detected. By subtracting this signal from both the reference signal and all measurements recorded during the experiment, the internally reflected signal is nullified.

One last factor to consider is that the signal will be stronger the closer the probe is to the reflective electrode surface. This is because the light is being collected by the same glass fibre that is emitting the light, rather than another fibre. However, while it is ideal for the fibre to be as close as possible to the reflective surface in order to achieve a high signal, there is a chance it can interfere with the mass transfer mechanisms of any redox materials that are produced at the electrode under investigation. If the probe is too close and penetrates the theoretical Nernst Diffusion Layer, then typical mass transfer behaviour would be hindered and absorbance measurements are likely to be lower than expected. Typical redox couples used so far have diffusion coefficients below 1×10^{-5} cm^2 s^{-1}, and products are rarely generated for over 60 s. The values can be used together with an equation that gives the Nernst Diffusion Layer thickness:

$$\delta = \sqrt{\frac{D}{t}} \qquad (6.2)$$

The Nernst Diffusion Layer thickness is given as 3.4 μm. Therefore, as long as the probe is placed 10–20 μm away from the surface of the electrode, the probe should not interfere with the mass transfer regime of any redox reactions. If an experiment calls to be carried out for longer time scales, then this distance will need to be reconsidered.

6.2 Example Scans

Several scans demonstrate the ability of the single fibre reflectance probe to detect smaller features than were possible with the bifurcated probe. Figure 6.3 shows a scan of a dual electrode, which consists of a 500 µm diameter electrode and a much smaller 50 µm diameter electrode. The scale of the signal measurements have been adjusted to better highlight the presence of the smaller electrode, since the larger electrode is highly reflected.

The dual electrode is insulated in glass, which does not reflect much light. Despite this, there is still quite a strong signal present when the probe is placed above the glass. The boundary of the glass insulation can be seen in the bottom-right hand corner, where the signal drops to 0.05. In this region, the probe is above nothing that is close enough to reflect light back to it, and yet a signal of 0.05 is still measured – compared to about 0.065 when the probe is above the glass. Therefore, most of the signal measured when the probe is over the glass is actually caused by internal reflection occurring at the tip of the probe.

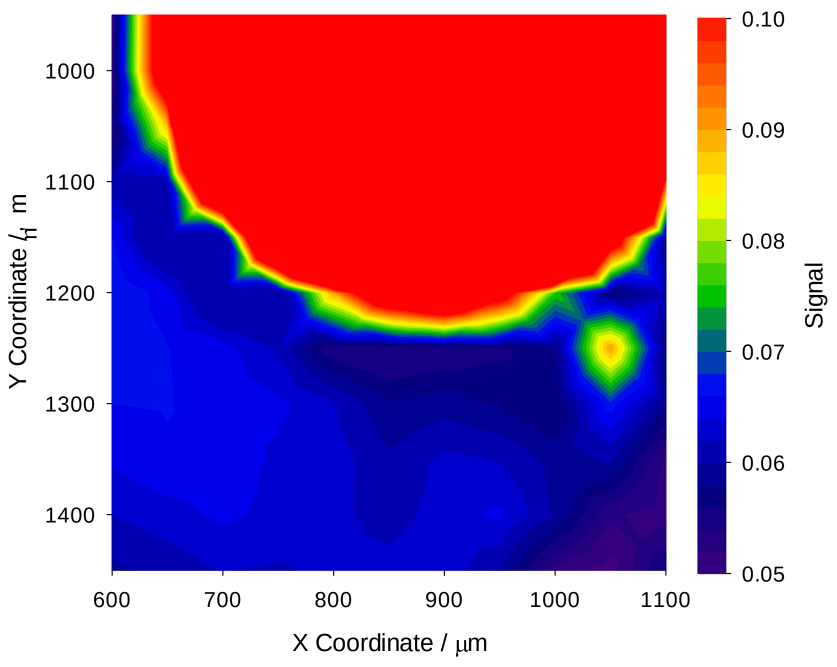


Figure 6.3: Scan of a dual electrode featuring a 500 µm diameter silver disk and a 50 µm diameter platinum disk, which was carried out using the single fibre reflectance probe. Measurements were taken 50 µm apart along the X and Y axes.

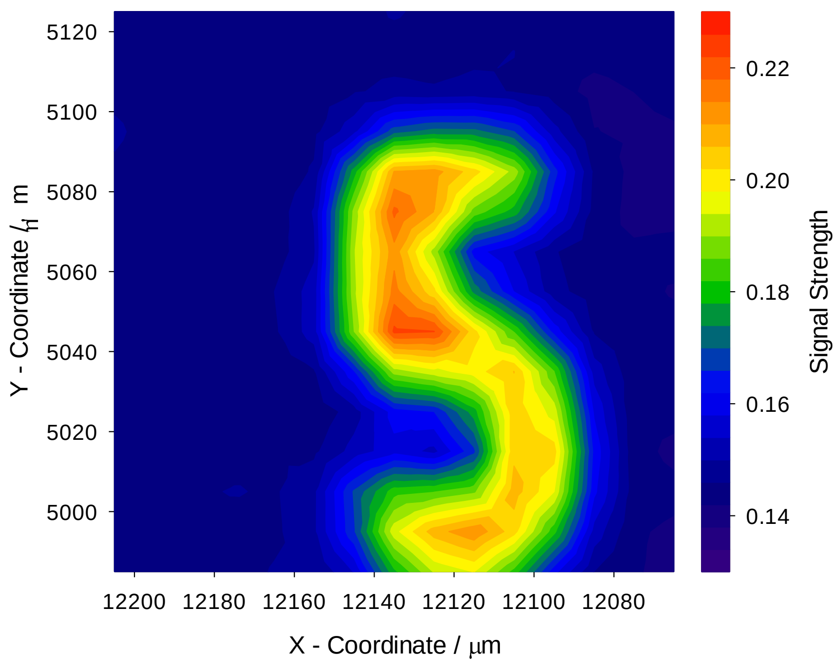


Figure 6.4: Image of a number '5' that has been etched into the substrate of the microwell arrays. Data points are separated by 10 μm along the x and y axes. Neither the array nor the probe are submerged in electrolyte.

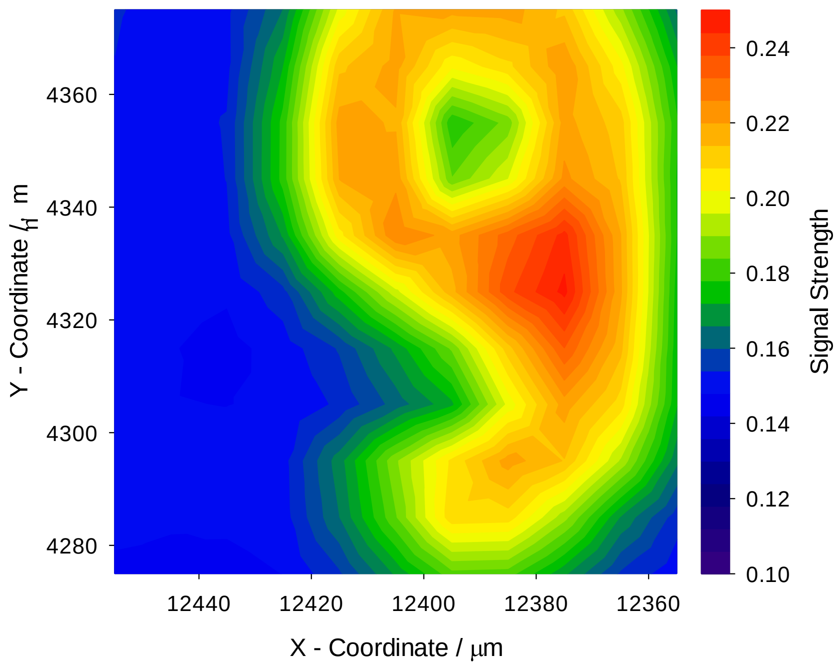


Figure 6.5: Image of a number '9' that has been etched into the substrate of the microwell arrays. Data points are separated by 10 μm along the x and y axes. Neither the array nor the probe are submerged in electrolyte.

Other demonstrations of the ability of the single fibre probe to pick out fine detail are shown in Figures 6.4 and 6.5. These figures show scans of numbers that are etched into the substrate of the microwell arrays. These numbers are used to indicate the number of individual wells relative to the output pins of the chip. As shown by the figures, these numbers are incredibly small (≈ 100 μm tall), but the probe is still able to detect them.

Figure 6.6 shows a scan of 50 μm diameter microwell with a silver chloride electrode at the base. The bright ring seen in the middle is from the conductive tracks that connect the electrode to an output pin at the base of the chip. The silver chloride electrode lies within this ring.

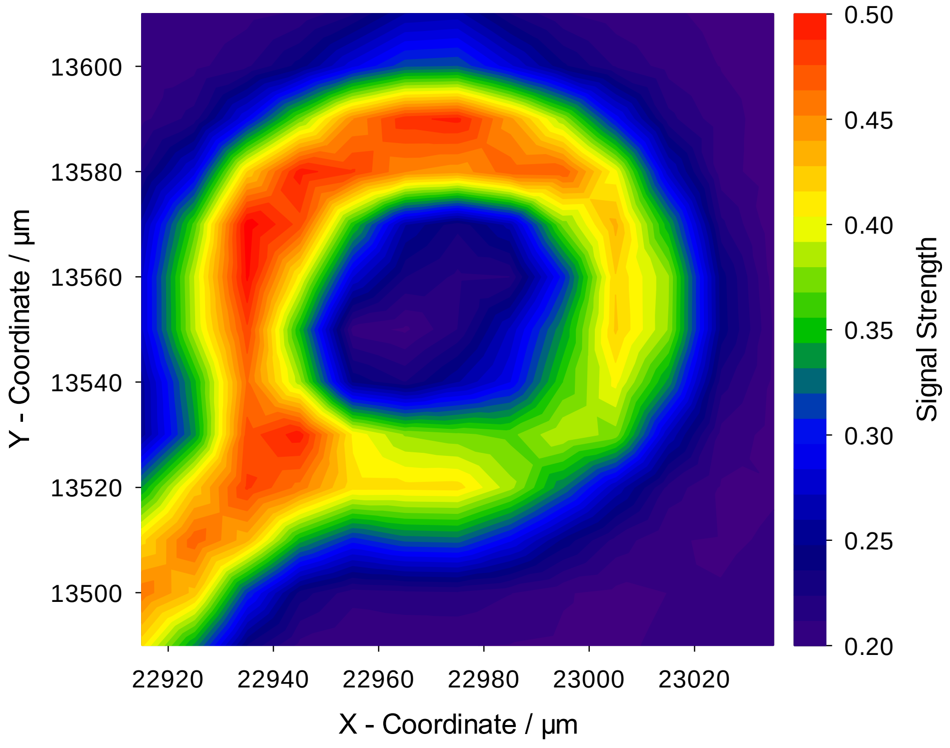


Figure 6.6: Scan of a 50 μm diameter silver-chloride microwell electrode. Data points are separated by 10 μm along the x and y axes. Neither the array nor the probe are submerged in electrolyte. The silver-chloride microwell lies within the bright ring. The bright ring is caused by the reflective conductive tracts connected to the microwell.

By comparison, Figure 6.7 shows a scan of a 50 μm diameter microwell with a platinum electrode at the base. Unlike the silver chloride electrode, the platinum is barely distinguishable from the conductive tracks that line the outside of the microwell; due to their similar reflective properties.

These two figures demonstrate that the single fibre probe is able to clearly identify the microwell, and should be suitable to inspect spectral changes occurring inside.

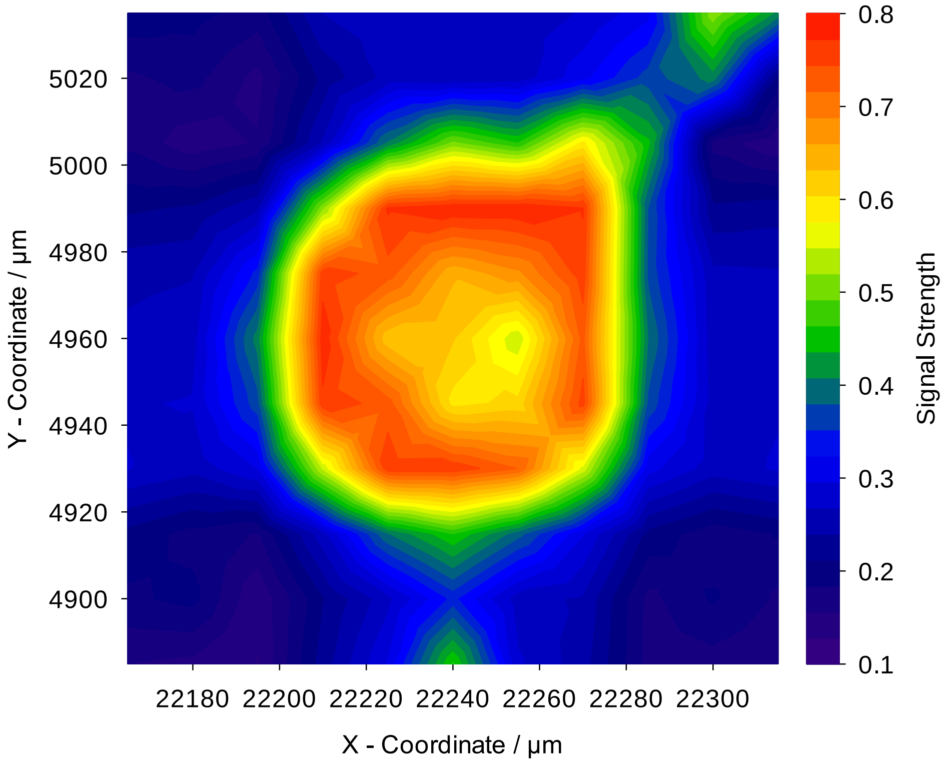


Figure 6.7: Scan of a 50 μm diameter platinum microwell electrode. Data points are separated by 10 μm along the x and y axes. Neither the array nor the probe are submerged in electrolyte. Note, in the top-right corner the conductive tract connected to the electrode can be seen.

6.3 Modelling

If the probe is being used to inspect a microwell with a diameter of 50 μm, the signal measured by the probe may be influenced by regions outside the microwell. In this instance, there will be a negative bias introduced to any absorbance measurements recorded. As such, it is important to know what size the area of inspection is.

As previously stated in Chapter 3, the profile of light intensity emitted an the optical fibre obeys a Gaussian distribution across a surface:

$$I(r,z) = I_0 \left(\frac{w_0}{w(z)}\right)^2 e^{\left(\frac{-2r^2}{w^2(z)}\right)} \qquad (6.3)$$

Using the same approach as with the bifurcated probe, the system can be simplified by assuming the electrode is a perfect reflector, and that a light emitting fibre is shining light on to a 62.5 μm optical from twice the distance between the probe and the electrode surface.

By using this same method, an initial estimation can be made to find out what proportion of light that is falling into the microwell is making up of the total light that is returned back to the single fibre probe.

Assuming that the beam emitted from the single fibre probe behaves similarly to the fibres of the bifurcated probe, the relationship between w(z) and z can be scaled down by an appropriate factor that reflects the differences in diameters of the fibres used.

For the bifurcated probe with 200 μm diameter fibres, w_0 was 40.8 μm. The equivalent value for a fibre with a 62.5 μm would be 12.75 μm. If the fibre is about 20 μm from the surface of the electrode, the value of w(z = 20 μm) is only going to be marginally larger; about 14.0 μm.

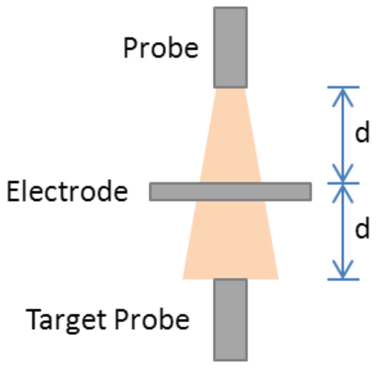


Figure 6.8: Schematic of how the light emitted from the probe can be ideally modelled to find how the probe behaves.

Using these figures, the percentage of light reflecting back to the probe that reached the 50 μm is about 98%. This is assuming that the SU-8 substrate that surrounds the microwell is of a similar brightness to the platinum electrode found inside the microwell.

While a lot of assumptions have been made to reach this figure, 98% is high enough to show that the single fibre probe is small enough to fully inspect a 50 μm diameter microwell without a significant amount of error.

6.4 Redox Results

Initial tests saw the single fibre probe used to investigate spectroscopic changes occurring in a platinum microwell electrode when an electrochemical redox reaction occurs. The redox couple in this case is the oxidation of ferrocyanide species to ferricyanide. Ferricyanide absorbs strongly at 450 nm wavelength compared to ferrocyanide.

Figure 6.10 shows cyclic voltammetry conducted on a 20 µm deep, 50 µm diameter platinum microwell electrode in an aqueous solution of 0.2 M $K_4Fe(CN)_6$ and 0.1 M Na_2SO_4. The graph shows that as the potential is increased beyond -0.3 V (*vs.* MMS), the $Fe(CN)_6^{4-}$ species stars to oxidise to the $Fe(CN)_6^{3-}$ species. When the potential reaches 0.0 V (*vs.* MMS), the reaction is solely controlled by the mass transfer mechanism, and the current reaches a steady state. The limiting current is equal to 0.7 µA.

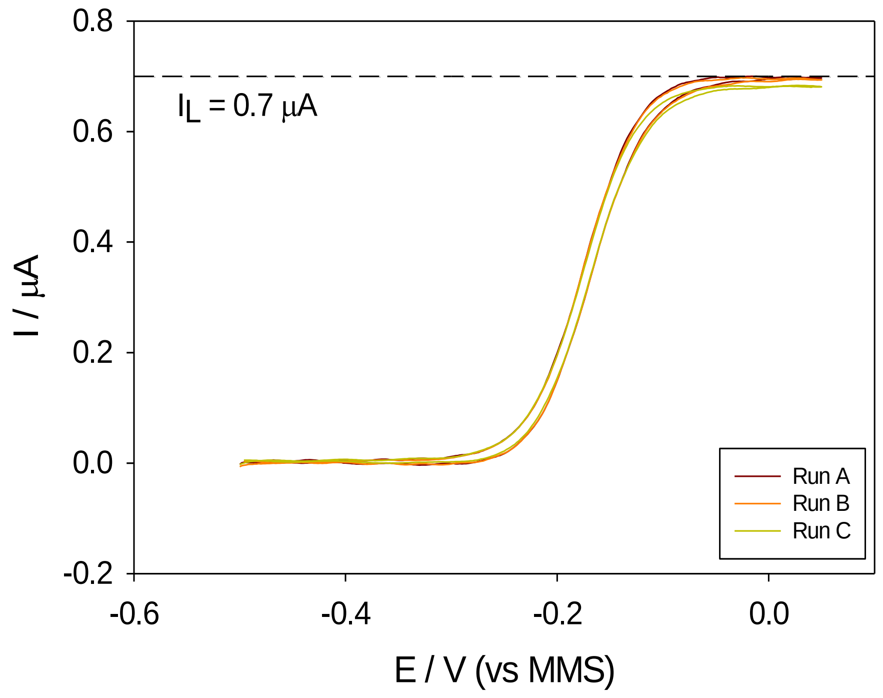


Figure 6.10: Cyclic voltammograms conducted on a 20 µm deep, 50 µm diameter Pt microwell at 5 mV s^{-1}, showing an approximate limiting current of 0.7 µA. The cell was filled with an aqueous 0.2 M $K_4Fe(CN)_6$ and 0.1 M Na_2SO_4 solution.

The limiting current passing through a microwell has been described by Bond *et al.* to obey the equation [67]:

$$i_{lim} = 4nFDCr\left(\frac{\pi r}{\pi r+4L}\right) \qquad (6.4)$$

Where all symbols have their usual meaning and L is equal to the depth of the microwell. Using the limiting current found in Figure 2, the diffusion coefficient for the $Fe(CN)_6^{4-}$ species is found to be 7.33 cm^2 s^{-1} – close to the value found in the literature [68].

To assess the ability of the single fibre probe to accurately detect absorbance changes inside the microwell, a series of similar potential steps were applied to the microwell electrode while the probe measured absorbance changes at different points above the well. For each set of potential steps, the probe was placed in a different position along an 11 $\times$ 11 grid above the microwell – similar to the technique used in Chapter 5.

Each data point for the 11 $\times$ 11 grid is separated by 15 μm along both x and y directions. The probe was placed at 100 μm from the insulating surface, so as to not interfere with the mass transfer mechanism occurring around the electrode.

Starting from coordinates (5170, 23170), the probe measured the absorbance after a potential step from −0.5 V to +0.05 V (*vs.* MMS) was applied to the microwell electrode for 6 seconds and the $Fe(CN)_6^{4-}$ species was oxidised. After the absorbance was measured, the potential was returned to −0.5 V, and the cell was left for 30 seconds to allow the cell to return to the initial state. Afterwards, the probe was moved along to the next position in the 11 $\times$ 11 grid and the process was repeated. Figure 6.11 shows how the absorbance above the microwells changes with the duration of the potential step.

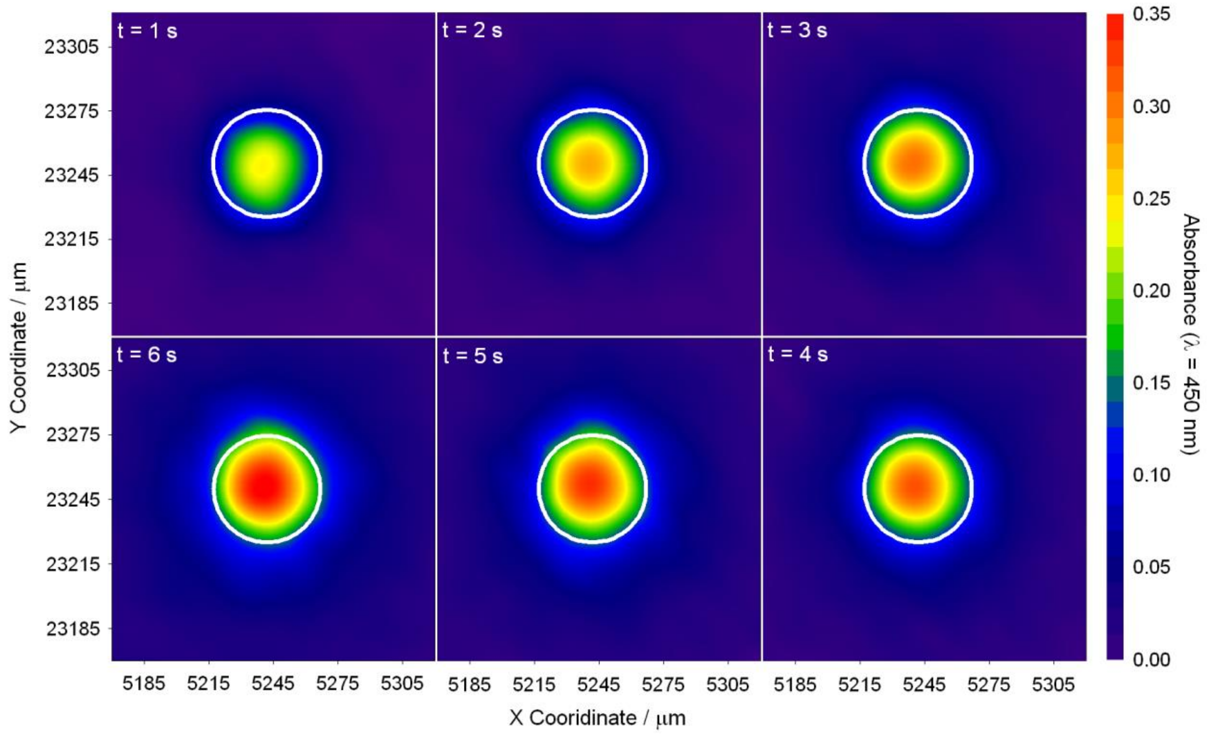


Figure 6.12: Absorbance plots for double potential step experiment performed at each position above the surface of the 50 µm diameter Pt microwell electrode and the substrate, after different time periods. The cell was filled with an aqueous 0.2 M $K_4Fe(CN)_6$ and 0.1 M Na_2SO_4 solution.

Over time, absorbance becomes stronger and radially diffuses away from the microwell as the potential step is applied for longer. The white line indicates the approximate perimeter of the microwell and it can be seen that the $Fe(CN)_6^{3-}$ species diffuses further from the microwell the longer the potential is applied for. The strength of absorbance at the centre of the microwell also increases.

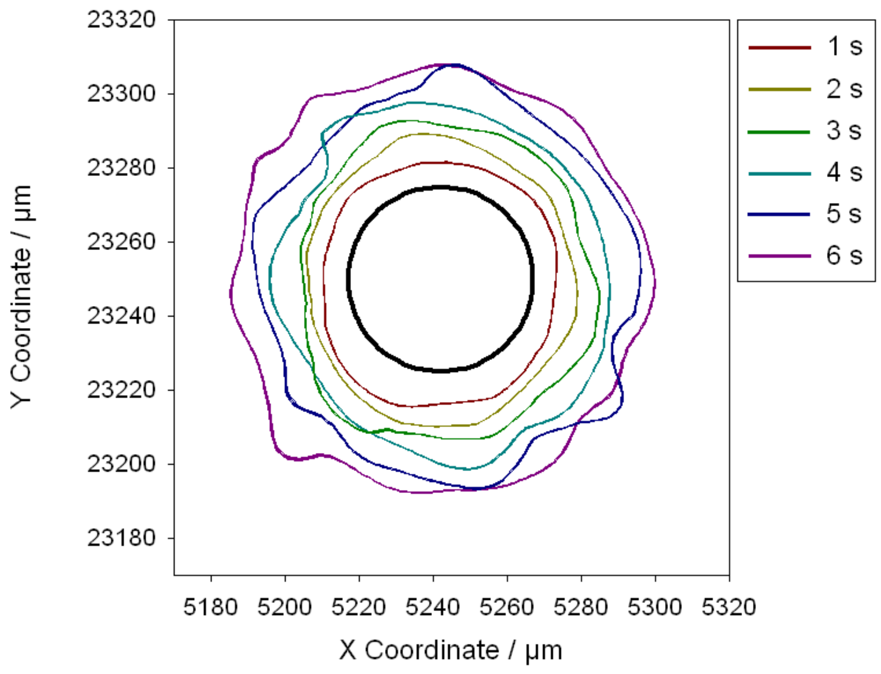


Figure 6.13: Plot showing where absorbance is equal to 0.05 for each of the absorbance plots shown in Figure 6.12.

Figure 6.13 shows overlapping contour lines where absorbance is equal to 0.05 for each of the plots in Figure 6.12. The plot illustrates the radial diffusion of $[Fe(CN)6]^{3-}$ species.

Overall, the data show that the single fibre probe is capable of measuring the diffusion behaviour of redox products produced at a microwell electrode. Similar results should be repeatable with an ordinary microelectrode disk, providing that the support is reflective enough such that noise has a negligible impact on the spectral data.

6.5 Silver Chloride Growth

When a silver electrode is placed in an electrolyte containing chloride molecules (e.g. an aqueous solution of sodium chloride) a driving potential can be used to from a layer of silver chloride:

$$Ag + NaCl \rightarrow AgCl + Na^{+} + e^{-}$$

Figure 6.14 show a voltammogram of a 0.5 mm diameter silver disk electrode in an aqueous solution of 0.01 M NaCl. Absorbance was measured using the bifurcated probe and was placed 2 mm from the electrode surface.

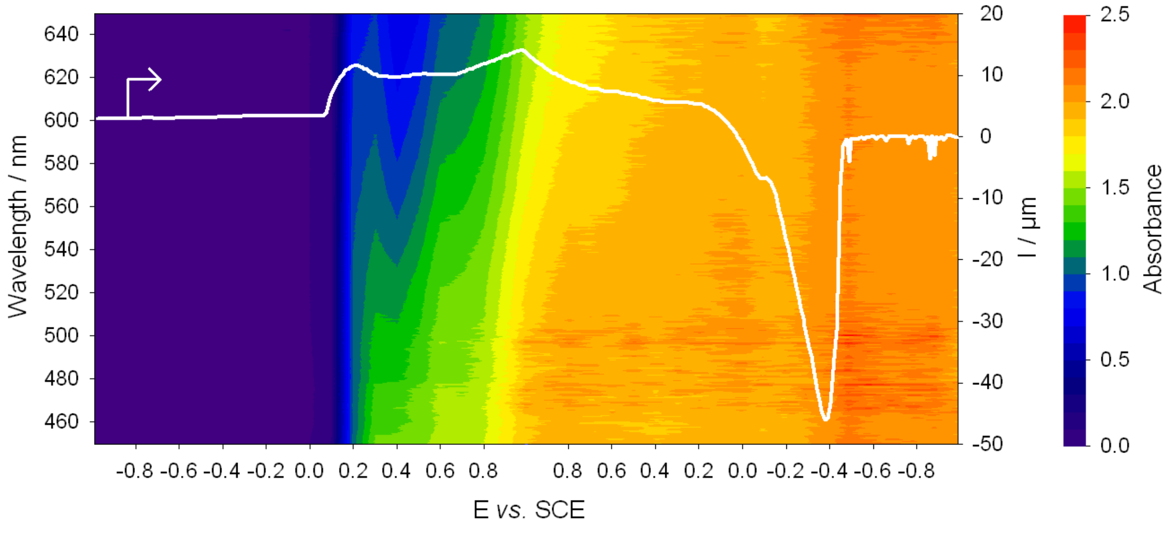


Figure 6.14: Cyclic voltammogram of 0.5 mm Ag disk electrode in 0.01 M NaCl at a sweep rate of 100 mV s^{-1}. Absorbance measurements were recorded using the bifurcated probe, which was placed 2 mm away from the electrode.

At +0.1 V (*vs.* SCE), the silver electrode starts to chloridise and an anodic current is observed. This is accompanied by a large increase in absorbance.

Beyond +0.4 V on the return sweep, the silver chloride is reduced back to silver. However, the absorbance remains high despite the absence of a chloride layer. This is most likely due caused by the lattice structure of the silver electrode changing in order to accommodate chloride ions. When the chloride ions are extracted, the surface of the electrode does not return to the smooth structure that it had at the start of the potential sweep.

The single fibre reflectance probe can be used to inspect how this silver chloride layer grows on the electrode. A scan was carried out on the electrode before and after a series of potential pulses. For each pulse, a potential of +0.2 V (*vs.* SCE) was applied to the electrode for 0.05 seconds, and then the potential was lowered to -0.75 V for 0.05 seconds. Figure 6.14 shows the current response from one of these potential pulses.

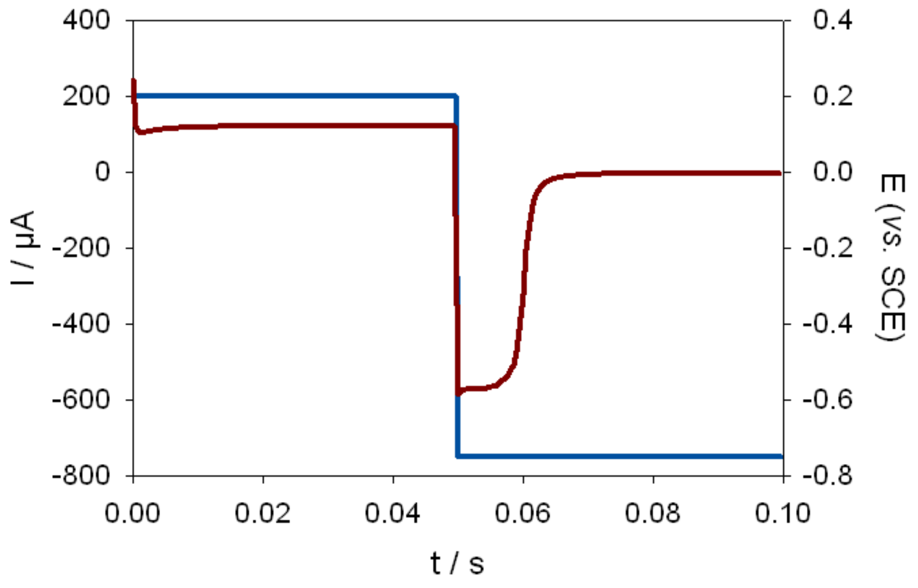


Figure 6.15: Example of current response (red) from potential step (blue) applied to 0.5 mm silver disk electrode in an aqueous solution of 0.01 M NaCl.

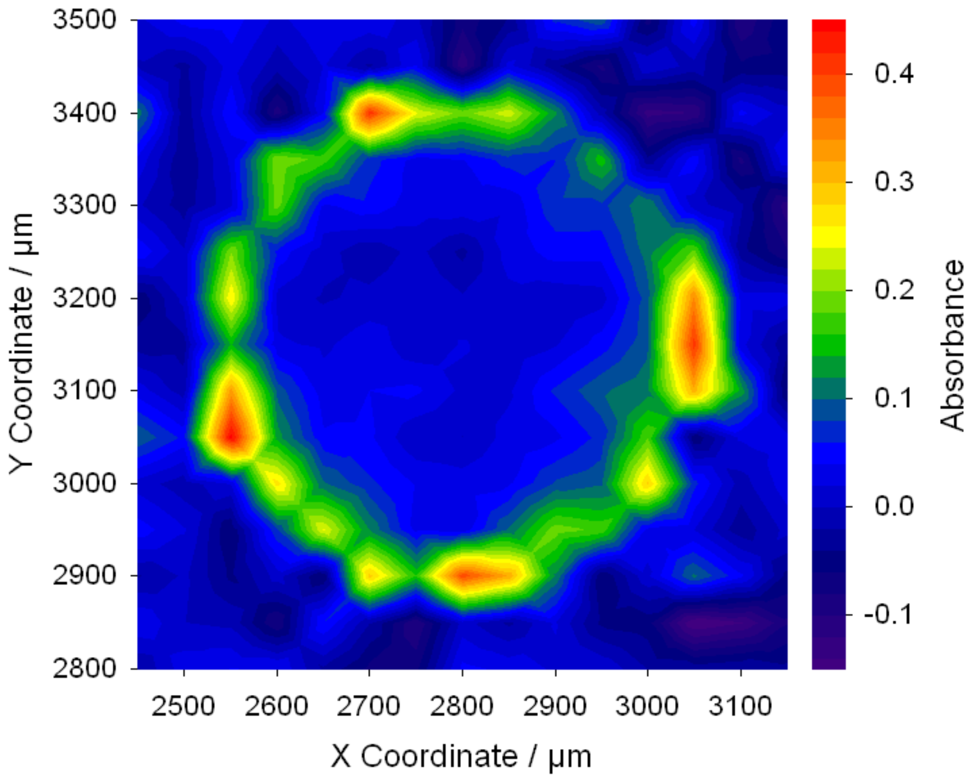


Figure 6.16: Absorbance plot across a 0.5 mm diameter silver electrode in an aqueous solution of 0.01 M NaCl after going through five potential pulses (+0.2 V for 0.05 s to −0.8 V for 0.05 s).

Figure 6.16 shows the absorbance plot after five of these potential pulses have been applied to the electrode. The plot shows a ring of high absorbance around the edge of the electrode. This suggests that the nucleation sites of silver chloride growth are occurring at the edge of the electrode, which would be explained by the greater rate of mass transport at the edge of electrode due to edge effects.

The experiment has shown that the single fibre reflectance probe is capable of inspecting processes that would have been otherwise difficult to observe when using the bifurcated probe.

6.6 Lipid Bilayer Measurements

For the purposes of DNA sequencing, Oxford Nano Technologies inspect their bilayers using fluorescent microscopes. By doing so, they can only inspect the microwells from directly above. Using this method, they are unable to find out how high each bilayer sits above the base of the microwell. Using the single fibre reflectance probe, this work aims to find out how high the bilayers generally sit above the base of the microwells.

To begin with, lipid bilayers are produced on an array of platinum microwell electrodes. To help identify the presence of a bilayer, the lipid solution used is dyes with a small amount of fluorescein, which strongly absorbs light at the 490 nm wavelength.

Figure 6.17 shows the spectrum recorded using the single fibre reflectance probe when positioned directly over wells 125 (left) and wells 126 to 128 (right), after the process of adding lipid bilayers.

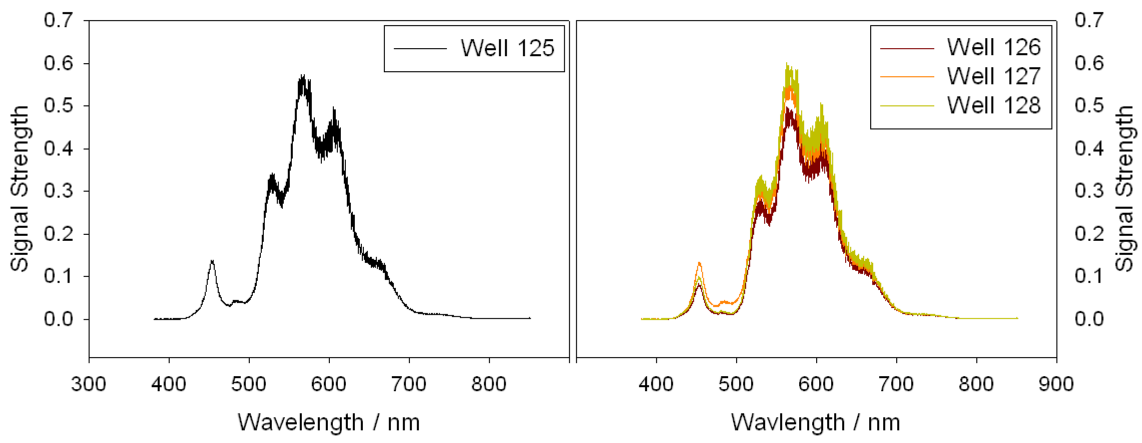


Figure 6.17: Signal measured over centre of 50 μm diameter Pt microwells using the single fibre reflectance probe.

The spectrum from well 125 can be used as a reference to find the relative absorbance spectra over the other wells:

$$A_n = \log\left(\frac{I_{125}}{I_n}\right) \qquad (6.5)$$

Figure 6.18 shows the absorbance spectra produced by applying Equation 6.5 to the spectra shown in Figure 6.17. The figure shows a large absorbance peak at the 490 nm wavelength for Well 126 and Well 128, suggesting the presence of fluorescein within those wells, and hence a bilayer. Had there not been a bilayer, the fluorescein would have been washed out of the microwell by the buffer flush stage, and there would be no absorbance peaks, like the spectra for Well 127.

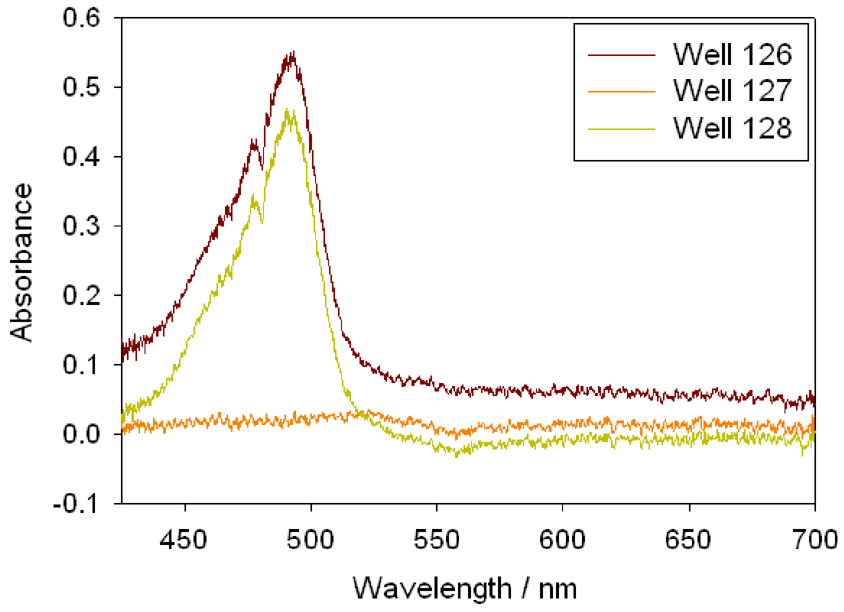


Figure 6.18: Absorbance spectra measured in Wells 126 to 128, using the signal from Well 125 as a reference.

It is worth noting that this is under the assumption that there is no fluorescein present in Well 125. If there was fluorescein present in Well 125 – and was still used as a reference – a well without fluorescein would produce a negative absorbance peak at the 490 nm wavelength.

As shown in Figure 6.18, the absorbance peaks seen in Wells 126 and 128 are of a similar size and shape, but are shifted to different positions along the absorbance axis. This is caused by each microwell electrode having its own unique reflectivity. For example, in Figure 6.17, the spectra for Wells 126 and 128 both have the same shape, suggesting they are the same colour; but the peaks are lower for Well 128, suggesting it is not as reflecting as much light as Well 126. Hence, in Figure 6.18, the absorbance spectra for these wells are separated by an absorbance value of approximately 0.1 across the wavelength range.

In order to qualitatively analyse these absorbance peaks, the spectra have to be corrected so that their unique reflective properties do not affect the absorbance spectra. Fluorescein absorbs a negligible amount of light between the 600 nm and 650 nm wavelength range. By correcting each absorbance spectrum so that the average absorbance measurement between 600 nm and 650 nm wavelengths is equal to 0, the effects of reflectivity are nullified, and a true absorbance spectrum is found.

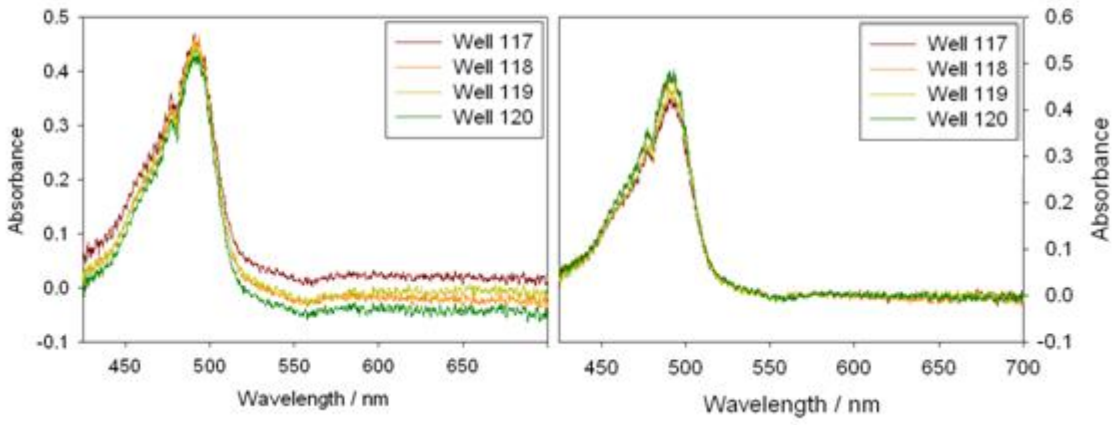


Figure 6.19: **(Right)** Absorbance spectra for wells 117 to 120 when using Well 125 as a reference. **(Left)** Corrected absorbance spectra so that the average absorbance measurements between 600 nm and 650 nm are equal to zero.

For example, Figure 6.19 shows 4 absorbance spectra that are all misaligned with one another. However, by setting the average signal between 600 nm and 650 nm wavelengths to 0, the spectra become aligned with another; making it easier them easier to analyse.

Table 6.1 shows the different values of absorbance at the 490 nm wavelength for over a dozen different wells that were found to contain lipid bilayers. The average value of absorbance was found to be equal to 0.45 with a standard deviation of 0.018.

In order to convert this absorbance value to a path length, the molar extinction coefficient of the fluorescein and lipid solution at 490 nm wavelength was found. Due to the difficulties of accurately measuring the weight of the small amounts of fluorescein used in making the lipid mixture, and hence the concentration, the molar extinction coefficient was find by diluting the same lipid mixture to a series of very low concentrations in an aqueous solution of 0.4 M KCl and 25 mM Trizma buffer, and measuring the absorbance. Figure 6.20 shows the absorbance of the lipid solution as a function of relative percentage concentration to that used in the microwell (not an absolute concentration).

Well No.	A(490 nm)	A(600-650nm)	Adjusted A (490 nm)
128	0.453623077	-0.007956369	0.461579446
126	0.532738462	0.058743711	0.473994751
124	0.493861538	0.02044662	0.473414918
123	0.343376923	-0.093081447	0.43645837
122	0.358846154	-0.107375472	0.466221626
121	0.331453846	-0.081976101	0.413429947
120	0.421169231	-0.041127673	0.462296904
119	0.435223077	-0.007733326	0.442956403
118	0.449884615	-0.020347733	0.470232348
117	0.451184615	0.020848537	0.430336078
116	0.3223	-0.140472013	0.462772013
115	0.415484615	-0.030221069	0.445705685
114	0.415353846	-0.033895597	0.449249444
113	0.400269231	-0.030853459	0.43112269
		Average	0.451412187
		Std Dev	0.018026531

Table 6.1: Raw absorbance and adjusted absorbance values for wells found to be containing fluorescein.

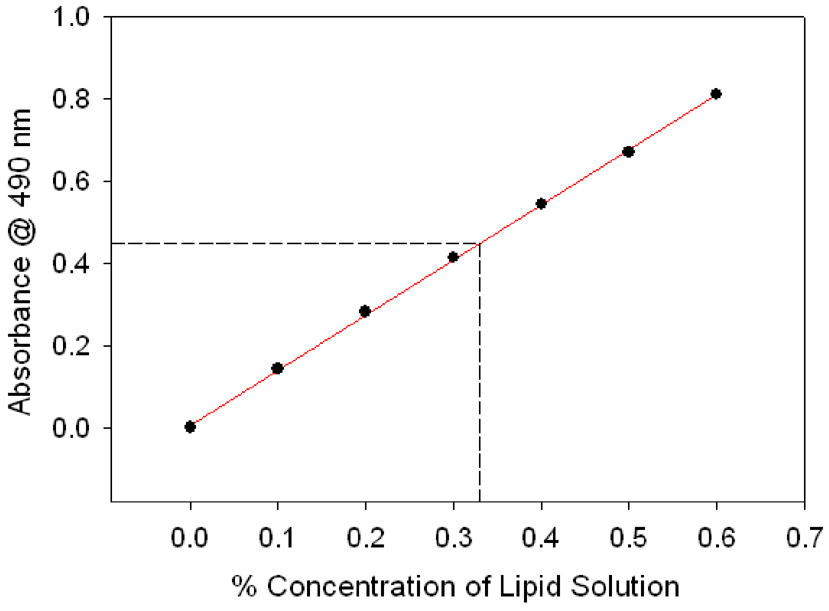


Figure 6.20: Absorbance measured as a function of percentage concentration of lipid solution used to make the bilayers. The lipid solution was mixed with a colourless aqueous solution of 0.4 M KCl and 25 mM Trizma buffer in 1 cm wide cuvettes. The aqueous solution of 0.4 M KCl and 25 mM Trizma buffer was used to produce the reference spectrum.

The average absorbance measured in the bilayers was found to be 0.45; and the equivalent absorbance in Figure 6.18 gives a 0.33% concentration of the lipid solution. The path length of light used to produce the absorbance values in Figure 6.18 was 1 cm. Therefore, using the Beer-Lambert Law:

$$A = \varepsilon c_1 l_1 = \varepsilon c_2 l_2 \qquad c_2 = 0.0033 \times c_1$$

$$l_1 = \frac{c_2 l_2}{c_1} = 0.0033 \times 1 = 0.0033\ \text{cm}$$

The path length that the light must pass through in the microwells is equal to 33.1 μm. Since the light passes through the bilayer twice for each absorbance measurement, the average bilayer must have an average height of 16.55 μm.

6.7 Conclusions

The single fibre reflectance probe is capable of resolving details that would not be otherwise be possible with a bifurcated probe that uses 200 µm diameter fibres. This ability has come at a cost of higher integration times in attempt to reduce noise, which means less data can be acquired over the course of an experiment.

However, the single fibre probe has been demonstrated to be able to map the diffusion field of redox product being produced inside a microwell electrode, locate nucleation sites, and identify the height of lipid bilayers sat inside a microwell. These results would not have been achieved through the use of the bifurcated reflectance probe.

Chapter 7: Conclusions

7.1 General Conclusions

The work presented here has demonstrated how spectral measurements can be quantitatively related to electrochemical measurements over longer time periods by recognising that edge effects reduce the absorbance measured over the centre of the electrode.

A novel method of imaging the diffusion field over time has been developed using readily available bifurcated reflectance probes. This technique allows for a greater understanding of how the behaviour of the electrochemical cell is affected by convection. However, investigations have shown that it would be difficult to use the data gathered in this manner quantitatively.

Finally, a fibre-optic splitter has been adapted for use in UV-Vis reflectance spectroscopy in microwells. The probe showed an excellent ability to investigate spectral changes on electrodes with diameters close to 50 μm. The probe was also able to pick out absorbance changes occurring in minute regions on macroelectrodes; something not achievable with the bifurcated probe.

7.2 Further Work

Unfortunately, this work has not been able to investigate any electrochemistry occurring inside the microwell while lipid bilayers are present. The bilayers were found to be incredibly sensitive to electrostatic charges and frequently collapsed when the electrode was connected to a potentiostat. There was difficulty in eliminating noise from current readings. There were also problems with inserting nanopores into the bilayers.

Another avenue of investigation would be to use the WSWE electrode in electrochemical experiments that utilise a form of controlled convection within the cell (e.g. flow cell). It may be possible to use this approach to quantitatively assess how redox products diffuse under different convection regimes.

Appendix: Visual Basic Codes

A1: Potential Sweep and Absorbance Acquisition

```
---form1---

Private Declare Function SendMessageStr Lib "user32" Alias "SendMessageA" (ByVal hwnd As
Long, ByVal wMsg As Long, ByVal wParam As Long, ByVal lParam As String) As Long


Dim i(5000)
'Dim filename As String
'Dim filesave As String
Dim datagraph(2)
Dim ep



Dim E(5000)
Dim Apredict(5000)
Dim lam(999)
Dim lamav(99)
Dim Aav(99)
Dim picLamda(1000)
Dim esq(1000)

Dim A(999)
Dim am(999, 1000)
Dim amav(99, 1000)
Dim amavback(99, 1000)
Dim amavsub(99, 1000)


Private Sub Command1_Click()
CWGraph1.ClearData
CWGraph3.ClearData
CWGraph1.ChartLength = 10000
CWGraph2.ClearData
'CWGraph3D1.ClearData
CWGraph3D2.ClearData
Erase E
Erase i
Erase lam
Erase lamav
Erase am
Erase amav


wey = 1
X = 0

For N = Text7 To Text4
Min = 0
Max = 0
pic = 0

    Filename = Text3.Text + Str(N) + ".dat"

    Open Filename For Input As #2

    Input #2, Er$
    Input #2, ir$
    E(N) = Val(Er$)

    If E(N) > 0 Then
    esq(N) = (E(N)) ^ 0.5
    Else
    End If

    i(N) = Val(ir$)



    Do
    Input #2, wave$
    Input #2, Al$



    lam(X) = Val(wave$) 'assign to array



    If lam(X) < Text1 Or lam(X) > Text2 Then
    A(X) = 0
    Else
    A(X) = Val(Al$) 'assign to array
    End If


    If lam(X) > Text6 - 3 And lam(X) < Text6 + 3 Then
    picLamda(N) = picLamda(N) + A(X)
    pic = pic + 1
    Else
    End If




    am(X, N) = A(X)
    X = X + 1
    Loop Until EOF(2)

    Close #2
picLamda(N) = picLamda(N) / pic


'now average over 10 points as splot cannot take 1000

w = 0
For Xa = 0 To 999 Step 10

For ww = 0 To 9
amav(w, N) = amav(w, N) + am(Xa + ww, N)
lamav(w) = lam(Xa + 5)
Next ww

amav(w, N) = amav(w, N) / 10
Aav(w) = amav(w, N)
w = w + 1
Next Xa


'now data loaded we analyse

X = 0
```

```
CWGraph1.ChartXvsY E(N), i(N)

CWGraph2.Axes(1).Minimum = CWSlide1.Pointers.Item(1).Value
CWGraph2.Axes(1).Maximum = CWSlide1.Pointers.Item(2).Value

CWGraph2.PlotXvsY lamav, Aav

Next N
terminate:


CWGraph3D2.Plot3DSurface lamav, E, amav




End Sub

Private Sub Command2_Click()

Dim r As Long
Dim PathArray() As String
Dim NewPathName As String
Dim FileFound As Integer
Dim NoOfFiles As Integer
Dim ListLine As String

PathArray = Split(Text3.Text, "\", -1, vbTextCompare)

For j = 1 To UBound(PathArray)
NewPathName = NewPathName + PathArray(j - 1) + "\"
Next j

NewPathName = NewPathName + "*.dat"

r = SendMessageStr(List1.hwnd, &H18D, &H20, NewPathName)

For k = 0 To (List1.ListCount - 1)
    ListLine = List1.List(k)
    FileFound = InStr(1, List1.List(k), PathArray(UBound(PathArray)), vbTextCompare)
    If FileFound > 0 Then
    NoOfFiles = NoOfFiles + 1
    End If
Next k

Text4.Text = CStr(NoOfFiles)
NoOfFiles = 0
List1.Clear
wey = 0
End Sub

Private Sub Command3_Click()
commondialog1.Flags = &H2&
commondialog1.DialogTitle = "Background data"
commondialog1.Filename = "*.dav"
commondialog1.InitDir = "C:\my documents\katie\data analysis\"
commondialog1.Action = 1
Open commondialog1.Filename For Input As #2

'Open FileName For Input As #2

    For xs = 0 To 98
    For xr = Text7 To Text4 - 1
        Input #2, back$
        amavback(xs, xr) = Val(back$)
    Next xr
    Input #2, back$
    amavback(99, Text4) = Val(back$)
    Next xs
Close #2


    For xs = 0 To 98
    For xr = Text7 To Text4 - 1
        amavsub(xs, xr) = amav(xs, xr) - amavback(xs, xr)
    Next xr
    amavsub(99, Text4) = amav(99, Text4) - amavback(99, Text4)
    Next xs


filesave = Text5.Text + ".sub"
    Open filesave For Output As #2

    For xs = 0 To 98
    For xr = Text7 To Text4 - 1

        Print #2, amavsub(xs, xr); ",";
    Next xr
    Print #2, amavsub(99, Text4)
    Next xs

End Sub

Private Sub Command4_Click()
Dim MaxE
Dim MinE
CWGraph3.ClearData

'Find MaxE and minE
MaxE = 0
MinE = 0
For N = Text7 To Text4
If E(N) > MaxE Then
MaxE = E(N)
End If
If E(N) < MinE Then
MinE = E(N)
End If
Next N


Q = 0
For N = Text7 To Text4
F = 96485
sr = Text8 / 1000 ' sweep rate
ep = Text9 * 1000 'epsilon
ne = Text10 ' number of electrons
area = (22 / 7) * ((Text11) ^ 2)
'timestep = Abs((E(N) - E(N - 1)) / sr)
'timestep = Text4 / ((2 * (MaxE - MinE)) / sr)
'dq = timestep * ((i(N) + i(N - 1)) / 2)
'Q = Q + dq 'sum charge
dQ = i(N) + i(N - 1)
Q = Q + dQ
Q2 = (Q * (MaxE - MinE)) / (Text4 * sr)



inter = Q2 / (ne * F * area)
Apredict(N) = 2 * ep * inter
```

```
 datagraph(0) = E(N)
 datagraph(1) = picLamda(N) + Text12
 datagraph(2) = Apredict(N)
 CWGraph3.ChartXY datagraph, bPlotPerRow = True



Next N



End Sub

Private Sub CWSlide1_PointerValueChanged(ByVal Pointer As Long, Value As Variant)
 Text1 = CWSlide1.Pointers.Item(1).Value
Text2 = CWSlide1.Pointers.Item(2).Value
 Text6 = CWSlide1.Pointers.Item(3).Value
End Sub

Private Sub Form_Load()
'Text3 = Form4.Text2
End Sub

Private Sub Save_Click()
'
    filesave = Text5.Text + ".dat"
    Open filesave For Output As #2

    For xs = 0 To 989 Step 10
    For xr = Text7 To Text4 - 1
        Print #2, am(xs, xr); ",";
    Next xr
    Print #2, am(999, Text4)
    Next xs


    Close #2

   filesave = Text5.Text + ".dav"
    Open filesave For Output As #2

    For xs = 0 To 98
    For xr = Text7 To Text4 - 1
        Print #2, amav(xs, xr); ",";
    Next xr
    Print #2, amav(99, Text4)
    Next xs


    Close #2



    filesave = Text5.Text + ".xyd"
    Open filesave For Output As #1

    For xs = Text7 To Text4
        Print #1, E(xs)
    Next xs

    For xs = 0 To 999 Step 10
        Print #1, Format$(lam(xs), "###.#")
    Next xs
    Close #1
    filesave = Text5.Text + ".pic"

    Open filesave For Output As #1

    For xs = Text7 To Text4
        Print #1, E(xs); ","; Format$(picLamda(xs), "#.####")
    Next xs

    Close #1







    filesave = Text5.Text + ".xya"
    Open filesave For Output As #1
    'Text4 = N
    For xs = Text7 To Text4
        Print #1, E(xs)
    Next xs

    For xs = 0 To 99
        Print #1, Format$(lamav(xs), "###.#")
    Next xs
    Close #1

    filesave = Text5.Text + ".cvd" ' current potential and absorbance
    Open filesave For Output As #1

    Print #1, Text6
    Print #1, ep / 1000

    For xs = Text7 To Text4
        Print #1, E(xs); ","; Format$(i(xs), "#.##e-##"); ","; Format$(picLamda(xs),
"#.####"); ","; Format$(Apredict(xs), "#.####")
    Next xs


    Close #1



End Sub

---form2---

Dim i(5000)
'Dim filename As String
'Dim filesave As String
Dim datagraph(2)



Dim E(5000)
Dim lam(999)
Dim lamav(99)
Dim picLamda(1000)
Dim esq(1000)

Dim A(999)
Dim am(999, 1000)
Dim amav(99, 1000)
Dim amavback(99, 1000)
Dim amavsub(99, 1000)


Private Sub Command1_Click()
CWGraph1.ClearData
CWGraph3.ClearData
CWGraph1.ChartLength = 10000
CWGraph2.ClearData
'CWGraph3D1.ClearData
CWGraph3D2.ClearData
Erase E
Erase i
Erase lam
Erase lamav
Erase am
Erase amav


wey = 1
X = 0

For N = Text7 To Text4
Min = 0
Max = 0
pic = 0
'Text7 = n

'Text3 = Form4.Text2

    Filename = Text3.Text + Str(N) + ".dat"

    Open Filename For Input As #2

    Input #2, Er$
    Input #2, ir$

    E(N) = Val(Er$)

    If E(N) > 0 Then
    esq(N) = (E(N)) ^ 0.5
    Else
    End If

    i(N) = Val(ir$)



    Do
    Input #2, wave$
    Input #2, Al$



    lam(X) = Val(wave$) 'assign to array

    If lam(X) < Text1 Or lam(X) > Text2 Then
    A(X) = 0
    Else
    A(X) = Val(Al$) 'assign to array
    End If


    If lam(X) > Text6 - 3 And lam(X) < Text6 + 3 Then
    picLamda(N) = picLamda(N) + A(X)
    pic = pic + 1
    Else
    End If




    am(X, N) = A(X)
    X = X + 1
    Loop Until EOF(2)

    Close #2
picLamda(N) = picLamda(N) / pic


'now average over 10 points as splot cannot take 1000

w = 0
For Xa = 0 To 999 Step 10

For ww = 0 To 9
amav(w, N) = amav(w, N) + am(Xa + ww, N)
lamav(w) = lam(Xa + 5)
Next ww

amav(w, N) = amav(w, N) / 10


w = w + 1
Next Xa




'now data loaded we analyse

X = 0
CWGraph1.ChartXvsY E(N), i(N)
CWGraph2.PlotXvsY lam, A
CWGraph3.ChartXvsY E(N), picLamda(N)


Next N
terminate:



' Plot surface
    'CWGraph3D1.Plot3DSurface lam, E, am
    CWGraph3D2.Plot3DSurface lamav, E, amav




End Sub

Private Sub Command2_Click()
wey = 0
End Sub

Private Sub Command3_Click()
commondialog1.Flags = &H2&
commondialog1.DialogTitle = "Background data"
commondialog1.Filename = "*.dav"
commondialog1.InitDir = "C:\my documents\katie\data analysis\"
commondialog1.Action = 1
Open commondialog1.Filename For Input As #2
```

```
'Open FileName For Input As #2

    For xs = 0 To 98
    For xr = Text7 To Text4 - 1
        Input #2, back$
        amavback(xs, xr) = Val(back$)
    Next xr
    Input #2, back$
    amavback(99, Text4) = Val(back$)
    Next xs
Close #2


    For xs = 0 To 98
    For xr = Text7 To Text4 - 1
        amavsub(xs, xr) = amav(xs, xr) - amavback(xs, xr)
    Next xr
    amavsub(99, Text4) = amav(99, Text4) - amavback(99, Text4)
    Next xs


filesave = Text5.Text + ".sub"
    Open filesave For Output As #2

    For xs = 0 To 98
    For xr = Text7 To Text4 - 1

        Print #2, amavsub(xs, xr); ",";
    Next xr
    Print #2, amavsub(99, Text4)
    Next xs






End Sub

Private Sub Command4_Click()
For N = Text7 To Text4
'CWGraph4.ChartXvsY (E(N)) ^ 0.5, picLamda(N)
esq(N) = E(N) ^ 0.5

CWStat1.LinFit esq, picLamda, z, slope, Intercept, mse
Next N

For N = Text7 To Text4
 datagraph(0) = esq(N)
 datagraph(1) = picLamda(N)
 datagraph(2) = z(N)
 CWGraph4.ChartXY datagraph, bPlotPerRow = True
Next N
c = Text9 / 1000
ep = Text10 * 1000

DR = (slope * ((22 / 7) ^ 0.5) / (4 * ep * c)) ^ 2

Text8 = Format(DR, "#.##e-#")




End Sub
```

```
Private Sub Form_Load()
Text3 = Form4.Text2
End Sub

Private Sub Save_Click()
'

'CommonDialog1.Flags = &H2&
    'CommonDialog1.DialogTitle = "Data Save"
    'CommonDialog1.FileName = "*.dat"
    'CommonDialog1.InitDir = "C:\my documents\katie\data analysis\"
    'commondialog1.Action = 2

    filesave = Text5.Text + ".dat"
    Open filesave For Output As #2

    For xs = 0 To 989 Step 10
    For xr = Text7 To Text4 - 1
        Print #2, am(xs, xr); ",";
    Next xr
    Print #2, am(999, Text4)
    Next xs


    Close #2


  'CommonDialog1.Flags = &H2&
    'CommonDialog1.DialogTitle = "Data Average Save"
    'CommonDialog1.FileName = "*.dav"
    'CommonDialog1.InitDir = "C:\my documents\katie\data analysis\"
    'CommonDialog1.Action = 2
    'Open CommonDialog1.FileName For Output As #2

   filesave = Text5.Text + ".dav"
    Open filesave For Output As #2

    For xs = 0 To 98
    For xr = Text7 To Text4 - 1
        Print #2, amav(xs, xr); ",";
    Next xr
    Print #2, amav(99, Text4)
    Next xs


    Close #2



    'CommonDialog1.Flags = &H2&
    'CommonDialog1.DialogTitle = "XY Save"
    'CommonDialog1.FileName = "*.xyd"
    'CommonDialog1.InitDir = "C:\my documents\katie\data analysis\"
    'CommonDialog1.Action = 2

    filesave = Text5.Text + ".xyd"
    Open filesave For Output As #1

    For xs = Text7 To Text4
        Print #1, E(xs)
    Next xs

    For xs = 0 To 999 Step 10
        Print #1, Format$(lam(xs), "###.#")
    Next xs
    Close #1


    'CommonDialog1.Flags = &H2&
    'CommonDialog1.DialogTitle = "XY average Save"
    'CommonDialog1.FileName = "*.xya"
    'CommonDialog1.InitDir = "C:\my documents\katie\data analysis\"
    'CommonDialog1.Action = 2
    filesave = Text5.Text + ".xya"
    Open filesave For Output As #1
    'Text4 = N
    For xs = Text7 To Text4
        Print #1, E(xs)
    Next xs

    For xs = 0 To 99
        Print #1, Format$(lamav(xs), "###.#")
    Next xs
    Close #1



    'CommonDialog1.Flags = &H2&
    'CommonDialog1.DialogTitle = "CV Save"
    'CommonDialog1.FileName = "*.cvd"
    'CommonDialog1.InitDir = "C:\my documents\katie\data analysis\"
    'CommonDialog1.Action = 2
    filesave = Text5.Text + ".cvd"
    Open filesave For Output As #1

    For xs = Text7 To Text4
        Print #1, E(xs); ","; i(xs)
    Next xs


    Close #1

End Sub

---form4---




Dim counter
Dim time(20000)
'Dim Data(20000)
Dim Datax(20000)
Dim DataY(20000)
Dim xtick
Dim data
Dim wey
Dim ymax(20000)
Dim ymin(20000)
Dim xmax(20000)
Dim xmin(20000)
Dim xma
Dim xmi
Dim yma
Dim ymi
Dim gai
Dim sweep
'Const NumPoints& = 200      ' Number of data points to collect
Const FirstPoint& = 0        ' set first element in buffer to transfer to array

Dim StopAq
Dim countersave

Dim dummy%
Dim LowChan%
Dim CBCount&
Dim CBRate&
Dim Options
Dim Gain
Dim TrigType%
Dim TrigCurrent!
Dim TrigVoltage!
Dim Range%
Dim TrigValue%
Dim Ulstat%
Dim N
Dim np
Dim Over
Dim noop
Dim avnum
Dim instname As ViSession
Dim devicestatusbyte As ViUInt8
Dim statuscode As ViStatus
Dim Filename
Dim wavelengthdata(2999) As ViReal64
Dim dataref(2999) As ViReal64
Dim databac(2999) As ViReal64
Dim dataA(2999)
Dim datab(2999)
Dim wavelengthdatabin(999)
Dim dataAbin(999)
Dim dataSBin(999)
Dim dataspec(2999) As ViReal64
Dim norev As Integer




Private Sub Add_Click()

    'determine if max number of datasets (5) is open noop = number open
    If noop = 5 Then
        MsgBox "Max No. of Datasets Open", vbExclamation + vbOKOnly, "Doh!!"
        Exit Sub
    End If

    'configure dialog box and form caption
    Form4.filebox.CancelError = True
    On Error GoTo ErrHandle
    Form4.filebox.Filename = "*.dat"
    Form4.filebox.InitDir = "C:\doug\data"
    Form4.filebox.Action = 1
    Form4.Caption = Form4.filebox.Filename

    'open file and input into variables
    Open Form4.filebox.Filename For Input As #4

    No = -1

    Do
        No = No + 1
        Input #4, E$
        Datax(No) = Val(E$)
        Input #4, i$
        DataY(No) = Val(i$)
    Loop Until EOF(4)
    Close #4

    noop = noop + 1

    CWGraphXY.PlotTemplate.MultiPlot = True
    CWGraphXY.Plots.Add

    'plot graph. norev is used if a previously added plot has been removed
    For nop = 0 To No
        If norev <> 0 Then CWGraphXY.Plots.Item(norev).ChartXvsY Datax(nop), DataY(nop)
        If norev = 0 Then CWGraphXY.Plots.Item(noop).ChartXvsY Datax(nop), DataY(nop)
    Next

    norev = 0

    'determine max and min values of data and display
    N = 0
    yma = DataY(0)
    ymi = DataY(0)
    xma = Datax(0)
    xmi = Datax(0)

    Do
        N = N + 1
        If DataY(N) > yma Then yma = DataY(N)
        If DataY(N) < ymi Then ymi = DataY(N)
        If Datax(N) > xma Then xma = Datax(N)
        If Datax(N) < xmi Then xmi = Datax(N)
    Loop Until N = No

    lblYmax.Caption = Format$(yma, "0.00E-00")
    lblYmin.Caption = Format$(ymi, "0.00E-00")
    lblXmax.Caption = Format$(xma, "0.000")
    lblXmin.Caption = Format$(xmi, "0.000")

ErrHandle:
End Sub

Private Sub AquireXY_Click()

    cmdStartXY_Click

End Sub



Private Sub AquireYt_Click()

    Form4.Visible = False
    form2.Visible = True
    'form2.CWButton1.Value = False

End Sub

Private Sub Blue_Click()

    CWGraphXY.Plots.Item(2).ClearData
    CWGraphXY.Plots.Item(2).ClearData
    noop = noop - 1

    If norev = 0 Or norev > 2 Then norev = 2



End Sub
```

```
Private Sub cmdStartXY_Click()
wey = 0
nc = 0
Filename = Text2.Text + "CV.dat"
Close #3
    Open Filename For Output As #3
'get the spectrometer setup

Test1 = SPX_init(ByVal "USB0::0x01313::0x0111::S/N M00231225::RAW", ByVal 2000,
instname)
Text6 = instname
inttime = CWKnob1.Value / 1000 ' sets integration time for the device
Test2% = SPX_setIntTime(ByVal instname, ByVal inttime)
test4% = SPX_getIntTime(ByVal instname, integrationtime) 'gets the actual value
Text4 = integrationtime
wave = SPX_getWavelengthData(ByVal instname, wavelengthdata(0), mini, Max)

'set system to continuous
Test3% = SPX_startScanCont(ByVal instname)



Dim TotChan0
Dim TotChan1
Dim X
Dim N
Dim LowChan%
Dim AveChan0
Dim AveChan1

    StopAq = 0
    gai = 1000000#
    sweep = 50
    noop = o
'    Shape1.Visible = False
'    Label2.Visible = False
'    Picture1.Visible = False
    Form4.Caption = "XY Program"




    For N = 0 To 20000
        DataY(N) = Empty
        Datax(N) = Empty
    Next N

    CWGraphXY.ClearData

    gai = InputBox("Enter Gain", "Enter Gain", gai)

    If IsNumeric(gai) Then
        sweep = InputBox("Enter Sweep Rate (mV/s)", "Sweep Rate", sweep)

        If IsNumeric(sweep) Then
            File.Enabled = False
            Aquire.Enabled = False
            CWButton2.Enabled = False
            cmdStartXY.Visible = False
            cmdStopXY.Visible = True
            lblAquiring.Visible = True
        Else
            MsgBox "Invalid Sweep Rate. Please Try Again", vbCritical, "Doh!!!"
        Exit Sub
```

```
            End If

        Else
            MsgBox "Invalid Gain Setting. Please Try Again", vbCritical, "Doh!!!"
            Exit Sub
        End If

       NumPoints = 2000 / 20
       N = 0
       countersave = 0



       Do
        N = N + 1

repeat:
        dummy% = DoEvents()

        If StopAq = 1 Then

            Exit Sub

        End If


        CWAIPoint1.SingleRead data
        vstart = data(0)

        Do ' sets up the wait state
        vloop = 0

        For wa = 1 To (NumPoints / 5)
        CWAIPoint1.SingleRead data
        vloop = vloop + data(0)
        Erase data
        Next wa
        vloop = vloop / (NumPoints / 5)
        DoEvents
        If wey = 1 Then Exit Sub

        Loop Until Abs(vstart - vloop) > 0.005
breakE:

       TotChan0 = 0
       TotChan1 = 0

        For w = 1 To NumPoints
        CWAIPoint1.SingleRead data
        TotChan0 = TotChan0 + data(0)
        TotChan1 = TotChan1 + data(1)
        Erase data
        Next w


        AveChan0 = -TotChan0 / (NumPoints)
        AveChan1 = TotChan1 / (NumPoints)



        If Check1.Value = 1 Then
        AveChan0 = AveChan0 * -1
        Else
        End If
```

```
    If Check2.Value = 1 Then
    AveChan1 = AveChan1 * -1
    Else
    End If



    Datax(N) = AveChan0
    DataY(N) = AveChan1 / gai





'avnum = CWSlide1.Value
Nsp = 0
Dim datasp(2999)
Erase datasp
Erase dataA

Do
test4 = SPX_getScanData(ByVal instname, dataspec(0))

For av = 0 To 2998
datasp(av) = datasp(av) + dataspec(av)
Next av
Nsp = Nsp + 1
Loop Until Nsp = avnum

'subtract background
For av = 0 To 2998
dataspec(av) = (datasp(av) / avnum) - datab(av)
If (dataspec(av) / dataref(av)) <= 0 Then
dataA(av) = 5
Else
dataA(av) = 2 - (Log(((dataspec(av) / dataref(av))) * 100) / 2.301)
End If
Next av



bin = 0
For av = 1 To 2999 Step 3
dataAbin(bin) = (dataA(av - 1) + dataA(av) + dataA(av + 1)) / 3
dataSBin(bin) = (datasp(av - 1) + datasp(av) + datasp(av + 1)) / (3 * avnum)
wavelengthdatabin(bin) = (wavelengthdata(av - 1) + wavelengthdata(av) +
wavelengthdata(av + 1)) / 3

'Set pointer 1 to be the active pointer
Set CWSlide2.ActivePointer = CWSlide2.Pointers.Item(1)

If wavelengthdatabin(bin) < CWSlide2.Value Then
dataAbin(bin) = 0
Else
End If

'Set pointer 2 to be the active pointer
Set CWSlide2.ActivePointer = CWSlide2.Pointers.Item(2)

If wavelengthdatabin(bin) > CWSlide2.Value Then
dataAbin(bin) = 0
Else
End If
```

```
bin = bin + 1
Next av



CWGraph2.PlotXvsY wavelengthdatabin, dataAbin
CWGraphXY.ChartXvsY Datax(N), DataY(N)
'nc = nc + 1

    Filename = Text2.Text + Str(N) + ".dat"
    Open Filename For Output As #2
    Print #2, Format(Datax(N), "0.0000"); ","; Format(DataY(N), "0.00e-00")
    For nsave = 0 To 999
    Print #2, Format(wavelengthdatabin(nsave), "000.0"); ","; Format(dataAbin(nsave),
"0.0000"); ","; Format(dataSBin(nsave), "0.0000")
    Next nsave
    Close #2


    Print #3, Format(Datax(N), "0.0000"); ","; Format(DataY(N), "0.00e-00")


    Text1 = N

    lblShowVolts.Caption = Format$(Datax(N), "0.000")
    lblShowCurrent.Caption = Format(DataY(N), "0.00E-00")
    countersave = countersave + 1


    Loop Until N = 20000
Close #3

End Sub

Private Sub cmdQuit_Click()
Dim yesno

yesno = MsgBox("Are you sure you want to quit?", vbQuestion + vbYesNo, "Quit")

If yesno = 6 Then End


End Sub

Private Sub cmdStopXY_Click()
noop = 1
StopAq = 1
    tmrXY.Enabled = False
    cmdStartXY.Visible = True
    lblAquiring.Visible = False
    File.Enabled = True
    Aquire.Enabled = True
    CWButton2.Enabled = True


    N = 0
    yma = DataY(1)
    ymi = DataY(1)
    xma = Datax(1)
    xmi = Datax(1)

    Do
        N = N + 1
        If DataY(N) > yma Then yma = DataY(N)
        If DataY(N) < ymi Then ymi = DataY(N)
        If Datax(N) > xma Then xma = Datax(N)
        If Datax(N) < xmi Then xmi = Datax(N)
    Loop Until N = countersave

    lblYmax.Caption = Format$(yma, "0.00E-00")
    lblYmin.Caption = Format$(ymi, "0.00E-00")

    lblXmax.Caption = Format$(xma, "0.000")
    lblXmin.Caption = Format$(xmi, "0.000")
    Test5% = SPX_close(ByVal instname)
 CWVisa1.Close  'actually stops crashing for the second time.
End Sub

Private Sub ExitXY_Click()

    Form4.Visible = False
    ' Form1.Visible = True

End Sub




Private Sub Command1_Click()

'this gets a dark background for correction

Test1 = SPX_init(ByVal "USB0::0x01313::0x0111::S/N M00231225::RAW", ByVal 2000,
instname)
Text6 = instname
inttime = CWKnob1.Value / 1000 ' sets integration time for the device
Test2% = SPX_setIntTime(ByVal instname, ByVal inttime)
test4% = SPX_getIntTime(ByVal instname, integrationtime) 'gets the actual value
Text4 = integrationtime
wave = SPX_getWavelengthData(ByVal instname, wavelengthdata(0), mini, Max)

'set system to continuous
Test3% = SPX_startScanCont(ByVal instname)

'get the data
avnum = CWSlide1.Value
Nbac = 0
Erase datab
Do
test4 = SPX_getScanData(ByVal instname, databac(0))
For av = 0 To 2999
datab(av) = datab(av) + databac(av)
Next av
Nbac = Nbac + 1
Loop Until Nbac = avnum

For av = 0 To 2999
datab(av) = datab(av) / avnum
Next av

Test5% = SPX_close(ByVal instname)
Text5 = Test5%

'save background data
filesavebac = Text2.Text + ".bac"
Open filesavebac For Output As #1

For av = 0 To 2999
Print #1, Format(wavelengthdata(av), "000.0"); ","; Format(datab(av), "0.00000")
```

```
Next av
Close #1

'save info

filesaveinf = Text2.Text + ".inf"
Open filesaveinf For Output As #1

Print #1, Date
Print #1, "Number of averaged spectra = "; avnum
Print #1, "Integration time/s = "; inttime


Close #1



CWVisa1.Close  'actually stops crashing for the second time.
Command2.Visible = True
End Sub

Private Sub Command2_Click()

Test1 = SPX_init(ByVal "USB0::0x01313::0x0111::S/N M00231225::RAW", ByVal 2000,
instname)
Text6 = instname
inttime = CWKnob1.Value / 1000 ' sets integration time for the device
Test2% = SPX_setIntTime(ByVal instname, ByVal inttime)
test4% = SPX_getIntTime(ByVal instname, integrationtime) 'gets the actual value
Text4 = integrationtime
wave = SPX_getWavelengthData(ByVal instname, wavelengthdata(0), mini, Max)

'set system to continuous
Test3% = SPX_startScanCont(ByVal instname)

'get the data
avnum = CWSlide1.Value
Nsp = 0

Dim datar(2999)
Erase datar

Do
test4 = SPX_getScanData(ByVal instname, dataref(0))
For av = 0 To 2998
datar(av) = datar(av) + dataref(av)
Next av
Nsp = Nsp + 1
Loop Until Nsp = avnum

For av = 0 To 2998
dataref(av) = (datar(av) / avnum) - datab(av)
If dataref(av) = 0 Then
dataref(av) = 0.0000000001
Else
End If

Next av

'save reference data
filesaveref = Text2.Text + ".ref"
Open filesaveref For Output As #1

For av = 0 To 2999
Print #1, Format(wavelengthdata(av), "000.0"); ","; Format(dataref(av), "0.00000")
Next av
```

```
Close #1



'CWGraph1.Caption = refence
CWGraph1.PlotXvsY wavelengthdata, dataref
Test5% = SPX_close(ByVal instname)
CWVisa1.Close  'actually stops crashing for the second time.
'leave running for later
Command2.Visible = False
cmdStartXY.Visible = True
End Sub

Private Sub Command3_Click()
resetit = SPX_reset(ByVal instname)
CWVisa1.Close
form1.Show

End Sub

Private Sub CWButton2_ValueChanged(ByVal Value As Boolean)

    Shape1.Visible = False
    Label2.Visible = False
    Picture1.Visible = False
    CWButton2.Enabled = True
    cmdStartXY.Enabled = True

    If CWButton2.Value = True Then
        Form4.Visible = True
        Form3.Visible = False
    End If

    If CWButton2.Value = False Then
        Form4.Visible = False
        Form3.Visible = True
        Form3.CWButton1.Value = False
    End If

End Sub




Private Sub CWButton3_ValueChanged(ByVal Value As Boolean)
If CWButton3.Value = True Then wey = 1
If CWButton3.Value = False Then wey = 0
End Sub

Private Sub CWSlide2_PointerValueChanged(ByVal Pointer As Long, Value As Variant)
CWGraph2.Axes(1).Minimum = CWSlide2.Pointers.Item(1).Value
CWGraph2.Axes(1).Maximum = CWSlide2.Pointers.Item(2).Value

End Sub

Private Sub Cyan_Click()
CWGraphXY.Plots.Item(5).ClearData
CWGraphXY.Plots.Item(5).ClearData
noop = noop - 1
If norev = 0 Then norev = 5


End Sub
```

```
Private Sub force_Click()
Form4.Visible = False
form2.Visible = False

    Form3.Visible = True
End Sub

Private Sub Form_Load()
CWAIPoint1.Device = 1
ChannelString = "0,1"
CWAIPoint1.Channels.Add ChannelString
End Sub

Private Sub Green_Click()
CWGraphXY.Plots.Item(1).ClearData
CWGraphXY.Plots.Item(1).ClearData
noop = noop - 1
If norev = 0 Or norev > 1 Then norev = 1

End Sub

Private Sub Open_Click()

Dim No
Dim E$
Dim i$
Dim nop

    Shape1.Visible = False
    Label2.Visible = False
    Picture1.Visible = False

    Form4.filebox.CancelError = True
    On Error GoTo ErrHandle
    Form4.filebox.Filename = "*.dat"
    Form4.filebox.InitDir = "C:\doug\data"
    Form4.filebox.Action = 1
    Form4.Caption = Form4.filebox.Filename
    Open Form4.filebox.Filename For Input As #4

    No = -1

    Do
        No = No + 1
        Input #4, E$
        Datax(No) = Val(E$)
        Input #4, i$
        DataY(No) = Val(i$)
    Loop Until EOF(4)

    Close #4
    'If noop = 0 Then
    'noop = 1
    'GoTo FirstOpen
    'End If

   'Over = MsgBox("Overlay?", vbYesNo, "Open File")

    'If Over = 7 Then
'FirstOpen:
noop = 1
    CWGraphXY.ClearData
     For nop = 0 To No
        CWGraphXY.Plots.Item(noop).ChartXvsY Datax(nop), DataY(nop)
    Next
    N = 0
    yma = DataY(0)
    ymi = DataY(0)
    xma = Datax(0)
    xmi = Datax(0)

    Do
        N = N + 1
        If DataY(N) > yma Then yma = DataY(N)
        If DataY(N) < ymi Then ymi = DataY(N)
        If Datax(N) > xma Then xma = Datax(N)
        If Datax(N) < xmi Then xmi = Datax(N)
    Loop Until N = No

    lblYmax.Caption = Format$(yma, "0.00E-00")
    lblYmin.Caption = Format$(ymi, "0.00E-00")
    lblXmax.Caption = Format$(xma, "0.000")
    lblXmin.Caption = Format$(xmi, "0.000")

    norev = 0



ErrHandle:     Exit Sub
End Sub



Private Sub optI0_Click()

End Sub

Private Sub optI1_Click()

End Sub

Private Sub optI2_Click()

End Sub

Private Sub optI3_Click()

End Sub



Private Sub Red_Click()
CWGraphXY.Plots.Item(3).ClearData
CWGraphXY.Plots.Item(3).ClearData
noop = noop - 1
If norev = 0 Or norev > 3 Then norev = 3


End Sub



Private Sub RemoveAll_Click()
yn = MsgBox("Are you sure you want to remove all plots?", vbQuestion + vbYesNo, "Remove
All Plots")
If yn = 6 Then
noop = 0
norev = 0
CWGraphXY.Plots.Item(1).ClearData
CWGraphXY.Plots.Item(2).ClearData
CWGraphXY.Plots.Item(3).ClearData
CWGraphXY.Plots.Item(4).ClearData
CWGraphXY.Plots.Item(5).ClearData
End If
End Sub

Private Sub Save_Click()

Dim ns

    Form4.filebox.CancelError = True
    On Error GoTo ErrHandle
    Form4.filebox.Flags = &H2&
    Form4.filebox.Filename = "*.dat"
    Form4.filebox.InitDir = "C:\doug\data"
    Form4.filebox.Action = 2
    Form4.Caption = Form4.filebox.Filename
    Open Form4.filebox.Filename For Output As #2

    ns = 0

    Do
        ns = ns + 1
        Print #2, Datax(ns); ","; DataY(ns)
    Loop Until ns = countersave

    Close #2

ErrHandle:     Exit Sub
End Sub




Private Sub Yellow_Click()
CWGraphXY.Plots.Item(4).ClearData
CWGraphXY.Plots.Item(4).ClearData
noop = noop - 1
If norev = 0 Or norev > 4 Then norev = 4


End Sub

---module1---

' -------------------------------------------------------------------------
'
'       Thorlabs SPx-USB - VB Header file for SP1-USB / SP2-USB Spectrometer instr.
driver
'  Do not modify the contents of this file.
'       Copyright:      Copyright(c) 2007, Thorlabs GmbH (www.thorlabs.com)
'
' -------------------------------------------------------------------------
'
'       Date:           Aug-21-2007
'       Software-Nr:   09.167.200
'       Version:        2.0
'       Changelog:              see 'readme.rtf'
'
' -------------------------------------------------------------------------
'
'       Disclaimer:
'
'       This library is free software; you can redistribute it and/or
'       modify it under the terms of the GNU Lesser General Public
'       License as published by the Free Software Foundation; either
'       version 2.1 of the License, or (at your option) any later version.
'
'       This library is distributed in the hope that it will be useful,
'       but WITHOUT ANY WARRANTY; without even the implied warranty of
'       MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
'       Lesser General Public License for more details.
'
'       You should have received a copy of the GNU Lesser General Public
'       License along with this library; if not, write to the Free Software
'       Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
'
' -------------------------------------------------------------------------


' -------------------------------------------------------------------------
' Functions
' -------------------------------------------------------------------------

' - Init/Close ---
Declare Function SPX_init& Lib "SPX_Drv_32.dll" (ByVal x1$, ByVal x2&, x3&)
Declare Function SPX_close& Lib "SPX_Drv_32.dll" (ByVal x1&)



' - Configuration Functions ---
Declare Function SPX_setIntTime& Lib "SPX_Drv_32.dll" (ByVal x1&, ByVal x2#)
Declare Function SPX_getIntTime& Lib "SPX_Drv_32.dll" (ByVal x1&, x2#)


' - Action/Status Functions ---
Declare Function SPX_startScan& Lib "SPX_Drv_32.dll" (ByVal x1&)
Declare Function SPX_startScanExtTrg& Lib "SPX_Drv_32.dll" (ByVal x1&)
Declare Function SPX_startScanCont& Lib "SPX_Drv_32.dll" (ByVal x1&)
Declare Function SPX_startScanContExtTrg& Lib "SPX_Drv_32.dll" (ByVal x1&)
Declare Function SPX_getDeviceStatus& Lib "SPX_Drv_32.dll" (ByVal x1&, ByVal x2$)


' - Data Functions ---
Declare Function SPX_getScanData& Lib "SPX_Drv_32.dll" (ByVal x1&, x2#)
Declare Function SPX_getRawScanData& Lib "SPX_Drv_32.dll" (ByVal x1&, x2%)
Declare Function SPX_getWavelengthData& Lib "SPX_Drv_32.dll" (ByVal x1&, x2#, x3#, x4#)


' - Utility Functions ---
Declare Function SPX_reset& Lib "SPX_Drv_32.dll" (ByVal x1&)
Declare Function SPX_errorMessage& Lib "SPX_Drv_32.dll" (ByVal x1&, ByVal x2&, ByVal
x3$)
Declare Function SPX_identificationQuery& Lib "SPX_Drv_32.dll" (ByVal x1&, ByVal x2$,
ByVal x3$, ByVal x4$, ByVal x5$)
Declare Function SPX_revisionQuery& Lib "SPX_Drv_32.dll" (ByVal x1&, ByVal x2$, ByVal
x3$)
Declare Function SPX_setUserText& Lib "SPX_Drv_32.dll" (ByVal x1&, ByVal x2$)
Declare Function SPX_getUserText& Lib "SPX_Drv_32.dll" (ByVal x1&, ByVal x2$)


' -------------------------------------------------------------------------
' Definitions
' -------------------------------------------------------------------------

' - Driver Error Codes ---
Global Const VI_WARN_SPX_DATA_NOT_READY = &H3FFC0901
Global Const VI_ERROR_READ_INCOMPLETE = &HBFFC0901


' - Buffers ---
```

```
Global Const SPX_BUFFER_SIZE = 256                          ' General buffer size
Global Const SPX_ERR_DESCR_BUFFER_SIZE = 512                    ' Buffer size for error
messages
Global Const SPX_TEXT_BUFFER_SIZE = 256                                         ' Buffer
size for texts from the LC1
Global Const SPX_NUM_PIXELS = 3000                                              ' this
is the number of effective pixels of CCD
Global Const SPX_MAX_USER_NAME_SIZE = 31                                ' this is max
number of characters for a user name

' - VISA strings ---
Global Const SP1_VI_FIND_RSC_PATTERN = "USB?*?{VI_ATTR_MANF_ID==0x1313 &&
VI_ATTR_MODEL_CODE==0x0111}"
Global Const SP2_VI_FIND_RSC_PATTERN = "USB?*?{VI_ATTR_MANF_ID==0x1313 &&
VI_ATTR_MODEL_CODE==0x0112}"
Global Const SPX_VI_FIND_RSC_PATTERN = "USB?*?{VI_ATTR_MANF_ID==0x1313 &&
((VI_ATTR_MODEL_CODE==0x0111) || (VI_ATTR_MODEL_CODE==0x0112))}"


' - Communication timeout ---
Global Const SPX_TIMEOUT_MIN = 1000
Global Const SPX_TIMEOUT_MAX = 60000

' - LC1 specific constants ---
Global Const SPX_MAX_INT_TIME = 0.2
' 200ms is the maximum integration time
Global Const SPX_MIN_INT_TIME = 0.000001                                        ' 1us
is the minimum integration time

' - scan states - these are the states reported by LC1_getDeviceStatus ---
Global Const SPX_SCAN_STATE_IDLE = &H0&
' LC waits for new scan to execute
Global Const SPX_SCAN_STATE_TRANSFER = &H20&                            ' scan is done,
waiting for data transfer to be finished
Global Const SPX_SCAN_STATE_IN_PROGRES = &H40&                  ' scan is in progres,
this message should never be get
Global Const SPX_SCAN_STATE_WAIT_TRIGGER = &H80&                ' same as IDLE except
that external trigger is armed
Global Const SPX_SCAN_STATE_UNKNOWN = &HFF&                             ' default value


' ---------------------------------------------------------------------------
' End of file
' ---------------------------------------------------------------------------
```

A2: Potential Step and Absorbance Acquisition

```
---form1---

Dim i(5000)
'Dim filename As String
'Dim filesave As String
Dim datagraph(2)

Dim datagraphi(2)


Dim Aav(99)
Dim E(5000)
Dim lam(999)
Dim lamav(99)
Dim picLamda(1000)
Dim esq(1000)
Dim ct(1000)
Dim icot(1000)
Dim za As Variant
Dim zi As Variant
Dim A(999)
Dim am(999, 1000)
Dim amav(99, 1000)
Dim amavback(99, 1000)
Dim amavsub(99, 1000)


Private Sub Command1_Click()


CWGraph1.ClearData
CWGraph3.ChartLength = 500
CWGraph3.ClearData
CWGraph5.ChartLength = 500
CWGraph5.ClearData
CWGraph1.ChartLength = 10000
CWGraph2.ClearData
'CWGraph3D1.ClearData
CWGraph3D2.ClearData
Erase E
Erase i
Erase icot
Erase lam
Erase lamav
Erase am
Erase picLamda
Erase amav
'Erase zi
'Erase za



wey = 1
X = 0

For N = Text7 To Text4
Min = 0
Max = 0
pic = 0

'Text7 = n
Text5 = Text3
'Text3 = Form4.Text2

    FileName = Text3.Text + Str(N) + ".dat"

    Open FileName For Input As #2

    Input #2, Er$
    Input #2, ir$

    E(N) = Val(Er$) '+ 0.01


    If E(N) > 0 Then
    esq(N) = (E(N)) ^ 0.5
    Else
    End If

    i(N) = Val(ir$)
    Do
    Input #2, wave$
    Input #2, Al$



    lam(X) = Val(wave$) 'assign to array

    If lam(X) < Text1 Or lam(X) > Text2 Then
    A(X) = 0
    Else
    A(X) = Val(Al$) 'assign to array
    End If


    If lam(X) > Text6 - 1 And lam(X) < Text6 + 1 Then
    picLamda(N) = picLamda(N) + A(X)
    pic = pic + 1
    Else
    End If




    am(X, N) = A(X)
    X = X + 1
    Loop Until EOF(2)

    Close #2

picLamda(N) = picLamda(N) / pic


'now average over 10 points as splot cannot take 1000

w = 0
For Xa = 0 To 999 Step 10

For ww = 0 To 9
amav(w, N) = amav(w, N) + am(Xa + ww, N)
lamav(w) = lam(Xa + 5)
Next ww
amav(w, N) = amav(w, N) / 10
Aav(w) = amav(w, N)
w = w + 1
Next Xa




'now data loaded we analyse

X = 0
CWGraph1.ChartXvsY E(N), i(N)

CWGraph2.PlotXvsY lamav, Aav

CWGraph3.ChartXvsY E(N), picLamda(N)


Next N
terminate:
```

```
' Plot surface
    'CWGraph3D1.Plot3DSurface lam, E, am
    CWGraph3D2.Plot3DSurface lamav, E, amav
    CWGraph1.PlotXvsY E, i



End Sub

Private Sub Command2_Click()
wey = 0
End Sub

Private Sub Command3_Click()
commondialog1.Flags = &H2&
commondialog1.DialogTitle = "Background data"
commondialog1.FileName = "*.dav"
commondialog1.InitDir = "C:\my documents\katie\data analysis\"
commondialog1.Action = 1
Open commondialog1.FileName For Input As #2

'Open FileName For Input As #2

    For xs = 0 To 98
    For xr = Text7 To Text4 - 1
        Input #2, back$
        amavback(xs, xr) = Val(back$)
    Next xr
    Input #2, back$
    amavback(99, Text4) = Val(back$)
    Next xs
Close #2


    For xs = 0 To 98
    For xr = Text7 To Text4 - 1
        amavsub(xs, xr) = amav(xs, xr) - amavback(xs, xr)
    Next xr
    amavsub(99, Text4) = amav(99, Text4) - amavback(99, Text4)
    Next xs


filesave = Text5.Text + ".sub"
    Open filesave For Output As #2

    For xs = 0 To 98
    For xr = Text7 To Text4 - 1

        Print #2, amavsub(xs, xr); ",";
    Next xr
    Print #2, amavsub(99, Text4)
    Next xs






End Sub

Private Sub Command4_Click()


CWGraph4.ClearData
```

```
CWGraph4.ChartLength = 500
w = 0 '
For N = Text7 To Text4
esq(N) = E(N) ^ 0.5 ' square root of time

'If E(N) > 1 Then
w = w + 1
ct(N) = 1 / (esq(N))
icot(N) = i(N)
'Else
'End If

Next N

CWStat1.LinFit esq, picLamda, za, slopea, intercepta, msea  'regression for A


For N = Text7 To Text4
 datagraph(0) = esq(N)
 datagraph(1) = picLamda(N)
 datagraph(2) = za(N)
 CWGraph4.ChartXY datagraph, bPlotPerRow = True
Next N

CWStat2.LinFit ct, icot, zi, Slopei, intercepti, msei  'regression for i

For N = Text7 + 1 To Text4
 datagraphi(0) = ct(N)
 datagraphi(1) = icot(N)
 datagraphi(2) = zi(N)
 CWGraph5.ChartXY datagraphi, bPlotPerRow = True
Next N


area = (22 / 7) * ((Text11) ^ 2)

c = Text9 / 1000
ep = Text10 * 1000

DR = (slopea * ((22 / 7) ^ 0.5) / (4 * ep * c)) ^ 2

F = 96485
ne = 1

DRi = (Slopei * ((22 / 7) ^ 0.5) / (ne * F * area * c)) ^ 2

Text8 = Format(DR, "#.##e-#")
Text12 = Format(DRi, "#.##e-#")

'Screw LinFit function and do it yourself!
Dim AggAbs
Dim AggTime
Dim AggAbsxTime
Dim AggTimeSq
Dim AggNo
Dim SimpleSlope

For k = 1 To UBound(esq)
AggNo = AggNo + 1
AggAbs = AggAbs + picLamda(k)
AggTime = AggTime + esq(k)
AggAsbxTime = AggAbsxTime + (esq(k) * picLamda(k))
AggTimeSq = AggTimeSq + (esq(k) * esq(k))
Next k
```

```
SimpleSlope = ((AggNo * AggAbsxTime) - (AggAbs * AggTime)) / ((AggNo * AggTimeSq) -
(AggTime * AggTime))

DR = (SimpleSlope * 1.77245385 / (4 * ep * c)) ^ 2
'Text13 = Format(slopea, "#.##e-#")


Dim Aggicot
Dim Aggct
Dim Aggicotxct
Dim Aggctxct
Dim AggNo2
Dim SimpleSlope2

For k = Text7 To Text4
Aggicot = Aggicot + icot(k)
Aggct = Aggct + ct(k)
Aggicotxct = Aggicotxct + (ct(k) * icot(k))
Aggctxct = Aggctxct + (ct(k) * ct(k))
Next k

SimpleSlope2 = (((Text4 - Text7) * Aggicotxct) - (Aggicot * Aggct)) / (((Text4 - Text7)
* Aggctxct) - (Aggct * Aggct))



DRi = (SimpleSlope2 * 1.77245385 / (ne * F * area * c)) ^ 2

'Text14 = Format(DRi, "#.##e-#")
'Text14 = CStr(Aggicotxct) + " , " + CStr(Aggicot) + " , " + CStr(Aggct) + " , " +
CStr(Aggctxct)
'Text14 = ct(100)
'Text14 = DRi

Erase zi, za


End Sub

Private Sub CWSlide1_PointerValueChanged(ByVal Pointer As Long, Value As Variant)
Text1 = CWSlide1.Pointers.Item(1).Value
Text2 = CWSlide1.Pointers.Item(2).Value
Text6 = CWSlide1.Pointers.Item(3).Value
CWGraph2.Axes(1).Minimum = CWSlide1.Pointers.Item(1).Value
CWGraph2.Axes(1).Maximum = CWSlide1.Pointers.Item(2).Value


End Sub

Private Sub Form_Load()
'Text3 = Form3.Text2
End Sub

Private Sub Save_Click()
'

    filesave = Text5.Text + ".dat"
    Open filesave For Output As #2

    For xs = 0 To 989 Step 10
    For xr = Text7 To Text4 - 1
        Print #2, am(xs, xr); ",";
    Next xr
    Print #2, am(999, Text4)
    Next xs
    Close #2



   filesave = Text5.Text + ".dav"
    Open filesave For Output As #2

    For xs = 0 To 98
    For xr = Text7 To Text4 - 1
        Print #2, amav(xs, xr); ",";
    Next xr
    Print #2, amav(99, Text4)
    Next xs


    Close #2





    filesave = Text5.Text + ".xyd"
    Open filesave For Output As #1

    For xs = Text7 To Text4
        Print #1, E(xs)
    Next xs

    For xs = 0 To 999 Step 10
        Print #1, Format$(lam(xs), "###.#")
    Next xs
    Close #1


    filesave = Text5.Text + ".xya"
    Open filesave For Output As #1

    For xs = Text7 To Text4
        Print #1, E(xs)
    Next xs

    For xs = 0 To 99
        Print #1, Format$(lamav(xs), "###.#")
    Next xs
    Close #1




    filesave = Text5.Text + ".cvd"
    Open filesave For Output As #1

    For xs = Text7 To Text4
        Print #1, E(xs); ","; i(xs)
    Next xs


    Close #1

End Sub

---form3---
```

```
Const BoardNum% = 1          ' Board number
'Const PortNum% = FIRSTPORTB     ' use first digital port
'Const Direction% = DIGITALOUT   ' program digital port A for output
Dim counter
Dim time(20000)
'Dim Data(20000)
Dim Datax(20000)
Dim DataY(20000)
Dim xtick
Dim eng!

Dim ymax(20000)
Dim ymin(20000)
Dim xmax(20000)
Dim xmin(20000)
Dim xma
Dim xmi
Dim yma
Dim ymi
Dim gai
Dim sweep
'Const NumPoints& = 2000      ' Number of data points to collect
Const FirstPoint& = 0        ' set first element in buffer to transfer to array
'Dim ADData%(NumPoints&)      ' dimension an array to hold the input values
'Dim MemHandle&               ' define a variable to contain the handle for
                             ' memory allocated by Windows through cbWinBufAlloc%()
'Dim HighChan%
'Dim EngUnits(NumPoints)
Dim StopAq
Dim countersave

Dim dummy%
Dim LowChan%
Dim CBCount&
Dim CBRate&
Dim Options
Dim Gain
Dim TrigType%
Dim TrigCurrent!
Dim TrigVoltage!
Dim Range%
Dim TrigValue%
Dim Ulstat%
Dim N
Dim np
Dim Over
Dim noop
Dim avnum
Dim instname As ViSession
Dim devicestatusbyte As ViUInt8
Dim statuscode As ViStatus
Dim FileName
Dim wavelengthdata(2999) As ViReal64
Dim dataref(2999) As ViReal64
Dim databac(2999) As ViReal64
Dim dataA(2999)
Dim datab(2999)
Dim wavelengthdatabin(999)
Dim dataAbin(999)
Dim TriggerActivate As Boolean

Dim dataspec(2999) As ViReal64
Dim norev As Integer
```

```
Private Sub Add_Click()

    'determine if max number of datasets (5) is open noop = number open
    If noop = 5 Then
        MsgBox "Max No. of Datasets Open", vbExclamation + vbOKOnly, "Doh!!"
        Exit Sub
    End If

    'configure dialog box and form caption
    Form4.filebox.CancelError = True
    On Error GoTo ErrHandle
    Form4.filebox.FileName = "*.dat"
    Form4.filebox.InitDir = "C:\doug\data"
    Form4.filebox.Action = 1
    Form4.Caption = Form4.filebox.FileName

    'open file and input into variables
    Open Form4.filebox.FileName For Input As #4

    No = -1

    Do
        No = No + 1
        Input #4, E$
        Datax(No) = Val(E$)
        Input #4, i$
        DataY(No) = Val(i$)
    Loop Until EOF(4)

    Close #4

    noop = noop + 1

    CWGraphXY.PlotTemplate.MultiPlot = True
    CWGraphXY.Plots.Add

    'plot graph. norev is used if a previously added plot has been removed
    For nop = 0 To No
        If norev <> 0 Then CWGraphXY.Plots.Item(norev).ChartXvsY Datax(nop), DataY(nop)
        If norev = 0 Then CWGraphXY.Plots.Item(noop).ChartXvsY Datax(nop), DataY(nop)
    Next

    norev = 0

    'determine max and min values of data and display
    N = 0
    yma = DataY(0)
    ymi = DataY(0)
    xma = Datax(0)
    xmi = Datax(0)

    Do
        N = N + 1
        If DataY(N) > yma Then yma = DataY(N)
        If DataY(N) < ymi Then ymi = DataY(N)
        If Datax(N) > xma Then xma = Datax(N)
        If Datax(N) < xmi Then xmi = Datax(N)
    Loop Until N = No

    lblYmax.Caption = Format$(yma, "0.00E-00")
    lblYmin.Caption = Format$(ymi, "0.00E-00")
    lblXmax.Caption = Format$(xma, "0.000")
    lblXmin.Caption = Format$(xmi, "0.000")
```

```
ErrHandle:
End Sub

Private Sub AquireXY_Click()

    cmdStartXY_Click

End Sub



Private Sub AquireYt_Click()

    Form4.Visible = False
    form2.Visible = True
    form2.CWButton1.Value = False

End Sub

Private Sub Blue_Click()

    CWGraphXY.Plots.Item(2).ClearData
    CWGraphXY.Plots.Item(2).ClearData
    noop = noop - 1

    If norev = 0 Or norev > 2 Then norev = 2



End Sub

Private Sub cmdOpenCom_Click()
' either open the com port or close it depending on button caption
    If cmdOpenCom.Caption = "Open com port" Then
        With objCommPort
            ' Set com port number based on user selection
            .CommPort = 5
            ' try to open the port
            .PortOpen = True
            ' Save current port settings
            OldHandShake = .Handshaking
            OldSettings = .Settings
            OldMode = .InputMode
            OldInputLen = .InputLen
            ' Set the new settings for Teckmo communications
            .Handshaking = comNone
            .Settings = "9600,N,8,1"
            .InputMode = comInputModeText
            .InputLen = 1
            .RThreshold = 1
            End With
        ' disable the port select area so the user can't try to change
        ' to a different com port while the current one is open
        'comboPortNum.Enabled = False
        ' change button caption to close com port
        cmdOpenCom.Caption = "Close com port"
        ' display a com port status message
        'lblStatus.Caption = "Com " & comboPortNum.Text & " is the current port."

        ' Note there is no need to make sure the receive buffer is empty
        ' since RTheshold is set to 1 above. This means whenever there is
        ' at least 1 byte in the receive buffer, the objCommPort_OnComm()
        ' subroutine will be triggered. this routine does error checking
        ' to remove stray bytes that are not part of a 6 byte reply

        ' Renumber all units
        'Call Renumber_All
    Else
        ' the "Close com port" button was clicked
        ' return comm port settings to the previous system values
        ' then close the port
        With objCommPort
            .Handshaking = OldHandShake
            .Settings = OldSettings
            .InputMode = OldInputMode
            .InputLen = OldInputLen
            .PortOpen = False
        End With
        ' enable open com port button
        cmdOpenCom.Caption = "Open com port"
        'comboPortNum.Enabled = True
        ' change the status message
        'lblStatus.Caption = "All com ports are currently closed."
    End If
    ' switch focus back to send button so enter key causes send
    'cmdSend.SetFocus
    objCommPort.Output = "/1ZR" + Chr(13)
End Sub

Private Sub cmdStartXY_Click()
TriggerActivate = False
nc = 0
cmdstartxy.Visible = False
Label2.Visible = True
Form3.Refresh


'get the spectrometer setup

Test1 = SPX_init(ByVal "USB0::0x01313::0x0111::S/N M00231225::RAW", ByVal 2000,
instname)
Text6 = instname
inttime = CWKnob1.Value / 1000 ' sets integration time for the device
Test2% = SPX_setIntTime(ByVal instname, ByVal inttime)
test4% = SPX_getIntTime(ByVal instname, integrationtime) 'gets the actual value
Text4 = integrationtime
wave = SPX_getWavelengthData(ByVal instname, wavelengthdata(0), mini, Max)

'set system to continuous
Test3% = SPX_startScanCont(ByVal instname)


'gai = InputBox("Enter Gain", "Enter Gain", gai)

gainEC = 10 ^ (CWKnob2.Value)


'put in arm channel 1

    Gain = BIP5VOLTS             ' set the gain

    'Ulstat% = cbAIn(BoardNum%, 1, Gain, DataValue%) 'example of analogue in
    'If Ulstat% = 30 Then MsgBox "Change the Gain argument to one supported by this
board.", 0, "Unsupported Gain"
    'If Ulstat% <> 0 Then Stop

    'Ulstat% = cbToEngUnits(BoardNum%, Gain, DataValue%, eng!)
    'If Ulstat% <> 0 Then Stop

estart = eng!
```

```
'Do

'Ulstat% = cbAIn(BoardNum%, 1, Gain, DataValue%)
'    If Ulstat% = 30 Then MsgBox "Change the Gain argument to one supported by this
board.", 0, "Unsupported Gain"
'    If Ulstat% <> 0 Then Stop

'    Ulstat% = cbToEngUnits(BoardNum%, Gain, DataValue%, eng!)
'    If Ulstat% <> 0 Then Stop




'Loop Until Abs(eng! - estart) > 0.1

NumPoints = 2000 / 20
    'Do ' sets up the wait state
    'vloop = 0

    'For wa = 1 To (NumPoints / 5)
    'CWAIPoint1.SingleRead Data
    'vloop = vloop + Data(0)
    'Erase Data
    'Next wa
    'vloop = vloop / (NumPoints / 5)


    'Loop Until Abs(vstart - vloop) > 0.005
'---
'CWAIPoint1.SingleRead Data
'    vstart = Data(1)

'    Do ' sets up the wait state
'    vloop = 0

'    For wa = 1 To (NumPoints / 5)
'    CWAIPoint1.SingleRead Data
'
'    vloop = Data(1)
'    Erase Data
'    Next wa
'    vloop = vloop / (NumPoints / 5)


'    Loop Until Abs(vstart - vloop) > 0.1
'---
'startt = Timer

CWDIO1.SingleWrite 0
tstartd = Timer
Do
Loop Until Timer - tstartd > 0.1
CWDIO1.SingleWrite 1
Do
Loop Until Timer - tstartd > 0.2
CWDIO1.SingleWrite 0
'Do
'Loop Until Timer - tstartd > 10.1
'CWDIO1.SingleWrite 1
'Do
'Loop Until Timer - tstartd > 10.2
'CWDIO1.SingleWrite 0

'objCommPort.Output = "/1gS10I2A1000S6I2A0G3R" + Chr(13)
```

```
Label2.Visible = False
Form3.Refresh





Dim TotChan0
Dim TotChan1
Dim X
Dim N
Dim LowChan%
Dim AveChan0
Dim AveChan1

    StopAq = 0
    gai = 1000000#
    sweep = 50
    noop = o
    startd = Timer


    'MemHandle& = cbWinBufAlloc(NumPoints&)      ' set aside memory to hold data
    'If MemHandle& = 0 Then Stop

    For N = 0 To 20000
        DataY(N) = Empty
        Datax(N) = Empty
    Next N

    CWGraphXY.ClearData

cmdStopXY.Visible = True



   N = 0
   countersave = 0



   Do
    N = N + 1

repeat:
    dummy% = DoEvents()

    If StopAq = 1 Then

        Exit Sub

    End If

    'MemHandle& = cbWinBufAlloc(NumPoints&)      ' set aside memory to hold data
    'If MemHandle& = 0 Then Stop
    ' Collect the values with cbAInScan%()
    ' Parameters:
    '   BoardNum%   :the number used by CB.CFG to describe this board
    '   LowChan%    :the first channel of the scan
    '   HighChan%   :the last channel of the scan
    '   CBCount&    :the total number of A/D samples to collect
    '   CBRate&     :sample rate
    '   Gain        :the gain for the board
    '   ADData%     :the array for the collected data values
    '   Options     :data collection options
```

```
     '   LowChan% = 0
      '  HighChan% = 1


    'CBCount& = NumPoints&          ' total number of data points to collect
    'CBRate& = sweep * 200              ' sampling rate (samples per second)
    'Options = CONVERTDATA     ' return data as 12-bit values
    Gain = BIP10VOLTS               ' set the gain

    'If MemHandle& = 0 Then Stop    ' check that a handle to a memory buffer exists
   'If CBRate& < 1000 Then CBRate& = 1000

    'Ulstat% = cbAInScan(BoardNum%, LowChan%, HighChan%, CBCount&, CBRate&, Gain,
MemHandle&, Options)

    'If Ulstat% = 30 Then MsgBox "Change the Gain argument to one supported by this
board.", 0, "Unsupported Gain"
    'If Ulstat% <> 0 And Ulstat% <> 91 Then Stop

    ' Transfer the data from the memory buffer set up by Windows to an array for use by
Visual Basic

    'Ulstat% = cbWinBufToArray(MemHandle&, ADData%(0), FirstPoint&, CBCount&)
    'If Ulstat% <> 0 Then Stop

  'Ulstat% = cbAIn(BoardNum%, 1, Gain, DataValue%)
   ' If Ulstat% = 30 Then MsgBox "Change the Gain argument to one supported by this
board.", 0, "Unsupported Gain"
   ' If Ulstat% <> 0 Then Stop

   ' Ulstat% = cbToEngUnits(BoardNum%, Gain, DataValue%, eng!)
    'If Ulstat% <> 0 Then Stop
    TotChan1 = 0

    For w = 1 To NumPoints
    CWAIPoint1.SingleRead Data

    TotChan1 = TotChan1 + Data(1)
    Erase Data
    Next w

    AveChan1 = TotChan1 / (NumPoints)


    Datax(N) = Timer - startd

    If Datax(N) > 10 Then
    '    If TriggerActivate = False Then
        CWDIO1.SingleWrite 1
    'tstarto = Timer
    'Do
    'Loop Until Timer - tstarto > 0.1
    'CWDIO1.SingleWrite 0
    '    TriggerActivate = True
    '    End If
        End If

    DataY(N) = AveChan1 / gainEC




'avnum = CWSlide1.Value
```

```
Nsp = 0
Dim datasp(2999)
Erase datasp
Erase dataA

Do
test4 = SPX_getScanData(ByVal instname, dataspec(0))

For av = 0 To 2998
datasp(av) = datasp(av) + dataspec(av)
Next av
Nsp = Nsp + 1
Loop Until Nsp = avnum

'subtract background
For av = 0 To 2998
dataspec(av) = (datasp(av) / avnum) - datab(av)
If (dataspec(av) / dataref(av)) <= 0 Then
dataA(av) = 5
Else
dataA(av) = 2 - (Log(((dataspec(av) / dataref(av))) * 100) / 2.301)
End If
Next av



bin = 0
For av = 1 To 2999 Step 3
dataAbin(bin) = (dataA(av - 1) + dataA(av) + dataA(av + 1)) / 3
wavelengthdatabin(bin) = (wavelengthdata(av - 1) + wavelengthdata(av) +
wavelengthdata(av + 1)) / 3

'Set pointer 1 to be the active pointer
Set CWSlide2.ActivePointer = CWSlide2.Pointers.Item(1)

If wavelengthdatabin(bin) < CWSlide2.Value Then
dataAbin(bin) = 0
Else
End If

'Set pointer 2 to be the active pointer
Set CWSlide2.ActivePointer = CWSlide2.Pointers.Item(2)

If wavelengthdatabin(bin) > CWSlide2.Value Then
dataAbin(bin) = 0
Else
End If



bin = bin + 1
Next av

CWGraph2.Axes.Item(1).Minimum = CWSlide2.Pointers.Item(1)
CWGraph2.Axes.Item(1).Maximum = CWSlide2.Pointers.Item(2)



CWGraph2.PlotXvsY wavelengthdatabin, dataAbin
CWGraphXY.ChartXvsY Datax(N), DataY(N)
'nc = nc + 1

    FileName = Text2.Text + Str(N) + ".dat"
    Open FileName For Output As #2
    Print #2, Format(Datax(N), "0.0000"); ","; Format(DataY(N), "0.00e-00")
    For nsave = 0 To 999
    Print #2, Format(wavelengthdatabin(nsave), "000.0"); ","; Format(dataAbin(nsave),
"0.0000")
    Next nsave
    Close #2
    Text1 = N


    lblShowVolts.Caption = Format$(Datax(N), "0.000")
    lblShowCurrent.Caption = Format(DataY(N), "0.00E-00")
    countersave = countersave + 1


    Loop Until N = 20000


End Sub

Private Sub cmdQuit_Click()
Dim yesno

yesno = MsgBox("Are you sure you want to quit?", vbQuestion + vbYesNo, "Quit")

If yesno = 6 Then End


End Sub

Private Sub cmdStopXY_Click()
noop = 190
StopAq = 1
    'tmrXY.Enabled = False
    'cmdstartxy.Visible = True
    'lblAquiring.Visible = False
    'File.Enabled = True
    'Aquire.Enabled = True
    'CWButton2.Enabled = True

    'Ulstat% = cbWinBufFree(MemHandle&)      ' Free up memory for use by
                                            ' other programs
    'If Ulstat% <> 0 Then Stop

    N = 0
    yma = DataY(1)
    ymi = DataY(1)
    xma = Datax(1)
    xmi = Datax(1)

    Do
        N = N + 1
        If DataY(N) > yma Then yma = DataY(N)
        If DataY(N) < ymi Then ymi = DataY(N)
        If Datax(N) > xma Then xma = Datax(N)
        If Datax(N) < xmi Then xmi = Datax(N)
    Loop Until N = countersave

    lblYmax.Caption = Format$(yma, "0.00E-00")
    lblYmin.Caption = Format$(ymi, "0.00E-00")

    lblXmax.Caption = Format$(xma, "0.000")
    lblXmin.Caption = Format$(xmi, "0.000")
    Test5% = SPX_close(ByVal instname)
 CWVisa1.Close  'actually stops crashing for the second time.
End Sub

Private Sub ExitXY_Click()
```

```
    Form4.Visible = False
    ' Form1.Visible = True

End Sub




Private Sub Command1_Click()

'this gets a dark background for correction

Test1 = SPX_init(ByVal "USB0::0x01313::0x0111::S/N M00231225::RAW", ByVal 2000,
instname)
Text6 = instname
inttime = CWKnob1.Value / 1000 ' sets integration time for the device
Test2% = SPX_setIntTime(ByVal instname, ByVal inttime)
test4% = SPX_getIntTime(ByVal instname, integrationtime) 'gets the actual value
Text4 = integrationtime
wave = SPX_getWavelengthData(ByVal instname, wavelengthdata(0), mini, Max)

'set system to continuous
Test3% = SPX_startScanCont(ByVal instname)

'get the data
avnum = CWSlide1.Value
Nbac = 0
Erase datab
Do
test4 = SPX_getScanData(ByVal instname, databac(0))
For av = 0 To 2999
datab(av) = datab(av) + databac(av)
Next av
Nbac = Nbac + 1
Loop Until Nbac = avnum

For av = 0 To 2999
datab(av) = datab(av) / avnum
Next av

Test5% = SPX_close(ByVal instname)
Text5 = Test5%

'save background data
filesavebac = Text2.Text + ".bac"
Open filesavebac For Output As #1

For av = 0 To 2999
Print #1, Format(wavelengthdata(av), "000.0"); ","; Format(datab(av), "0.00000")
Next av
Close #1

'save info

filesaveinf = Text2.Text + ".inf"
Open filesaveinf For Output As #1

Print #1, Date
Print #1, "Number of averaged spectra = "; avnum
Print #1, "Integration time/s = "; inttime


Close #1
```

```
CWVisa1.Close  'actually stops crashing for the second time.
Command2.Visible = True
End Sub

Private Sub Command2_Click()

Test1 = SPX_init(ByVal "USB0::0x01313::0x0111::S/N M00231225::RAW", ByVal 2000,
instname)
Text6 = instname
inttime = CWKnob1.Value / 1000 ' sets integration time for the device
Test2% = SPX_setIntTime(ByVal instname, ByVal inttime)
test4% = SPX_getIntTime(ByVal instname, integrationtime) 'gets the actual value
Text4 = integrationtime
wave = SPX_getWavelengthData(ByVal instname, wavelengthdata(0), mini, Max)

'set system to continuous
Test3% = SPX_startScanCont(ByVal instname)

'get the data
avnum = CWSlide1.Value
Nsp = 0

Dim datar(2999)
Erase datar

Do
test4 = SPX_getScanData(ByVal instname, dataref(0))
For av = 0 To 2998
datar(av) = datar(av) + dataref(av)
Next av
Nsp = Nsp + 1
Loop Until Nsp = avnum

For av = 0 To 2998
dataref(av) = (datar(av) / avnum) - datab(av)
If dataref(av) = 0 Then
dataref(av) = 0.0000000001
Else
End If

Next av

'save reference data
filesaveref = Text2.Text + ".ref"
Open filesaveref For Output As #1

For av = 0 To 2999
Print #1, Format(wavelengthdata(av), "000.0"); ","; Format(dataref(av), "0.00000")
Next av
Close #1



'CWGraph1.Caption = refence
CWGraph1.PlotXvsY wavelengthdata, dataref
Test5% = SPX_close(ByVal instname)
CWVisa1.Close  'actually stops crashing for the second time.
'leave running for later
Command2.Visible = False
cmdstartxy.Visible = True
'cmdStartXY.Visible = True
End Sub

Private Sub Command3_Click()
```

```
resetit = SPX_reset(ByVal instname)
CWVisa1.Close

form1.Show

End Sub







Private Sub CWSlide1_PointerValueChanged(ByVal Pointer As Long, Value As Variant)
Text3 = CWSlide1.Pointers.Item(1).Value


End Sub

Private Sub CWSlide2_PointerValueChanged(ByVal Pointer As Long, Value As Variant)
CWGraph1.Axes(1).Minimum = CWSlide2.Pointers.Item(1).Value
CWGraph1.Axes(1).Maximum = CWSlide2.Pointers.Item(2).Value
CWGraph2.Axes(1).Minimum = CWSlide2.Pointers.Item(1).Value
CWGraph2.Axes(1).Maximum = CWSlide2.Pointers.Item(2).Value
End Sub

Private Sub Cyan_Click()
CWGraphXY.Plots.Item(5).ClearData
CWGraphXY.Plots.Item(5).ClearData
noop = noop - 1
If norev = 0 Then norev = 5


End Sub

Private Sub force_Click()
Form4.Visible = False
form2.Visible = False

    Form3.Visible = True
End Sub

Private Sub Form_Load()
noop = 0
CWAIPoint1.Device = 1
ChannelString = "0,1"
CWAIPoint1.Channels.Add ChannelString
CWDIO1.Ports(0).Assignment = cwdioOutput
End Sub

Private Sub Green_Click()
CWGraphXY.Plots.Item(1).ClearData
CWGraphXY.Plots.Item(1).ClearData
noop = noop - 1
If norev = 0 Or norev > 1 Then norev = 1

End Sub

Private Sub Open_Click()

Dim No
Dim E$
Dim i$
Dim nop
```

```
    Shape1.Visible = False
    Label2.Visible = False
    Picture1.Visible = False

    form2.filebox.CancelError = True
    On Error GoTo ErrHandle
    Form4.filebox.FileName = "*.dat"
    Form4.filebox.InitDir = "C:\doug\data"
    Form4.filebox.Action = 1
    Form4.Caption = Form4.filebox.FileName
    Open Form4.filebox.FileName For Input As #4

    No = -1

    Do
        No = No + 1
        Input #4, E$
        Datax(No) = Val(E$)
        Input #4, i$
        DataY(No) = Val(i$)
    Loop Until EOF(4)

    Close #4
    'If noop = 0 Then
    'noop = 1
    'GoTo FirstOpen
    'End If

   'Over = MsgBox("Overlay?", vbYesNo, "Open File")

    'If Over = 7 Then
'FirstOpen:
noop = 1
    CWGraphXY.ClearData
     For nop = 0 To No
        CWGraphXY.Plots.Item(noop).ChartXvsY Datax(nop), DataY(nop)
    Next

    N = 0
    yma = DataY(0)
    ymi = DataY(0)
    xma = Datax(0)
    xmi = Datax(0)

    Do
        N = N + 1
        If DataY(N) > yma Then yma = DataY(N)
        If DataY(N) < ymi Then ymi = DataY(N)
        If Datax(N) > xma Then xma = Datax(N)
        If Datax(N) < xmi Then xmi = Datax(N)
    Loop Until N = No

    lblYmax.Caption = Format$(yma, "0.00E-00")
    lblYmin.Caption = Format$(ymi, "0.00E-00")
    lblXmax.Caption = Format$(xma, "0.000")
    lblXmin.Caption = Format$(xmi, "0.000")

    norev = 0



ErrHandle:     Exit Sub
End Sub
```

```
Private Sub optI0_Click()

End Sub

Private Sub optI1_Click()

End Sub

Private Sub optI2_Click()

End Sub

Private Sub optI3_Click()

End Sub



Private Sub Red_Click()
CWGraphXY.Plots.Item(3).ClearData
CWGraphXY.Plots.Item(3).ClearData
noop = noop - 1
If norev = 0 Or norev > 3 Then norev = 3


End Sub



Private Sub RemoveAll_Click()
yn = MsgBox("Are you sure you want to remove all plots?", vbQuestion + vbYesNo, "Remove
All Plots")
If yn = 6 Then
noop = 0
norev = 0
CWGraphXY.Plots.Item(1).ClearData
CWGraphXY.Plots.Item(2).ClearData
CWGraphXY.Plots.Item(3).ClearData
CWGraphXY.Plots.Item(4).ClearData
CWGraphXY.Plots.Item(5).ClearData
End If
End Sub

Private Sub Save_Click()

Dim ns

    Form4.filebox.CancelError = True
    On Error GoTo ErrHandle
    Form4.filebox.Flags = &H2&
    Form4.filebox.FileName = "*.dat"
    Form4.filebox.InitDir = "C:\doug\data"
    Form4.filebox.Action = 2
    Form4.Caption = Form4.filebox.FileName
    Open Form4.filebox.FileName For Output As #2

    ns = 0

    Do
        ns = ns + 1
        Print #2, Datax(ns); ","; DataY(ns)
    Loop Until ns = countersave

    Close #2
```

```
ErrHandle:     Exit Sub
End Sub




Private Sub Yellow_Click()
CWGraphXY.Plots.Item(4).ClearData
CWGraphXY.Plots.Item(4).ClearData
noop = noop - 1
If norev = 0 Or norev > 4 Then norev = 4


End Sub
---module 1---

' -------------------------------------------------------------------------
'
'       Thorlabs SPx-USB - VB Header file for SP1-USB / SP2-USB Spectrometer instr.
driver
'  Do not modify the contents of this file.
'       Copyright:      Copyright(c) 2007, Thorlabs GmbH (www.thorlabs.com)
'
' -------------------------------------------------------------------------
'
'       Date:           Aug-21-2007
'       Software-Nr:   09.167.200
'       Version:        2.0
'       Changelog:              see 'readme.rtf'
'
' -------------------------------------------------------------------------
'
'       Disclaimer:
'
'       This library is free software; you can redistribute it and/or
'       modify it under the terms of the GNU Lesser General Public
'       License as published by the Free Software Foundation; either
'       version 2.1 of the License, or (at your option) any later version.
'
'       This library is distributed in the hope that it will be useful,
'       but WITHOUT ANY WARRANTY; without even the implied warranty of
'       MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
'       Lesser General Public License for more details.
'
'       You should have received a copy of the GNU Lesser General Public
'       License along with this library; if not, write to the Free Software
'       Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
'
' -------------------------------------------------------------------------


' -------------------------------------------------------------------------
' Functions
' -------------------------------------------------------------------------

' - Init/Close ---
Declare Function SPX_init& Lib "SPX_Drv_32.dll" (ByVal x1$, ByVal x2&, x3&)
Declare Function SPX_close& Lib "SPX_Drv_32.dll" (ByVal x1&)



' - Configuration Functions ---
Declare Function SPX_setIntTime& Lib "SPX_Drv_32.dll" (ByVal x1&, ByVal x2#)
Declare Function SPX_getIntTime& Lib "SPX_Drv_32.dll" (ByVal x1&, x2#)
```

```
' - Action/Status Functions ---
Declare Function SPX_startScan& Lib "SPX_Drv_32.dll" (ByVal x1&)
Declare Function SPX_startScanExtTrg& Lib "SPX_Drv_32.dll" (ByVal x1&)
Declare Function SPX_startScanCont& Lib "SPX_Drv_32.dll" (ByVal x1&)
Declare Function SPX_startScanContExtTrg& Lib "SPX_Drv_32.dll" (ByVal x1&)
Declare Function SPX_getDeviceStatus& Lib "SPX_Drv_32.dll" (ByVal x1&, ByVal x2$)


' - Data Functions ---
Declare Function SPX_getScanData& Lib "SPX_Drv_32.dll" (ByVal x1&, x2#)
Declare Function SPX_getRawScanData& Lib "SPX_Drv_32.dll" (ByVal x1&, x2%)
Declare Function SPX_getWavelengthData& Lib "SPX_Drv_32.dll" (ByVal x1&, x2#, x3#, x4#)


' - Utility Functions ---
Declare Function SPX_reset& Lib "SPX_Drv_32.dll" (ByVal x1&)
Declare Function SPX_errorMessage& Lib "SPX_Drv_32.dll" (ByVal x1&, ByVal x2&, ByVal
x3$)
Declare Function SPX_identificationQuery& Lib "SPX_Drv_32.dll" (ByVal x1&, ByVal x2$,
ByVal x3$, ByVal x4$, ByVal x5$)
Declare Function SPX_revisionQuery& Lib "SPX_Drv_32.dll" (ByVal x1&, ByVal x2$, ByVal
x3$)
Declare Function SPX_setUserText& Lib "SPX_Drv_32.dll" (ByVal x1&, ByVal x2$)
Declare Function SPX_getUserText& Lib "SPX_Drv_32.dll" (ByVal x1&, ByVal x2$)
```

```
' ---------------------------------------------------------------------------
' Definitions
' ---------------------------------------------------------------------------

' - Driver Error Codes ---
Global Const VI_WARN_SPX_DATA_NOT_READY = &H3FFC0901
Global Const VI_ERROR_READ_INCOMPLETE = &HBFFC0901


' - Buffers ---
Global Const SPX_BUFFER_SIZE = 256                          ' General buffer size
Global Const SPX_ERR_DESCR_BUFFER_SIZE = 512                    ' Buffer size for error
messages
Global Const SPX_TEXT_BUFFER_SIZE = 256                                         ' Buffer
size for texts from the LC1
Global Const SPX_NUM_PIXELS = 3000                                              ' this
is the number of effective pixels of CCD
Global Const SPX_MAX_USER_NAME_SIZE = 31                                ' this is max
number of characters for a user name

' - VISA strings ---
Global Const SP1_VI_FIND_RSC_PATTERN = "USB?*?{VI_ATTR_MANF_ID==0x1313 &&
VI_ATTR_MODEL_CODE==0x0111}"
Global Const SP2_VI_FIND_RSC_PATTERN = "USB?*?{VI_ATTR_MANF_ID==0x1313 &&
VI_ATTR_MODEL_CODE==0x0112}"
Global Const SPX_VI_FIND_RSC_PATTERN = "USB?*?{VI_ATTR_MANF_ID==0x1313 &&
((VI_ATTR_MODEL_CODE==0x0111) || (VI_ATTR_MODEL_CODE==0x0112))}"


' - Communication timeout ---
Global Const SPX_TIMEOUT_MIN = 1000
Global Const SPX_TIMEOUT_MAX = 60000
```

```
' - LC1 specific constants ---
Global Const SPX_MAX_INT_TIME = 0.2
' 200ms is the maximum integration time
Global Const SPX_MIN_INT_TIME = 0.000001                                        ' 1us
is the minimum integration time

' - scan states - these are the states reported by LC1_getDeviceStatus ---
Global Const SPX_SCAN_STATE_IDLE = &H0&
' LC waits for new scan to execute
Global Const SPX_SCAN_STATE_TRANSFER = &H20&                            ' scan is done,
waiting for data transfer to be finished
Global Const SPX_SCAN_STATE_IN_PROGRES = &H40&                  ' scan is in progres,
this message should never be get
Global Const SPX_SCAN_STATE_WAIT_TRIGGER = &H80&                ' same as IDLE except
that external trigger is armed
Global Const SPX_SCAN_STATE_UNKNOWN = &HFF&                             ' default value


' --------------------------------------------------------------------------
' End of file
' --------------------------------------------------------------------------
```

A3: Scanning Software

```
---Acquisition ----

Dim XRange As Integer
Dim YRange As Integer
'Dim zRange As Integer

Dim Timewin

Dim LowChan%
Dim HighChan%
Dim yesno As Integer

Dim N As Single

Dim counter As Integer
Dim Total()
Dim Ave()
Dim Max()
Dim Maxp()
Dim MaxAve

Dim Total_bef()
Dim Ave_bef()
Dim Max_bef()
Dim Maxp_bef()
Dim MaxAve_bef
Dim Avep_bef()
Dim Total_aft()
Dim Ave_aft()
Dim Max_aft()
Dim Maxp_aft()
Dim MaxAve_aft()
Dim Avep_aft()
Dim Max_diff()
Dim W600_aft()
Dim W600_bef()
Dim W600_diff()
Dim Abs490nm()

Dim OriginX
Dim OriginY

Dim xmax As Integer
Dim ymax As Integer
Dim zmax As Integer
Dim xmin As Integer
Dim ymin As Integer
Dim zmin As Integer

Dim Avep()
Dim MinAve
Dim MinMax
Dim MaxMax
 Dim X
 Dim Y
 Dim Z
 Dim FileName
 Dim numpositions

 Dim BeforePulse As Boolean
 Dim Wavelength600nm
 Dim Wavelength490nm
 Dim Well_125_490nm


Static Sub cmdQuit_Click()
Dim yesno

yesno = MsgBox("Are you sure you want to quit?", vbQuestion + vbYesNo, "Quit")

If yesno = 6 Then End
End Sub

Static Sub cmdReset_Click()
counter = 0
lblNoDatasets.Caption = 0
End Sub

Static Sub cmdSave_Click()
Dim ns


 Form1.filebox.CancelError = True

    On Error GoTo ErrHandle
    Form1.filebox.Flags = &H2&
    Form1.filebox.FileName = "*.dat"
    Form1.filebox.InitDir = "C:\WINDOWS\Desktop"
    Form1.filebox.Action = 2
    Form1.Caption = Form1.filebox.FileName
    Open Form1.filebox.FileName For Output As #2

    ns = -1

    For X = xmin To xmax
     For Y = ymin To ymax
        Print #2, X; ","; Y; ","; Ave(X, Y)
     Next Y
    Next X
    Close #2



ErrHandle:     Exit Sub
End Sub

Static Sub cmdStart_Click()
StackList.Text1.Text = StackList.Text1.Text + "Acquisition.cmdStart_Click" + Chr(13) +
Chr(10)
XRange = XYZController.Combo2.Text - XYZController.Combo1.Text + 1
YRange = XYZController.Combo4.Text - XYZController.Combo3.Text + 1
'zRange = XYZController.Combo6.Text - XYZController.Combo5.Text + 1

'RepeatLoop:

If XYZController.NumEditCounter > 0 Then
'If BeforePulse = False Then
GoTo notfirst
'End If
End If

'   If PulseBox.Value = 0 Then
'   CWGraph3D1.Height = 6975
'   CWGraph3D2.Height = 2295
'   CWGraph3D2.Top = 7320
'   End If
   CWGraph1.ChartLength = NumEditSamplestoSave.Value
    MaxAve = Empty
    MaxMax = Empty


ReDim Ave(XRange, YRange)
ReDim Avep(XRange, YRange)
ReDim Total(XRange, YRange)
ReDim Max(XRange, YRange)
ReDim Maxp(XRange, YRange)
ReDim Ave_bef(XRange, YRange)
ReDim Avep_bef(XRange, YRange)
ReDim Total_bef(XRange, YRange)
ReDim Max_bef(XRange, YRange)
ReDim Maxp_bef(XRange, YRange)
ReDim Ave_aft(XRange, YRange)
ReDim Avep_aft(XRange, YRange)
ReDim Total_aft(XRange, YRange)
ReDim Max_aft(XRange, YRange)
ReDim Maxp_aft(XRange, YRange)
ReDim Max_diff(XRange, YRange)
ReDim W600_aft(XRange, YRange)
ReDim W600_bef(XRange, YRange)
ReDim W600_diff(XRange, YRange)
ReDim Abs490nm(XRange, YRange)

    For X = 0 To (XRange - 1)
      For Y = 0 To (YRange - 1)

      Ave(X, Y) = Empty
      Avep(X, Y) = Empty
      Total(X, Y) = Empty
      Max(X, Y) = Empty
      Maxp(X, Y) = Empty
      Ave_bef(X, Y) = Empty
      Avep_bef(X, Y) = Empty
      Total_bef(X, Y) = Empty
      Max_bef(X, Y) = Empty
      Maxp_bef(X, Y) = Empty
      Ave_aft(X, Y) = Empty
      Avep_aft(X, Y) = Empty
      Total_aft(X, Y) = Empty
      Max_aft(X, Y) = Empty
      Maxp_aft(X, Y) = Empty
      Max_diff(X, Y) = Empty
      W600_aft(X, Y) = Empty
      W600_bef(X, Y) = Empty
      W600_diff(X, Y) = Empty
      Abs490nm(X, Y) = Empty
      'Shape1(21 * Y + X).FillColor = RGB(255, 255, 255)
        'Shape2(21 * Y + X).FillColor = RGB(255, 255, 255)

      Next Y
      Next X
MinAve = 1E+19
MinMax = 1E+19
Acquisition.filebox.CancelError = True
    'On Error GoTo ErrSave

    Acquisition.filebox.Flags = &H2&
    Acquisition.filebox.FileName = "*.dat"
    Acquisition.filebox.InitDir = "C:\WINDOWS\Desktop\a.dat"
    Acquisition.filebox.Action = 2
    Acquisition.Caption = Acquisition.filebox.FileName
    FileName = Acquisition.filebox.FileName

OriginX = Int(XYZController.NumEditPosition(0) + 0.5)
OriginY = Int(XYZController.NumEditPosition(1) + 0.5)

CWGraph3D1.Axes(1).Minimum = OriginX
CWGraph3D1.Axes(1).Maximum = OriginX + (XYZController.Combo2 *
XYZController.NumEditResolution(0))
CWGraph3D1.Axes(2).Minimum = OriginY
CWGraph3D1.Axes(2).Maximum = OriginY + (XYZController.Combo4 *
XYZController.NumEditResolution(1))

CWGraph3D2.Axes(1).Minimum = OriginX
CWGraph3D2.Axes(1).Maximum = OriginX + (XYZController.Combo2 *
XYZController.NumEditResolution(0))
CWGraph3D2.Axes(2).Minimum = OriginY
CWGraph3D2.Axes(2).Maximum = OriginY + (XYZController.Combo4 *
XYZController.NumEditResolution(1))


Timewin = NumEditTimewin.Value

notfirst:
Errx:
'here is the problem
X = (Round(XYZController.NumEditRelPosition(0), 0) /
(XYZController.NumEditResolution(0))) + (((XRange - 1) / 2) -
Int((Val(XYZController.Combo2.Text) + Val(XYZController.Combo1.Text)) / 2))

Erry:
Y = (Round(XYZController.NumEditRelPosition(1), 0) /
(XYZController.NumEditResolution(1))) + (((YRange - 1) / 2) -
Int((Val(XYZController.Combo3.Text) + Val(XYZController.Combo4.Text)) / 2))

'Z = (Round(XYZController.NumEditRelPosition(2), 0) /
(XYZController.NumEditResolution(2))) + (10 - Int((Val(XYZController.Combo5.Text) +
Val(XYZController.Combo6.Text)) / 2))

If counter = 0 Then
xmax = X
xmin = X
ymax = Y
ymin = Y
'zmax = Z
'zmin = Z
End If

If X < xmin Then xmin = X
If X > xmax Then xmax = X

If Y < ymin Then ymin = Y
If Y > ymax Then ymax = Y

If Z < zmin Then zmin = Z
If Z > zmax Then zmax = Z

Select Case Combo1.Text

Case "X vs Y"
Combo2.Clear
For i = zmin - (10 - Int((Val(XYZController.Combo5.Text) +
Val(XYZController.Combo6.Text)) / 2)) To zmax - (10 -
Int((Val(XYZController.Combo5.Text) + Val(XYZController.Combo6.Text)) / 2))
```

```
Combo2.AddItem i
Next i

Case "X vs Z"
Combo2.Clear
For i = ymin - (10 - Int((Val(XYZController.Combo3.Text) +
Val(XYZController.Combo4.Text)) / 2)) To ymax - (10 -
Int((Val(XYZController.Combo3.Text) + Val(XYZController.Combo4.Text)) / 2))
Combo2.AddItem i
Next i

Case "Y vs Z"
Combo2.Clear
For i = xmin - (10 - Int((Val(XYZController.Combo2.Text) +
Val(XYZController.Combo1.Text)) / 2)) To xmax - (10 -
Int((Val(XYZController.Combo2.Text) + Val(XYZController.Combo1.Text)) / 2))
Combo2.AddItem i
Next i

End Select

     Call plotit
    ' GoTo RepeatLoop
    StackList.Text1.Text = StackList.Text1.Text + "...Completed Loop..." + Chr(13) +
Chr(10)
    End Sub
Static Sub plotit()
repeatspec: 'come back and gather another spectra

StackList.Text1.Text = StackList.Text1.Text + "Acquisition.PlotIt" + Chr(13) + Chr(10)
Dim instname As ViSession ' the 'name' of the spectrometer
Dim wavelengthdata(2999) As ViReal64 ' where the wavelength goes
Dim data(2999) As ViReal64 ' the data array
Dim dataA(2999) As ViReal64 ' ratio
Dim dataaf(2999) As ViReal64
Dim KInt As Integer
Dim datatot(2999)

If PulseBox.Value = 1 Then
If BeforePulse = False Then
'Pulse here and then take a different spectrum triggers the step on positive edge
'set a step and allow 0.5 s to equilibriate
CWDIO1.SingleWrite 1
tstarto = Timer
Do
Loop Until Timer - tstarto > CWSlide1.Value
CWDIO1.SingleWrite 0
Else
End If
End If
Label8.Caption = CStr(Timer - tstarto)

'----
For i = 0 To 2999
    datatot(i) = 0
Next i
For k = 0 To CWSlide2.Value - 1

'----
spetro = SPX_init(ByVal "USB0::0x01313::0x0111::S/N M00231225::RAW", ByVal 2000,
instname)
'Gives you the name back
'Text1 = instname
' sets integration time for the device
inttime = CWKnob1.Value / 1000
```

```
spetro = SPX_setIntTime(ByVal instname, ByVal inttime)
'check the intigration time is OK
'gets the actual value
spetro = SPX_getIntTime(ByVal instname, integrationtime)
'Text3 = integrationtime
'find the wavelength, max and min if wanted
Wave = SPX_getWavelengthData(ByVal instname, wavelengthdata(0), mini, Maxw)

'If PulseBox.Value = 1 Then
'If BeforePulse = False Then
'Pulse here and then take a different spectrum triggers the step on positive edge
'set a step and allow 0.5 s to equilibriate
'CWDIO1.SingleWrite 1
'tstarto = Timer
'Do
'Loop Until Timer - tstarto > CWSlide1.Value
'CWDIO1.SingleWrite 0
'Else
'End If
'End If

spetro = SPX_startScan(ByVal instname)
'get the data. CARE on the array name - seems to have to be with '(0)'
spetro = SPX_getScanData(ByVal instname, data(0))
'close
spetro = SPX_close(ByVal instname)
CWVisa1.Close  'actually stops crashing for the second time.

'If PulseBox.Value = 1 Then
'If BeforePulse = False Then
'Pulse here and then take a different spectrum triggers the step on positive edge
'set a step and allow at least 0.25 s to equilibriate
'CWDIO1.SingleWrite 1
'tstarto = Timer
'Do
'Loop Until Timer - tstarto > 0.25
'CWDIO1.SingleWrite 0
'Else
'End If
'End If


For i = 0 To 2999
    If wavelengthdata(i) - 490 > 0 Then
        If counter = 0 Then
        Well_125_490nm = data(i)
        Else
        Wavelength490nm = data(i)
        End If
    GoTo Got490
    End If
Next i

Got490:

For i = 0 To 2999
    If wavelengthdata(i) - 600 > 0 Then
    For j = i - 10 To i + 10
    Wavelength600nm = Wavelength600nm + data(j)
    Next j
    Wavelength600nm = Wavelength600nm / 21
    GoTo Got600
    End If
Next i

Got600:
```

```
'----
For i = 0 To 2999
    datatot(i) = datatot(i) + data(i)
Next i

CWGraph1.PlotXvsY wavelengthdata, datatot
Next k

If PulseBox.Value = 1 Then
If BeforePulse = False Then
'Pulse here and then take a different spectrum triggers the step on positive edge
'set a step and allow at least 0.25 s to equilibriate
CWDIO1.SingleWrite 1
tstarto = Timer
Do
Loop Until Timer - tstarto > 0.25
CWDIO1.SingleWrite 0
Else
End If
End If


For i = 0 To 2999
    data(i) = datatot(i) / CWSlide2.Value
Next i



'----

s = -1
    Total(X, Y) = Empty
    Ave(X, Y) = Empty
    Max(X, Y) = Empty
    Maxp(X, Y) = Empty

    Do
        s = s + 1


        Total(X, Y) = Total(X, Y) + data(s) 'puts our spectro data into the system

        If Abs(data(s)) > Maxp(X, Y) Then
        Max(X, Y) = data(s) '1 '(ScaledData(s))
        Maxp(X, Y) = data(s) '1 'Abs(ScaledData(s))
        End If
    Loop Until s >= 2999 'NumEditNoPoints.Value - 1



        Ave(X, Y) = (Total(X, Y) / 2999) ' / NumEditGain.Value
        Avep(X, Y) = Abs((Total(X, Y) / 2999))
        Max(X, Y) = Max(X, Y) ' / NumEditGain.Value
        KInt = 0

        Dim Xv()
        Dim Yv()
        'Dim Zv()
        ReDim Xv(XRange)
        ReDim Yv(YRange)
        'ReDim Zv(zRange)
        For w = 0 To (XRange - 1)
        'Xv(w) = XYZController.Combo1.Text + (w * XYZController.NumEditResolution(0))
        Xv(w) = OriginX + (w * XYZController.NumEditResolution(0))
        Next w

        For w = 0 To (YRange - 1)
        'Yv(w) = XYZController.Combo3.Text + (w * XYZController.NumEditResolution(1))
        Yv(w) = OriginY + (w * XYZController.NumEditResolution(1))
        Next w

        'For w = 0 To (zRange - 1)
        'Zv(w) = XYZController.Combo6.Text + (w * XYZController.NumEditResolution(2))
        'Next w




        Label2.Caption = Format$(Ave(X, Y), "0.00E-00")
        'Label6.Caption = Format$(Max(X, Y), "0.00E-00")
        If Avep(X, Y) > MaxAve Then
        MaxAve = 1.01 * Avep(X, Y)
        End If

        If Avep(X, Y) < MinAve Then
        MinAve = Avep(X, Y)
        End If

        If Maxp(X, Y) > MaxMax Then
        MaxMax = 1.01 * Maxp(X, Y)
        End If

        If Maxp(X, Y) < MinMax Then
        MinMax = Maxp(X, Y)
        End If



If BeforePulse = True Then
Ave_bef(X, Y) = Ave(X, Y)
Avep_bef(X, Y) = Avep(X, Y)
Total_bef(X, Y) = Total(X, Y)
Max_bef(X, Y) = Max(X, Y)
Maxp_bef(X, Y) = Maxp(X, Y)
W600_bef(X, Y) = Wavelength600nm
Else
Ave_aft(X, Y) = Ave(X, Y)
Avep_aft(X, Y) = Avep(X, Y)
Total_aft(X, Y) = Total(X, Y)
Max_aft(X, Y) = Max(X, Y)
Maxp_aft(X, Y) = Maxp(X, Y)
W600_aft(X, Y) = Wavelength600nm
W600_diff(X, Y) = W600_bef(X, Y) - W600_aft(X, Y)
End If
    If PulseBox.Value = 1 Then
        If BeforePulse = True Then
        CWGraph3D1.Plot3DSurface Xv, Yv, Max_bef
        Refresh
        Else
        CWGraph3D1.Plot3DSurface Xv, Yv, Max_aft
        CWGraph3D2.Plot3DSurface Xv, Yv, W600_diff
        Refresh
        End If
    Else
        CWGraph3D1.Plot3DSurface Xv, Yv, Max
    End If
```

```
'--== Bilayer-Detection ==--
If Check1.Value = 1 Then
    If counter > 0 Then
    Abs490nm(X, Y) = (Log(Well_125_490nm / Wavelength490nm)) - Wavelength600nm
    Else
    Abs490nm(X, Y) = 0
    End If
    CWGraph3D2.Plot3DSurface Xv, Yv, Abs490nm
    Refresh
End If







  Text1 = CStr(Max_aft(X, Y))
  Text2 = CStr(Max_bef(X, Y))


  Acquisition.Refresh

If PulseBox.Value = 1 Then
 If BeforePulse = True Then
       Open FileName & "_Before_Pulse.dat" For Output As #2
       Open FileName & "_Before_Pulse" & counter & ".dat" For Output As #3
 Else
       Open FileName & "_After_Pulse.dat" For Output As #2
       Open FileName & "_After_Pulse" & counter & ".dat" For Output As #3
 End If
Else
       Open FileName & ".dat" For Output As #2
       Open FileName & counter & ".dat" For Output As #3
End If


    ns = -1

   '===Round Up/Down Co-ordinate values===
 If XYZController.NumEditPosition(0) - Int(XYZController.NumEditPosition(0)) > 0.5 Then
 XYZController.NumEditPosition(0) = Int(XYZController.NumEditPosition(0)) + 1
 Else
 XYZController.NumEditPosition(0) = Int(XYZController.NumEditPosition(0))
 End If

 If XYZController.NumEditPosition(1) - Int(XYZController.NumEditPosition(1)) > 0.5 Then
 XYZController.NumEditPosition(1) = Int(XYZController.NumEditPosition(1)) + 1
 Else
 XYZController.NumEditPosition(1) = Int(XYZController.NumEditPosition(1))
 End If



    For b = xmin To xmax
     For a = ymin To ymax
     If Ave(b, a) <> Empty Then
     'Print #2, Yv(b) + Int(XYZController.NumEditPosition(1)) -
(Val(XYZController.Combo4.Text) * Val(XYZController.NumEditResolution(1))); ","; Xv(A) +
(XYZController.NumEditPosition(0)) - (Val(XYZController.Combo2.Text) *
Val(XYZController.NumEditResolution(0))); ","; Ave(A, b); ","; Max(A, b)
     Print #2, Yv(a); ","; Xv(b); ","; Ave(b, a); ","; Max(b, a)
     '","; W450NM(A, b); ","; W475NM(A, b); ","; W500NM(A, b); ","; W525NM(A, b); ",";
W550NM(A, b); ","; W575NM(A, b); ","; W600NM(A, b); ","; W625NM(A, b); ","; W650NM(A,
b); ","; W675NM(A, b); ","; W700NM(A, b)
     End If
     Next a
    Next b
    'Print #2, Wavelength600nm
    Close #2

     Print #3, XYZController.NumEditPosition(2); ","; XYZController.NumEditPosition(1);
","; XYZController.NumEditPosition(0)
     For i = 0 To 2999
       Print #3, wavelengthdata(i); ","; data(i)
       Next i

       Close #3



'Repeat
If PulseBox.Value = 1 Then
If BeforePulse = True Then
BeforePulse = False
Label6.Caption = "After"
GoTo repeatspec:
Else
BeforePulse = True
Label6.Caption = "Before"
End If


XYZController.objCommPort2.Output = "/1gS3I2A3000S3I2A0G6R" + Chr(13)

tstarto = Timer
Do
Loop Until Timer - tstarto > 20
End If



counter = counter + 1

'Call XYZController.cmdTTL_Click
XYZController.NumEditCounter = counter
Label1.Caption = XYZController.NumEditPositions.Value - (counter + 1)
'If Label1.Caption = 0 Then Command1.Visible = True
Call XYZController.cmdScan_Click    '<-- Stack space error

End Sub

Private Sub Command1_Click()
XYZController.Visible = True
End Sub


Static Sub Form_Load()
StackList.Show
SaveStart = 0
Combo1.AddItem "X vs Y"
Combo1.AddItem "X vs Z"
Combo1.AddItem "Y vs Z"

BeforePulse = True
Label6.Caption = "Before"
CWAIPoint1.Device = 1
ChannelString = "0,1"
```

```
CWAIPoint1.Channels.Add ChannelString
CWDIO1.SingleWrite 0
End Sub

--- frmSplash---


Option Explicit

Private Sub Form_KeyPress(KeyAscii As Integer)
    Unload Me
    Main.Show
End Sub


Private Sub Frame1_Click()
    Unload Me
    Main.Show
End Sub

---Main--


Private Sub MDIForm_Load()
Acquisition.Show
XYZController.Show
End Sub

---Stacklist---

---XYZ Controller

 'TeckmoDemo Program - Visual Basic 6.0
' Last updated Feb 15, 2001
' For the most recent updates and product information please check our website
' Zaber Technologies Inc - www.zaber.com

' global variables
Dim iCommand(1 To 6) As Integer     ' 6 byte instruction
Dim t_now As Double                 ' current timer value
Dim t_last As Double                ' timer value when last byte was received
Dim bytecount As Integer            ' number of bytes in receive buffer
Dim bReply(1 To 6) As Byte          ' 6 byte reply from Teckmo chain
Dim dReplyData As Double            ' value of 4 data bytes in reply
Dim dReplyDataUm As Double          ' as above but in units of um
Dim Zero(0 To 2)
Dim Coords(0 To 2, 0 To 99999)
Dim XCoord As String
Dim YCoord As String
Dim ZCoord As String
Dim No
Dim Scan
Dim nn
Dim N
Dim NewRelPosition(0 To 2)
Dim countpoints
'***Edit***
Dim SaveStart As Integer


' system settings to restore before closing com port
Dim OldHandShake As Integer
Dim OldSettings As String
Dim OldMode As Integer
Dim OldInputLen As Integer
```

```
Static Sub cmdGenerateCoords_Click()
'For c = 0 To (((Val(Combo2.Text)) - (Val(Combo1.Text))) + 1) * (((Val(Combo4.Text)) -
(Val(Combo3.Text))) + 1)
For a = 0 To 2
For b = 0 To 99999
Coords(a, b) = Empty
Next b
Next a



No = 0
For f = Val(Combo5.Text) To Val(Combo6.Text)
For d = Val(Combo3.Text) To Val(Combo4.Text)
For e = Val(Combo1.Text) To Val(Combo2.Text)

Coords(0, No) = e
Coords(1, No) = d
Coords(2, No) = f
List1.AddItem Coords(0, No) & "," & Coords(1, No) & "," & Coords(2, No)
No = No + 1
Next e
Next d
Next f
NumEditPositions.Value = No + 1
End Sub

Static Sub cmdHome_Click()
' the "send instruction" button was clicked
    If (objCommPort.PortOpen = True) Then
        ' if the com port has already been openned
        ' convert the user data and send the instruction
        'Call Position_to_bytes(Val(txtData.Text))
        ' put all entered values in format to send to com port
        For Y = 1 To 3
        'Call Position_to_bytes(NumEditNewPosition(y - 1).Value)
        'unit = Chr(comboUnitNum.Text)
        unit = Chr(0)
        inst = Chr(1)

        'databytes = Chr$(iCommand(3)) + Chr$(iCommand(4)) + Chr$(iCommand(5)) +
Chr$(iCommand(6))
        objCommPort.Output = unit + inst
        Next Y
    Else
        ' the send button was pressed before the com port was open
        MsgBox "You must open the com port before sending instructions."
    End If
End Sub

Static Sub cmdMoveRelative_Click()
 ' the "send instruction" button was clicked
    If (objCommPort.PortOpen = True) Then
        ' if the com port has already been openned
        ' convert the user data and send the instruction
        'Call Position_to_bytes(Val(txtData.Text))
        ' put all entered values in format to send to com port
        For Y = 1 To 3
        Call Position_to_bytes(NumEditNewPosition(Y - 1).Value)
        'unit = Chr(comboUnitNum.Text)
        unit = Chr(Y)
        'inst = Chr(comboCommandNum.Text)
        inst = Chr(21)
        databytes = Chr$(iCommand(3)) + Chr$(iCommand(4)) + Chr$(iCommand(5)) +
Chr$(iCommand(6))
        objCommPort.Output = unit + inst + databytes
        Next Y
    Else
        ' the send button was pressed before the com port was open
        MsgBox "You must open the com port before sending instructions."
    End If
End Sub

Static Sub cmdReturn_Click(Index As Integer)
NumEditNewPosition(Index).Value = 0
  ' the "send instruction" button was clicked
    If (objCommPort.PortOpen = True) Then
        ' if the com port has already been openned
        ' convert the user data and send the instruction
        'Call Position_to_bytes(Val(txtData.Text))
        ' put all entered values in format to send to com port

        Call Position_to_bytes(Zero(Index) + NumEditNewPosition(Index).Value)
        'unit = Chr(comboUnitNum.Text)
        unit = Chr(Index + 1)
        'inst = Chr(comboCommandNum.Text)
        inst = Chr(20)
        databytes = Chr$(iCommand(3)) + Chr$(iCommand(4)) + Chr$(iCommand(5)) +
Chr$(iCommand(6))
        objCommPort.Output = unit + inst + databytes

    Else
        ' the send button was pressed before the com port was open
        MsgBox "You must open the com port before sending instructions."
    End If
End Sub

Static Sub cmdReturntoOrigin_Click()
For b = 0 To 2
NumEditNewPosition(b).Value = 0
Next b
cmdSend_Click
End Sub

Static Sub cmdSet_Click(Index As Integer)
Zero(Index) = NumEditPosition(Index)
NumEditRelPosition(Index) = NumEditPosition(Index) - Zero(Index)
End Sub

Static Sub cmdSetZero_Click()
For X = 0 To 2
Zero(X) = NumEditPosition(X)
NumEditRelPosition(X) = NumEditPosition(X) - Zero(X)
Next X

file = Text1
    Open Text1 For Output As #2
    Print #2, NumEditPosition(0)
    Print #2, NumEditPosition(1)
    Print #2, NumEditPosition(2)
    Close #2

End Sub
```

```
Static Sub cmdLoad_Click()
filebox.CancelError = True
    'On Error GoTo ErrHandle


   filebox.Action = 1

    Open filebox.FileName For Input As #1

     No = -1

    Do
        No = No + 1
        Input #1, XCoord$
        Coords(0, No) = Val(XCoord$)
        Input #1, YCoord$
        Coords(1, No) = Val(YCoord$)
        Input #1, ZCoord$
        Coords(2, No) = Val(ZCoord$)
    List1.AddItem (Coords(0, No) & "," & Coords(1, No) & "," & Coords(2, No))
    Loop Until EOF(1)

    Close #1

    NumEditPositions.Value = No + 1

End Sub

Static Sub cmdScan_Click()

StackList.Text1.Text = StackList.Text1.Text + "XYZController" + Chr(13) + Chr(10)
 XYZController.Hide
 '***Edit***
SaveStart = SaveStart + 1
StackList.Text1.Text = StackList.Text1.Text + CStr(SaveStart) + Chr(13) + Chr(10)
Acquisition.NumEditAcquired = 0
nn = 0
Scan = 1
countpoints = countpoints + 1
If (countpoints - 1) >= No Then
countpoints = 0
NumEditCounter = 0
Scan = 0
Exit Sub
End If


Do

'Label3.Caption = countpoints & "," & nn
NumEditNewPosition(nn).Value = (Coords(nn, countpoints - 1) *
NumEditResolution(nn).Value) - NumEditRelPosition(nn).Value
 NewRelPosition(nn) = NumEditRelPosition(nn).Value + NumEditNewPosition(nn).Value
 ' the "send instruction" button was clicked
    If (objCommPort.PortOpen = True) Then
        ' if the com port has already been openned
        ' convert the user data and send the instruction
        'Call Position_to_bytes(Val(txtData.Text))
        ' put all entered values in format to send to com port
        'For y = 1 To 3
        Call Position_to_bytes(NumEditNewPosition(nn).Value)
        'unit = Chr(comboUnitNum.Text)
        unit = Chr(nn + 1)
        'inst = Chr(comboCommandNum.Text)
        inst = Chr(21)
        databytes = Chr$(iCommand(3)) + Chr$(iCommand(4)) + Chr$(iCommand(5)) +
Chr$(iCommand(6))
        objCommPort.Output = unit + inst + databytes
        'Next y
    Else
        ' the send button was pressed before the com port was open
        MsgBox "You must open the com port before sending instructions."
    End If
    'Call Delay(1)

    ' wait for 1 seconds
    t = Timer
    While (Timer - t) < 0.5
    Wend

  'Call objCommPort_OnComm

nn = nn + 1
Loop Until nn = 3
StackList.Text1.Text = StackList.Text1.Text + "***Scan Complete***" + Chr(13) + Chr(10)
End Sub

Static Sub cmdTransmit_Click()
' the "send instruction" button was clicked
    If (objCommPort.PortOpen = True) Then
        ' if the com port has already been openned
        ' convert the user data and send the instruction
        Call Position_to_bytes(Val(txtData.Text))
        ' put all entered values in format to send to com port

        unit = Chr(comboUnitNum.Text)

        inst = Chr(comboCommandNum.Text)

        databytes = Chr$(iCommand(3)) + Chr$(iCommand(4)) + Chr$(iCommand(5)) +
Chr$(iCommand(6))
        objCommPort.Output = unit + inst + databytes

    Else
        ' the send button was pressed before the com port was open
        MsgBox "You must open the com port before sending instructions."
    End If
End Sub

Static Sub Command1_Click()
    file = Text1
    Open Text1 For Input As #2
    Input #2, Xorig$
    NumEditNewPosition(0) = Val(Xorig$)
    Input #2, Yorig$
    NumEditNewPosition(1) = Val(Yorig$)
    Input #2, zorig$
    NumEditNewPosition(2) = Val(zorig$)
    Close #2

End Sub

Private Sub Command2_Click()
objCommPort.Output = Chr$(1) + Chr$(20) + Chr$(1) + Chr$(1) + Chr$(0) + Chr$(0)
End Sub

Private Sub Command3_Click()
 ' either open the com port or close it depending on button caption
    If Check1.Value = 1 Then
        With objCommPort2
```

```
            ' Set com port number based on user selection
            .CommPort = 5
            ' try to open the port
            .PortOpen = True
            ' Save current port settings
            OldHandShake = .Handshaking
            OldSettings = .Settings
            OldMode = .InputMode
            OldInputLen = .InputLen
            ' Set the new settings for Teckmo communications
            .Handshaking = comNone
            .Settings = "9600,N,8,1"
            .InputMode = comInputModeText
            .InputLen = 1
            .RThreshold = 1
            .Output = "/1ZR" + Chr(13)
            End With
        ' disable the port select area so the user can't try to change
        ' to a different com port while the current one is open
        'comboPortNum.Enabled = False
        ' change button caption to close com port
        'cmdOpenCom.Caption = "Close com port"
        ' display a com port status message
        'lblStatus.Caption = "Com " & comboPortNum.Text & " is the current port."

        ' Note there is no need to make sure the receive buffer is empty
        ' since RTheshold is set to 1 above. This means whenever there is
        ' at least 1 byte in the receive buffer, the objCommPort_OnComm()
        ' subroutine will be triggered. this routine does error checking
        ' to remove stray bytes that are not part of a 6 byte reply

        ' Renumber all units
        'Call Renumber_All
    Else
        ' the "Close com port" button was clicked
        ' return comm port settings to the previous system values
        ' then close the port
        With objCommPort2
            .Handshaking = OldHandShake
            .Settings = OldSettings
            .InputMode = OldInputMode
            .InputLen = OldInputLen
'            .PortOpen = False
        End With
        ' enable open com port button
        'cmdOpenCom.Caption = "Open com port"
        'comboPortNum.Enabled = True
        ' change the status message
        'lblStatus.Caption = "All com ports are currently closed."
    End If
    ' switch focus back to send button so enter key causes send
    'cmdSend.SetFocus
End Sub

Static Sub Form_Load()
    ' Initialize combo box choices
    With comboPortNum
        .AddItem 1
        .AddItem 2
        .AddItem 3
        .AddItem 4
    End With
    With comboUnitNum
        For i = 0 To 254
        .AddItem i
        Next i
    End With
    With comboCommandNum
        For i = 0 To 63
        .AddItem i
        Next i
    End With
    With Combo1
        For i = 0 To 50
        .AddItem i
        Next i
    End With
     With Combo2
        For i = 0 To 50
        .AddItem i
        Next i
    End With
     With Combo3
        For i = 0 To 50
        .AddItem i
        Next i
    End With
     With Combo4
        For i = 0 To 50
        .AddItem i
        Next i
    End With

    With Combo5
        For i = 0 To 50
        .AddItem i
        Next i
    End With
     With Combo6
        For i = 0 To 50
        .AddItem i
        Next i
    End With
    ' disable close port button since the com port is not open yet
    cmdOpenCom.Enabled = True
    cmdOpenCom.Caption = "Open com port and renumber"
    comboPortNum.Enabled = True
    ' display com port status message
    lblStatus.Caption = "All com ports are currently closed."
    ' clear the text Display area and add headings
    lstDisplay.Clear
    txtHeadings.Text = "Unit#       Command#       Data (bits)      Data (um)"
    ' set default input mode to um units
     'DIO.Ports(0).Assignment = cwdioOutput

     'DIO.SingleWrite 0



End Sub

Static Sub cmdOpenCom_Click()
    ' either open the com port or close it depending on button caption
    If cmdOpenCom.Caption = "Open com port and renumber" Then
        With objCommPort
            ' Set com port number based on user selection
            .CommPort = Val(comboPortNum.Text)
            ' try to open the port
            .PortOpen = True
            ' Save current port settings
            OldHandShake = .Handshaking
            OldSettings = .Settings
            OldMode = .InputMode
            OldInputLen = .InputLen
            ' Set the new settings for Teckmo communications
            .Handshaking = comNone
            .Settings = "9600,N,8,1"
            .InputMode = comInputModeText
            .InputLen = 1
            .RThreshold = 1
        End With
        ' disable the port select area so the user can't try to change
        ' to a different com port while the current one is open
        comboPortNum.Enabled = False
        ' change button caption to close com port
        cmdOpenCom.Caption = "Close com port"
        ' display a com port status message
        lblStatus.Caption = "Com " & comboPortNum.Text & " is the current port."

        ' Note there is no need to make sure the receive buffer is empty
        ' since RTheshold is set to 1 above. This means whenever there is
        ' at least 1 byte in the receive buffer, the objCommPort_OnComm()
        ' subroutine will be triggered. this routine does error checking
        ' to remove stray bytes that are not part of a 6 byte reply

        ' Renumber all units
        If Check1.Value = 1 Then
        With objCommPort2
            ' Set com port number based on user selection
            .CommPort = 5
            ' try to open the port
            .PortOpen = True
            ' Save current port settings
            OldHandShake = .Handshaking
            OldSettings = .Settings
            OldMode = .InputMode
            OldInputLen = .InputLen
            ' Set the new settings for Teckmo communications
            .Handshaking = comNone
            .Settings = "9600,N,8,1"
            .InputMode = comInputModeText
            .InputLen = 1
            .RThreshold = 1
            .Output = "/1ZR" + Chr(13)
            End With
        End If
        ' disable the port select area so the user can't try to change
        ' to a different com port while the current one is open
        'comboPortNum.Enabled = False
        ' change button caption to close com port
        'cmdOpenCom.Caption = "Close com port"
        ' display a com port status message
        'lblStatus.Caption = "Com " & comboPortNum.Text & " is the current port."

        ' Note there is no need to make sure the receive buffer is empty
        ' since RTheshold is set to 1 above. This means whenever there is
        ' at least 1 byte in the receive buffer, the objCommPort_OnComm()
        ' subroutine will be triggered. this routine does error checking
        ' to remove stray bytes that are not part of a 6 byte reply

        ' Renumber all units
        'Call Renumber_All

        Call Renumber_All
    Else
        ' the "Close com port" button was clicked
        ' return comm port settings to the previous system values
```

```
        ' then close the port
        With objCommPort
            .Handshaking = OldHandShake
            .Settings = OldSettings
            .InputMode = OldInputMode
            .InputLen = OldInputLen
            .PortOpen = False
        End With
        ' enable open com port button
        cmdOpenCom.Caption = "Open com port and renumber"
        comboPortNum.Enabled = True
        ' change the status message
        lblStatus.Caption = "All com ports are currently closed."
    End If
    ' switch focus back to send button so enter key causes send

 ' either open the com port or close it depending on button caption

    ' switch focus back to send button so enter key causes send
    'cmdSend.SetFocus




    cmdSend.SetFocus


End Sub

Static Sub cmdClearScreen_Click()
    lstDisplay.Clear
    ' switch focus back to send button so enter key causes send
    cmdSend.SetFocus
End Sub

Static Sub Renumber_All()
    ' renumber all units
    objCommPort.Output = Chr$(0) + Chr$(2) + Chr$(0) + Chr$(0) + Chr$(0) + Chr$(0)
    ' wait while renumbering
    'Call Delay(1.5)
    ' wait for d seconds
    t = Timer
    While (Timer - t) < 0.5
    Wend
End Sub

Static Sub Form_KeyPress(KeyAscii As Integer)
    ' if a key is pressed check if it is the enter key. if so, send instruction
    If KeyAscii = 13 Then Call cmdSend_Click
End Sub
Static Sub cmdSend_Click()
    ' the "send instruction" button was clicked
    If (objCommPort.PortOpen = True) Then
        ' if the com port has already been openned
        ' convert the user data and send the instruction
        'Call Position_to_bytes(Val(txtData.Text))
        ' put all entered values in format to send to com port
        For Y = 1 To 3
        Call Position_to_bytes(Zero(Y - 1) + NumEditNewPosition(Y - 1).Value)
        'unit = Chr(comboUnitNum.Text)
        unit = Chr(Y)
        'inst = Chr(comboCommandNum.Text)
        inst = Chr(20)
        databytes = Chr$(iCommand(3)) + Chr$(iCommand(4)) + Chr$(iCommand(5)) +
Chr$(iCommand(6))
```

```
        objCommPort.Output = unit + inst + databytes
        Next Y
    Else
        ' the send button was pressed before the com port was open
        MsgBox "You must open the com port before sending instructions."
    End If
End Sub
Static Sub objCommPort_OnComm()
StackList.Text1.Text = StackList.Text1.Text + "objCommPort_OnComm" + Chr(13) + Chr(10)
    ' This subroutine is triggered whenever there is a com event
    ' The comEvReceive event occurs whenever the number of bytes
    ' in the receive buffer reaches that specified by objCommPort.RTheshold
    ' (which is set to 1 for our purposes so that whenever there is a byte
    ' in the receive buffer, this code is executed)
    If objCommPort.CommEvent = comEvReceive Then
        ' The com event was triggered because there is a byte in the
        ' receive buffer.
        ' Determine elapsed time since the last byte was received
        t_last = t_now
        t_now = Timer
        If (t_now - t_last) > 0.1 Then
            ' 100 ms or more has passed since the previous byte
            ' was received so ignore all bytes except the current one
            ' This is necessary to trap erroneous bytes due to noise
            ' or other sources. Using this method, if an erroneous byte
            ' occurs it may corrupt the reply but it should not upset
            ' further communications
            bytecount = 0
        End If
        ' count the byte that was just received
        bytecount = bytecount + 1
        ' read the byte into the reply array


        bReply(bytecount) = Asc(objCommPort.Input)


        ' if this byte is the sixth then display the reply
        If bytecount = 6 Then
            ' calculate the value of the 4 byte data in the reply
            Call Bytes_to_position
            ' display the reply
            Call Display_reply
            ' reset the bytecount to zero
            bytecount = 0
        End If
    End If

End Sub

Static Sub Display_reply()
StackList.Text1.Text = StackList.Text1.Text + "Display_Reply" + Chr(13) + Chr(10)
    ' make sure there are enough lines in the display list
    ' to display this reply on the appropriate line number
    While lstDisplay.ListCount <= bReply(1)
    lstDisplay.AddItem ""
    Wend
    ' display text with appropriate column spacing
    ' determine text lengths
    L1 = Len(Str(bReply(1)))
    L2 = Len(Str(bReply(2)))
    L3 = Len(Str(dReplyData))
    ' determine needed spacing
    S1 = Left$("                    ", 13 - L1)
    S2 = Left$("                    ", 16 - L2)
    S3 = Left$("                    ", 18 - L3)
    ' determine the complete line of display text
    DisplayString = bReply(1) & S1 & bReply(2) & S2 & dReplyData & S3 & dReplyDataUm
    NumEditPosition(bReply(1) - 1) = dReplyDataUm       '<----Error Here
    NumEditRelPosition(bReply(1) - 1) = NumEditPosition(bReply(1) - 1) - Zero(bReply(1)
- 1)
    If bReply(2) = "255" Then MsgBox "Error! Data out of range.", vbCritical + vbOKOnly,
"Error"
    'MsgBox "Error.  Unit " & bReply(1) & " data out of range.", vbCritical + vbOKOnly,
"Error"
    ' display the text on the appropriate line number
    lstDisplay.List(bReply(1)) = DisplayString

    If Round(NumEditRelPosition(0), 0) = NewRelPosition(0) And
Round(NumEditRelPosition(1), 0) = NewRelPosition(1) And Round(NumEditRelPosition(2), 0)
= NewRelPosition(2) And Scan = 1 Then
    '***Edit***
    'If SaveStart = 1 Then
     Call Acquisition.cmdStart_Click
    ' End If
    'If Acquisition.NumEditAcquired = 1 Then Call cmdScan_Click
    End If
End Sub

Static Sub Bytes_to_position()
    ' convert the 4 received data bytes to a single value
    If bReply(6) > 127 Then         ' negative
    dReplyData = -4294967296# + 256# * 256# * 256# * bReply(6) + 256# * 256# * bReply(5)
+ 256# * bReply(4) + bReply(3)
    Else
    dReplyData = 256# * 256# * 256# * bReply(6) + 256# * 256# * bReply(5) + 256# *
bReply(4) + bReply(3)
    End If

    ' also determine the value in units of um
    ' ReplyData in um = ReplyData in usteps * 6.35 um/step / 64 usteps/step
    dReplyDataUm = Round(dReplyData * 6.35 / 64, 2)
End Sub

Static Sub Position_to_bytes(ByVal Position As Double)
    ' convert the user input data into 4 data bytes to send
    If (optMicrons.Value = True) Then
        ' convert data from microns to microsteps before sending it
        ' position [usteps] = position [um] * 64 [usteps/step] / 6.35 [um/step]
        Position = Round(Position * 64 / 6.35, 0)
        ' since the position in microsteps must be a whole number while
        ' the position in microns is not, there will be an error of up to
        ' +/- 0.5 microsteps (.05 um) between the requested value in microns
        ' and the actual position the program instructs the unit to move to
    End If
    If Position < 0 Then
        'negative value entered by user
        Position = 4294967296# + Position
    End If
        iCommand(6) = Int(Position / (256# * 256# * 256#))
        Position = Position - 256# * 256# * 256# * iCommand(6)
        iCommand(5) = Int(Position / (256# * 256#))
        Position = Position - 256# * 256# * iCommand(5)
        iCommand(4) = Int(Position / 256#)
        Position = Position - 256# * iCommand(4)
        iCommand(3) = Int(Position)
End Sub


--Module 1---
' ---------------------------------------------------------------------
```

```
'
'       Thorlabs SPx-USB - VB Header file for SP1-USB / SP2-USB Spectrometer instr.
driver
'  Do not modify the contents of this file.
'       Copyright:      Copyright(c) 2007, Thorlabs GmbH (www.thorlabs.com)
'
' -------------------------------------------------------------------------
'
'       Date:           Aug-21-2007
'       Software-Nr:   09.167.200
'       Version:        2.0
'       Changelog:              see 'readme.rtf'
'
' -------------------------------------------------------------------------
'
'       Disclaimer:
'
'       This library is free software; you can redistribute it and/or
'       modify it under the terms of the GNU Lesser General Public
'       License as published by the Free Software Foundation; either
'       version 2.1 of the License, or (at your option) any later version.
'
'       This library is distributed in the hope that it will be useful,
'       but WITHOUT ANY WARRANTY; without even the implied warranty of
'       MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
'       Lesser General Public License for more details.
'
'       You should have received a copy of the GNU Lesser General Public
'       License along with this library; if not, write to the Free Software
'       Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
'
' -------------------------------------------------------------------------


' -------------------------------------------------------------------------
' Functions
' -------------------------------------------------------------------------

' - Init/Close ---
Declare Function SPX_init& Lib "SPX_Drv_32.dll" (ByVal x1$, ByVal x2&, x3&)
Declare Function SPX_close& Lib "SPX_Drv_32.dll" (ByVal x1&)



' - Configuration Functions ---
Declare Function SPX_setIntTime& Lib "SPX_Drv_32.dll" (ByVal x1&, ByVal x2#)
Declare Function SPX_getIntTime& Lib "SPX_Drv_32.dll" (ByVal x1&, x2#)


' - Action/Status Functions ---
Declare Function SPX_startScan& Lib "SPX_Drv_32.dll" (ByVal x1&)
Declare Function SPX_startScanExtTrg& Lib "SPX_Drv_32.dll" (ByVal x1&)
Declare Function SPX_startScanCont& Lib "SPX_Drv_32.dll" (ByVal x1&)
Declare Function SPX_startScanContExtTrg& Lib "SPX_Drv_32.dll" (ByVal x1&)
Declare Function SPX_getDeviceStatus& Lib "SPX_Drv_32.dll" (ByVal x1&, ByVal x2$)


' - Data Functions ---
Declare Function SPX_getScanData& Lib "SPX_Drv_32.dll" (ByVal x1&, x2#)
Declare Function SPX_getRawScanData& Lib "SPX_Drv_32.dll" (ByVal x1&, x2%)
Declare Function SPX_getWavelengthData& Lib "SPX_Drv_32.dll" (ByVal x1&, x2#, x3#, x4#)


' - Utility Functions ---
Declare Function SPX_reset& Lib "SPX_Drv_32.dll" (ByVal x1&)
Declare Function SPX_errorMessage& Lib "SPX_Drv_32.dll" (ByVal x1&, ByVal x2&, ByVal
x3$)
Declare Function SPX_identificationQuery& Lib "SPX_Drv_32.dll" (ByVal x1&, ByVal x2$,
ByVal x3$, ByVal x4$, ByVal x5$)
Declare Function SPX_revisionQuery& Lib "SPX_Drv_32.dll" (ByVal x1&, ByVal x2$, ByVal
x3$)
Declare Function SPX_setUserText& Lib "SPX_Drv_32.dll" (ByVal x1&, ByVal x2$)
Declare Function SPX_getUserText& Lib "SPX_Drv_32.dll" (ByVal x1&, ByVal x2$)



' ---------------------------------------------------------------------------
' Definitions
' ---------------------------------------------------------------------------

' - Driver Error Codes ---
Global Const VI_WARN_SPX_DATA_NOT_READY = &H3FFC0901
Global Const VI_ERROR_READ_INCOMPLETE = &HBFFC0901


' - Buffers ---
Global Const SPX_BUFFER_SIZE = 256                          ' General buffer size
Global Const SPX_ERR_DESCR_BUFFER_SIZE = 512                    ' Buffer size for error
messages
Global Const SPX_TEXT_BUFFER_SIZE = 256                                         ' Buffer
size for texts from the LC1
Global Const SPX_NUM_PIXELS = 3000                                              ' this
is the number of effective pixels of CCD
Global Const SPX_MAX_USER_NAME_SIZE = 31                                ' this is max
number of characters for a user name

' - VISA strings ---
Global Const SP1_VI_FIND_RSC_PATTERN = "USB?*?{VI_ATTR_MANF_ID==0x1313 &&
VI_ATTR_MODEL_CODE==0x0111}"
Global Const SP2_VI_FIND_RSC_PATTERN = "USB?*?{VI_ATTR_MANF_ID==0x1313 &&
VI_ATTR_MODEL_CODE==0x0112}"
Global Const SPX_VI_FIND_RSC_PATTERN = "USB?*?{VI_ATTR_MANF_ID==0x1313 &&
((VI_ATTR_MODEL_CODE==0x0111) || (VI_ATTR_MODEL_CODE==0x0112))}"


' - Communication timeout ---
Global Const SPX_TIMEOUT_MIN = 1000
Global Const SPX_TIMEOUT_MAX = 60000

' - LC1 specific constants ---
Global Const SPX_MAX_INT_TIME = 0.2
' 200ms is the maximum integration time
Global Const SPX_MIN_INT_TIME = 0.000001                                        ' 1us
is the minimum integration time

' - scan states - these are the states reported by LC1_getDeviceStatus ---
Global Const SPX_SCAN_STATE_IDLE = &H0&
' LC waits for new scan to execute
Global Const SPX_SCAN_STATE_TRANSFER = &H20&                            ' scan is done,
waiting for data transfer to be finished
Global Const SPX_SCAN_STATE_IN_PROGRES = &H40&                  ' scan is in progres,
this message should never be get
Global Const SPX_SCAN_STATE_WAIT_TRIGGER = &H80&                ' same as IDLE except
that external trigger is armed
Global Const SPX_SCAN_STATE_UNKNOWN = &HFF&                             ' default value


' ---------------------------------------------------------------------------
' End of file
' ---------------------------------------------------------------------------
```

A4: Scanning and Potential Step Acquisition

```
---XYZ Controller ---

 'TeckmoDemo Program - Visual Basic 6.0
' Last updated Feb 15, 2001
' For the most recent updates and product information please check our website
' Zaber Technologies Inc - www.zaber.com

' global variables
Dim iCommand(1 To 6) As Integer     ' 6 byte instruction
Dim t_now As Double                 ' current timer value
Dim t_last As Double                ' timer value when last byte was received
Dim bytecount As Integer            ' number of bytes in receive buffer
Dim bReply(1 To 6) As Byte          ' 6 byte reply from Teckmo chain
Dim dReplyData As Double            ' value of 4 data bytes in reply
Dim dReplyDataUm As Double          ' as above but in units of um
Dim Zero(0 To 2)
Dim Coords(0 To 2, 0 To 99999)
Dim XCoord As String
Dim YCoord As String
Dim ZCoord As String
Dim No
Dim Scan
Dim nn
Dim N
Dim NewRelPosition(0 To 2)
Dim countpoints
'***Edit***
Dim SaveStart As Integer


' system settings to restore before closing com port
Dim OldHandShake As Integer
Dim OldSettings As String
Dim OldMode As Integer
Dim OldInputLen As Integer




Static Sub cmdGenerateCoords_Click()
'For c = 0 To (((Val(Combo2.Text)) - (Val(Combo1.Text))) + 1) * (((Val(Combo4.Text)) -
(Val(Combo3.Text))) + 1)
For a = 0 To 2
For b = 0 To 99999
Coords(a, b) = Empty
Next b
Next a



No = 0
For f = Val(Combo5.Text) To Val(Combo6.Text)
For d = Val(Combo3.Text) To Val(Combo4.Text)
For E = Val(Combo1.Text) To Val(Combo2.Text)

Coords(0, No) = E
Coords(1, No) = d
Coords(2, No) = f
List1.AddItem Coords(0, No) & "," & Coords(1, No) & "," & Coords(2, No)
No = No + 1
Next E
Next d
Next f
NumEditPositions.Value = No + 1
End Sub

Static Sub cmdHome_Click()
' the "send instruction" button was clicked
    If (objCommPort.PortOpen = True) Then
        ' if the com port has already been openned
        ' convert the user data and send the instruction
        'Call Position_to_bytes(Val(txtData.Text))
        ' put all entered values in format to send to com port
        For Y = 1 To 3
        'Call Position_to_bytes(NumEditNewPosition(y - 1).Value)
        'unit = Chr(comboUnitNum.Text)
        unit = Chr(0)
        inst = Chr(1)

        'databytes = Chr$(iCommand(3)) + Chr$(iCommand(4)) + Chr$(iCommand(5)) +
Chr$(iCommand(6))
        objCommPort.Output = unit + inst
        Next Y
    Else
        ' the send button was pressed before the com port was open
        MsgBox "You must open the com port before sending instructions."
    End If
End Sub

Static Sub cmdMoveRelative_Click()
```

```
 ' the "send instruction" button was clicked
    If (objCommPort.PortOpen = True) Then
        ' if the com port has already been openned
        ' convert the user data and send the instruction
        'Call Position_to_bytes(Val(txtData.Text))
        ' put all entered values in format to send to com port
        For Y = 1 To 3
        Call Position_to_bytes(NumEditNewPosition(Y - 1).Value)
        'unit = Chr(comboUnitNum.Text)
        unit = Chr(Y)
        'inst = Chr(comboCommandNum.Text)
        inst = Chr(21)
        databytes = Chr$(iCommand(3)) + Chr$(iCommand(4)) + Chr$(iCommand(5)) +
Chr$(iCommand(6))
        objCommPort.Output = unit + inst + databytes
        Next Y
    Else
        ' the send button was pressed before the com port was open
        MsgBox "You must open the com port before sending instructions."
    End If
End Sub

Static Sub cmdReturn_Click(Index As Integer)
NumEditNewPosition(Index).Value = 0
  ' the "send instruction" button was clicked
    If (objCommPort.PortOpen = True) Then
        ' if the com port has already been openned
        ' convert the user data and send the instruction
        'Call Position_to_bytes(Val(txtData.Text))
        ' put all entered values in format to send to com port

        Call Position_to_bytes(Zero(Index) + NumEditNewPosition(Index).Value)
        'unit = Chr(comboUnitNum.Text)
        unit = Chr(Index + 1)
        'inst = Chr(comboCommandNum.Text)
        inst = Chr(20)
        databytes = Chr$(iCommand(3)) + Chr$(iCommand(4)) + Chr$(iCommand(5)) +
Chr$(iCommand(6))
        objCommPort.Output = unit + inst + databytes

    Else
        ' the send button was pressed before the com port was open
        MsgBox "You must open the com port before sending instructions."
    End If
End Sub

Static Sub cmdReturntoOrigin_Click()
For b = 0 To 2
NumEditNewPosition(b).Value = 0
Next b
cmdSend_Click
End Sub

Static Sub cmdSet_Click(Index As Integer)
Zero(Index) = NumEditPosition(Index)
NumEditRelPosition(Index) = NumEditPosition(Index) - Zero(Index)
End Sub

Static Sub cmdSetZero_Click()
For X = 0 To 2
Zero(X) = NumEditPosition(X)
NumEditRelPosition(X) = NumEditPosition(X) - Zero(X)
Next X

File = Text1
    Open Text1 For Output As #2
    Print #2, NumEditPosition(0)
    Print #2, NumEditPosition(1)
    Print #2, NumEditPosition(2)
    Close #2

End Sub




Static Sub cmdLoad_Click()
filebox.CancelError = True
    'On Error GoTo ErrHandle


   filebox.Action = 1

    Open filebox.FileName For Input As #1

     No = -1

    Do
        No = No + 1
        Input #1, XCoord$
        Coords(0, No) = Val(XCoord$)
        Input #1, YCoord$
        Coords(1, No) = Val(YCoord$)
        Input #1, ZCoord$
        Coords(2, No) = Val(ZCoord$)
    List1.AddItem (Coords(0, No) & "," & Coords(1, No) & "," & Coords(2, No))
    Loop Until EOF(1)

    Close #1

    NumEditPositions.Value = No + 1

End Sub

Static Sub cmdScan_Click()

'StackList.Text1.Text = StackList.Text1.Text + "XYZController" + Chr(13) + Chr(10)
 XYZController.Hide
 '***Edit***
SaveStart = SaveStart + 1
'StackList.Text1.Text = StackList.Text1.Text + CStr(SaveStart) + Chr(13) + Chr(10)
Acquisition.NumEditAcquired = 0
nn = 0
Scan = 1
countpoints = countpoints + 1
If (countpoints - 1) >= No Then
countpoints = 0
NumEditCounter = 0
Scan = 0
Exit Sub
End If


Do

'Label3.Caption = countpoints & "," & nn
NumEditNewPosition(nn).Value = (Coords(nn, countpoints - 1) *
NumEditResolution(nn).Value) - NumEditRelPosition(nn).Value
 NewRelPosition(nn) = NumEditRelPosition(nn).Value + NumEditNewPosition(nn).Value
 ' the "send instruction" button was clicked
    If (objCommPort.PortOpen = True) Then
```

```
        ' if the com port has already been openned
        ' convert the user data and send the instruction
        'Call Position_to_bytes(Val(txtData.Text))
        ' put all entered values in format to send to com port
        'For y = 1 To 3
        Call Position_to_bytes(NumEditNewPosition(nn).Value)
        'unit = Chr(comboUnitNum.Text)
        unit = Chr(nn + 1)
        'inst = Chr(comboCommandNum.Text)
        inst = Chr(21)
        databytes = Chr$(iCommand(3)) + Chr$(iCommand(4)) + Chr$(iCommand(5)) +
Chr$(iCommand(6))
        objCommPort.Output = unit + inst + databytes
        'Next y
    Else
        ' the send button was pressed before the com port was open
        MsgBox "You must open the com port before sending instructions."
    End If
    'Call Delay(1)

    ' wait for 1 seconds
    t = Timer
    While (Timer - t) < 0.5
    Wend

  'Call objCommPort_OnComm

nn = nn + 1
Loop Until nn = 3
'StackList.Text1.Text = StackList.Text1.Text + "***Scan Complete***" + Chr(13) + Chr(10)
End Sub

Static Sub cmdTransmit_Click()
' the "send instruction" button was clicked
    If (objCommPort.PortOpen = True) Then
        ' if the com port has already been openned
        ' convert the user data and send the instruction
        Call Position_to_bytes(Val(txtData.Text))
        ' put all entered values in format to send to com port

        unit = Chr(comboUnitNum.Text)

        inst = Chr(comboCommandNum.Text)

        databytes = Chr$(iCommand(3)) + Chr$(iCommand(4)) + Chr$(iCommand(5)) +
Chr$(iCommand(6))
        objCommPort.Output = unit + inst + databytes

    Else
        ' the send button was pressed before the com port was open
        MsgBox "You must open the com port before sending instructions."
    End If
End Sub

Static Sub Command1_Click()
    File = Text1
    Open Text1 For Input As #2
    Input #2, Xorig$
    NumEditNewPosition(0) = Val(Xorig$)
    Input #2, Yorig$
    NumEditNewPosition(1) = Val(Yorig$)
    Input #2, zorig$
    NumEditNewPosition(2) = Val(zorig$)
    Close #2

End Sub
```

```
Private Sub Command2_Click()
objCommPort.Output = Chr$(1) + Chr$(20) + Chr$(1) + Chr$(1) + Chr$(0) + Chr$(0)
End Sub

Private Sub Command3_Click()
 ' either open the com port or close it depending on button caption
    If Check1.Value = 1 Then
        With objCommPort2
            ' Set com port number based on user selection
            .CommPort = 5
            ' try to open the port
            .PortOpen = True
            ' Save current port settings
            OldHandShake = .Handshaking
            OldSettings = .Settings
            OldMode = .InputMode
            OldInputLen = .InputLen
            ' Set the new settings for Teckmo communications
            .Handshaking = comNone
            .Settings = "9600,N,8,1"
            .InputMode = comInputModeText
            .InputLen = 1
            .RThreshold = 1
            .Output = "/1ZR" + Chr(13)
            End With
        ' disable the port select area so the user can't try to change
        ' to a different com port while the current one is open
        'comboPortNum.Enabled = False
        ' change button caption to close com port
        'cmdOpenCom.Caption = "Close com port"
        ' display a com port status message
        'lblStatus.Caption = "Com " & comboPortNum.Text & " is the current port."

        ' Note there is no need to make sure the receive buffer is empty
        ' since RTheshold is set to 1 above. This means whenever there is
        ' at least 1 byte in the receive buffer, the objCommPort_OnComm()
        ' subroutine will be triggered. this routine does error checking
        ' to remove stray bytes that are not part of a 6 byte reply

        ' Renumber all units
        'Call Renumber_All
    Else
        ' the "Close com port" button was clicked
        ' return comm port settings to the previous system values
        ' then close the port
        With objCommPort2
            .Handshaking = OldHandShake
            .Settings = OldSettings
            .InputMode = OldInputMode
            .InputLen = OldInputLen
'            .PortOpen = False
        End With
        ' enable open com port button
        'cmdOpenCom.Caption = "Open com port"
        'comboPortNum.Enabled = True
        ' change the status message
        'lblStatus.Caption = "All com ports are currently closed."
    End If
    ' switch focus back to send button so enter key causes send
    'cmdSend.SetFocus
End Sub

Static Sub Form_Load()
    ' Initialize combo box choices
    With comboPortNum
        .AddItem 1
        .AddItem 2
        .AddItem 3
        .AddItem 4
    End With
    With comboUnitNum
        For i = 0 To 254
        .AddItem i
        Next i
    End With
    With comboCommandNum
        For i = 0 To 63
        .AddItem i
        Next i
    End With
    With Combo1
        For i = 0 To 50
        .AddItem i
        Next i
    End With
     With Combo2
        For i = 0 To 50
        .AddItem i
        Next i
    End With
     With Combo3
        For i = 0 To 50
        .AddItem i
        Next i
    End With
     With Combo4
        For i = 0 To 50
        .AddItem i
        Next i
    End With

    With Combo5
        For i = 0 To 50
        .AddItem i
        Next i
    End With
     With Combo6
        For i = 0 To 50
        .AddItem i
        Next i
    End With
    ' disable close port button since the com port is not open yet
    cmdOpenCom.Enabled = True
    cmdOpenCom.Caption = "Open com port and renumber"
    comboPortNum.Enabled = True
    ' display com port status message
    lblStatus.Caption = "All com ports are currently closed."
    ' clear the text Display area and add headings
    lstDisplay.Clear
    txtHeadings.Text = "Unit#       Command#       Data (bits)      Data (um)"
    ' set default input mode to um units
     'DIO.Ports(0).Assignment = cwdioOutput

     'DIO.SingleWrite 0



End Sub

Static Sub cmdOpenCom_Click()
```

```
    ' either open the com port or close it depending on button caption
    If cmdOpenCom.Caption = "Open com port and renumber" Then
        With objCommPort
            ' Set com port number based on user selection
            .CommPort = Val(comboPortNum.Text)
            ' try to open the port
            .PortOpen = True
            ' Save current port settings
            OldHandShake = .Handshaking
            OldSettings = .Settings
            OldMode = .InputMode
            OldInputLen = .InputLen
            ' Set the new settings for Teckmo communications
            .Handshaking = comNone
            .Settings = "9600,N,8,1"
            .InputMode = comInputModeText
            .InputLen = 1
            .RThreshold = 1
        End With
        ' disable the port select area so the user can't try to change
        ' to a different com port while the current one is open
        comboPortNum.Enabled = False
        ' change button caption to close com port
        cmdOpenCom.Caption = "Close com port"
        ' display a com port status message
        lblStatus.Caption = "Com " & comboPortNum.Text & " is the current port."

        ' Note there is no need to make sure the receive buffer is empty
        ' since RTheshold is set to 1 above. This means whenever there is
        ' at least 1 byte in the receive buffer, the objCommPort_OnComm()
        ' subroutine will be triggered. this routine does error checking
        ' to remove stray bytes that are not part of a 6 byte reply

        ' Renumber all units
        If Check1.Value = 1 Then
        With objCommPort2
            ' Set com port number based on user selection
            .CommPort = 5
            ' try to open the port
            .PortOpen = True
            ' Save current port settings
            OldHandShake = .Handshaking
            OldSettings = .Settings
            OldMode = .InputMode
            OldInputLen = .InputLen
            ' Set the new settings for Teckmo communications
            .Handshaking = comNone
            .Settings = "9600,N,8,1"
            .InputMode = comInputModeText
            .InputLen = 1
            .RThreshold = 1
            .Output = "/1ZR" + Chr(13)
            End With
        End If
        ' disable the port select area so the user can't try to change
        ' to a different com port while the current one is open
        'comboPortNum.Enabled = False
        ' change button caption to close com port
        'cmdOpenCom.Caption = "Close com port"
        ' display a com port status message
        'lblStatus.Caption = "Com " & comboPortNum.Text & " is the current port."

        ' Note there is no need to make sure the receive buffer is empty
        ' since RTheshold is set to 1 above. This means whenever there is
        ' at least 1 byte in the receive buffer, the objCommPort_OnComm()
        ' subroutine will be triggered. this routine does error checking
        ' to remove stray bytes that are not part of a 6 byte reply

        ' Renumber all units
        'Call Renumber_All

        Call Renumber_All
    Else
        ' the "Close com port" button was clicked
        ' return comm port settings to the previous system values
        ' then close the port
        With objCommPort
            .Handshaking = OldHandShake
            .Settings = OldSettings
            .InputMode = OldInputMode
            .InputLen = OldInputLen
            .PortOpen = False
        End With
        ' enable open com port button
        cmdOpenCom.Caption = "Open com port and renumber"
        comboPortNum.Enabled = True
        ' change the status message
        lblStatus.Caption = "All com ports are currently closed."
    End If
    ' switch focus back to send button so enter key causes send

 ' either open the com port or close it depending on button caption

    ' switch focus back to send button so enter key causes send
    'cmdSend.SetFocus




    cmdSend.SetFocus


End Sub

Static Sub cmdClearScreen_Click()
    lstDisplay.Clear
    ' switch focus back to send button so enter key causes send
    cmdSend.SetFocus
End Sub

Static Sub Renumber_All()
    ' renumber all units
    objCommPort.Output = Chr$(0) + Chr$(2) + Chr$(0) + Chr$(0) + Chr$(0) + Chr$(0)
    ' wait while renumbering
    'Call Delay(1.5)
    ' wait for d seconds
    t = Timer
    While (Timer - t) < 0.5
    Wend
End Sub

Static Sub Form_KeyPress(KeyAscii As Integer)
    ' if a key is pressed check if it is the enter key. if so, send instruction
    If KeyAscii = 13 Then Call cmdSend_Click
End Sub
Static Sub cmdSend_Click()
    ' the "send instruction" button was clicked
    If (objCommPort.PortOpen = True) Then
        ' if the com port has already been openned
        ' convert the user data and send the instruction
        'Call Position_to_bytes(Val(txtData.Text))
        ' put all entered values in format to send to com port
        For Y = 1 To 3
        Call Position_to_bytes(Zero(Y - 1) + NumEditNewPosition(Y - 1).Value)
        'unit = Chr(comboUnitNum.Text)
        unit = Chr(Y)
        'inst = Chr(comboCommandNum.Text)
        inst = Chr(20)
        databytes = Chr$(iCommand(3)) + Chr$(iCommand(4)) + Chr$(iCommand(5)) +
Chr$(iCommand(6))
        objCommPort.Output = unit + inst + databytes
        Next Y
    Else
        ' the send button was pressed before the com port was open
        MsgBox "You must open the com port before sending instructions."
    End If
End Sub
Static Sub objCommPort_OnComm()
'StackList.Text1.Text = StackList.Text1.Text + "objCommPort_OnComm" + Chr(13) + Chr(10)
    ' This subroutine is triggered whenever there is a com event
    ' The comEvReceive event occurs whenever the number of bytes
    ' in the receive buffer reaches that specified by objCommPort.RTheshold
    ' (which is set to 1 for our purposes so that whenever there is a byte
    ' in the receive buffer, this code is executed)
    If objCommPort.CommEvent = comEvReceive Then
        ' The com event was triggered because there is a byte in the
        ' receive buffer.
        ' Determine elapsed time since the last byte was received
        t_last = t_now
        t_now = Timer
        If (t_now - t_last) > 0.1 Then
            ' 100 ms or more has passed since the previous byte
            ' was received so ignore all bytes except the current one
            ' This is necessary to trap erroneous bytes due to noise
            ' or other sources. Using this method, if an erroneous byte
            ' occurs it may corrupt the reply but it should not upset
            ' further communications
            bytecount = 0
        End If
        ' count the byte that was just received
        bytecount = bytecount + 1
        ' read the byte into the reply array


        bReply(bytecount) = Asc(objCommPort.Input)


        ' if this byte is the sixth then display the reply
        If bytecount = 6 Then
            ' calculate the value of the 4 byte data in the reply
            Call Bytes_to_position
            ' display the reply
            Call Display_reply
            ' reset the bytecount to zero
            bytecount = 0
        End If
    End If

End Sub

Static Sub Display_reply()
'StackList.Text1.Text = StackList.Text1.Text + "Display_Reply" + Chr(13) + Chr(10)
    ' make sure there are enough lines in the display list
    ' to display this reply on the appropriate line number
    While lstDisplay.ListCount <= bReply(1)
    lstDisplay.AddItem ""
    Wend
    ' display text with appropriate column spacing
    ' determine text lengths
    L1 = Len(Str(bReply(1)))
    L2 = Len(Str(bReply(2)))
    L3 = Len(Str(dReplyData))
    ' determine needed spacing
    S1 = Left$("                    ", 13 - L1)
    S2 = Left$("                    ", 16 - L2)
    S3 = Left$("                    ", 18 - L3)
    ' determine the complete line of display text
    DisplayString = bReply(1) & S1 & bReply(2) & S2 & dReplyData & S3 & dReplyDataUm
    NumEditPosition(bReply(1) - 1) = dReplyDataUm       '<----Error Here
    NumEditRelPosition(bReply(1) - 1) = NumEditPosition(bReply(1) - 1) - Zero(bReply(1)
- 1)
    If bReply(2) = "255" Then MsgBox "Error! Data out of range.", vbCritical + vbOKOnly,
"Error"
    'MsgBox "Error.  Unit " & bReply(1) & " data out of range.", vbCritical + vbOKOnly,
"Error"
    ' display the text on the appropriate line number
    lstDisplay.List(bReply(1)) = DisplayString

    If Round(NumEditRelPosition(0), 0) = NewRelPosition(0) And
Round(NumEditRelPosition(1), 0) = NewRelPosition(1) And Round(NumEditRelPosition(2), 0)
= NewRelPosition(2) And Scan = 1 Then
    '***Edit***
    'If SaveStart = 1 Then
     Call Acquisition.cmdStart_Click
    ' End If
    'If Acquisition.NumEditAcquired = 1 Then Call cmdScan_Click
    End If
End Sub

Static Sub Bytes_to_position()
    ' convert the 4 received data bytes to a single value
    If bReply(6) > 127 Then         ' negative
    dReplyData = -4294967296# + 256# * 256# * 256# * bReply(6) + 256# * 256# * bReply(5)
+ 256# * bReply(4) + bReply(3)
    Else
    dReplyData = 256# * 256# * 256# * bReply(6) + 256# * 256# * bReply(5) + 256# *
bReply(4) + bReply(3)
    End If

    ' also determine the value in units of um
    ' ReplyData in um = ReplyData in usteps * 6.35 um/step / 64 usteps/step
    dReplyDataUm = Round(dReplyData * 6.35 / 64, 2)
End Sub

Static Sub Position_to_bytes(ByVal Position As Double)
    ' convert the user input data into 4 data bytes to send
    If (optMicrons.Value = True) Then
        ' convert data from microns to microsteps before sending it
        ' position [usteps] = position [um] * 64 [usteps/step] / 6.35 [um/step]
        Position = Round(Position * 64 / 6.35, 0)
        ' since the position in microsteps must be a whole number while
        ' the position in microns is not, there will be an error of up to
        ' +/- 0.5 microsteps (.05 um) between the requested value in microns
        ' and the actual position the program instructs the unit to move to
    End If
    If Position < 0 Then
        'negative value entered by user
        Position = 4294967296# + Position
    End If
        iCommand(6) = Int(Position / (256# * 256# * 256#))
        Position = Position - 256# * 256# * 256# * iCommand(6)
        iCommand(5) = Int(Position / (256# * 256#))
        Position = Position - 256# * 256# * iCommand(5)
        iCommand(4) = Int(Position / 256#)
        Position = Position - 256# * iCommand(4)
        iCommand(3) = Int(Position)
End Sub
```

```
---Main---

Private Sub MDIForm_Load()
Acquisition.Show
XYZController.Show
Form3.Show
End Sub


---frmSplash---


Option Explicit

Private Sub Form_KeyPress(KeyAscii As Integer)
    Unload Me
    Main.Show
End Sub


Private Sub Frame1_Click()
    Unload Me
    Main.Show
End Sub

---Form 3---


Const BoardNum% = 1           ' Board number
'Const PortNum% = FIRSTPORTB      ' use first digital port
'Const Direction% = DIGITALOUT    ' program digital port A for output
Dim counter
Dim time(20000)
'Dim Data(20000)
Dim Datax(20000)
Dim DataY(20000)
Dim xtick
Dim eng!

Dim ymax(20000)
Dim ymin(20000)
Dim xmax(20000)
Dim xmin(20000)
Dim xma
Dim xmi
Dim yma
Dim ymi
Dim gai
Dim sweep
'Const NumPoints& = 2000       ' Number of data points to collect
Const FirstPoint& = 0         ' set first element in buffer to transfer to array
'Dim ADData%(NumPoints&)       ' dimension an array to hold the input values
'Dim MemHandle&                ' define a variable to contain the handle for
                              ' memory allocated by Windows through cbWinBufAlloc%()
'Dim HighChan%
```

```
'Dim EngUnits(NumPoints)
Dim StopAq
Dim countersave

Dim dummy%
Dim LowChan%
Dim CBCount&
Dim CBRate&
Dim Options
Dim Gain
Dim TrigType%
Dim TrigCurrent!
Dim TrigVoltage!
Dim Range%
Dim TrigValue%
Dim Ulstat%
Dim N
Dim np
Dim Over
Dim noop
Dim avnum
Dim instname As ViSession
Dim devicestatusbyte As ViUInt8
Dim statuscode As ViStatus
Dim FileName
Dim wavelengthdata(2999) As ViReal64
Dim dataref(2999) As ViReal64
Dim databac(2999) As ViReal64
Dim dataA(2999)
Dim datab(2999)
Dim wavelengthdatabin(999)
Dim dataAbin(999)
Dim TriggerActivate As Boolean

Dim dataspec(2999) As ViReal64
Dim norev As Integer




Private Sub Add_Click()

    'determine if max number of datasets (5) is open noop = number open
    If noop = 5 Then
        MsgBox "Max No. of Datasets Open", vbExclamation + vbOKOnly, "Doh!!"
        Exit Sub
    End If

    'configure dialog box and form caption
    Form4.filebox.CancelError = True
    On Error GoTo ErrHandle
    Form4.filebox.FileName = "*.dat"
    Form4.filebox.InitDir = "C:\doug\data"
    Form4.filebox.Action = 1
    Form4.Caption = Form4.filebox.FileName

    'open file and input into variables
    Open Form4.filebox.FileName For Input As #4

    No = -1

    Do
        No = No + 1
        Input #4, E$
        Datax(No) = Val(E$)
        Input #4, i$
        DataY(No) = Val(i$)
    Loop Until EOF(4)

    Close #4

    noop = noop + 1

    CWGraphXY.PlotTemplate.MultiPlot = True
    CWGraphXY.Plots.Add

    'plot graph. norev is used if a previously added plot has been removed
    For nop = 0 To No
        If norev <> 0 Then CWGraphXY.Plots.Item(norev).ChartXvsY Datax(nop), DataY(nop)
        If norev = 0 Then CWGraphXY.Plots.Item(noop).ChartXvsY Datax(nop), DataY(nop)
    Next

    norev = 0

    'determine max and min values of data and display
    N = 0
    yma = DataY(0)
    ymi = DataY(0)
    xma = Datax(0)
    xmi = Datax(0)

    Do
        N = N + 1
        If DataY(N) > yma Then yma = DataY(N)
        If DataY(N) < ymi Then ymi = DataY(N)
        If Datax(N) > xma Then xma = Datax(N)
        If Datax(N) < xmi Then xmi = Datax(N)
    Loop Until N = No

    lblYmax.Caption = Format$(yma, "0.00E-00")
    lblYmin.Caption = Format$(ymi, "0.00E-00")
    lblXmax.Caption = Format$(xma, "0.000")
    lblXmin.Caption = Format$(xmi, "0.000")

ErrHandle:
End Sub

Private Sub AquireXY_Click()

    cmdStartXY_Click

End Sub



Private Sub AquireYt_Click()

    Form4.Visible = False
    form2.Visible = True
    form2.CWButton1.Value = False

End Sub

Private Sub Blue_Click()

    CWGraphXY.Plots.Item(2).ClearData
    CWGraphXY.Plots.Item(2).ClearData
    noop = noop - 1

    If norev = 0 Or norev > 2 Then norev = 2
```

```
End Sub

Private Sub cmdOpenCom_Click()
' either open the com port or close it depending on button caption
    If cmdOpenCom.Caption = "Open com port" Then
        With objCommPort
            ' Set com port number based on user selection
            .CommPort = 5
            ' try to open the port
            .PortOpen = True
            ' Save current port settings
            OldHandShake = .Handshaking
            OldSettings = .Settings
            OldMode = .InputMode
            OldInputLen = .InputLen
            ' Set the new settings for Teckmo communications
            .Handshaking = comNone
            .Settings = "9600,N,8,1"
            .InputMode = comInputModeText
            .InputLen = 1
            .RThreshold = 1
            End With
        ' disable the port select area so the user can't try to change
        ' to a different com port while the current one is open
        'comboPortNum.Enabled = False
        ' change button caption to close com port
        cmdOpenCom.Caption = "Close com port"
        ' display a com port status message
        'lblStatus.Caption = "Com " & comboPortNum.Text & " is the current port."

        ' Note there is no need to make sure the receive buffer is empty
        ' since RTheshold is set to 1 above. This means whenever there is
        ' at least 1 byte in the receive buffer, the objCommPort_OnComm()
        ' subroutine will be triggered. this routine does error checking
        ' to remove stray bytes that are not part of a 6 byte reply

        ' Renumber all units
        'Call Renumber_All
    Else
        ' the "Close com port" button was clicked
        ' return comm port settings to the previous system values
        ' then close the port
        With objCommPort
            .Handshaking = OldHandShake
            .Settings = OldSettings
            .InputMode = OldInputMode
            .InputLen = OldInputLen
            .PortOpen = False
        End With
        ' enable open com port button
        cmdOpenCom.Caption = "Open com port"
        'comboPortNum.Enabled = True
        ' change the status message
        'lblStatus.Caption = "All com ports are currently closed."
    End If
    ' switch focus back to send button so enter key causes send
    'cmdSend.SetFocus
    objCommPort.Output = "/1ZR" + Chr(13)
End Sub

Private Sub cmdStartXY_Click()
StackList.Text1.Text = StackList.Text1.Text + "Arm" + Chr(13) + Chr(10)
TriggerActivate = False
nc = 0
```

```
cmdstartxy.Visible = False
Label2.Visible = True
Form3.Refresh
N = 0

'get the spectrometer setup

Test1 = SPX_init(ByVal "USB0::0x01313::0x0111::S/N M00231225::RAW", ByVal 2000,
instname)
Text6 = instname
inttime = CWKnob1.Value / 1000 ' sets integration time for the device
Test2% = SPX_setIntTime(ByVal instname, ByVal inttime)
test4% = SPX_getIntTime(ByVal instname, integrationtime) 'gets the actual value
Text4 = integrationtime
wave = SPX_getWavelengthData(ByVal instname, wavelengthdata(0), mini, Max)

'set system to continuous
Test3% = SPX_startScanCont(ByVal instname)
StackList.Text1.Text = StackList.Text1.Text + "Spectro switched on" + Chr(13) + Chr(10)

'gai = InputBox("Enter Gain", "Enter Gain", gai)

gainEC = 10 ^ (CWKnob2.Value)


'put in arm channel 1

    Gain = BIP5VOLTS             ' set the gain

    'Ulstat% = cbAIn(BoardNum%, 1, Gain, DataValue%) 'example of analogue in
    'If Ulstat% = 30 Then MsgBox "Change the Gain argument to one supported by this
board.", 0, "Unsupported Gain"
    'If Ulstat% <> 0 Then Stop

    'Ulstat% = cbToEngUnits(BoardNum%, Gain, DataValue%, eng!)
    'If Ulstat% <> 0 Then Stop

estart = eng!

'Do

'Ulstat% = cbAIn(BoardNum%, 1, Gain, DataValue%)
'    If Ulstat% = 30 Then MsgBox "Change the Gain argument to one supported by this
board.", 0, "Unsupported Gain"
'    If Ulstat% <> 0 Then Stop

'    Ulstat% = cbToEngUnits(BoardNum%, Gain, DataValue%, eng!)
'    If Ulstat% <> 0 Then Stop




'Loop Until Abs(eng! - estart) > 0.1

NumPoints = 2000 / 20
    'Do ' sets up the wait state
    'vloop = 0

    'For wa = 1 To (NumPoints / 5)
    'CWAIPoint1.SingleRead Data
    'vloop = vloop + Data(0)
    'Erase Data
    'Next wa
    'vloop = vloop / (NumPoints / 5)
    'Loop Until Abs(vstart - vloop) > 0.005
'---
'CWAIPoint1.SingleRead Data
'    vstart = Data(1)

'    Do ' sets up the wait state
'    vloop = 0

'    For wa = 1 To (NumPoints / 5)
'    CWAIPoint1.SingleRead Data
'
'    vloop = Data(1)
'    Erase Data
'    Next wa
'    vloop = vloop / (NumPoints / 5)


'    Loop Until Abs(vstart - vloop) > 0.1
'---
'startt = Timer

CWDIO1.SingleWrite 0
tstartd = Timer
Do
Loop Until Timer - tstartd > 0.1
CWDIO1.SingleWrite 1
Do
Loop Until Timer - tstartd > 0.2
CWDIO1.SingleWrite 0
'Do
'Loop Until Timer - tstartd > 10.1
'CWDIO1.SingleWrite 1
'Do
'Loop Until Timer - tstartd > 10.2
'CWDIO1.SingleWrite 0

'objCommPort.Output = "/1gS10I2A1000S6I2A0G3R" + Chr(13)


Label2.Visible = False
Form3.Refresh





Dim TotChan0
Dim TotChan1
Dim X
'Dim N
Dim LowChan%
Dim AveChan0
Dim AveChan1

    StopAq = 0
    gai = 1000000#
    sweep = 50
    'noop = o
    noop = 0
    startd = Timer


    'MemHandle& = cbWinBufAlloc(NumPoints&)      ' set aside memory to hold data
    'If MemHandle& = 0 Then Stop
    For N = 0 To 20000
        DataY(N) = Empty
        Datax(N) = Empty
    Next N

    CWGraphXY.ClearData

cmdStopXY.Visible = True

'StackList.Text1.Text = StackList.Text1.Text + "Abs Started" + Chr(13) + Chr(10)

   N = 0
   countersave = 0

   Do
    N = N + 1

repeat:
'DoEvents




    If StopAq = 1 Then

        Exit Sub

    End If






    'MemHandle& = cbWinBufAlloc(NumPoints&)      ' set aside memory to hold data
    'If MemHandle& = 0 Then Stop
    ' Collect the values with cbAInScan%()
    ' Parameters:
    '   BoardNum%   :the number used by CB.CFG to describe this board
    '   LowChan%    :the first channel of the scan
    '   HighChan%   :the last channel of the scan
    '   CBCount&    :the total number of A/D samples to collect
    '   CBRate&     :sample rate
    '   Gain        :the gain for the board
    '   ADData%     :the array for the collected data values
    '   Options     :data collection options


     '   LowChan% = 0
      '  HighChan% = 1


    'CBCount& = NumPoints&          ' total number of data points to collect
    'CBRate& = sweep * 200              ' sampling rate (samples per second)
    'Options = CONVERTDATA     ' return data as 12-bit values
    Gain = BIP10VOLTS               ' set the gain

    'If MemHandle& = 0 Then Stop    ' check that a handle to a memory buffer exists
   'If CBRate& < 1000 Then CBRate& = 1000

    'Ulstat% = cbAInScan(BoardNum%, LowChan%, HighChan%, CBCount&, CBRate&, Gain,
MemHandle&, Options)

    'If Ulstat% = 30 Then MsgBox "Change the Gain argument to one supported by this
board.", 0, "Unsupported Gain"
    'If Ulstat% <> 0 And Ulstat% <> 91 Then Stop
    ' Transfer the data from the memory buffer set up by Windows to an array for use by
Visual Basic

    'Ulstat% = cbWinBufToArray(MemHandle&, ADData%(0), FirstPoint&, CBCount&)
    'If Ulstat% <> 0 Then Stop

  'Ulstat% = cbAIn(BoardNum%, 1, Gain, DataValue%)
   ' If Ulstat% = 30 Then MsgBox "Change the Gain argument to one supported by this
board.", 0, "Unsupported Gain"
   ' If Ulstat% <> 0 Then Stop

   ' Ulstat% = cbToEngUnits(BoardNum%, Gain, DataValue%, eng!)
    'If Ulstat% <> 0 Then Stop
    TotChan1 = 0

    For w = 1 To NumPoints
    CWAIPoint1.SingleRead data

    TotChan1 = TotChan1 + data(1)
    Erase data
    Next w

    AveChan1 = TotChan1 / (NumPoints)


    Datax(N) = Timer - startd

    If Datax(N) > 15 Then
        If TriggerActivate = False Then
        CWDIO1.SingleWrite 1
        tstarto = Timer
            Do
            Loop Until Timer - tstarto > 0.1
        CWDIO1.SingleWrite 0
        TriggerActivate = True
        End If
    End If

    DataY(N) = AveChan1 / gainEC




'avnum = CWSlide1.Value
Nsp = 0
Dim datasp(2999)
Erase datasp
Erase dataA

Do
test4 = SPX_getScanData(ByVal instname, dataspec(0))

For av = 0 To 2998
datasp(av) = datasp(av) + dataspec(av)
Next av
Nsp = Nsp + 1
Loop Until Nsp = avnum

'subtract background
For av = 0 To 2998
'dataspec(av) = (datasp(av) / avnum) - datab(av)
dataspec(av) = (datasp(av) / avnum)
If (dataspec(av) / dataref(av)) <= 0 Then
dataA(av) = 5
Else
```

```
dataA(av) = 2 - (Log(((dataspec(av) / dataref(av))) * 100) / 2.301)
End If
Next av



bin = 0
For av = 1 To 2999 Step 3
dataAbin(bin) = (dataA(av - 1) + dataA(av) + dataA(av + 1)) / 3
wavelengthdatabin(bin) = (wavelengthdata(av - 1) + wavelengthdata(av) +
wavelengthdata(av + 1)) / 3

'Set pointer 1 to be the active pointer
Set CWSlide2.ActivePointer = CWSlide2.Pointers.Item(1)

If wavelengthdatabin(bin) < CWSlide2.Value Then
dataAbin(bin) = 0
Else
End If

'Set pointer 2 to be the active pointer
Set CWSlide2.ActivePointer = CWSlide2.Pointers.Item(2)

If wavelengthdatabin(bin) > CWSlide2.Value Then
dataAbin(bin) = 0
Else
End If



bin = bin + 1
Next av

CWGraph2.Axes.Item(1).Minimum = CWSlide2.Pointers.Item(1)
CWGraph2.Axes.Item(1).Maximum = CWSlide2.Pointers.Item(2)



CWGraph2.PlotXvsY wavelengthdatabin, dataAbin
CWGraphXY.ChartXvsY Datax(N), DataY(N)
'nc = nc + 1

    FileName = Text2.Text + Str(N) + ".dat"
    Open FileName For Output As #2
    Print #2, Format(Datax(N), "0.0000"); ","; Format(DataY(N), "0.00e-00")
    For nsave = 0 To 999
    Print #2, Format(wavelengthdatabin(nsave), "000.0"); ","; Format(dataAbin(nsave),
"0.0000")
    Next nsave
    Close #2
    Text1 = N


    lblShowVolts.Caption = Format$(Datax(N), "0.000")
    lblShowCurrent.Caption = Format(DataY(N), "0.00E-00")
    countersave = countersave + 1

    CWAIPoint1.SingleRead data


    'If Abs(data(0)) < 0.5 Then
    'Call cmdStopXY_Click
    'End If

    If N = 200 Then
    Call cmdStopXY_Click
    End If
    Loop Until N = 20000
StackList.Text1.Text = StackList.Text1.Text + "Abs Finished" + Chr(13) + Chr(10)

End Sub

Private Sub cmdQuit_Click()
Dim yesno

yesno = MsgBox("Are you sure you want to quit?", vbQuestion + vbYesNo, "Quit")

If yesno = 6 Then End


End Sub

Private Sub cmdStopXY_Click()
StackList.Text1.Text = StackList.Text1.Text + "Stop" + Chr(13) + Chr(10)
noop = 190
StopAq = 1
    'tmrXY.Enabled = False
    'cmdstartxy.Visible = True
    'lblAquiring.Visible = False
    'File.Enabled = True
    'Aquire.Enabled = True
    'CWButton2.Enabled = True

    'Ulstat% = cbWinBufFree(MemHandle&)      ' Free up memory for use by
                                            ' other programs
    'If Ulstat% <> 0 Then Stop

    N = 0
    yma = DataY(1)
    ymi = DataY(1)
    xma = Datax(1)
    xmi = Datax(1)

    Do
        N = N + 1
        If DataY(N) > yma Then yma = DataY(N)
        If DataY(N) < ymi Then ymi = DataY(N)
        If Datax(N) > xma Then xma = Datax(N)
        If Datax(N) < xmi Then xmi = Datax(N)
    Loop Until N = countersave

    lblYmax.Caption = Format$(yma, "0.00E-00")
    lblYmin.Caption = Format$(ymi, "0.00E-00")

    lblXmax.Caption = Format$(xma, "0.000")
    lblXmin.Caption = Format$(xmi, "0.000")
    Test5% = SPX_close(ByVal instname)
 CWVisa1.Close  'actually stops crashing for the second time.
End Sub

Private Sub ExitXY_Click()

    Form4.Visible = False
    ' Form1.Visible = True

End Sub



Private Sub Command1_Click()
```

```
StackList.Text1.Text = StackList.Text1.Text + "Background" + Chr(13) + Chr(10)
Text2.Text = Acquisition.Text3.Text
'this gets a dark background for correction

Test1 = SPX_init(ByVal "USB0::0x01313::0x0111::S/N M00231225::RAW", ByVal 2000,
instname)
Text6 = instname
inttime = CWKnob1.Value / 1000 ' sets integration time for the device
Test2% = SPX_setIntTime(ByVal instname, ByVal inttime)
test4% = SPX_getIntTime(ByVal instname, integrationtime) 'gets the actual value
Text4 = integrationtime
wave = SPX_getWavelengthData(ByVal instname, wavelengthdata(0), mini, Max)

'set system to continuous
Test3% = SPX_startScanCont(ByVal instname)

'get the data
avnum = CWSlide1.Value
Nbac = 0
Erase datab
Do
test4 = SPX_getScanData(ByVal instname, databac(0))
For av = 0 To 2999
datab(av) = datab(av) + databac(av)
Next av
Nbac = Nbac + 1
Loop Until Nbac = avnum

For av = 0 To 2999
datab(av) = datab(av) / avnum
Next av

Test5% = SPX_close(ByVal instname)
Text5 = Test5%

'save background data
filesavebac = Text2.Text + ".bac"
Open filesavebac For Output As #1

For av = 0 To 2999
Print #1, Format(wavelengthdata(av), "000.0"); ","; Format(datab(av), "0.00000")
Next av
Close #1

'save info

filesaveinf = Text2.Text + ".inf"
Open filesaveinf For Output As #1

Print #1, Date
Print #1, "Number of averaged spectra = "; avnum
Print #1, "Integration time/s = "; inttime


Close #1



CWVisa1.Close  'actually stops crashing for the second time.
Command2.Visible = True
Call Command2_Click
End Sub

Private Sub Command2_Click()
```

```
'    Do ' sets up the wait state
'   CWAIPoint1.SingleRead data


'    Loop Until Abs(data(0)) > 0.5


StackList.Text1.Text = StackList.Text1.Text + "Reference" + Chr(13) + Chr(10)
Test1 = SPX_init(ByVal "USB0::0x01313::0x0111::S/N M00231225::RAW", ByVal 2000,
instname)
Text6 = instname
inttime = CWKnob1.Value / 1000 ' sets integration time for the device
Test2% = SPX_setIntTime(ByVal instname, ByVal inttime)
test4% = SPX_getIntTime(ByVal instname, integrationtime) 'gets the actual value
Text4 = integrationtime
wave = SPX_getWavelengthData(ByVal instname, wavelengthdata(0), mini, Max)

'set system to continuous
Test3% = SPX_startScanCont(ByVal instname)

'get the data
avnum = CWSlide1.Value
Nsp = 0

Dim datar(2999)
Erase datar

Do
test4 = SPX_getScanData(ByVal instname, dataref(0))
For av = 0 To 2998
datar(av) = datar(av) + dataref(av)
Next av
Nsp = Nsp + 1
Loop Until Nsp = avnum

For av = 0 To 2998
dataref(av) = (datar(av) / avnum)
'dataref(av) = (datar(av) / avnum) - datab(av)
If dataref(av) = 0 Then
dataref(av) = 0.0000000001
Else
End If

Next av

'save reference data
filesaveref = Text2.Text + ".ref"
Open filesaveref For Output As #1

For av = 0 To 2999
Print #1, Format(wavelengthdata(av), "000.0"); ","; Format(dataref(av), "0.00000")
Next av
Close #1



'CWGraph1.Caption = refence
CWGraph1.PlotXvsY wavelengthdata, dataref
Test5% = SPX_close(ByVal instname)
CWVisa1.Close  'actually stops crashing for the second time.
'leave running for later
Command2.Visible = False
cmdstartxy.Visible = True
'cmdStartXY.Visible = True
Call cmdStartXY_Click
End Sub
```

```
Private Sub Command3_Click()
resetit = SPX_reset(ByVal instname)
CWVisa1.Close

form1.Show

End Sub







Private Sub CWSlide1_PointerValueChanged(ByVal Pointer As Long, Value As Variant)
Text3 = CWSlide1.Pointers.Item(1).Value


End Sub

Private Sub CWSlide2_PointerValueChanged(ByVal Pointer As Long, Value As Variant)
CWGraph1.Axes(1).Minimum = CWSlide2.Pointers.Item(1).Value
CWGraph1.Axes(1).Maximum = CWSlide2.Pointers.Item(2).Value
CWGraph2.Axes(1).Minimum = CWSlide2.Pointers.Item(1).Value
CWGraph2.Axes(1).Maximum = CWSlide2.Pointers.Item(2).Value
End Sub

Private Sub Cyan_Click()
CWGraphXY.Plots.Item(5).ClearData
CWGraphXY.Plots.Item(5).ClearData
noop = noop - 1
If norev = 0 Then norev = 5


End Sub

Private Sub force_Click()
Form4.Visible = False
form2.Visible = False

    Form3.Visible = True
End Sub

Private Sub Form_Load()
noop = 0
CWAIPoint1.Device = 1
ChannelString = "0,1"
CWAIPoint1.Channels.Add ChannelString
CWDIO1.Ports(0).Assignment = cwdioOutput
End Sub

Private Sub Green_Click()
CWGraphXY.Plots.Item(1).ClearData
CWGraphXY.Plots.Item(1).ClearData
noop = noop - 1
If norev = 0 Or norev > 1 Then norev = 1

End Sub

Private Sub Open_Click()

Dim No
Dim E$
Dim i$
Dim nop

    Shape1.Visible = False
    Label2.Visible = False
    Picture1.Visible = False

    form2.filebox.CancelError = True
    On Error GoTo ErrHandle
    Form4.filebox.FileName = "*.dat"
    Form4.filebox.InitDir = "C:\doug\data"
    Form4.filebox.Action = 1
    Form4.Caption = Form4.filebox.FileName
    Open Form4.filebox.FileName For Input As #4

    No = -1

    Do
        No = No + 1
        Input #4, E$
        Datax(No) = Val(E$)
        Input #4, i$
        DataY(No) = Val(i$)
    Loop Until EOF(4)

    Close #4
    'If noop = 0 Then
    'noop = 1
    'GoTo FirstOpen
    'End If

   'Over = MsgBox("Overlay?", vbYesNo, "Open File")

    'If Over = 7 Then
'FirstOpen:
noop = 1
    CWGraphXY.ClearData
     For nop = 0 To No
        CWGraphXY.Plots.Item(noop).ChartXvsY Datax(nop), DataY(nop)
    Next

    N = 0
    yma = DataY(0)
    ymi = DataY(0)
    xma = Datax(0)
    xmi = Datax(0)

    Do
        N = N + 1
        If DataY(N) > yma Then yma = DataY(N)
        If DataY(N) < ymi Then ymi = DataY(N)
        If Datax(N) > xma Then xma = Datax(N)
        If Datax(N) < xmi Then xmi = Datax(N)
    Loop Until N = No

    lblYmax.Caption = Format$(yma, "0.00E-00")
    lblYmin.Caption = Format$(ymi, "0.00E-00")
    lblXmax.Caption = Format$(xma, "0.000")
    lblXmin.Caption = Format$(xmi, "0.000")

    norev = 0



ErrHandle:     Exit Sub
End Sub
```

```
Private Sub optI0_Click()

End Sub

Private Sub optI1_Click()

End Sub

Private Sub optI2_Click()

End Sub

Private Sub optI3_Click()

End Sub



Private Sub Red_Click()
CWGraphXY.Plots.Item(3).ClearData
CWGraphXY.Plots.Item(3).ClearData
noop = noop - 1
If norev = 0 Or norev > 3 Then norev = 3


End Sub



Private Sub RemoveAll_Click()
yn = MsgBox("Are you sure you want to remove all plots?", vbQuestion + vbYesNo, "Remove
All Plots")
If yn = 6 Then
noop = 0
norev = 0
CWGraphXY.Plots.Item(1).ClearData
CWGraphXY.Plots.Item(2).ClearData
CWGraphXY.Plots.Item(3).ClearData
CWGraphXY.Plots.Item(4).ClearData
CWGraphXY.Plots.Item(5).ClearData
End If
End Sub

Private Sub Save_Click()

Dim ns

    Form4.filebox.CancelError = True
    On Error GoTo ErrHandle
    Form4.filebox.Flags = &H2&
    Form4.filebox.FileName = "*.dat"
    Form4.filebox.InitDir = "C:\doug\data"
    Form4.filebox.Action = 2
    Form4.Caption = Form4.filebox.FileName
    Open Form4.filebox.FileName For Output As #2

    ns = 0

    Do
        ns = ns + 1
        Print #2, Datax(ns); ","; DataY(ns)
    Loop Until ns = countersave
    Close #2

ErrHandle:     Exit Sub

End Sub



Private Sub Yellow_Click()
CWGraphXY.Plots.Item(4).ClearData
CWGraphXY.Plots.Item(4).ClearData
noop = noop - 1
If norev = 0 Or norev > 4 Then norev = 4


End Sub

Public Sub PotentialStep()
Call Command1_Click
End Sub

---Acquisition---

Dim XRange As Integer
Dim YRange As Integer
'Dim zRange As Integer

Dim Timewin

Dim LowChan%
Dim HighChan%
Dim yesno As Integer

Dim N As Single

Dim counter As Integer
Dim Total()
Dim Ave()
Dim Max()
Dim Maxp()
Dim MaxAve

Dim Total_bef()
Dim Ave_bef()
Dim Max_bef()
Dim Maxp_bef()
Dim MaxAve_bef
Dim Avep_bef()
Dim Total_aft()
Dim Ave_aft()
Dim Max_aft()
Dim Maxp_aft()
Dim MaxAve_aft()
Dim Avep_aft()
Dim Max_diff()
Dim W600_aft()
Dim W600_bef()
Dim W600_diff()
Dim Abs490nm()

Dim OriginX
Dim OriginY

Dim xmax As Integer
Dim ymax As Integer
Dim zmax As Integer
Dim xmin As Integer
Dim ymin As Integer
Dim zmin As Integer

Dim Avep()
Dim MinAve
Dim MinMax
Dim MaxMax
 Dim X
 Dim Y
 Dim Z
 Dim FileName
 Dim numpositions

 Dim BeforePulse As Boolean
 Dim Wavelength600nm
 Dim Wavelength490nm
 Dim Well_125_490nm


Static Sub cmdQuit_Click()
Dim yesno

yesno = MsgBox("Are you sure you want to quit?", vbQuestion + vbYesNo, "Quit")

If yesno = 6 Then End
End Sub

Static Sub cmdReset_Click()
counter = 0
lblNoDatasets.Caption = 0
End Sub

Static Sub cmdSave_Click()
Dim ns


 form1.filebox.CancelError = True

    On Error GoTo ErrHandle
    form1.filebox.Flags = &H2&
    form1.filebox.FileName = "*.dat"
    form1.filebox.InitDir = "C:\WINDOWS\Desktop"
    form1.filebox.Action = 2
    form1.Caption = form1.filebox.FileName
    Open form1.filebox.FileName For Output As #2

    ns = -1

    For X = xmin To xmax
     For Y = ymin To ymax
        Print #2, X; ","; Y; ","; Ave(X, Y)
     Next Y
    Next X
    Close #2



ErrHandle:     Exit Sub
End Sub

Static Sub cmdStart_Click()
'StackList.Text1.Text = StackList.Text1.Text + "Acquisition.cmdStart_Click" + Chr(13) +
Chr(10)
XRange = XYZController.Combo2.Text - XYZController.Combo1.Text + 1
```

```
YRange = XYZController.Combo4.Text - XYZController.Combo3.Text + 1
'zRange = XYZController.Combo6.Text - XYZController.Combo5.Text + 1

'RepeatLoop:

If XYZController.NumEditCounter > 0 Then
'If BeforePulse = False Then
GoTo notfirst
'End If
End If

'   If PulseBox.Value = 0 Then
'   CWGraph3D1.Height = 6975
'   CWGraph3D2.Height = 2295
'   CWGraph3D2.Top = 7320
'   End If
   CWGraph1.ChartLength = NumEditSamplestoSave.Value
    MaxAve = Empty
    MaxMax = Empty


ReDim Ave(XRange, YRange)
ReDim Avep(XRange, YRange)
ReDim Total(XRange, YRange)
ReDim Max(XRange, YRange)
ReDim Maxp(XRange, YRange)
ReDim Ave_bef(XRange, YRange)
ReDim Avep_bef(XRange, YRange)
ReDim Total_bef(XRange, YRange)
ReDim Max_bef(XRange, YRange)
ReDim Maxp_bef(XRange, YRange)
ReDim Ave_aft(XRange, YRange)
ReDim Avep_aft(XRange, YRange)
ReDim Total_aft(XRange, YRange)
ReDim Max_aft(XRange, YRange)
ReDim Maxp_aft(XRange, YRange)
ReDim Max_diff(XRange, YRange)
ReDim W600_aft(XRange, YRange)
ReDim W600_bef(XRange, YRange)
ReDim W600_diff(XRange, YRange)
ReDim Abs490nm(XRange, YRange)

    For X = 0 To (XRange - 1)
      For Y = 0 To (YRange - 1)

      Ave(X, Y) = Empty
      Avep(X, Y) = Empty
      Total(X, Y) = Empty
      Max(X, Y) = Empty
      Maxp(X, Y) = Empty
      Ave_bef(X, Y) = Empty
      Avep_bef(X, Y) = Empty
      Total_bef(X, Y) = Empty
      Max_bef(X, Y) = Empty
      Maxp_bef(X, Y) = Empty
      Ave_aft(X, Y) = Empty
      Avep_aft(X, Y) = Empty
      Total_aft(X, Y) = Empty
      Max_aft(X, Y) = Empty
      Maxp_aft(X, Y) = Empty
      Max_diff(X, Y) = Empty
      W600_aft(X, Y) = Empty
      W600_bef(X, Y) = Empty
      W600_diff(X, Y) = Empty
      Abs490nm(X, Y) = Empty
      'Shape1(21 * Y + X).FillColor = RGB(255, 255, 255)
```

```
        'Shape2(21 * Y + X).FillColor = RGB(255, 255, 255)

      Next Y
      Next X
MinAve = 1E+19
MinMax = 1E+19
Acquisition.filebox.CancelError = True
    'On Error GoTo ErrSave

    Acquisition.filebox.Flags = &H2&
    Acquisition.filebox.FileName = "*.dat"
    Acquisition.filebox.InitDir = "C:\WINDOWS\Desktop\a.dat"
    Acquisition.filebox.Action = 2
    Acquisition.Caption = Acquisition.filebox.FileName
    FileName = Acquisition.filebox.FileName

OriginX = Int(XYZController.NumEditPosition(0) + 0.5)
OriginY = Int(XYZController.NumEditPosition(1) + 0.5)

CWGraph3D1.Axes(1).Minimum = OriginX
CWGraph3D1.Axes(1).Maximum = OriginX + (XYZController.Combo2 *
XYZController.NumEditResolution(0))
CWGraph3D1.Axes(2).Minimum = OriginY
CWGraph3D1.Axes(2).Maximum = OriginY + (XYZController.Combo4 *
XYZController.NumEditResolution(1))

CWGraph3D2.Axes(1).Minimum = OriginX
CWGraph3D2.Axes(1).Maximum = OriginX + (XYZController.Combo2 *
XYZController.NumEditResolution(0))
CWGraph3D2.Axes(2).Minimum = OriginY
CWGraph3D2.Axes(2).Maximum = OriginY + (XYZController.Combo4 *
XYZController.NumEditResolution(1))


Timewin = NumEditTimewin.Value

notfirst:
Errx:
'here is the problem
X = (Round(XYZController.NumEditRelPosition(0), 0) /
(XYZController.NumEditResolution(0))) + (((XRange - 1) / 2) -
Int((Val(XYZController.Combo2.Text) + Val(XYZController.Combo1.Text)) / 2))

Erry:
Y = (Round(XYZController.NumEditRelPosition(1), 0) /
(XYZController.NumEditResolution(1))) + (((YRange - 1) / 2) -
Int((Val(XYZController.Combo3.Text) + Val(XYZController.Combo4.Text)) / 2))

'Z = (Round(XYZController.NumEditRelPosition(2), 0) /
(XYZController.NumEditResolution(2))) + (10 - Int((Val(XYZController.Combo5.Text) +
Val(XYZController.Combo6.Text)) / 2))

If counter = 0 Then
xmax = X
xmin = X
ymax = Y
ymin = Y
'zmax = Z
'zmin = Z
End If

If X < xmin Then xmin = X
If X > xmax Then xmax = X

If Y < ymin Then ymin = Y
If Y > ymax Then ymax = Y

If Z < zmin Then zmin = Z
If Z > zmax Then zmax = Z

Select Case Combo1.Text

Case "X vs Y"
Combo2.Clear
For i = zmin - (10 - Int((Val(XYZController.Combo5.Text) +
Val(XYZController.Combo6.Text)) / 2)) To zmax - (10 -
Int((Val(XYZController.Combo5.Text) + Val(XYZController.Combo6.Text)) / 2))
Combo2.AddItem i
Next i

Case "X vs Z"
Combo2.Clear
For i = ymin - (10 - Int((Val(XYZController.Combo3.Text) +
Val(XYZController.Combo4.Text)) / 2)) To ymax - (10 -
Int((Val(XYZController.Combo3.Text) + Val(XYZController.Combo4.Text)) / 2))
Combo2.AddItem i
Next i

Case "Y vs Z"
Combo2.Clear
For i = xmin - (10 - Int((Val(XYZController.Combo2.Text) +
Val(XYZController.Combo1.Text)) / 2)) To xmax - (10 -
Int((Val(XYZController.Combo2.Text) + Val(XYZController.Combo1.Text)) / 2))
Combo2.AddItem i
Next i

End Select

     Call plotit
    ' GoTo RepeatLoop
    'StackList.Text1.Text = StackList.Text1.Text + "...Completed Loop..." + Chr(13) +
Chr(10)
    End Sub
Static Sub plotit()
repeatspec: 'come back and gather another spectra

'StackList.Text1.Text = StackList.Text1.Text + "Acquisition.PlotIt" + Chr(13) + Chr(10)
Dim instname As ViSession ' the 'name' of the spectrometer
Dim wavelengthdata(2999) As ViReal64 ' where the wavelength goes
Dim data(2999) As ViReal64 ' the data array
Dim dataA(2999) As ViReal64 ' ratio
Dim dataaf(2999) As ViReal64
Dim KInt As Integer
Dim datatot(2999)

If PulseBox.Value = 1 Then
If BeforePulse = False Then
'Pulse here and then take a different spectrum triggers the step on positive edge
'set a step and allow 0.5 s to equilibriate
CWDIO1.SingleWrite 1
tstarto = Timer
Do
Loop Until Timer - tstarto > CWSlide1.Value
CWDIO1.SingleWrite 0
Else
End If
End If
Label8.Caption = CStr(Timer - tstarto)

'----
For i = 0 To 2999
```

```
    datatot(i) = 0
Next i
For k = 0 To CWSlide2.Value - 1

'----
spetro = SPX_init(ByVal "USB0::0x01313::0x0111::S/N M00231225::RAW", ByVal 2000,
instname)
'Gives you the name back
'Text1 = instname
' sets integration time for the device
inttime = CWKnob1.Value / 1000
spetro = SPX_setIntTime(ByVal instname, ByVal inttime)
'check the intigration time is OK
'gets the actual value
spetro = SPX_getIntTime(ByVal instname, integrationtime)
'Text3 = integrationtime
'find the wavelength, max and min if wanted
wave = SPX_getWavelengthData(ByVal instname, wavelengthdata(0), mini, Maxw)

'If PulseBox.Value = 1 Then
'If BeforePulse = False Then
'Pulse here and then take a different spectrum triggers the step on positive edge
'set a step and allow 0.5 s to equilibriate
'CWDIO1.SingleWrite 1
'tstarto = Timer
'Do
'Loop Until Timer - tstarto > CWSlide1.Value
'CWDIO1.SingleWrite 0
'Else
'End If
'End If

spetro = SPX_startScan(ByVal instname)
'get the data. CARE on the array name - seems to have to be with '(0)'
spetro = SPX_getScanData(ByVal instname, data(0))
'close
spetro = SPX_close(ByVal instname)
CWVisa1.Close  'actually stops crashing for the second time.

'If PulseBox.Value = 1 Then
'If BeforePulse = False Then
'Pulse here and then take a different spectrum triggers the step on positive edge
'set a step and allow at least 0.25 s to equilibriate
'CWDIO1.SingleWrite 1
'tstarto = Timer
'Do
'Loop Until Timer - tstarto > 0.25
'CWDIO1.SingleWrite 0
'Else
'End If
'End If


For i = 0 To 2999
    If wavelengthdata(i) - 490 > 0 Then
        If counter = 0 Then
        Well_125_490nm = data(i)
        Else
        Wavelength490nm = data(i)
        End If
    GoTo Got490
    End If
Next i

Got490:
```

```
For i = 0 To 2999
    If wavelengthdata(i) - 600 > 0 Then
    For j = i - 10 To i + 10
    Wavelength600nm = Wavelength600nm + data(j)
    Next j
    Wavelength600nm = Wavelength600nm / 21
    GoTo Got600
    End If
Next i

Got600:


'----
For i = 0 To 2999
    datatot(i) = datatot(i) + data(i)
Next i

CWGraph1.PlotXvsY wavelengthdata, datatot
Next k

If PulseBox.Value = 1 Then
If BeforePulse = False Then
'Pulse here and then take a different spectrum triggers the step on positive edge
'set a step and allow at least 0.25 s to equilibriate
CWDIO1.SingleWrite 1
tstarto = Timer
Do
Loop Until Timer - tstarto > 0.25
CWDIO1.SingleWrite 0
Else
End If
End If


For i = 0 To 2999
    data(i) = datatot(i) / CWSlide2.Value
Next i



'----

s = -1
    Total(X, Y) = Empty
    Ave(X, Y) = Empty
    Max(X, Y) = Empty
    Maxp(X, Y) = Empty

    Do
        s = s + 1


        Total(X, Y) = Total(X, Y) + data(s) 'puts our spectro data into the system

        If Abs(data(s)) > Maxp(X, Y) Then
        Max(X, Y) = data(s) '1 '(ScaledData(s))
        Maxp(X, Y) = data(s) '1 'Abs(ScaledData(s))
        End If
    Loop Until s >= 2999 'NumEditNoPoints.Value - 1



        Ave(X, Y) = (Total(X, Y) / 2999) ' / NumEditGain.Value
        Avep(X, Y) = Abs((Total(X, Y) / 2999))
        Max(X, Y) = Max(X, Y) ' / NumEditGain.Value
        KInt = 0

        Dim Xv()
        Dim Yv()
        'Dim Zv()
        ReDim Xv(XRange)
        ReDim Yv(YRange)
        'ReDim Zv(zRange)


        For w = 0 To (XRange - 1)
        'Xv(w) = XYZController.Combo1.Text + (w * XYZController.NumEditResolution(0))
        Xv(w) = OriginX + (w * XYZController.NumEditResolution(0))
        Next w

        For w = 0 To (YRange - 1)
        'Yv(w) = XYZController.Combo3.Text + (w * XYZController.NumEditResolution(1))
        Yv(w) = OriginY + (w * XYZController.NumEditResolution(1))
        Next w

        'For w = 0 To (zRange - 1)
        'Zv(w) = XYZController.Combo6.Text + (w * XYZController.NumEditResolution(2))
        'Next w





        Label2.Caption = Format$(Ave(X, Y), "0.00E-00")
        'Label6.Caption = Format$(Max(X, Y), "0.00E-00")
        If Avep(X, Y) > MaxAve Then
        MaxAve = 1.01 * Avep(X, Y)
        End If

        If Avep(X, Y) < MinAve Then
        MinAve = Avep(X, Y)
        End If

        If Maxp(X, Y) > MaxMax Then
        MaxMax = 1.01 * Maxp(X, Y)
        End If

        If Maxp(X, Y) < MinMax Then
        MinMax = Maxp(X, Y)
        End If



If BeforePulse = True Then
Ave_bef(X, Y) = Ave(X, Y)
Avep_bef(X, Y) = Avep(X, Y)
Total_bef(X, Y) = Total(X, Y)
Max_bef(X, Y) = Max(X, Y)
Maxp_bef(X, Y) = Maxp(X, Y)
W600_bef(X, Y) = Wavelength600nm
Else
Ave_aft(X, Y) = Ave(X, Y)
Avep_aft(X, Y) = Avep(X, Y)
Total_aft(X, Y) = Total(X, Y)
Max_aft(X, Y) = Max(X, Y)
Maxp_aft(X, Y) = Maxp(X, Y)
W600_aft(X, Y) = Wavelength600nm
W600_diff(X, Y) = W600_bef(X, Y) - W600_aft(X, Y)
End If
```

```
    If PulseBox.Value = 1 Then
        If BeforePulse = True Then
        CWGraph3D1.Plot3DSurface Xv, Yv, Max_bef
        Refresh
        Else
        CWGraph3D1.Plot3DSurface Xv, Yv, Max_aft
        CWGraph3D2.Plot3DSurface Xv, Yv, W600_diff
        Refresh
        End If
    Else
        CWGraph3D1.Plot3DSurface Xv, Yv, Max
    End If

'--== Bilayer-Detection ==--
If Check1.Value = 1 Then
    If counter > 0 Then
    Abs490nm(X, Y) = (Log(Well_125_490nm / Wavelength490nm)) - Wavelength600nm
    Else
    Abs490nm(X, Y) = 0
    End If
    CWGraph3D2.Plot3DSurface Xv, Yv, Abs490nm
    Refresh
End If








  Text1 = CStr(Max_aft(X, Y))
  Text2 = CStr(Max_bef(X, Y))


  Acquisition.Refresh

If PulseBox.Value = 1 Then
 If BeforePulse = True Then
       Open FileName & "_Before_Pulse.dat" For Output As #2
       Open FileName & "_Before_Pulse" & counter & ".dat" For Output As #3
 Else
       Open FileName & "_After_Pulse.dat" For Output As #2
       Open FileName & "_After_Pulse" & counter & ".dat" For Output As #3
 End If
Else
       Open FileName & ".dat" For Output As #2
       Open FileName & counter & ".dat" For Output As #3
End If


    ns = -1

   '===Round Up/Down Co-ordinate values===
 If XYZController.NumEditPosition(0) - Int(XYZController.NumEditPosition(0)) > 0.5 Then
 XYZController.NumEditPosition(0) = Int(XYZController.NumEditPosition(0)) + 1
 Else
 XYZController.NumEditPosition(0) = Int(XYZController.NumEditPosition(0))
 End If

 If XYZController.NumEditPosition(1) - Int(XYZController.NumEditPosition(1)) > 0.5 Then
 XYZController.NumEditPosition(1) = Int(XYZController.NumEditPosition(1)) + 1
 Else
 XYZController.NumEditPosition(1) = Int(XYZController.NumEditPosition(1))
 End If
```

```
    For b = xmin To xmax
     For a = ymin To ymax
     If Ave(b, a) <> Empty Then
     'Print #2, Yv(b) + Int(XYZController.NumEditPosition(1)) -
(Val(XYZController.Combo4.Text) * Val(XYZController.NumEditResolution(1))); ","; Xv(A) +
(XYZController.NumEditPosition(0)) - (Val(XYZController.Combo2.Text) *
Val(XYZController.NumEditResolution(0))); ","; Ave(A, b); ","; Max(A, b)
     Print #2, Yv(a); ","; Xv(b); ","; Ave(b, a); ","; Max(b, a)
     '","; W450NM(A, b); ","; W475NM(A, b); ","; W500NM(A, b); ","; W525NM(A, b); ",";
W550NM(A, b); ","; W575NM(A, b); ","; W600NM(A, b); ","; W625NM(A, b); ","; W650NM(A,
b); ","; W675NM(A, b); ","; W700NM(A, b)
     End If
     Next a
    Next b
    'Print #2, Wavelength600nm
    Close #2

     Print #3, XYZController.NumEditPosition(2); ","; XYZController.NumEditPosition(1);
","; XYZController.NumEditPosition(0)
     For i = 0 To 2999
       Print #3, wavelengthdata(i); ","; data(i)
       Next i

       Close #3



'Repeat
If PulseBox.Value = 1 Then
If BeforePulse = True Then
BeforePulse = False
Label6.Caption = "After"
GoTo repeatspec:
Else
BeforePulse = True
Label6.Caption = "Before"
End If
End If


'XYZController.objCommPort2.Output = "/1gS3I2A3000S3I2A0G6R" + Chr(13)


'End If


Text3.Text = FileName + "," + CStr(XYZController.NumEditPosition(0)) + "," +
CStr(XYZController.NumEditPosition(1)) + ","

'StackList.Text1.Text = StackList.Text1.Text + "Acquisition" + Chr(13) + Chr(10)
Call Form3.PotentialStep
'Call Form3.Command2_Click
'StackList.Text1.Text = StackList.Text1.Text + "Form3Ref" + Chr(13) + Chr(10)
'Call Form3.cmdStartXY_Click
'Call Form3.cmdStopXY_Click
'StackList.Text1.Text = StackList.Text1.Text + "Form3Start" + Chr(13) + Chr(10)
'tstarto = Timer
'Do
'Loop Until Timer - tstarto > 5
tstarto = Timer
Do
Loop Until Timer - tstarto > 60
counter = counter + 1
```

```
'Call XYZController.cmdTTL_Click
XYZController.NumEditCounter = counter
Label1.Caption = XYZController.NumEditPositions.Value - (counter + 1)
'If Label1.Caption = 0 Then Command1.Visible = True
Call XYZController.cmdScan_Click    '<-- Stack space error

End Sub

Private Sub Command1_Click()
XYZController.Visible = True
End Sub


Static Sub Form_Load()
StackList.Show
SaveStart = 0
Combo1.AddItem "X vs Y"
Combo1.AddItem "X vs Z"
Combo1.AddItem "Y vs Z"

BeforePulse = True
Label6.Caption = "Before"
CWAIPoint1.Device = 1
ChannelString = "0,1"
CWAIPoint1.Channels.Add ChannelString
CWDIO1.SingleWrite 0
End Sub

--- Module 1---
' -------------------------------------------------------------------------
'
'       Thorlabs SPx-USB - VB Header file for SP1-USB / SP2-USB Spectrometer instr.
driver
'  Do not modify the contents of this file.
'       Copyright:      Copyright(c) 2007, Thorlabs GmbH (www.thorlabs.com)
'
' -------------------------------------------------------------------------
'
'       Date:           Aug-21-2007
'       Software-Nr:   09.167.200
'       Version:        2.0
'       Changelog:              see 'readme.rtf'
'
' -------------------------------------------------------------------------
'
'       Disclaimer:
'
'       This library is free software; you can redistribute it and/or
'       modify it under the terms of the GNU Lesser General Public
'       License as published by the Free Software Foundation; either
'       version 2.1 of the License, or (at your option) any later version.
'
'       This library is distributed in the hope that it will be useful,
'       but WITHOUT ANY WARRANTY; without even the implied warranty of
'       MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
'       Lesser General Public License for more details.
'
'       You should have received a copy of the GNU Lesser General Public
'       License along with this library; if not, write to the Free Software
'       Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
'
' -------------------------------------------------------------------------


' -------------------------------------------------------------------------
' Functions
' -----------------------------------------------------------------------

' - Init/Close ---
Declare Function SPX_init& Lib "SPX_Drv_32.dll" (ByVal x1$, ByVal x2&, x3&)
Declare Function SPX_close& Lib "SPX_Drv_32.dll" (ByVal x1&)



' - Configuration Functions ---
Declare Function SPX_setIntTime& Lib "SPX_Drv_32.dll" (ByVal x1&, ByVal x2#)
Declare Function SPX_getIntTime& Lib "SPX_Drv_32.dll" (ByVal x1&, x2#)


' - Action/Status Functions ---
Declare Function SPX_startScan& Lib "SPX_Drv_32.dll" (ByVal x1&)
Declare Function SPX_startScanExtTrg& Lib "SPX_Drv_32.dll" (ByVal x1&)
Declare Function SPX_startScanCont& Lib "SPX_Drv_32.dll" (ByVal x1&)
Declare Function SPX_startScanContExtTrg& Lib "SPX_Drv_32.dll" (ByVal x1&)
Declare Function SPX_getDeviceStatus& Lib "SPX_Drv_32.dll" (ByVal x1&, ByVal x2$)


' - Data Functions ---
Declare Function SPX_getScanData& Lib "SPX_Drv_32.dll" (ByVal x1&, x2#)
Declare Function SPX_getRawScanData& Lib "SPX_Drv_32.dll" (ByVal x1&, x2%)
Declare Function SPX_getWavelengthData& Lib "SPX_Drv_32.dll" (ByVal x1&, x2#, x3#, x4#)


' - Utility Functions ---
Declare Function SPX_reset& Lib "SPX_Drv_32.dll" (ByVal x1&)
Declare Function SPX_errorMessage& Lib "SPX_Drv_32.dll" (ByVal x1&, ByVal x2&, ByVal
x3$)
Declare Function SPX_identificationQuery& Lib "SPX_Drv_32.dll" (ByVal x1&, ByVal x2$,
ByVal x3$, ByVal x4$, ByVal x5$)
Declare Function SPX_revisionQuery& Lib "SPX_Drv_32.dll" (ByVal x1&, ByVal x2$, ByVal
x3$)
Declare Function SPX_setUserText& Lib "SPX_Drv_32.dll" (ByVal x1&, ByVal x2$)
Declare Function SPX_getUserText& Lib "SPX_Drv_32.dll" (ByVal x1&, ByVal x2$)








' -----------------------------------------------------------------------
' Definitions
' -----------------------------------------------------------------------

' - Driver Error Codes ---
Global Const VI_WARN_SPX_DATA_NOT_READY = &H3FFC0901
Global Const VI_ERROR_READ_INCOMPLETE = &HBFFC0901


' - Buffers ---
Global Const SPX_BUFFER_SIZE = 256                          ' General buffer size
Global Const SPX_ERR_DESCR_BUFFER_SIZE = 512                    ' Buffer size for error
messages
Global Const SPX_TEXT_BUFFER_SIZE = 256                                         ' Buffer
size for texts from the LC1
Global Const SPX_NUM_PIXELS = 3000                                              ' this
is the number of effective pixels of CCD
Global Const SPX_MAX_USER_NAME_SIZE = 31                                ' this is max
number of characters for a user name
```

```
' - VISA strings ---
Global Const SP1_VI_FIND_RSC_PATTERN = "USB?*?{VI_ATTR_MANF_ID==0x1313 &&
VI_ATTR_MODEL_CODE==0x0111}"
Global Const SP2_VI_FIND_RSC_PATTERN = "USB?*?{VI_ATTR_MANF_ID==0x1313 &&
VI_ATTR_MODEL_CODE==0x0112}"
Global Const SPX_VI_FIND_RSC_PATTERN = "USB?*?{VI_ATTR_MANF_ID==0x1313 &&
((VI_ATTR_MODEL_CODE==0x0111) || (VI_ATTR_MODEL_CODE==0x0112))}"


' - Communication timeout ---
Global Const SPX_TIMEOUT_MIN = 1000
Global Const SPX_TIMEOUT_MAX = 60000

' - LC1 specific constants ---
Global Const SPX_MAX_INT_TIME = 0.2
' 200ms is the maximum integration time
Global Const SPX_MIN_INT_TIME = 0.000001                                        ' 1us
is the minimum integration time

' - scan states - these are the states reported by LC1_getDeviceStatus ---
Global Const SPX_SCAN_STATE_IDLE = &H0&
' LC waits for new scan to execute
Global Const SPX_SCAN_STATE_TRANSFER = &H20&                           ' scan is done,
waiting for data transfer to be finished
Global Const SPX_SCAN_STATE_IN_PROGRES = &H40&                 ' scan is in progres,
this message should never be get
Global Const SPX_SCAN_STATE_WAIT_TRIGGER = &H80&               ' same as IDLE except
that external trigger is armed
Global Const SPX_SCAN_STATE_UNKNOWN = &HFF&                            ' default value


' -------------------------------------------------------------------------
' End of file
' -------------------------------------------------------------------------
```

List of References

[1] Bard, A. J., Faulkner, L. R.; *Electrochemical Methods* (2nd Edition), p680 (2001)

[2] Heineman, W.R.; *Anal. Chem.* **50**, 390 (1978)

[3] Koch, D. F. A.; *Nature* **202**, 387 (1964)

[4] Koch, D. F. A., *Scaife, D. E.; J. Electrochem. Soc.* **113**, 302 (1966)

[5] Walker, D. C.; *Can. J. Chem.* **45**, 807 (1967)

[6] Bewick, A., Tuxford, A. M.; *Symp. Fara. Soc.* **4**, 114 (1970)

[7] Aylmer-Keller, A. W. B. *et al.*; *Fara. Disc. Chem. Soc.* **56** 96 (1973)

[8] Kuwana, T., Darlington, R. K., Leedy, D. W.; *Anal. Chem.* **36**, 2023 (1964)

[9] Murray, R. W., Heineman, W. R., O'Dom, G. W.; *Anal. Chem.* **39**,1666 (1967)

[10] Kuwana, T., Heineman, W. R., *Acc. Chem. Res*. **9**, 241(1976)

[11] Steckhan, E., Kuwana, T., *Ber. Bunsenges. Phys. Chem.* **78**, 253 (1974)

[12] Yildiz, A., Kissinger, P.T., Reilley, C. N.; *Anal Chem.* **40**, 1018 (1968)

[13] Strojek, I. W., Kuwana, T., Feldberg, S. W., *J. Amer. Chem. Soc*. **90**, 1353 (1968)

[14] Bard, A. J., Faulkner, L. R.; *Electrochemical Methods* (2nd Edition), p66 (2001)

[15] Bard, A. J., Faulkner, L. R.; *Electrochemical Methods* (2nd Edition), p29 (2001)

[16] Bard, A. J., Faulkner, L. R.; *Electrochemical Methods* (2nd Edition), p163 (2001)

[17] Bard, A. J., Faulkner, L. R.; *Electrochemical Methods* (2nd Edition), p682 (2001)

[18] Winograd, N., *Blount*, H. N., Kuwana, T.; *J. Phys. Chem*. **73**, 3456 (1969)

[19] Li, C., Wilson, G. S.; *Anal. Chem.* **45**, 2370 (1973)

[20] Lee, Y. F., Kirchhoff, J. R.; *Anal. Chem.* **65**, 3430 (1993)

[21] Zak, J., Porter, M. D., Kuwana, T.; *Anal. Chem.*, **55**, 2219 (1983)

[22] Heineman, W. R., Norris, B. J., Goelz, J. F.; *Anal. Chem.* **47** 79 (1975)

[23] Norris, B. J., Meckstroth, M. L., Heineman, W. R.; *Anal. Chem.* **48,** 630 (1976)

[24] Curran, D. J.; *Anal. Chem.* **64**, 2688 (1992)

[25] Pyun, C-H., Park, S-M.; *Anal. Chem.* **58**, 251 (1986)

[26] Shim, Y.-B. *et al.*; *J. Electrochem. Soc.* **137**, 538 (1990)

[27] Mho, S. *et al.*; *Bull. Korean Chem. Soc.* **15**, 739 (1994)

[28] Cai, M., Park S-M.; *J. Electrochem. Soc.* **143**, 2125 (1996)

[29] Shaw, M. J. *et al.*; *Inorg. Chem.* **49**, 9590 (2010)

[30] Zhao, M., Scherson, D. A.; *Anal. Chem.* **64***,* 3064 (1992)

[31] Tolmachev, Y. V. *et al.*; *Anal. Chem.* **70**, 1149 (1998)

[32] Schroll, C. A. *et al.*; *Anal. Chem.* **83**, 4214 (2011)

[33] Clarke, J. *et al.*; *Nature Nano.* **4**, 265 (2009)

[34] Syvitski, J. P. M.; *Principles, Methods and Application of Particle Size Analysis.* p109 (2007)

[35] Song, L. *et al.*; *Science* **274** 1859 (1996)

[36] Howorka, S., Cheley, S., Bayley, H.; *Nat. Biotechnol.* **19,** 636 (2001)

[37] Meller, A. and Branton, D.; *Electrophoresis* **23**, 2583 (2002)

[38] Astier, Y., Braha, O., Bayley H.; J. Am. Chem. Soc. **128**, 1705 (2006)

[39] Cockroft, S. L. *et al.*; *J. Am. Chem. Soc.* **130**, 818 (2008)

[40] Benner, S. *et al.*; *Nature Nanotech.* **2**, 718 (2007)

[41] Lieberman, K. R. *et al.*; *J. Am. Chem. Soc.***132**, 17961 (2010)

[42] B.M. Venkatesan, R. Bashir, *Nat. Nanotechnol.* **6,** 615 (2011)

[43] Branton, D. *et al.*; *Nat. Biotechnol.* **26,** 1146 (2008)

[44] Majd, S. *et al.*; *Curr. Opin. Biotechnol.* **21,** 439 (2010)

[45] Alberts, B. *et al.*; *The Lipid Bilayer,Molecular Biology of the Cell* (4th Edition), Garland Science, New York, (2002)

[46] Culshaw, B.,Dakin, J.;*Optical Fiber Sensors–Systems and Applications* (1989)

[47] J. B. Faria, *IEEE Transactions on Instrumentation and Measurement* **47**, 742 (1998)

[48] R. Reif, O Amar and I Bigio, *Applied Optics* **46**, No. 29, (2007)

[49] Puangmali, K. Althoefer and L. Seneviratne, *IEEE Transactions on Instrumentation and Measurement* **59**, No. 2, (2010)

[50] Zoski, Z. G.; *Handbook of Electrochemistry*, p592 (2007)

[51] Petrovic, S.; *The Chem. Edu.* **5**, 231 (2000)

[52] Jørgensen, K.; *Acta Chemica Scandanavia* **10**, 500 (1956)

[53] Jørgensen, K.; *Acta Chemica Scandanavia* **10**, 518 (1956)

[54] Macpherson, J. V.; Jones, C. E.; Unwin, P. R. *J. Phys. Chem. B,* 9891 (1998)

[55] Watkins, J. J.; Henry S. W.; *Langmuir* **20***,* 5474 (2004)

[56] Pletcher, D.; *A First Course in Electrode Processes* (2nd Ed.), P7 (2009)

[57] Atkinson, Kendall E.; *An Introduction to Numerical Analysis* (2nd Ed.), (1989)

[58] Mahon, P., Oldham, K.; *Anal. Chem.* **77**, 6100 (2005)

[59] Shoup D., Szabo, A.; *J. Electroanal. Chem.* **140**, 236 (1982)

[60] Flanagan, J. B., Marcoux, L.; *J. Phys. Chem.* **77,** 1051–1055 (1973)

[61] Vogel, A. I.; *A text book of quantitative inorganic analysis – theory and practice* (2nd ed.), (1951)

[62] Kaye, G. W. C.; Laby, T. H. *Tables of Physical and Chemical Constants and some Mathematical Functions* (12th ed.), p 231 (1959)

[63] Yang, Y. F., Denuault, G.; *J. Electroanal. Chem.* **418*,*** 99 (1996)

[64] Yang, Y. F., Denuault, G.; *J. Electroanal. Chem.* ***443***, 273 (1998)

[65] Yang, Y. F.; Denuault, G. *J. Chem. Soc. Faraday Trans.* ***92***, 3791 (1996)

[66] Pletcher, D., Sotiropoulos, S.; *J. Chem. Soc. Faraday Trans.* **90**, 3663 (1994)

[67] Bond, A. M. *et al.*; *J. Electroanal. Chem.* **249**, 1 (1988)

[68] Konopka, S. J. and McDuffie, B.; *Anal. Chem.*, **42**, 1741 (1970)