UNIVERSITY OF ALBERTA

Investigations Into Exploding Droplets

BY

Darcy B. Hager ©

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of **Doctor of Philosophy**.

DEPARTMENT OF CHEMISTRY

EDMONTON, ALBERTA

FALL 1997

0-612-22991-2

UNIVERSITY OF ALBERTA

LIBRARY RELEASE FORM

NAME OF AUTHOR:	Darcy B. Hager
TITLE OF THESIS:	Investigations Into Exploding Droplets
DEGREE:	Doctor of Philosophy
YEAR THIS DEGREE GRANTED:	1997

[illegible]

4535-33A Avenue
Edmonton, Alberta T6L 4X2

Date May 2, 1997

UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES AND RESEARCH

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled **Investigations Into Exploding Droplets** submitted by **Darcy B. Hager** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy**.

[illegible]
Dr. N.J. Dovichi

[illegible]
Dr. G. Horlick

[illegible]
Dr. D. J. Harrison

[illegible]
Dr. P. Kebarle

[illegible]
Dr. F. Pasutto

[illegible]
Dr. J. D. Henion

Date April 21, 1997

In loving memory of my brother

Abstract

The research presented stems from the direct visual observation of electrospray occurring as microscopic droplets pass through an electric discharge produced by a strong electrostatic field. This phenomenon has been called droplet electrospray or DES. An extremely fine and highly reproducible aerosol can be produced in this manner from each droplet.

In DES, a stream of uncharged droplets with a controlled radius between 6 and 50 μm and at frequencies in the 100 kllz range is produced by vibrating the capillary tip with a piezoelectric buzzer. The droplets are then charged by letting the droplet stream pass close to a 50 μm diameter Pt rod electrode that is at 3kV and is producing an electric discharge in the ambient air. The charged droplets are destabilized by the surface charge and emit a fine spray of offspring droplets. These events are regular and reproducible and can therefore be observed with a stroboscopic technique. The observed droplet fission is very similar in appearance to that reported by Gomez and Tang. The gas phase ions produced by the droplet fission (droplet spray) were detected with a mass spectrometer and their relative abundances determined vs. their spatial relationships with the spraying droplets. The mass spectra of the DES-produced ions are close to identical to spectra observed with conventional electrospray and were found to be well separated in space from the ions produced by the corona discharge. The present DES results provide some important insights into the mechanism of conventional ES.

Acknowledgments

I would like to thank all the people who have helped me with the research I've done during my time at the University of Alberta.

Above all, I would like to thank Dr. Norman Dovichi, my supervisor. I owe this man a great debt for his patience, trust, guidance, and encouragement. He never gave up on me and I will never forget that.

I would like to thank Dr. Paul Kebarle for his help, reassurance and many stimulating hallway conversations. The other members of my Ph.D. committee, Dr. Gary Horlick, Dr. Jed Harrison, Dr. Frank Pasutto, and Dr. Jack Henion have been very helpful with their comments and encouragement; I wish to thank them all.

Each of the past and present members of The Northern Lights Laser Lab are due a load of thanks for their help, friendship, and inspiration. Thanks guys. I would also like to thank Dr. Art Blades and Dr. John Klassen. These two gentlemen spent several hours helping me in the basement with the mass spectrometer.

Finally, I thank with all my heart, my family. My parents, Dewey and Twyla have shown steadfast support over the years in every way possible. I thank my little sister Rebecca for her unwavering faith in me. Most of all, my wife Sally deserves a million thanks for providing love, moral support and encouragement throughout my degree program. I appreciate all the sacrifices she selflessly made in support of my goal, and the inspiration she gave me to reach for it.

Table of Contents

List of Figures

Symbols and Abbreviations

% RA	relative abundance expressed as a percentage
AC	alternating current
AMU	atomic mass unit
API	atmospheric pressure ionization
BNC	baby *N* connector
CCD	charge coupled device
CI	chemical ionization
CID	collision induced dissociation
CPS	counts per second
DC	direct current
DES	droplet electrospray
DLI	direct liquid introduction
EI	electron impact
ES	electrospray
ESI	electrospray ionization
ESMS	electrospray mass spectrometry
HPLC	high performance liquid chromatography
LED	light-emitting diode
LYS	lysine
MS	mass spectrometry
MS/MS	Tandem MS

PHE	phenylalanine
PTFE	polytetrafluoroethylene
SDS	sodium dodecylsulfate
TTL	transistor-transistor logic
UV	ultraviolet
VCR	video cassette recorder

CHAPTER 1:

GENERAL INTRODUCTION

1.1 INTRODUCTION

It has been known for a long time that droplets with sufficient electrostatic charge become unstable and will shed some of this destabilizing charge in a rapid explosive event. The resulting daughter droplets are much smaller than the original parent, are themselves highly charged and, if they lose any mass through evaporation, are prone to the same fate as their parent.

The relatively new ionization method called electrospray ionization (ESI) is based on carrying the process of evaporation and sudden droplet disruption to its ultimate conclusion: the production of bare ions in the gas phase. Strictly speaking, ESI is not an ionization technique as the ions produced are already present in solution as ions. It is, however, an elegant way of extracting a useable quantity of such ions from solution without imparting a large amount of energy into the ion, which could lead to the breaking of molecular bonds. Because ESI is an important technique and because it relies on an electrostatically induced cascade of droplet disruptions, the work described in this dissertation endeavored to examine this disruption phenomenon.

Chapter 2 describes a technique developed to control and observe the phenomenon of droplet disruption over a range of sizes and introduces the concept of droplet electrospray, or DES. Control was achieved by using a mechanism that produced an extremely regular stream of identically sized droplets. The droplets were charged by passing them through a corona discharge. The observations were made by using stroboscopic illumination and video microscopy and provided detailed visual data showing the changes that a droplet undergoes before, during, and after its explosive disruption. The images also provided evidence of the universality of the disruption phenomenon for droplets of different sizes.

When ions that were present in the solution of the spraying droplet were detected by mass spectrometry, further experimentation showed that DES was useful as a source of ions for mass spectrometry. The results showed sensitivity comparable to conventional ES and are presented in Chapter 3.

Because the point in space where the droplet disruption event took place could be precisely controlled, measurements of the type and abundance of gas phase ions in the vicinity of the spraying droplet were possible. These measurements were made using mass spectrometry and showed graphically where and when solution ions in different degrees of solvation appeared around the disrupting droplets. Chapter 4 presents the results of these measurements. It also describes the steps necessary to stabilize the DES apparatus sufficiently to allow for the time-consuming measurements necessary to map the ion abundances. The work of Chapter 5 benefited directly from these increases in the stability of the instrumentation.

Chapter 5 describes preliminary results using the DES apparatus as an interface between packed-column high performance liquid chromatography (HPLC) and mass spectrometry (MS).

Chapters 3, 4 and 5 are all based on measurements made using MS results. Chapter 3 uses MS to assess DES as a source of ions, Chapter 4 uses MS to map the ion abundances present near an exploding droplet and Chapter 5 uses MS as a detector for high performance liquid chromatography. Mass spectrometry is introduced in the next section.

1.2 MASS SPECTROMETRY

1.2.1 Ion Source

In a mass spectrometer, a sample is ionized, and then the positive or negative ions are separated according to their mass-to-charge ratio and detected. All these operations are carried out in a high vacuum, typically 10^{-5} - 10^{-9} torr. This low pressure gives the ions a large enough mean free path for them to traverse the instrument with a low likelihood of colliding with any other molecule or ion.

Figure 1.1 shows a typical electron impact (EI) ion source. In this ion source, the sample is bombarded by high-energy electrons. Any ions produced are extracted by the repeller, and they then enter the mass analyzer. This type of ion source generates ions with a large amount of internal energy. Because of this excess energy, most of the ions will immediately break apart, giving rise to a pattern of fragments that will be unique for each substance.

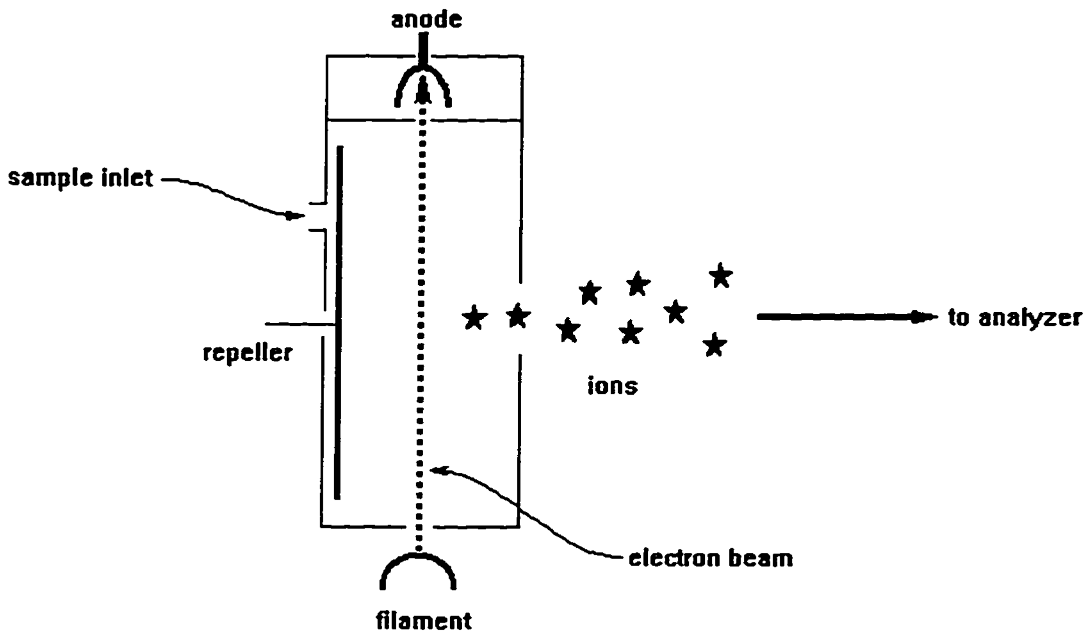

Figure 1.1 Electron Impact Ion Source. Neutral molecules entering through the sample inlet are charged when subjected to bombardment with high-energy electrons. The ions produced are pushed out of the ion source by the repeller. This method of ionization imparts substantial internal energy to the ions usually resulting in their rapid fragmentation.

Many types of molecules are particularly susceptible to fragmentation — so much so that an intact molecular ion may not be detected when an EI source is used. In cases like these, a second method of ionization is commonly used. This type of ion source, called chemical ionization (CI), makes use of a reagent gas to transfer charge to the analyte molecules and in so doing can be much gentler, resulting in an abundant molecular ion. In the usual case where methane is used as the reagent gas, electron impact ionization of the neutral methane produces a reagent plasma containing CH_5^+ and $C_2H_5^+$ ions. The analyte molecules, M, react with this plasma to produce mostly MH^+ ions by proton exchange. Because of the high ion source pressures needed in CI, this technique requires higher pumping capacity than EI mass spectrometry.

Soft ionization sources, such as CI, are valuable when the molecular ion is absent or weak in EI. This situation is common with the large, fragile molecules found in samples of biological interest. Other soft ionization sources for mass spectrometry include field desorption, field ionization[1], fast atom bombardment and laser desorption.

1.2.2 Mass Analysis

The second component of a mass spectrometer is the mass analyzer. It provides the means for separating the ions produced in the ion source according to their mass-to-charge ratio (m/q). In normal operation, only one nominal mass (an integral number of atomic mass units) is allowed to pass through the analyzer and reach the detector. Through appropriate adjustment of some instrumental parameter, the mass analyzer can be scanned over a large range, typically 10-1000 atomic mass units. Several designs of mass analyzer are used. Historically the two most popular have been the magnetic sector and the quadrupole mass analyzer.

Figure 1.2 shows the magnetic sector mass analyzer. Ions enter from the ion source and are accelerated by a high voltage V. This accelerating voltage imparts kinetic energy to the ion according to:

$$\frac{1}{2}mv^2 = Vq \qquad (1)$$

where: m = mass
v = velocity
V = voltage
q = charge

The high energy ions then enter the magnetic sector where they are deflected by a magnetic field B through an arc of radius R. The centrifugal force exactly balances the force due to the magnetic field:

$$\frac{mv^2}{R} = qVB \qquad (2)$$

Substituting (1) into (2) and solving for kinetic energy gives:

$$\frac{1}{2}mv^2 = \frac{q^2B^2R^2}{2m} \qquad (3)$$

Substituting (1) into (3) and rearranging gives the working equation for the magnetic sector mass analyzer:

$$\frac{m}{q} = \frac{B^2R^2}{2V} \qquad (4)$$

In the typical instrument, B and R are kept constant, and the m/q allowed to pass through the mass analyzer is selected by varying the accelerating voltage V.

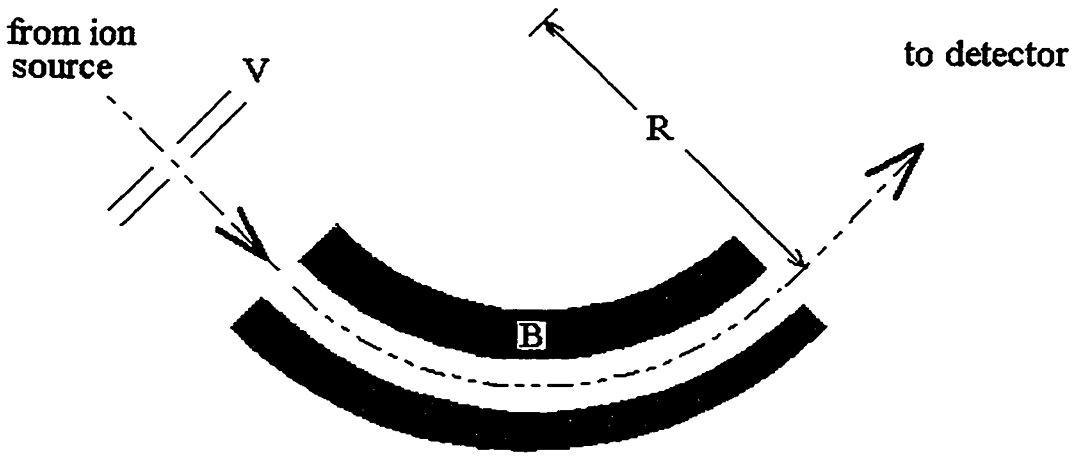

Figure 1.2 Magnetic Sector Mass Analyzer. Through a combination of acceleration through a voltage, V, and deflection through a magnetic field, **B**, ions of a specific mass/charge ratio are separated from the others and delivered to the detector.

The other common mass analyzer used in chromatography-mass spectrometry is the quadrupole mass analyzer. Figure 1.3 shows the design of this system. The ion is accelerated only moderately (~10eV) towards the four electrodes. The opposite pairs of electrodes are connected electrically and the voltage applied between each pair has a DC component U and an AC component $V_0\cos\omega t$. The ion entering the space between the electrodes will experience an electric field given by:

$$f = \frac{(U + V_0 \cos\omega t)(x^2 - y^2)}{r_0^2} \qquad (5)$$

where $2r_0$ = electrode spacing

and x and y are coordinates in the plane of the hyperbolae

This equation applies to electrodes of hyperbolic cross-section. However, circular electrodes give similar results and are easier to manufacture to the high tolerances required. For a given value of U and V_0 , ions of certain m/q will have stable, oscillatory trajectories through the quadrupole while all others will hit an electrode and not reach the detector. The mass analyzer is scanned across a range of m/q by varying U and V_0, while keeping U/V_0 constant.

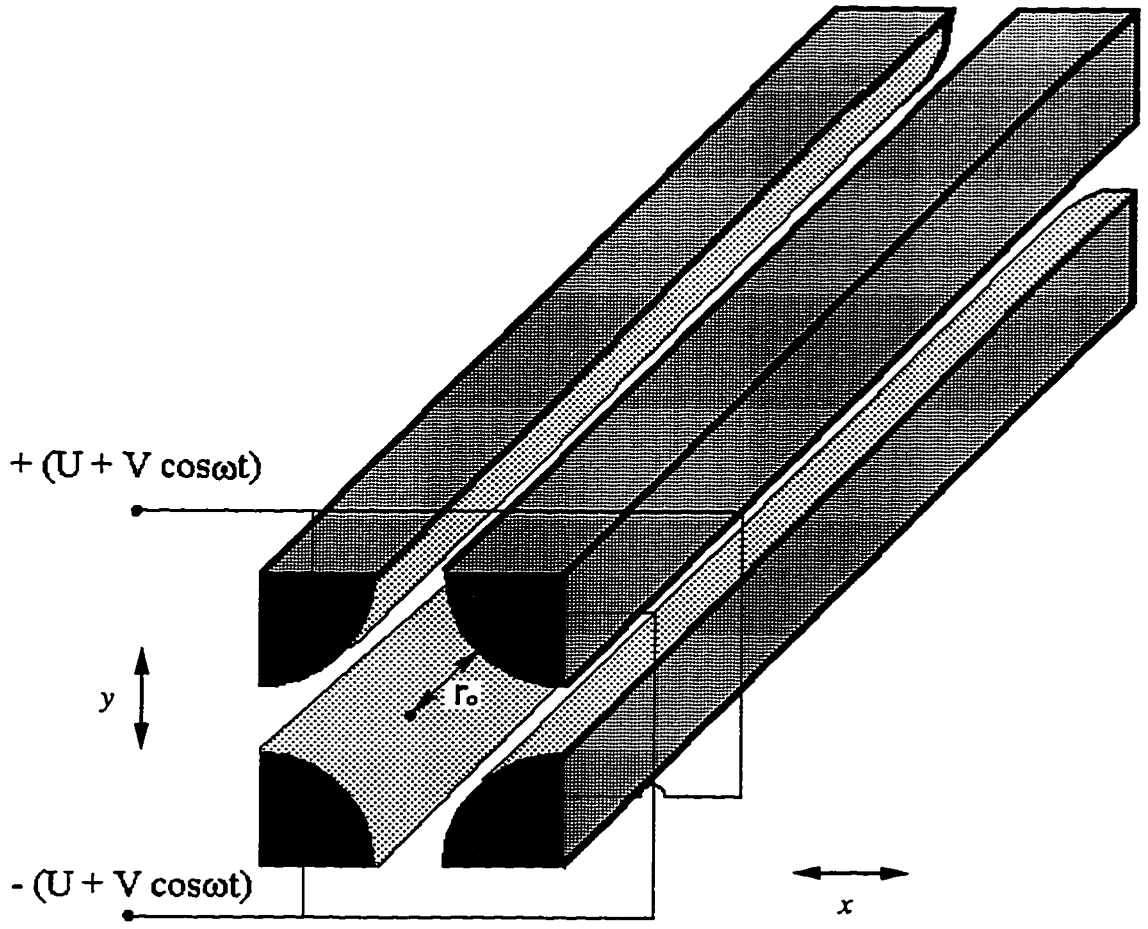

Figure 1.3 Quadrupole Mass Analyzer. Certain values of DC voltage, U, and AC amplitude, V, will allow ions of a very narrow range of mass-charge ratio to assume a stable oscillatory trajectory and thus be able to traverse the quadrupole and reach the detector. A mass spectrum is obtained by scanning these values.

Another mass analyzer that has become increasingly popular is the quadrupole ion trap.[2] In this design, ions are held in a chamber defined by three special electrodes: two end cap electrodes and a ring electrode. The end cap electrodes are normally held at ground potential while a radio frequency potential is applied to the ring electrode to produce a quadrupole electric field. Ions in the trap follow stable trajectories if their m/q ratio falls within a region of stability defined by the operating voltage applied to the ring electrode. By scanning the amplitude of this voltage, ions of increasing m/q adopt unstable trajectories and are ejected from the trap and detected using an ion detector. The ion trap has some very attractive features including high sensitivity, small size and the ability to do tandem MS. Tandem MS, or MS/MS, is the technique of using a mass analyzer to select an ionized species, subjecting the selected parent ion to some means of raising its internal energy causing it to fragment, then analyzing the daughter ions. In an ion trap, the parent ion is retained while all other ions are ejected. A collision gas can then be introduced to cause fragmentation after which another mass analysis scan can be performed. The process may be repeated many times in a technique called MS^n and is very useful in the structure elucidation of large complex molecules.

When used as a chromatographic detector, the mass spectrometer is usually only required to resolve to the nearest nominal mass. It must, however, be able to scan through its entire mass range in a time period less than the width of a typical chromatographic peak, which can be as short as 2 seconds. The quadrupole mass analyzer is usually preferred over the magnetic sector in benchtop instruments because of the former's relatively smaller size and faster scanning rate. Because of its high sensitivity, relatively compact construction, and ability to do MS^n, the ion trap mass analyzer is now becoming popular.

1.2.3 Ion Detection

Ions selected by the mass analyzer, either sector, quadrupole or ion trap, are detected by an electron multiplier. In this device, the ions impact an electrode, causing secondary electron emission. The signal is amplified by either a series of dynodes or a specially prepared channel that multiplies the initial electron pulse by as much as 8 orders of magnitude (see Figure 1.4). The surface of the dynodes or the channel electron multiplier is coated with a compound, usually BeO, GaP, or CsSb, that ejects several electrons when subjected to the impact of a high energy electron or ion. Like the mass analyzer, the detector must also have a rapid response. Typical electron multipliers have a response time in the 0.5-2 nsec range. Arrays of the channel-type of electron multiplier, or *channeltron*, have been used for simultaneous mass spectral detection.[3]

1.2.4 Vacuum System

All three sections of the mass spectrometer—ion source, mass analyzer, and detector— are kept at about 10^{-5} - 10^{-9} torr. This pressure is accomplished by means of one or more of either a diffusion pump, turbo-molecular pump, or cryogenic pump. Since the samples are introduced into the mass spectrometer via the ion source, most of the pumping capacity is devoted to this region. When interfaced to a chromatograph, the pumping system must have sufficient capacity to handle the load imposed by the mobile phase.

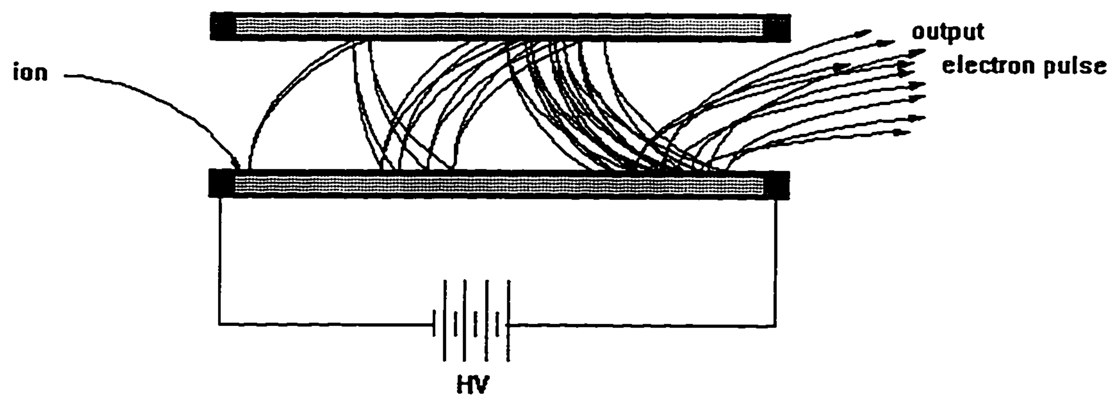

Figure 1.4 Channel Type of Electron Multiplier. Ions striking the inside surface of this type of detector, also known as the channeltron, cause emission of 2 or more electrons. These electrons will also strike the surface farther down the tube each causing another emission. The process repeats itself, each time magnifying the total charge traveling toward the detector. The total magnification may be as high as 8 orders of magnitude.

1.3 ELECTROSPRAY IONIZATION

Invented by Dole[4] in 1968, and reintroduced in 1984 by Fenn[5], electrospray ionization for mass spectrometry (ESI) has become very popular. It provides a simple means for the transfer of ionic species from solution to the gas phase without requiring unusual liquid matrices, such as the glycerol used in fast atom bombardment (FAB)[6] ion sources and without imparting significant internal energy into the resulting ions. This last point is important when working with large and fragile biomolecules.

The apparatus required for producing an electrospray is typically quite simple. The solution of interest is pumped through a metal capillary of 0.2 mm o.d. and 0.1 mm i.d. that is connected to a high voltage DC power supply. A voltage of ±2-3 kV is applied to the capillary and the solution emerging from the capillary tip is drawn into a cone by the electric field. A jet of small charged droplets is emitted from the tip of this cone and the droplets rapidly disperse into an aerosol.

The aerosol droplets drift away from the capillary under the influence of the electric field, losing mass by evaporation. As they shrink, their surface charge becomes more concentrated eventually reaching a point where the surface tension holding the droplet together is overcome by coulombic repulsion.

The point at which this occurs is given by the Rayleigh equation:

$$Q_R^2 = 64\pi^2\varepsilon_0\gamma R_R^3 \qquad (6)$$

where Q_R is the maximum charge the droplet can support, ε_0 is the permittivity of vacuum and γ is the surface tension of the liquid forming the droplet. As the droplet shrinks, Q_R decreases, eventually approaching Q, the actual charge on the droplet. The droplet in this situation then undergoes a rapid fission whereby about 2% of its mass and 15% of its charge are lost.[7,8]

The mechanism of electrospray ionization as applied to mass spectrometry was the subject of a recent review by Kebarle and Tang.[11]

Horlick and Agnes have recently applied ESI-MS to quantitative and qualitative elemental analyses[12-14]. In their studies, they have developed methodologies for increasing the linear range over which various inorganic ions may be determined.

1.4 DROPLET PRODUCTION BY THE FORCED BREAKUP OF A LIQUID JET

Schneider and Hendricks described a method for the production of streams of liquid droplets that were extremely uniform both in size and spacing relative to one another.[15] The method used was based on theory developed by Rayleigh[16]. Rayleigh showed that a cylinder of liquid will collapse to a collection of droplets when subjected to an axially symmetric sinusoidal perturbation in the column's radius if the wavelength of the perturbation is greater than the circumference of the liquid cylinder. This theory also applies to a moving jet of liquid and was presented in detail by

Schneider and Lindblad in 1965[16]. The important results from reference 16 are summarized below.

In order to produce a jet of liquid by forcing the liquid through a capillary, the velocity of the liquid exiting the capillary must be great enough to provide sufficient energy to create fresh surface. The minimum average velocity v necessary to form a jet of radius a of a liquid having density ρ and surface tension T is given by:

$$v > 2\left(\frac{T}{a\rho}\right)^{1/2} \qquad (7)$$

Liquid exiting the capillary at a velocity less than this value will form a hanging drop instead of a jet. The jet diameter $2a$ is determined by the inside diameter D of the capillary and is on the order of 0.8*D.

An important consequence of using small diameter capillaries and high flow rates is the high pressure drop across the length of the capillary. The Poiseuille equation relates pressure drop, flow, and viscosity for a capillary of length l and internal radius a:

$$F = \frac{\pi a^4}{8\mu l}(P_1 - P_2) \qquad (8)$$

where F is the volume flow rate, μ is the viscosity, and P_1 and P_2 are the mean pressure values at each end of the capillary. For example, pumping methanol (viscosity 0.547 cp) at a flow of 0.500 mL/min through a capillary of radius 10.5 μm and length 1.00 cm creates a pressure differential of 9.55 MPa (1390 psi) across the length of the capillary.

A piezoelectric transducer is commonly used to apply a mechanical disturbance to the capillary that transmits it to the liquid jet. A periodic disturbance results in undulations in the surface of the jet that rapidly progress into the breakup of the jet into a train of regularly spaced droplets. Each undulation gives rise to a droplet, therefore the frequency of the signal used to drive the piezoelectric vibrator is equal to the frequency of droplet formation. The volume of liquid V in each droplet produced by this method is therefore:

$$V = \frac{F}{f} \tag{9}$$

where f is the vibration frequency. The droplet's radius r is given by the formula for the volume of a sphere:

$$r = \left(\frac{3V}{4\pi}\right)^{1/3} \tag{10}$$

or, by substitution:

$$r = \left(\frac{3F}{4\pi f}\right)^{1/3} \tag{11}$$

and the velocity of the jet v and hence the initial velocity of the droplets is given by:

$$v = \frac{F}{\pi a^2} \tag{12}$$

Since electronic function generators having frequency accuracy on the order of 1 ppm are readily available, the size of the droplets produced can be determined to an accuracy limited by the accuracy of the volumetric flow rate, typically <1%.

While the size of the droplets can be changed by changing flow rate and vibration frequency, many factors influence the sizes achievable in practice. For a given capillary, minimum useable flow is set by the requirement of jet formation and limited to a maximum determined by the pressure handling capability of the liquid delivery system. Therefore, capillary size must be changed to produce droplets over a wide range of sizes. If droplet velocity needs to be considered, it may place additional constraints on the apparatus as velocity is also determined by capillary size and liquid flow rate.

There are limits to the vibration frequencies that can be used as well. These limits are determined by the natural resonances present in the transducer and capillary and must be chosen by experiment. Many vibration frequencies result in the formation of satellite droplets between the main droplets. These small droplets form from the collapse of a filament of liquid connecting adjacent droplets as they are forming from the jet. Slight adjustment of the vibration frequency or flow rate can either eliminate the formation of satellite droplets or cause them to form close enough to the main droplet to be quickly absorbed.

Finally, there is another problem that becomes significant when very small inside diameter capillaries are used: plugging. Care must be taken to eliminate insoluble material from solutions pumped through fine capillaries.

1.5 HPLC-MS INTERFACES

1.5.1 General Interfacing Considerations

Modern high resolution analytical chromatographic instrumentation provides the means to quickly and efficiently separate small amounts of very complex mixtures into their components. Through careful choice of the chromatographic system — liquid, gas, or supercritical fluid, a very wide range of samples can be separated, from the volatile and highly stable to the involatile or thermally labile. By using a mass spectrometer to detect the components as they elute from the chromatograph, each sample's mass spectrum can be obtained. This information is useful in the identification of unknowns, confirmation of peak purity, and highly selective detection. In addition, the sensitivity of the mass spectrometer often exceeds conventional chromatographic detectors.

An ideal interface takes the eluant from the chromatograph, removes the chromatographic mobile phase (except where that is used as reagent in chemical ionization), and quantitatively introduces the components into the ion source of the mass spectrometer. It does all this without degrading the performance of the chromatograph or the mass spectrometer. The interface should also facilitate the changing of the chromatographic column without requiring a shutdown and restart of the attached mass spectrometer.

1.5.2 High Performance Liquid Chromatography

Fourteen years ago[17-19], only two types of LC-MS interface were available: the moving belt and the direct liquid introduction (DLI) methods. In the moving belt design, column effluent was deposited onto a belt and passed by an infrared heater that helped to evaporate the solvent. The sample was first carried through a two-stage vacuum lock into the high vacuum of the ion source housing and then vaporized near the electron beam. A powerful heater removed any trace of sample remaining on the belt before the belt looped around and was reused. The belt was made of polyimide and could withstand temperatures of up to 400°C. This interface can be used with ion sources in either EI or CI modes and can handle column flows ranging from 0.4 mL/min for water to 2 mL/min for volatile organic solvents. The chromatographic performance of the moving belt interface was studied by Lankmayr and coworkers[18] and found to be affected by type of solvent, flow rate, mode of sample transfer onto the belt, belt speed, and vaporizer temperature. They found that the mode of sample transfer onto the belt was the most critical of these parameters and that dramatic improvements in peak shape could be obtained by using a spray deposition method. Upon optimization of all parameters, they were able to obtain total-ion chromatograms that were virtually identical to those recorded by HPLC-UV.

A common fault of the moving belt interface was memory effects caused by a) incomplete removal of the deposited sample and b) deposition of sample on sealing surfaces contacted by the belt as it went from the high pressure to low pressure sections of the interface. Mizuno and coworkers[20] developed a double belt LC-MS interface that eliminated this memory effect. In this design, the LC effluent was deposited on a belt and the solvent evaporated as before. A second belt was laid down on top of the first and kept the coated surface from contacting and contaminating the sealing surfaces. The two belts were separated again once they entered the high vacuum. After sample desorption the belts were wound on take-up spools for disposal and not recycled. Other problems with the moving belt design remain, however. It tends to give poor mass spectra with polar compounds due to thermal decomposition. In addition, volatile samples are not transferred quantitatively.

The second approach to LC-MS interfacing, direct liquid introduction (DLI), was reviewed by Niessen[21,22] in detail. The first component found in DLI interfaces is a restrictor that is used to limit the flow of column effluent into the mass spectrometer. Early restrictor designs were prone to plugging so modern DLI interfaces use a thin diaphragm, sometimes augmented by piezoelectric crystal vibrators, to overcome this problem. Often, the column effluent is split; only a portion of the effluent passes through the pinhole and the bulk bypasses to waste.

The second component in the DLI interface is the desolvation chamber. This chamber is where the aerosol droplets produced by the restrictor outlet evaporate. Heat must be added to the system, as it was calculated that in an entirely adiabatic process, a droplet of acetonitrile at room temperature will freeze (at -92°C) after the evaporation of only 23% of the liquid[20]. The vaporized column effluent then enters the ion source. Because of the large amount of solvent introduced into the ion source, this interface necessitates use of chemical ionization, with the chromatographic mobile phase acting as reagent gas.

To generate EI spectra, the "particle beam interface" was developed[23] (See Figure 1.6.). This design is similar to the DLI design up to the desolvation chamber. At the downstream end of this chamber, instead of entering the ion source directly, the dried particles of analyte are formed into a supersonic beam by a small diameter

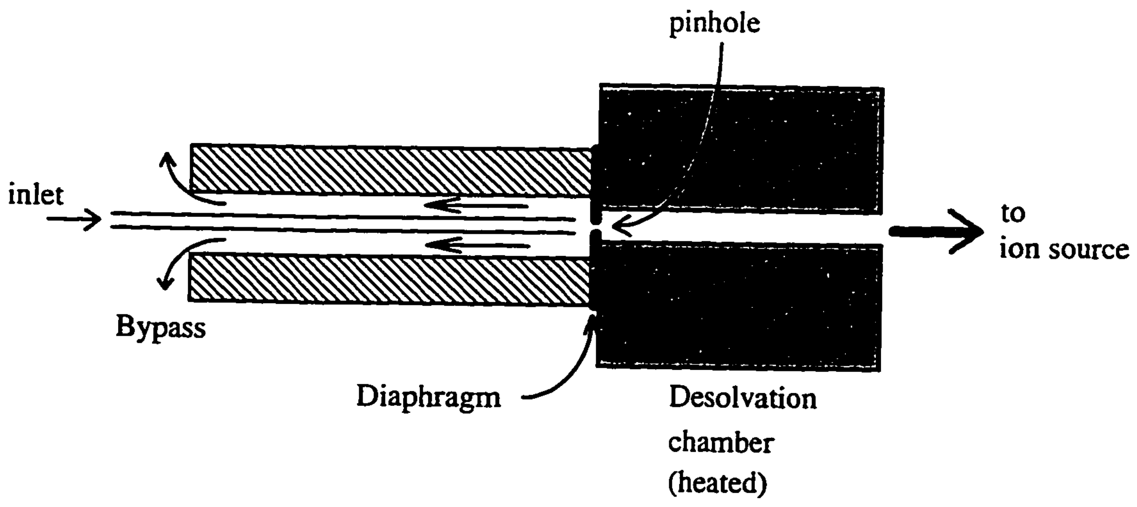

Figure 1.5 DLI Restrictor/Splitter. A fraction of the column effluent passes through the pinhole where it is vaporized in the heated desolvation chamber and introduced into the ion source of the mass spectrometer. The rest of the column effluent bypasses the pinhole. The split ratio can be adjusted by changing the pinhole size or the solvent flow rate.

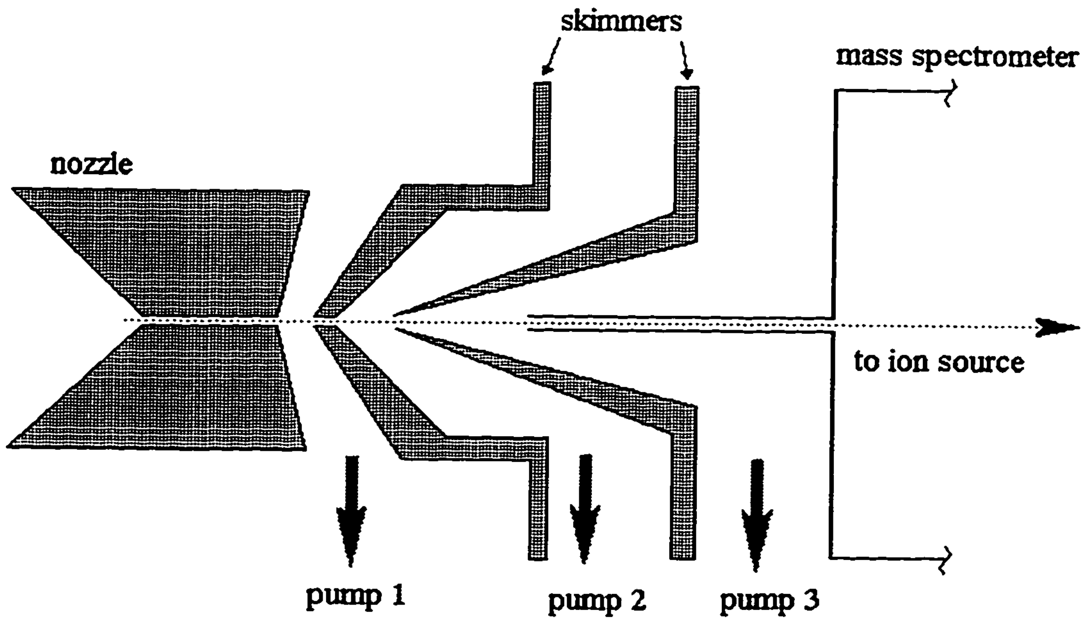

Figure 1.6 Particle Beam Interface. This design is similar to the DLI design up to the desolvation chamber. At the downstream end of this chamber, instead of entering the ion source directly, the dried particles of analyte are formed into a supersonic beam by a small diameter nozzle. The analyte is preferentially introduced into the mass spectrometer by directing the beam though two skimmers with strong pumping between the nozzle and the first skimmer and between the first and second skimmers. This design passes a substantial fraction of solute while removing sufficient solvent to allow generation of EI mass spectra.

nozzle. The analyte is preferentially introduced into the mass spectrometer by directing the beam though two skimmers with strong pumping between the nozzle and the first skimmer and between the first and second skimmers. This design passes a substantial fraction of solute while removing sufficient solvent to allow generation of EI mass spectra.

Recently, atmospheric pressure ionization has become popular as a source of ions[24]. This method, which encompasses thermospray, ion spray, and electrospray techniques, combines the interface and the ion source into one unit that operates under ambient pressure. Flow rates used are on the order of 40 μL/min, typical for micro-LC. Higher flow rates, such as those used with conventional HPLC columns, can be accommodated with a post-column split, but not as Henion describes "without some experimental care".[24] Chapter 5 presents work done in assessing the viability of using droplet electrospray (DES) as the basis of a HPLC-MS interface using conventional HPLC columns.

1.5.3 Computing Considerations

The successful interfacing of chromatography and mass spectrometry makes available a huge amount of data. Collecting a complete mass spectrum from m/q = 50 to 500 every second for a 30 minute chromatogram requires a computer with sufficient storage capacity for 800 kilobytes per chromatographic analysis, at 8 bit resolution. In addition, most chromatography data systems provide graphical display of the total ion chromatogram, selected ion chromatogram(s), and the mass spectrum of any point throughout the chromatographic run. Provisions are usually made in the software to subtract the background spectrum obtained from the baseline near the peak of interest. Quantitative analyses can be done with the assistance of peak area integration routines and, because the raw data is saved, reintegration can be done readily using different parameters. Smoothing and peak-finding algorithms can be applied after the analysis helping the analyst to extract more useful information from the raw data. The data may also be easily and safely archived indefinitely on magnetic or optical media.

Often mass spectral libraries are available for immediate searching to compare the mass spectra of unknown peaks to those of known compounds. The availability of fast microprocessors and inexpensive mass storage has put the finishing touch on this powerful technique.

1.6 REFERENCES

1. Beckey, H. D., *Principles Of Field Ionization And Field Desorption Mass Spectrometry*, Pergamon Press, NY (1977)
2. Cooks, R. G.; Glish, G. L.; McLuckey, S. A.; Kaiser, R. E., *C&E News*, March 25, 26-41 (1991)
3. Tuithof, H. H.; Boerboom, A. J. H.; Meuzelaar, H. L. C., *Int. J. Mass Spectrom. Ion Phys.*, **17**, #3, 299-307 (1975)
4. Dole, M; Mack, L. L.; Hines, R. L.; Mobley, R. C.; Ferguson, L. D.; Alice, M. B., *J. Chem. Phys.*, **49**, 2240 (1968)
5. Fenn, J. B.; Yamashita, M., *J. Phys. Chem.*, **88**, 4451-4459 (1984)
6. Niessen, W. M. A., *J. Chromatogr. Libr.*, **59**, 8–9 (1996)
7. Gomez, A.; Tang, K., *Physics of Fluids*, **6**, 404-414. (1994)
8. de la Mora, J. F., *J. Coll. Int. Sci.*, **178**, 209-218 (1996)
9. Blades, A. T.; Jayaweera, P.; Ikonomou, M. G.; Kebarle, P., *Int. J. Mass Spectrom. and Ion Processes*, **102**, 251 (1990)
10. Blades, A. T.; Jayaweera, P.; Ikonomou, M. G.; Kebarle, P., *Int. J. Mass Spectrom. and Ion Processes*, **101**, 325 (1990)
11. Kebarle, P.; Tang, L., *Anal. Chem.*, **65**, 972A-986A (1993)
12. Agnes, G. R.; Horlick, G., *Appl. Spectrosc.*, **46**, 401 (1992)
13. Agnes, G. R.; Horlick, G., *Appl. Spectrosc.*, **48**, 649-654 (1992)
14. Agnes, G. R.; Horlick, G., *Appl. Spectrosc.*, **48**, 655-661 (1992)

15. Schneider, J. M.; Hendricks, C. D., *Rev. Sci. Ins.*, **35,** 1349-1350 (1964)

16. Schneider, J. M.; Lindblad, N.R., *Rev. Sci. Ins.*, **42,** 635-638 (1971)

17. Willard, H. H.; Merritt, L. L.; Dean, J. A.; Settle, F. A., *Instrumental Methods Of Analysis*, 585-591, Wadsworth, (1981)

18. Lankmayr, E. P.; Hayes, M. J.; Karger, B. L.; Vouros, P.; and McGuire, J. M., *Int. J. Mass Spec. Ion Phys.*, **46,** 177-180 (1983)

19. Dobberstein, P.; Korte, E.; Meyerhoff, G.; Pesch, R., *Int. J. Mass Spec. Ion Phys.*, **46,** 185-188 (1983)

20. Mizuno, T.; Azuma, K.; Otsuka, K., *Analytical Sciences*, **4,** 541-246 (1988)

21. Niessen, W. M. A., *Chromatographia*, **21,** 277-287 (1986)

22. Niessen, W. M. A., *Chromatographia*, **21**, 342-354 (1986)

23. Woodfin, V. L.; Dorn, S. B., *Anal. Chem.*, **62,** 2573-2580 (1990)

24. Henion, J. D.; Wachs, T. W.; Conboy, J. C.; Garcia, F., *J. Chrom. Sci.*, **29**, 357-366 (1991)

CHAPTER 2:

DROPLET CHARGING AND DISRUPTION

2.1 INTRODUCTION

In the typical electrospray ion source, a liquid solution is pumped through a metal capillary to which a high voltage has been applied. Upon leaving the tip of the capillary, the solution forms a thin jet. After a short distance, this jet destabilizes and breaks into a fine aerosol that disperses rapidly since each droplet carries a charge of the same polarity. This phenomenon, also known as *electrohydrodynamic nebulization*, has been described by Smith and others.[1-3] The resulting charged droplets can give rise to ions for which a model called *ion evaporation* has been proposed by Iribarne and Thomson.[4,5] After production, the droplets in the electrospray aerosol undergo further reductions in size, ultimately leading to ions. Fenn and coworkers developed this technology into an ion source for mass spectrometry.[6,7] The mechanism by which the charge is acquired by the aerosol has been described by Kebarle and coworkers.[8]

In this chapter, I report a method whereby the droplet formation and subsequent droplet charging and disruption are decoupled. By vibrating the tip of a capillary at high frequency, a reproducible droplet stream is formed. The droplet stream passes an electrode held at high potential; droplets passing this electrode rapidly acquire charge and undergo electrostatic disruption. Finally, a microscope is used to observe the droplet stream; when illuminated with a pulsed light source that is phase referenced to the droplet formation rate, a stable stroboscopic image is generated of the droplet disruption processes.

2.2 EXPERIMENTAL

Figure 2.1 is a schematic diagram of the system. Droplets were produced from a stream of acetone emitted from a 10 μm inner diameter fused silica capillary by vibrating the capillary with a piezoelectric buzzer.[9,10] Droplet production rate equaled the buzzer frequency. By controlling the flow with a grounded high pressure syringe pump and the vibration frequency with a high precision function generator, the volume of each droplet could be calculated. Typical flows were in the range of 2 mL/hr with droplet production rates on the order of a few hundred kHz, the exact frequency being determined by the natural resonance of the capillary/transducer assembly. Droplets were typically 1-5 pL in volume (6-11 μm radius spheres).

The stream of droplets was directed past a 50 μm diameter platinum wire that had been cut with a razor blade at an angle of approximately 45^{o}. Several methods of cutting the Pt wire were tried including shears, wire cutters and scalpel blades. The decision to slice the wire at a 45^{o} angle was based on the poor reproducibility of the other methods and their tendency to leave the wire bent and having a jagged tip. This electrode was attached to a high voltage power supply. The electrode was fixed to the laboratory bench while the droplet generator assembly was attached to a 3-axis translation stage.

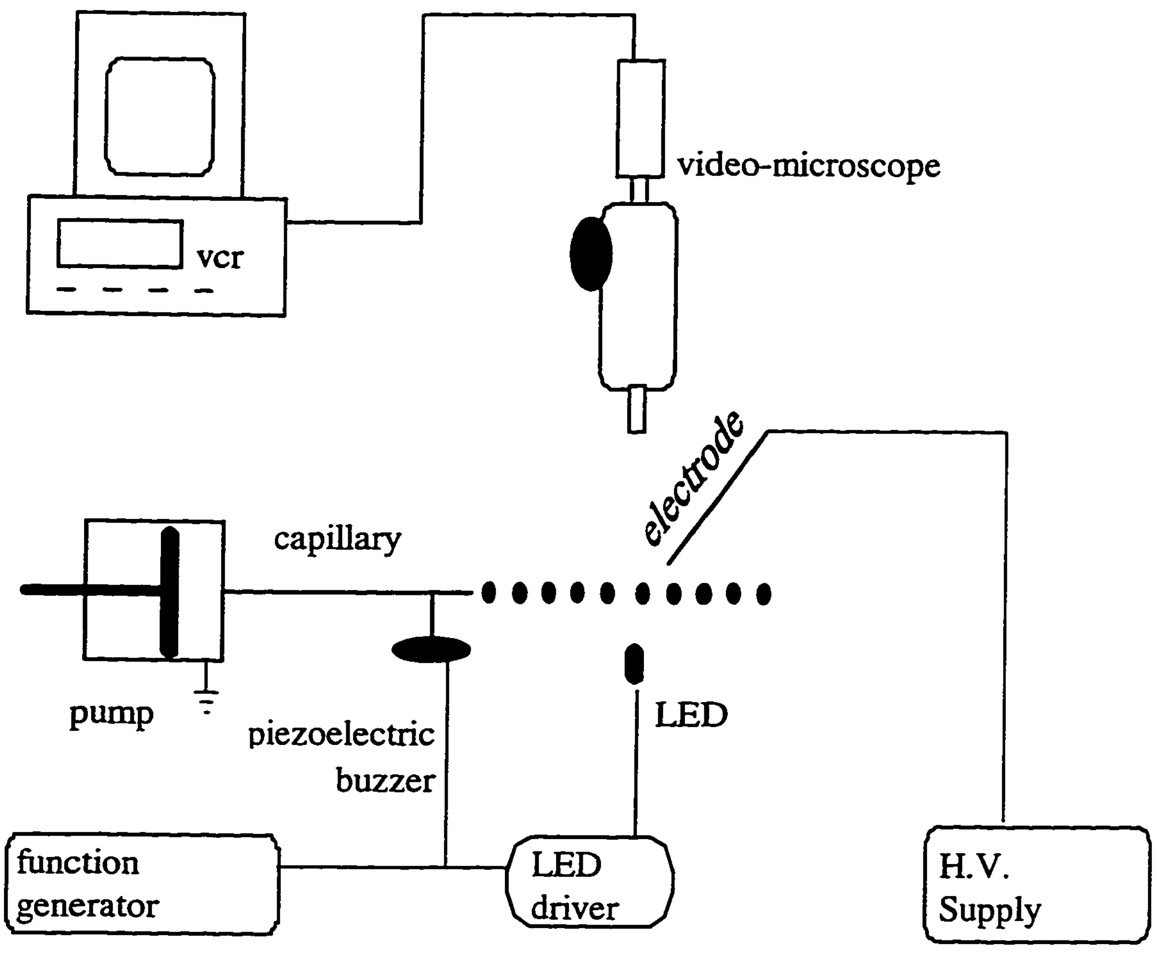

Figure 2.1 Apparatus for generation and observation of droplet electrospray. The droplet stream passing in front of the video microscope is illuminated by light from the light-emitting diode (LED). This light is pulsed by the LED driver and the pulses syncronized with the signal driving the piezoelectric buzzer. The video images were displayed on a monitor and recorded on tape using a video cassette recorder (VCR).

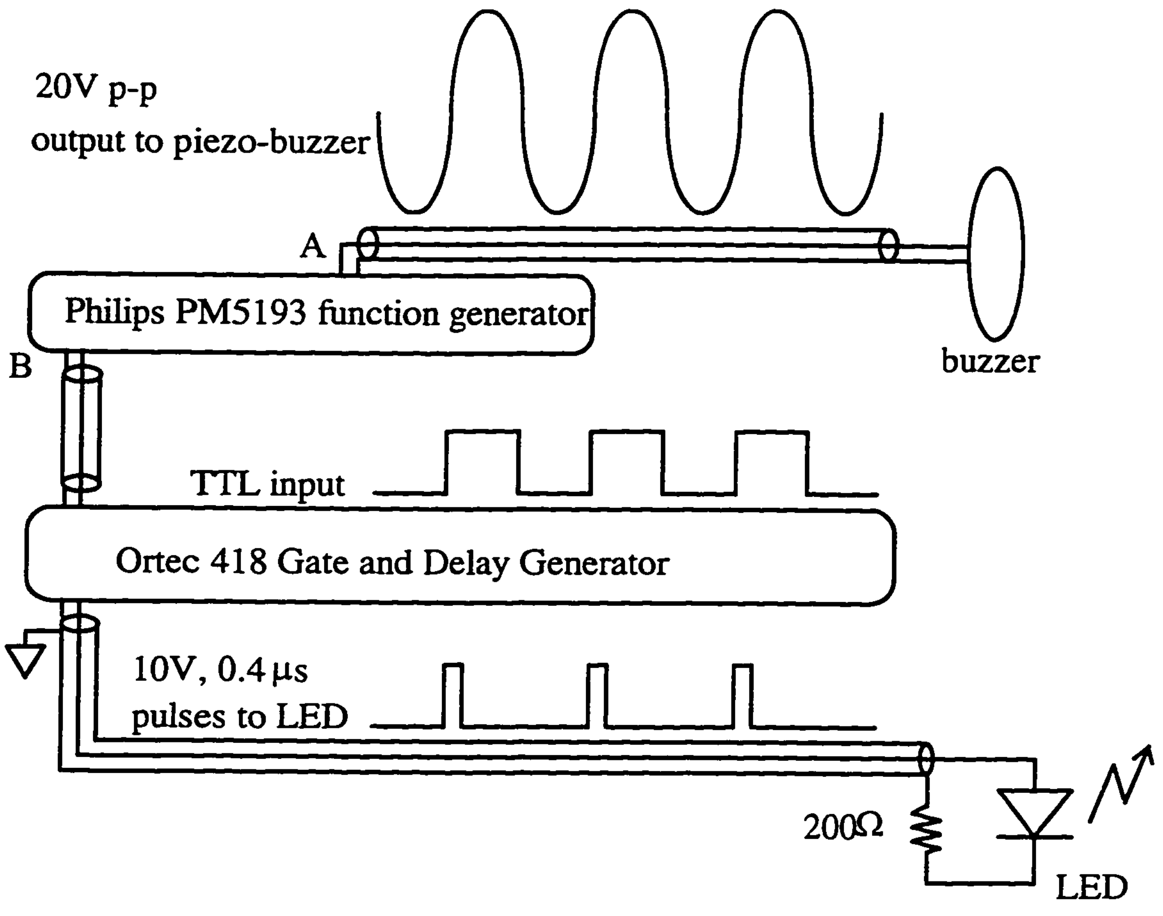

Figure 2.2 Electrical signals used in the droplet generation apparatus.

A microscope was used to observe the droplet stream. The microscope body was machined from Delrin. Delrin was used because it was not electrically conductive and would therefore not deflect the charged droplets. A 4X, 0.08 numerical aperture microscope objective lens was fitted to one end of the Delrin tube and a Sony model XC-73 CCD video camera attached to the other end. The video signal was fed to a Magnavox model VR3213AT01 video cassette recorder. The video images were viewed on a Panasonic model CT-110-MCA video monitor.

The 20V p-p sine wave output, A, of a Philips model PM5193 programmable synthesizer/function generator was used to drive the piezoelectric transducer. The function generator also had a TTL compatible output, B, that was used to trigger an Ortec model 416 gate and delay generator, labeled LED Driver in Figure 2.1. This LED driver produced 10V, 0.4 μs pulses that were synchronized with the droplet generator. These pulses were used to power a red light emitting diode (LED). This flashing light source provided stroboscopic illumination.

Figure 2.2 illustrates the electrical interconnections and Figures 2.3-2.7 show the images obtained. The stroboscopic back-lighting provided by the LED made the droplets appear dark against a bright background.

A videographic record was made on VHS video tape. Selected frames from the video were digitized with a Scion model LG-3 frame grabber and Scion Image 1.55 software on a Macintosh computer. In addition to the video images, photographs were taken with a 35 mm camera on black and white film. For these, the CCD camera was replaced with the film camera and Kodak Plus-X ISO 100 film was exposed for 15 seconds.

To better visualize the fine aerosols emitted by the disrupting droplets, the LED was switched off and a laser beam introduced from the side, along the path of the droplet stream. The light beam was small enough to illuminate only the aerosol droplets therefore, when photographed in a darkened room, the only source of light for the image was laser light reflected from the droplets. The intensity of the light beam was sufficient to make visible the fine spray plume emitted by the exploding droplets. Figure 2.5 shows the aerosol illuminated by laser light. In this picture, the droplets appear as bright images on a dark background. The laser beam was pulsed using an acoustooptic modulator and was phase-referenced to the droplet production frequency. Therefore a stroboscopic image was obtained showing the evolution of the aerosol sprayed off the exploding droplets. Figure 2.8 depicts the optical arrangement used for this image. The aerosol generated by each exploding droplet is shown much more clearly using this form of illumination.

2.3 RESULTS

At negative electrode potentials, the droplet stream was attracted towards the electrode in proportion to the voltage. At voltages more negative than about -2 kV, any liquid that contacted the electrode was sprayed off the tip in the same manner as a classical electrospray.

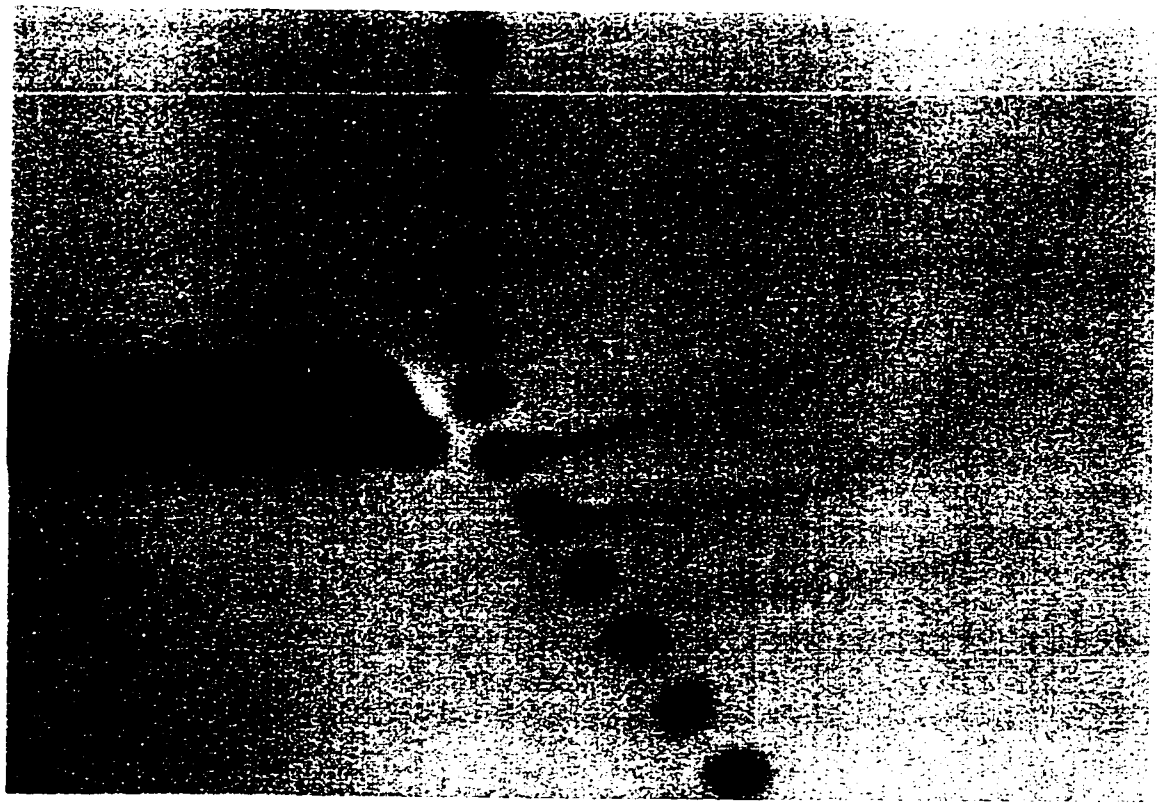

Figure 2.3a Droplet electrospray from 16-μm diameter droplets of acetone. Droplets are generated at 380 kHz and at a flow rate of 2.0 mL/hr. The electrode is at 6,000 volt potential. The droplets pass through the corona discharge, barely visible at the electrode tip, picking up charge. The charged droplets are oriented and deflected by the electric field. Slightly below the electrode, the droplets generate a pointed end, and a fine spray is emitted from that end. The spray is generated over a 50-μm distance down steam from the electrode. The spray appears to detach from the droplet and drift away. No spray is observed further downstream.

The most interesting phenomena occurred when positive voltages were applied. Below about +2.5 kV, the droplet stream was attracted towards the electrode in proportion to the voltage. However, above +2.5 kV, which was also the point where an electric discharge began to appear, the behavior of the droplets became more complex. Droplets that were upstream from the electrode were still attracted toward the electrode; this attraction was relatively minor and only could be observed when the droplet velocity was low. However, once past the electrode, the droplets were repelled from the electrode.

Above about +3.5 kV, as the droplets approached close to the tip of the electrode, a very small jet of liquid appeared on the side of the droplet opposite to the electrode, Figure 2.3 and Figure 2.7. The path followed by a low velocity droplet is shown in Figure 2.4. The slight initial attraction and subsequent repulsion are illustrated by the lines drawn over the photograph.

The high velocity droplets shown spraying in Figure 2.7 underwent much less perturbation of their trajectory when they encountered the high electric field near the electrode. These fast moving droplets were also well past the electrode when the electrospray jet formed and it was noted that the point on the droplet where the jet originated did not substantially rotate around the droplet as was the case with the slow moving droplets when their jets formed near the electrode, Figure 2.3 and Figure 2.6.

After moving out of the region of high field, the jet stopped, the aerosol detached from the droplet and drifted away while the droplet continued along its new trajectory. The aerosol plume emitted from the spraying droplet was best seen with laser illumination as in Figure 2.5.

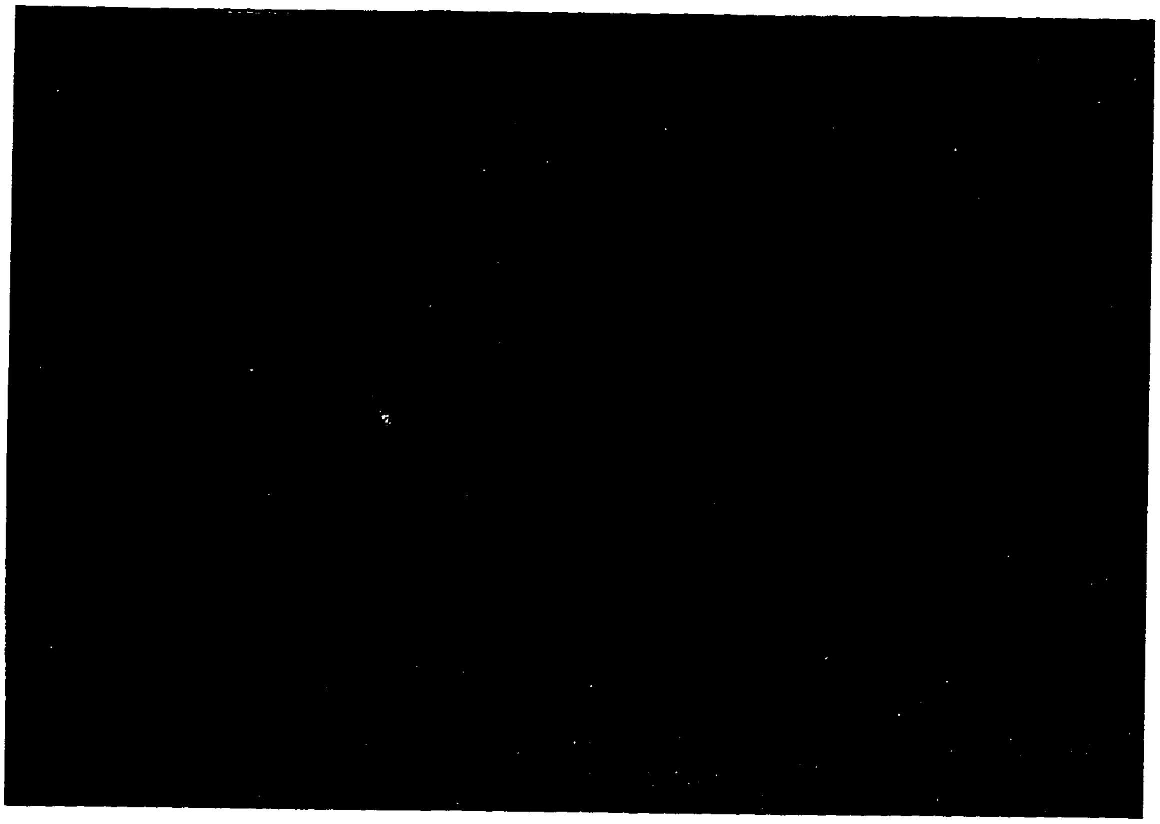

Figure 2.3b Droplet electrospray from 21-μm diameter acetone droplets. Droplets are generated at 254 kHz and at a flow rate of 4.0 mL/hr. The electrode is at 4,000 volt potential. Electrospray from adjacent droplets is generated at nearly 90°.

Figure 2.6 presents 11 images of a droplet in various stages of charging, distorting, spraying, and tumbling. In the first 4 images, the droplet is acquiring charge and beginning to distort, developing a cone-shaped protrusion on the side of the droplet opposite the electrode. The electrospray jet originates at the tip of the cone then shoots off in the direction of the electric field at that point. During the lifetime of the jet, the emission may form an arc, as seen in the ninth image of Figure 2.6.

When the droplet's velocity past the electrode was slow enough that the spray began near the electrode, the position of the emission cone was observed to rotate as the droplet moved so as to maintain its alignment with the electric field. This rotation resulted in the production of the arc-shaped spray and also left the droplet tumbling after the spray stopped.

Methanol droplets were also produced and exhibited the same behavior near the high voltage as acetone. The droplet stream was traced downstream for several centimeters. No subsequent electrospray phenomenon was observed.

2.4 DISCUSSION

In conventional electrospray, the high electric field at the tip of the capillary leads to both formation of a jet of droplets and the disruption of the droplets through formation of the electrospray. The wide distribution of droplet size and the large region in space where disruption occurs makes it difficult to visualize the disruption process.

On the other hand, the droplet electrospray described in this paper separates droplet formation, droplet charging, and droplet disruption in three distinct stages. Because of the periodic formation of the droplet and controlled charging by the electrode, the droplet disruption is remarkably reproducible, allowing stroboscopic observation of the phenomenon. The photographs in Figures 2.3a and 2.3b represent accumulated exposures from roughly 10^5 experiments; the sharp features reflect the stability of the system. Furthermore, adjustment of the stroboscope phase allows direct visualization of the spray's evolution. Figure 2.6 shows in a sequence of images how the shape of the droplet changes throughout the process of being charged, spraying, and relaxing after the spray plume detaches. Beginning at the left of the figure, the droplet's shape becomes elongated as it encounters the electrostatic field and the flow of positive ions emanating from the corona discharge. The fourth image from the left shows a Taylor cone beginning to form on the upper surface of the droplet. Subsequent images illustrate the progression of the droplet electrospray event. The elapsed time between images is approximately 2μs. This same behavior was found over all droplet sizes. The velocity of the droplet stream did have an affect on the direction of spray plume; the emission from fast moving droplets remained oriented in the same direction for the duration of the spray while the direction of the spray from slower moving droplets was found to rotate. Since the slower droplets began spraying while still quite close to the electrode, the direction of the electric field was strong enough to swing the polarized droplet around as it passed by the electrode. This motion produced an arc-shaped electrospray aerosol and left the parent droplet tumbling afterward. Fast droplets, on the other hand, did not spray until well past the electrode, where the field was not as strong, Figure 2.6. The spray was straighter, the images more stable for longer periods of time, and the trajectories of the parent droplets were much less affected by the whole process.

We believe that the droplet is charged by passage through the corona discharge at the tip of the positive electrode. According to Bailey, drops or particles may be charged by passage near a corona discharge in two ways: diffusion charging and field charging.[11] The former mechanism is a process of ion attachment based on random encounters between the particle and airborne ions in the absence of an external electric field. In the presence of a uniform electric field, such as that found some distance from the most common corona discharge sources, the effective cross-sectional area of the particle available for the capture of ions is increased by the field's influence on both the particle and the ion cloud. Even though the electric field experienced by the droplets was non-uniform due to their proximity to the charging electrode, it was reasonable to assume that this field played a significant role in rapidly charging the droplets to a sufficient extent to lead to their immediate disruption.

The ions acquired by the droplets have been shown to be produced in a sequence of ion/molecule reactions, initiated by the ionization of moist air.[12-14] Beginning with ionized oxygen and nitrogen molecules, hydronium ion–water clusters are rapidly produced via the following reaction scheme:

$$e^- + N_2 \Rightarrow N_2^+ + N_2 \Rightarrow N_4^+ + O_2 \Rightarrow O_2^+ \qquad (1)$$

$$e^- + O_2 \Rightarrow O_2^+ + O_2 \Rightarrow O_4^+ \qquad (2)$$

$O_2^+, O_n^+ + H_2O \Rightarrow O_2^+(H_2O)$ (3)

$O_2^+(H_2O) + H_2O \Rightarrow O_2^+(H_2O)_2 \Rightarrow H_3O^+(OH) + O_2$ (4)

$H_3O^+(OH) + H_2O \Rightarrow H_3O^+(H_2O) + OH$ (5)

The successive addition of water molecules to the hydronium ion–water clusters results in a distribution of ions of the form $H_3O^+(H_2O)_n$ where n is reported to range from 5 to 8 at thermal equilibrium in moist air.[13] In practice, other ions were also observed in the mass spectrum of the corona discharge and were attributed to solvent vapours and other contaminants present in the laboratory environment. Examples included $CH_3OH_2^+$, $NH_4^+(H_2O)$, $NH_4^+(CH_3OH)$, and other combinations of protonated ammonia, water and methanol. In addition to this charging process, a strong electric field must also be present before the droplet electrospray phenomenon is observed by our stroboscopic technique. This experimental condition ensures a stable, reproducible, highly oriented spray.

The attraction of droplets to both the negative and positive electrode can be explained by the polarization of the droplet in the non-uniform electric field near the electrode tip. Attraction was best observed at voltages below that which induced the droplet electrospray phenomenon, however as illustrated in Figure 2.4, can also be seen in slower moving droplets that did undergo disruption. The attraction due to droplet polarization is much weaker than the repulsion which results after the droplets accumulate some positive charge.

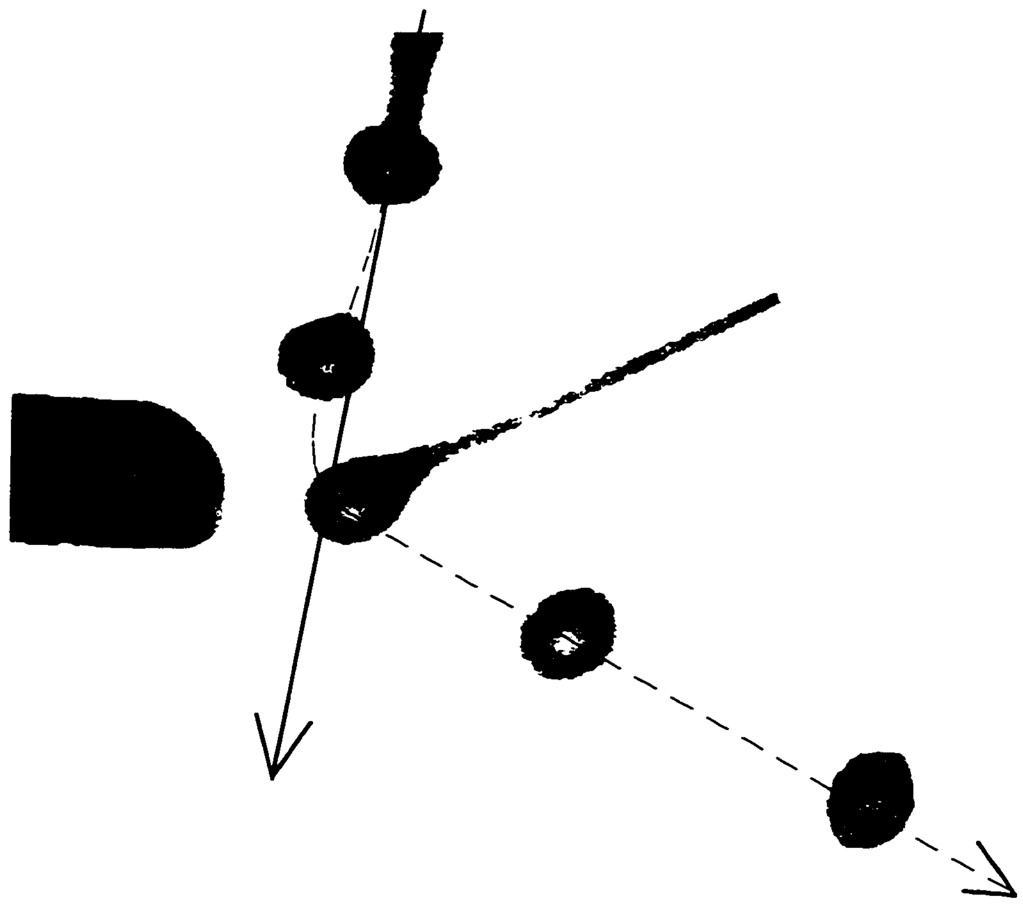

Figure 2.4 The path followed by slow droplets. In this figure the lines show the initial attraction and subsequent repulsion of the droplet by the charging electrode.

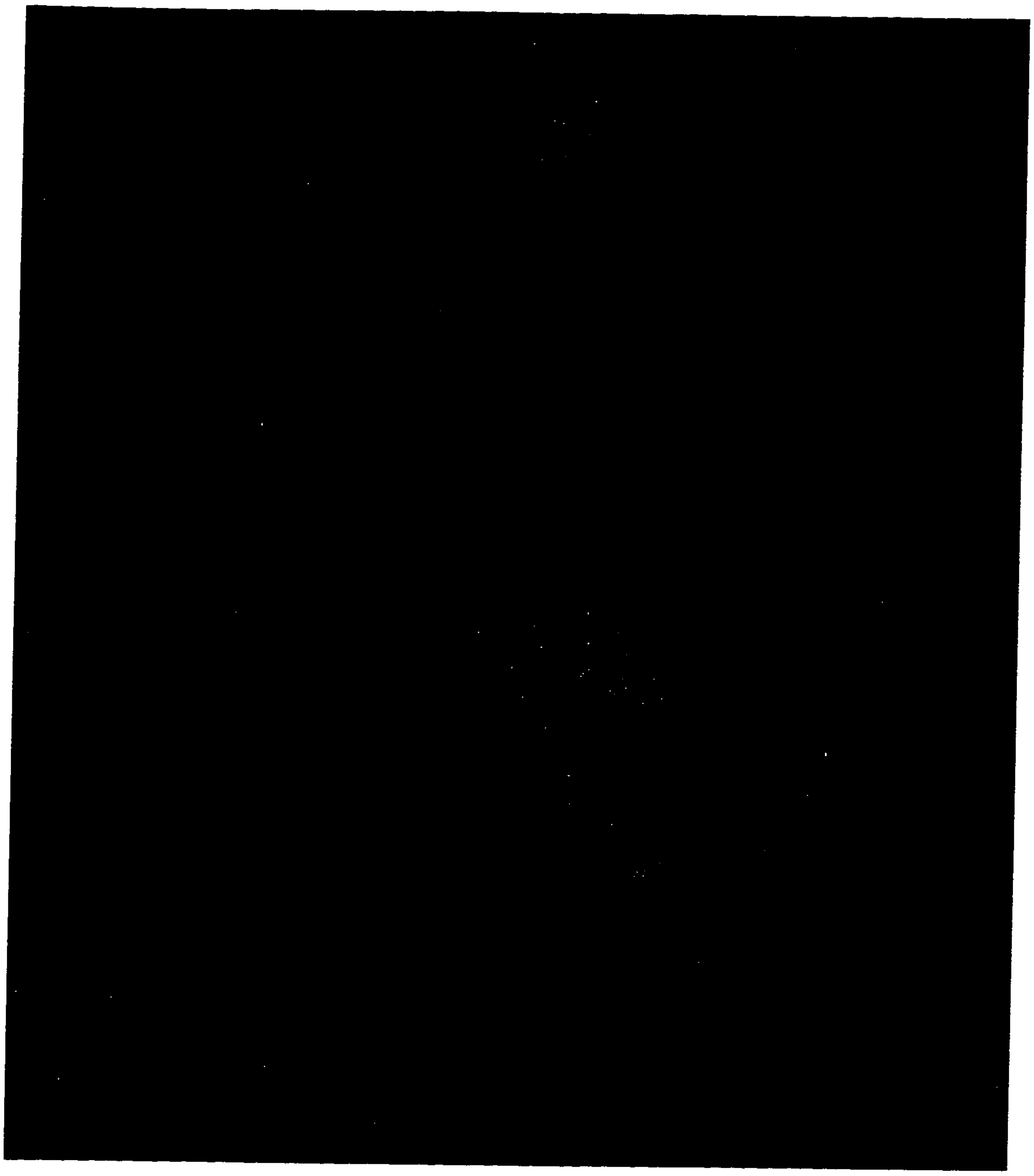

Figure 2.5 Droplet electrospray illuminated with pulsed laser light. The droplets and the spray aerosol are seen as bright images on a dark background.

Negative potentials resulted in the production of an electric discharge at a substantially lower voltage than did positive potentials. Negative potentials also failed to charge the droplets; under negative potentials, we failed to observe droplet deflection from the vicinity of the electrode. We presume that the droplets are not charged under negative potentials, which may reflect differences in the ion chemistry of the discharge or may simply be a consequence of the lower electric field strengths attainable with negative potentials.

The droplet polarization that occurs under the influence of the electric field, combined with the acquisition of charge from its passage through the corona lead to the observation of the droplet electrospray event. The fact that each droplet sprays in the same direction implies the electric field is orienting them. Observation of the Taylor cone's reorientation in slower moving droplets is consistent with this explanation. This behavior is only possible because they are polarized. A consequence of this polarization is that the droplets must be disrupting at well below the Rayleigh limit of charge. The charge on the droplet surface is concentrated around the point of lowest field strength and locally becomes sufficient to cause the spray to erupt. Overall, however, the entire charge on the droplet would not suffice to cause its disruption were it evenly distributed over the droplet's surface.

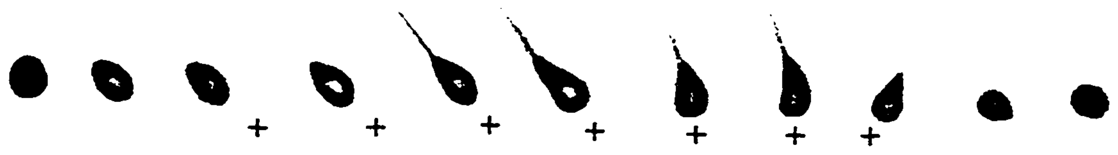

Figure 2.6 A sequence showing the evolution of a spraying droplet charged by the corona discharge electrode. Where the droplet is close to the charging electrode, the position of the electrode is indicated in the figure with a + symbol. The direction of droplet motion is from the left of the figure to the right.

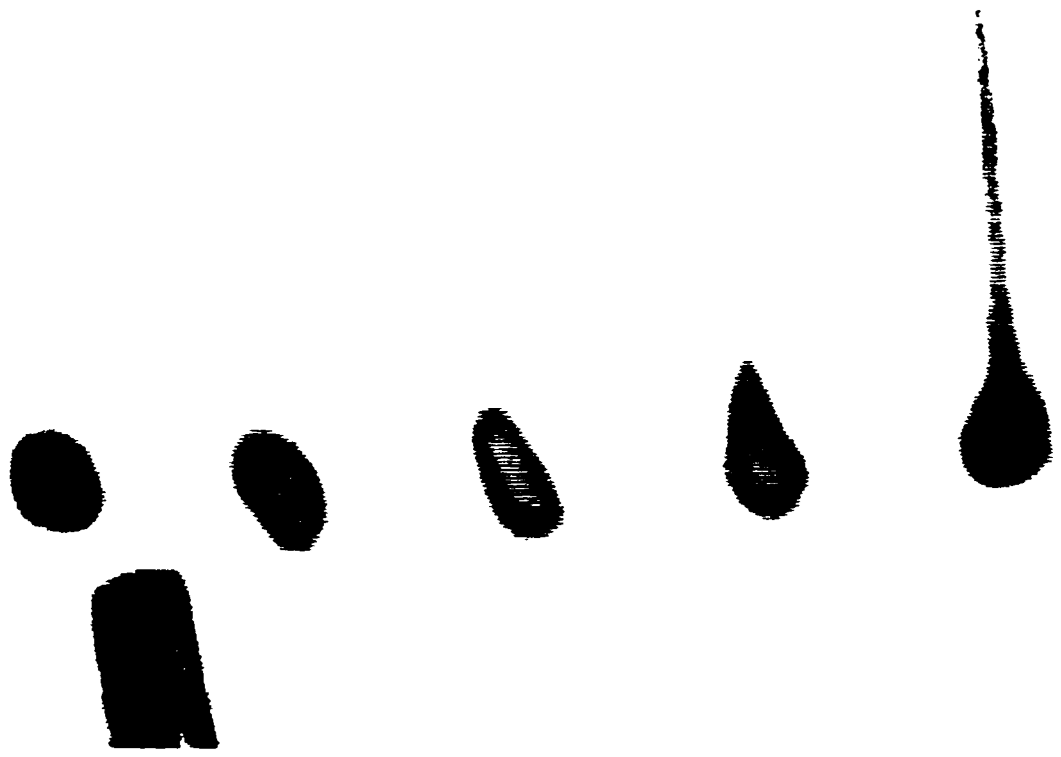

Figure 2.7 Droplet electrospray from higher velocity droplets show the spray beginning farther away from the electrode. The spray from these droplets did not change direction like that from the slower droplets. The direction of droplet motion is from the left of the figure to the right.

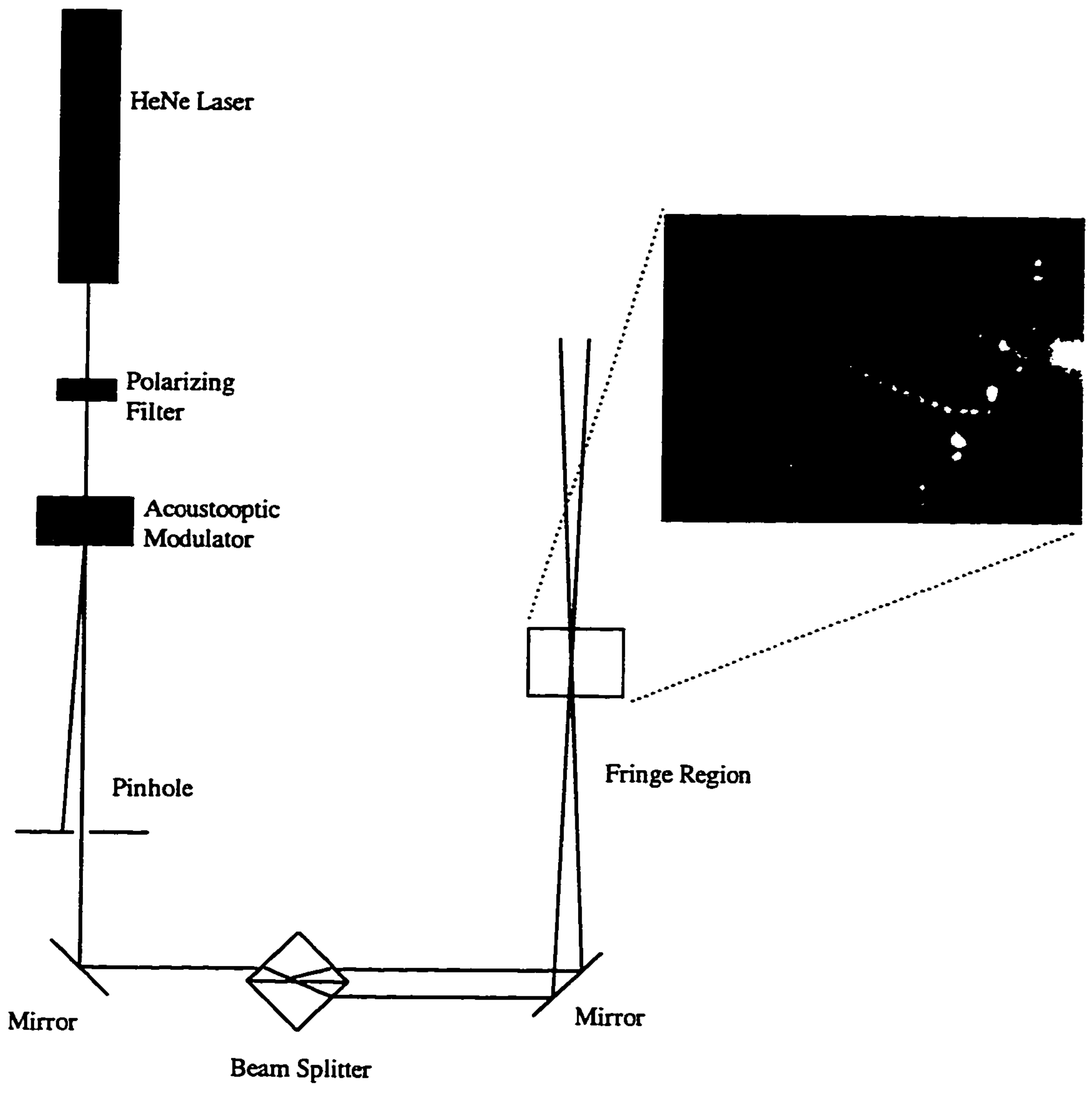

Figure 2.8 The optical arrangement for stroboscopic laser fringe illumination of droplet electrospray. The interference fringes were produced using two crossed laser beams. The beams were chopped by the acoustooptic modulator in synchronization with the droplet production frequency. Much better images of the DES aerosol were obtained with this arrangement. See Figure 2.5.

2.5 REFERENCES

1. Smith, D.P.H., *IEEE Trans. Ind. Appl.*, **IA-22**, 527 - 535. (1986)
2. Hayati, I.; Bailey, A.I.; Tadros, T.F., *J. Coll. Int. Sci.*, **117**, 205 - 221. (1987)
3. Cloupeau, M.; Prunet-Foch, B., *J. Electrostat*, **25**, 165 - 184. (1990)
4. Iribarne, J.V.; Thomson, B. A., *J. Chem. Phys.*, **64**, 2287 - 2294. (1976)
5. Thomson, B.A; Iribarne, J.V., *J. Chem. Phys.*, **71**, 4451 - 4463. (1979)
6. Yamashita, M.; Fenn, J.B., *J. Phys. Chem.*, **88**, 4451 - 4459. (1984)
7. Yamashita, M.; Fenn, J.B., *J. Phys. Chem.*, **88**, 4671 - 4675. (1984)
8. Blades, A.T.; Ikonomou, M.G.; Kebarle, P., *Anal. Chem.*, **63** , 2109 - 2114. (1991)
9. Galley, P.J.; Hieftje, G.M., *Appl. Spectrosc.*, **46**, 1460 - 1463. (1992)
10. Childers, A.G.; Hieftje, G.M., *Appl. Spectrosc.*, **40**, 688 - 691. (1986)
11. Bailey, A.G. *Electrostatic Spraying of Liquids*, John Wiley & Sons, (1988).
12. Good, A.; Durden, D. A.; Kebarle, P., *J. Chem. Phys.*, **52** , 222. (1970)
13. Sunner, J.; Nicol, G.; Kebarle, P., *Anal. Chem.*, **60** , 1300 - 1307. (1988)
14. Sunner, J.; Ikonomou, M. G.; Kebarle, P., *Anal. Chem.*, **60** , 1308 - 1313. (1988)

CHAPTER 3:

DROPLET ELECTROSPRAY MASS SPECTROMETRY

3.1 INTRODUCTION

The droplet electrospray (DES) method developed recently[1] has the potential of becoming an electrospray mass spectrometry (ESMS) technique for analytical purposes. The method also provides results that are of interest concerning the nature of the conventional electrospray (ES) mechanism. The present work, which reports the first experimental results in which ions produced by DES are detected with a mass spectrometer, provides data and discussion that relate to both of the above topics.

In conventional electrospray, sample is pumped through a metal capillary held at high potential; upon exiting the capillary, the sample forms a fine spray, which yields analyte ions. DES provides a unique method to test models for electrospray production. In DES, the solution is passed through a capillary and the capillary tip is at the same potential as the surroundings. Uncharged droplets are produced by vibrating the capillary tip at a high frequency ν, typically $\nu = 100$ kHz. Droplets of highly uniform radius (monodisperse) are obtained. A desired radius is selected by control of the flow rate, the capillary internal diameter and the frequency. So far, controlled droplet radii in the range 6-50 μm have been produced[1]. The number of droplets per second corresponds to the frequency ν. The relationship between droplet radius r, flow rate f and frequency is given by,

$$\nu\left(\frac{4}{3}\right)\pi r^3 = f \qquad (1)$$

The droplets are charged downstream of the capillary tip by placing a 50 μm diameter platinum rod electrode near the droplet stream. The electrode is at a positive potential (~3 kV) relative to ground and a small corona discharge current (~0.3 μA) is present due to this potential, in the absence of the droplet stream. When the electrode is brought within charging distance of the droplet stream, typically ~20 μm between electrode tip and droplet stream, the droplets become positively charged by the positive ion current of the discharge. The charge acquired by the droplets leads to destabilization of the droplets and they are observed to emit a fine spray that we will call droplet spray. When the sampling aperture of an atmospheric pressure mass spectrometer is near the droplet spray, ions due to positive electrolyte ions present in the solution are detected.

3.2 EXPERIMENTAL

The droplet electrospray apparatus was described briefly previously[1]. Here, we provide a more complete account.

A schematic of the setup is shown in Figure 3.1. The solution was pumped by a high pressure syringe pump (ISCO model 314) through a fused silica capillary of 10 μm to 30 μm inner diameter. The end of the capillary was held by a stainless steel union some 2 cm from the capillary tip. About 1.5 cm from the union, the capillary was vibrated by a short length of copper wire, 14 mm long 0.2 mm diameter, which was soldered to a piezoelectric buzzer[2].

In the absence of vibration, an unbroken liquid jet emerged from the capillary tip and persisted over a considerable distance. In the presence of vibration, the liquid jet was observed to break into droplets at a distance that depended on the flow rate, surface tension, and viscosity of the solvent and the frequency of the buzzer. Typically, the distance was equal to ~15 jet diameters from the capillary tip.

The jet and droplets were observed through a microscope. Stroboscopic illumination was provided by a red light emitting diode (LED), whose frequency was identical to the frequency of the buzzer, typically 100 kHz and with an exposure of 400 nsec per flash. The light from the LED shone directly into the objective lens of the microscope, providing dark images of the droplets against a bright background. Photographs could be taken with a 35 mm camera and video images could be obtained with a charge coupled device (CCD) camera, when either of these was attached to the microscope.

Charging of the droplets was obtained with a corona discharge by placing a 50 μm diameter platinum rod electrode within a distance of ~20 μm of the droplet stream. The electrode was mounted on an XY-translation stage that allowed accurate positioning of the electrode relative to the droplet stream. The application of a positive potential of ~3 kV led to a corona discharge of ~0.3 μA. The discharge current leaving the Pt electrode was measured by means of a microamp meter, see Figure 3.2. The charged droplets emitted a spray of very much smaller charged droplets that could be observed with the microscope and the video, see Figure 3.3.

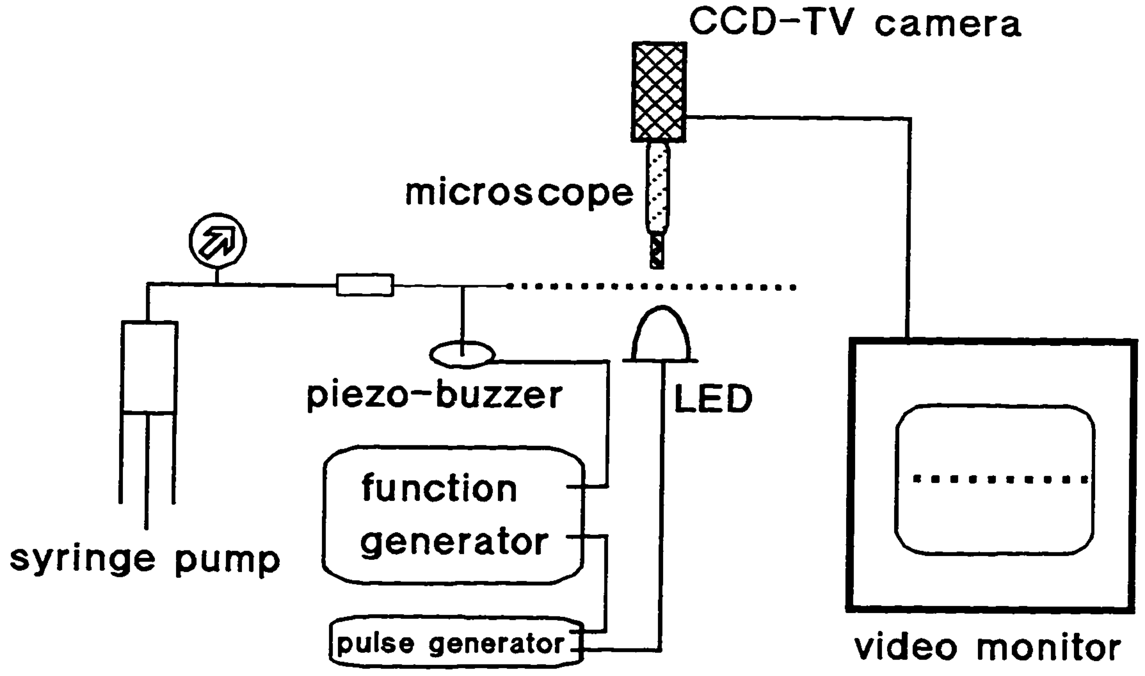

Figure 3.1 Apparatus used for the generation and observation of the droplet stream. Solution is pumped by a high pressure syringe pump through the fused silica capillary. The free end of the capillary is vibrated at ~100 kHz by a piezo electric buzzer. The droplets are illuminated by a light emitting diode (LED) that emits 0.4 μs long pulses at a frequency equal to that of the buzzer. Droplets are observed with a low power microscope and displayed on video monitor.

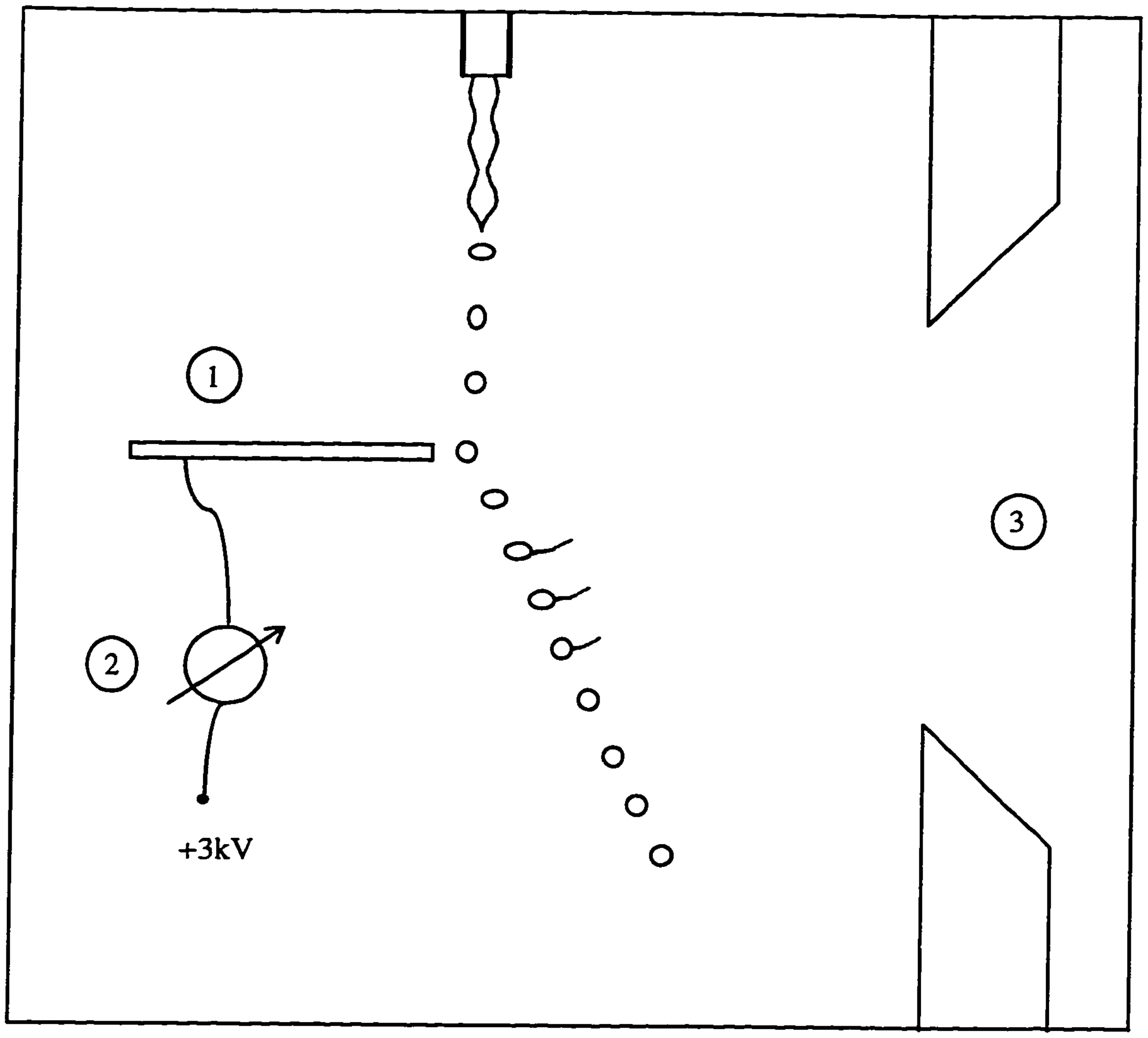

Figure 3.2 Diagram of droplet spray process showing: liquid jet, separating droplets, droplet charging by corona discharge from platinum (Pt) electrode and spray from charged droplets. The position of 4 mm hole leading to mass spectrometer ion sampling orifice is shown on the left side. Drawing of droplet stream, Pt wire and ion sampling orifice is approximately to scale, except for the distance between the ion sampling orifice and the droplet stream that was reduced by a factor of 20. Pt electrode, 1. Discharge current meter 2. Mass spectrometer orifice 3.

When the capillary and Pt electrode arrangement was mounted in front of an atmospheric pressure mass spectrometer, (SCIEX TAGA 6000E) and a droplet spray was induced to occur in front of the ion aperture, ions could be detected with the mass spectrometer. A schematic of this arrangement is shown in Figure 3.2. For a description of the TAGA mass spectrometer sampling of electrospray produced ions, see Ikonomou *et al.*[3]. To observe ions, the 4 mm diameter ion sampling orifice in the TAGA interface plate was positioned within ~6 cm of the droplet stream. The direction of the droplet stream was parallel to the face of the interface plate. The discharge rod was perpendicular to the face of the interface plate and approximately coaxial with the ion sampling orifice, see Figure 3.2.

Solutions were prepared in methanol. Reagents were analytical reagent grade.

3.3 RESULTS AND DISCUSSION

3.3.1 Droplet Electrospray, Mass Spectral Data and Origin of the Detected Ions.

The low magnification photograph, Figure 3.3, gives an overview of the droplet electrospray (DES) process. The forced vibration of the capillary that carries the solution induces bead formation in the emerging liquid jet that is followed by break-up of the beads into separate droplets. The break-up induces an oscillation of the droplets into prolate and oblate shapes. The droplets acquire charge when they are in the vicinity of the charging electrode. Due to this charge, the droplet stream is deflected away from the electrode. The charged droplets emit a spray when they are just past the electrode.

The bright area at the tip of the charging electrode was due to the bright spot at the center of the light emitting diode used for back lighting. Because of the narrow depth of field, some sections of the photograph appear out of focus. At the top of the picture, the liquid jet emerges from the capillary tip too close to the camera and the droplet stream in the lower right corner of Figure 3.3 is too far from the camera to be in focus. When in focus, droplets that were not charged enough to spray gave slightly sharper images than those that had sprayed, indicating some variability in the charging or spraying process. More detailed pictures obtained with higher magnification are shown in Figures 3.4 and 3.5. Figure 3.4 gives details into the droplet oscillation induced by the break up of the neutral liquid. Just before complete separation, the droplets are connected by a thin filament. On separation this filament can give rise to small satellite droplets as is the case in Figure 3.4. The formation of such small satellite droplets can be prevented by working at different frequencies.

An enlarged photograph of the droplet spray produced after the droplet has been charged is shown in Figure 3.5. Approximate distance measurements are possible with these photographs because the electrode diameter of 50 μm provides a distance reference. The distance between the electrode tip and the droplet stream is approximately 20 μm. The frequency used was 91 kHz so that the flight time between successive droplets in Figure 3.5 is ~11 μs. The charging of the droplets is followed by an elongation of the droplet away from the electrode. This elongation grows to a sharp point that emits a fine spray of charged offspring droplets.

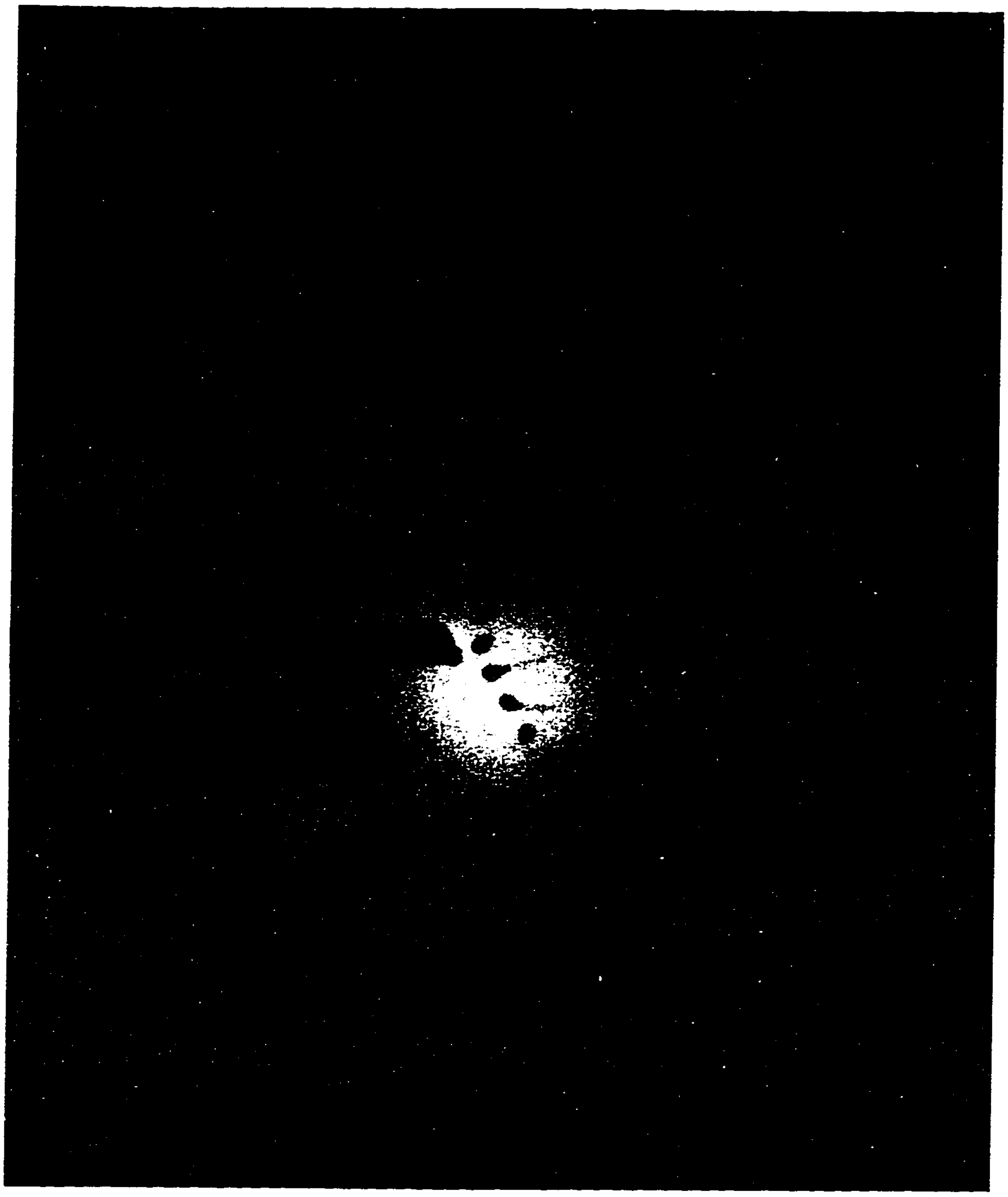

Figure 3.3 Photograph of vibrated capillary tip showing separation of liquid jet into droplets, droplet charging by Pt electrode, and production of a fine spray of much smaller offspring droplets by parent droplets, just past the Pt electrode.

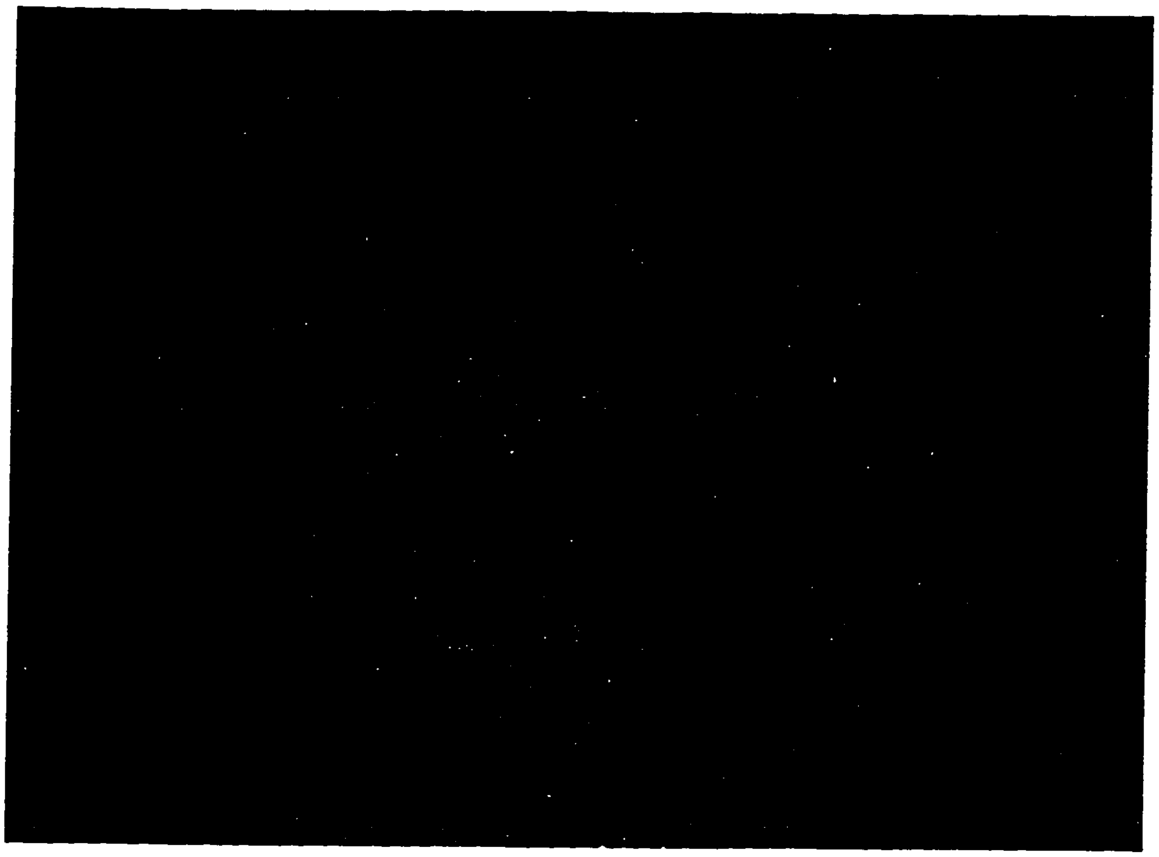

Figure 3.4 Detail of break up of uncharged liquid stream into droplets. The break up initiates droplet oscillation into oblate and prolate shapes. The liquid filament that connects the droplets just before break up can lead to formation of a much smaller satellite droplet. Formation of satellite droplets can be eliminated by tuning the flow rate and frequency.

The appearance of the spraying droplets in Figures 3.3 and 3.5 is very similar to the charged droplet fission observed by Gomez and Tang[4]. These authors were photographing droplets from conventional electrospray using a shadow-graph technique with 25 nsec flashes combined with a CCD camera. Their shadow-graphs were obtained with single flashes and showed droplets caught in the act of fission, see Figures 3.5 and 3.6 in reference 4. In the present work, the shadow-graphs were obtained by a stroboscopic technique and a given frame represents the superposition of 3000 consecutive images of 3000 droplets arriving consecutively to the same places when the flashes occur. The relatively sharp pictures of the droplets illustrates the remarkable reproducibility of the DES process from droplet to droplet. Droplets that are in the process of spray (fission) are expected to lead to "smeared out" spray plumes when the present technique is used, because the positions of the fine, daughter droplets in the plume are not synchronized with the illumination. Aside from this drawback, the present technique is seen to provide a very clear illustration of the evolution of the process.

At positive charging electrode potentials, electrons stripped from nearby air molecules and collected by the electrode tip generated a tiny current, which may be thought of as a positive ion current emitted by the electrode. This current was measured by the picoammeter shown diagrammatically in Figure 3.2 as the device labeled 2. The positive current leaving the Pt electrode in the absence of a nearby droplet stream was typically 0.3 μA, when the electrode was at 3 kV. This current always decreased to ~0.18 μA when the droplet stream was brought to within a short

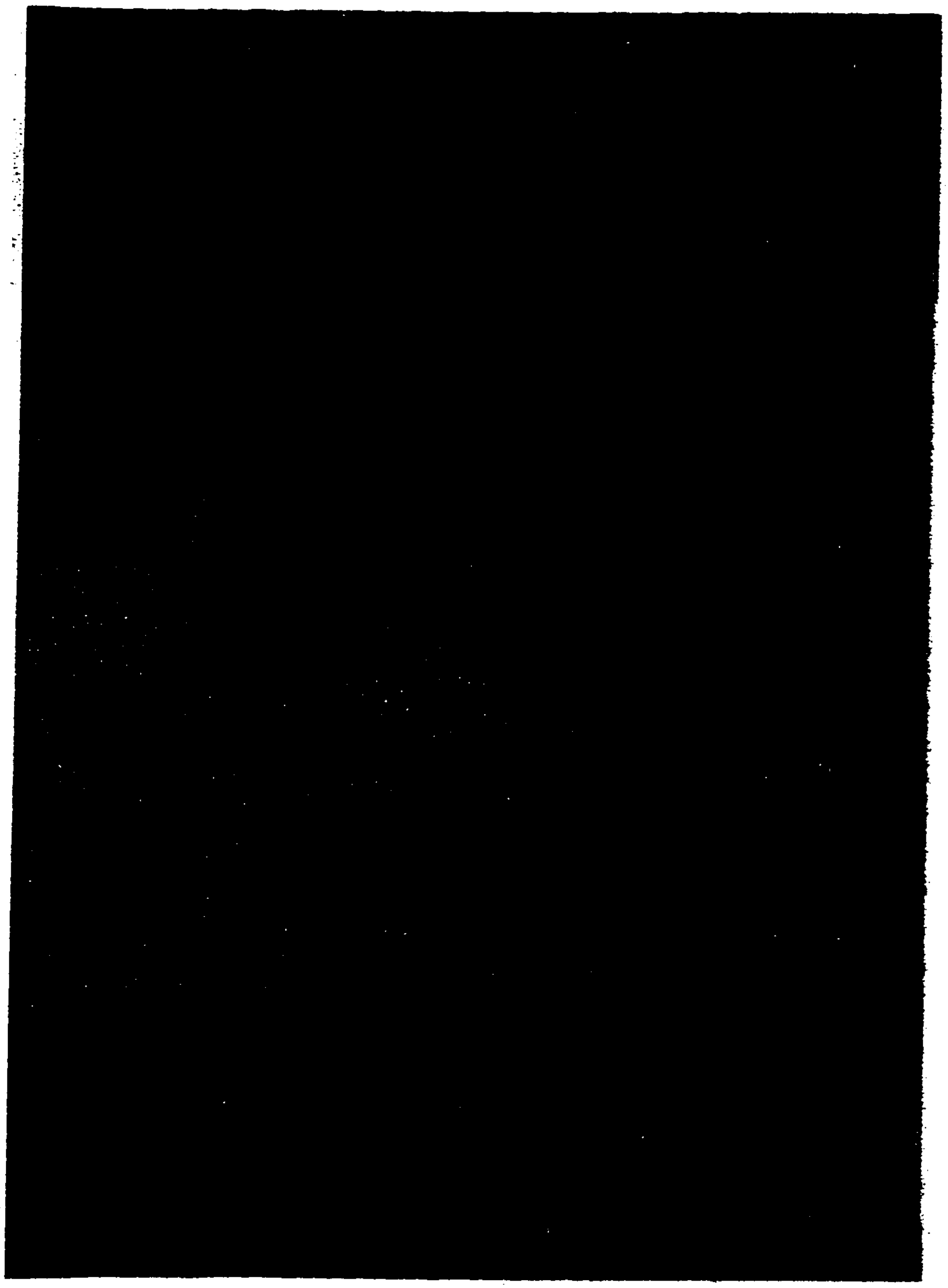

Figure 3.5 Enlarged view of 50 μm diameter platinum rod electrode, droplets, near electrode and droplet spray past electrode. Droplet diameter 44 μm, 93 kHz, 3 kV, discharge current 0.16 μA. Solution: 10^{-4} mol/L tetraethylammonium bromide in methanol.

distance of the elecrode such that droplet charging and droplet spray occurred. The exact reason for the current decrease is not known but it is probable that the polarization of droplets when in the vicinity of the electrode reduces the electric field at the electrode tip and this is followed by a decrease of the discharge current. The rapid sweeping away of the air molecules by the stream of fast moving droplets may also contribute to the decrease in current.[11]

Mass spectra obtained with the arrangement shown in Figure 3.2 are given in Figures 3.6-3.8. The mass spectrum in Figure 3.6 was obtained when the droplet stream, composed of a solution of 10^{-4} M NaCl in methanol, was kept at a larger distance from the Pt electrode so that a droplet spray was not observed with the microscope. For this case, there is only a corona discharge present near the mass spectrometer aperture. The ions observed are due to: $CH_3OH_2^+$, NH_4^+, $NH_4^+ \bullet H_2O$ and $NH_4^+ \bullet CH_3OH$. As discussed in Chapter 2, these species are expected products from discharge atmospheric pressure ionization (API)[5]. The original API produced ions, N_2^+ and O_2^+, are converted to H_3O^+ and $H_3O^+(H_2O)_n$ in the presence of water vapor. When methanol and ammonia are present in the air, proton transfer from the proton hydrates leads to the ions observed in Figure 3.6. For a detailed description of the processes involved, see reference 5. The presence of methanol in the present case is due to evaporation of the solvent-methanol. The presence of ammonia in the laboratory air often leads to the formation of NH_4^+ in conventional API experiments conducted in our laboratory.

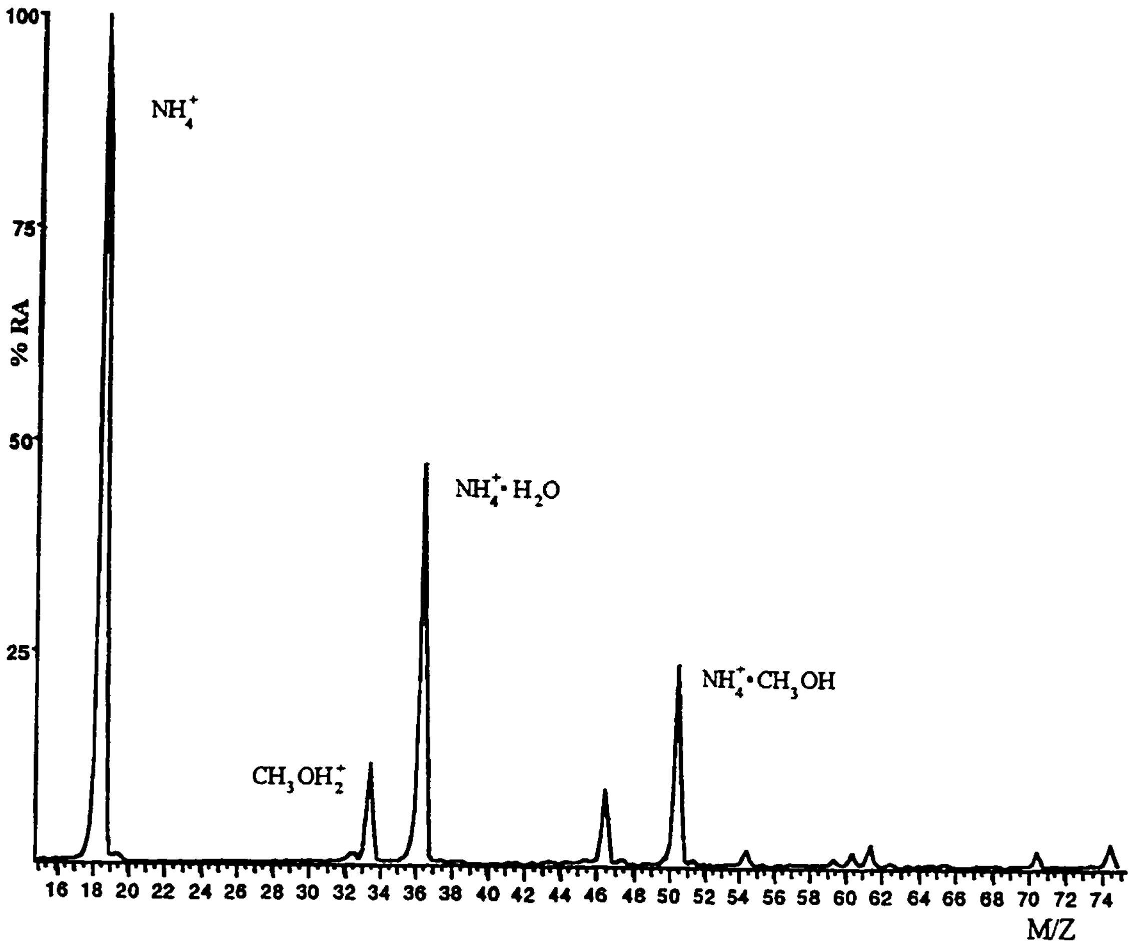

Figure 3.6 Mass spectrum observed with a deliberately misaligned droplet stream such that stream passed ~200 μm below the Pt electrode. Droplets were observed not to spray. Only ions due to the corona discharge are observed. Potential, 3 kV. Discharge current, 0.3 μA. Intensity equal to 100 corresponds to 2.5×10^5 counts/s.

The mass spectrum shown in Figure 3.7 was obtained with the same solution (10^{-4} M NaCl in methanol) as used for Figure 3.6, but now the droplet stream was brought close to the stationary Pt electrode such that droplet spray was observed with the microscope. The spectrum is mostly due to Na^+ and Na^+H_2O ions that are due to the electrolyte ions present in the droplets. Some ions due to the electric discharge, such as NH_4^+, $NH_4^+ \bullet H_2O$, $H_3O^+(H_2O)_2$ at low intensities are also observed. The spectrum obviously is largely due to electrospray, DES in this case.

The spectrum shown in Figure 3.8 was obtained with 10^{-4} mol/L solution of 1, 12-dodecyldiamine in methanol and droplet spray. The major ion is the doubly protonated base, an ion that could not be produced by gas phase electric discharge ionization. Most of the singly protonated base intensity is probably also due to ions from the solution.

Several other analytes in methanol solution such as potassium crown ether complexes (Figure 3.9), tetraalkylamines (Figure 3.10) and doubly charged transition metal ions such as Co^{++} complexed to DMSO (Figure 3.11) were also examined and the DES spectra obtained were found to be close to identical to spectra obtained previously in this laboratory with conventional electrospray.

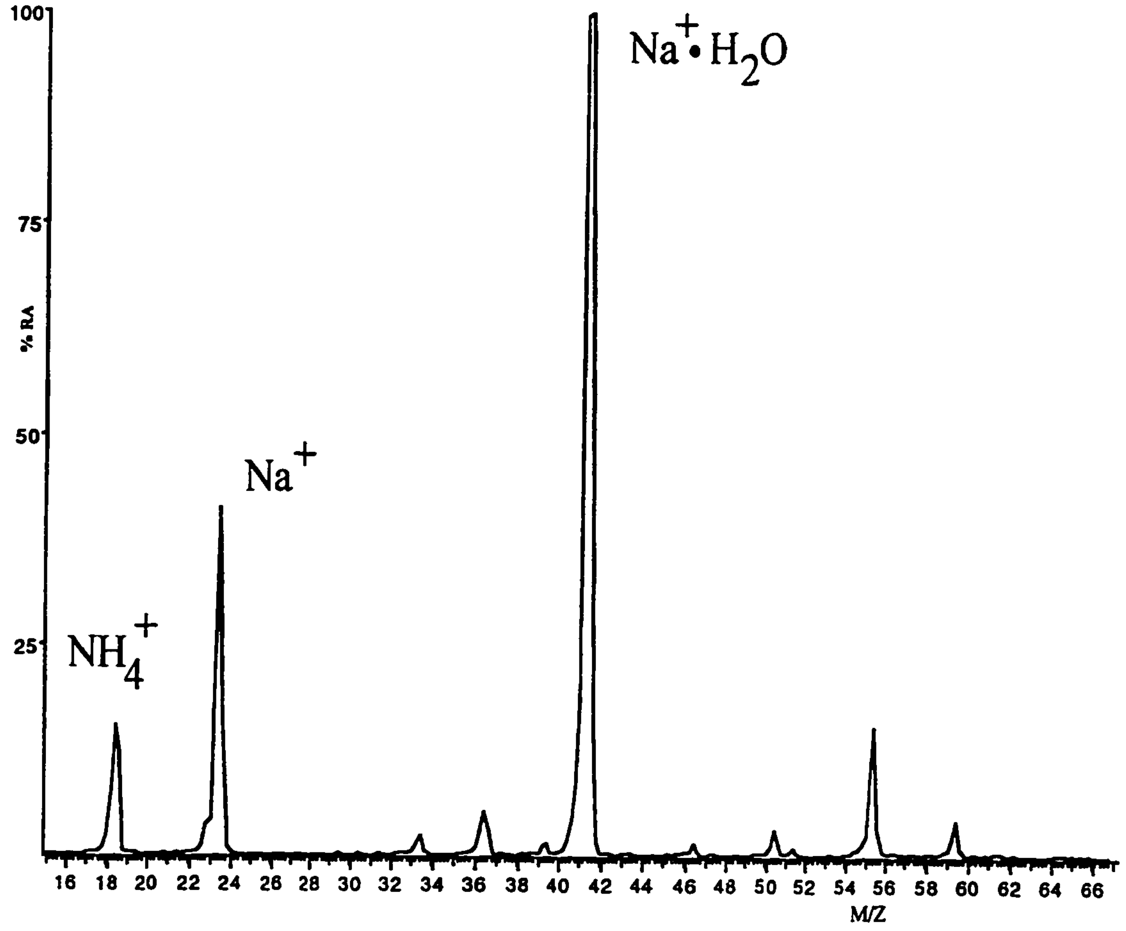

Figure 3.7 Mass spectrum observed when droplet stream was aligned with Pt electrode, as in Figure 3.5, and a droplet spray was observed. Ions, $Na^+(H_2O)_n$, observed in spectrum are due to electrolyte ions present in solution; 10^{-4} mol/L NaCl in methanol. Potential 3 kV. Discharge current; 0.18 μA. Intensity of 100 corresponds to 1.4×10^5 counts/s.

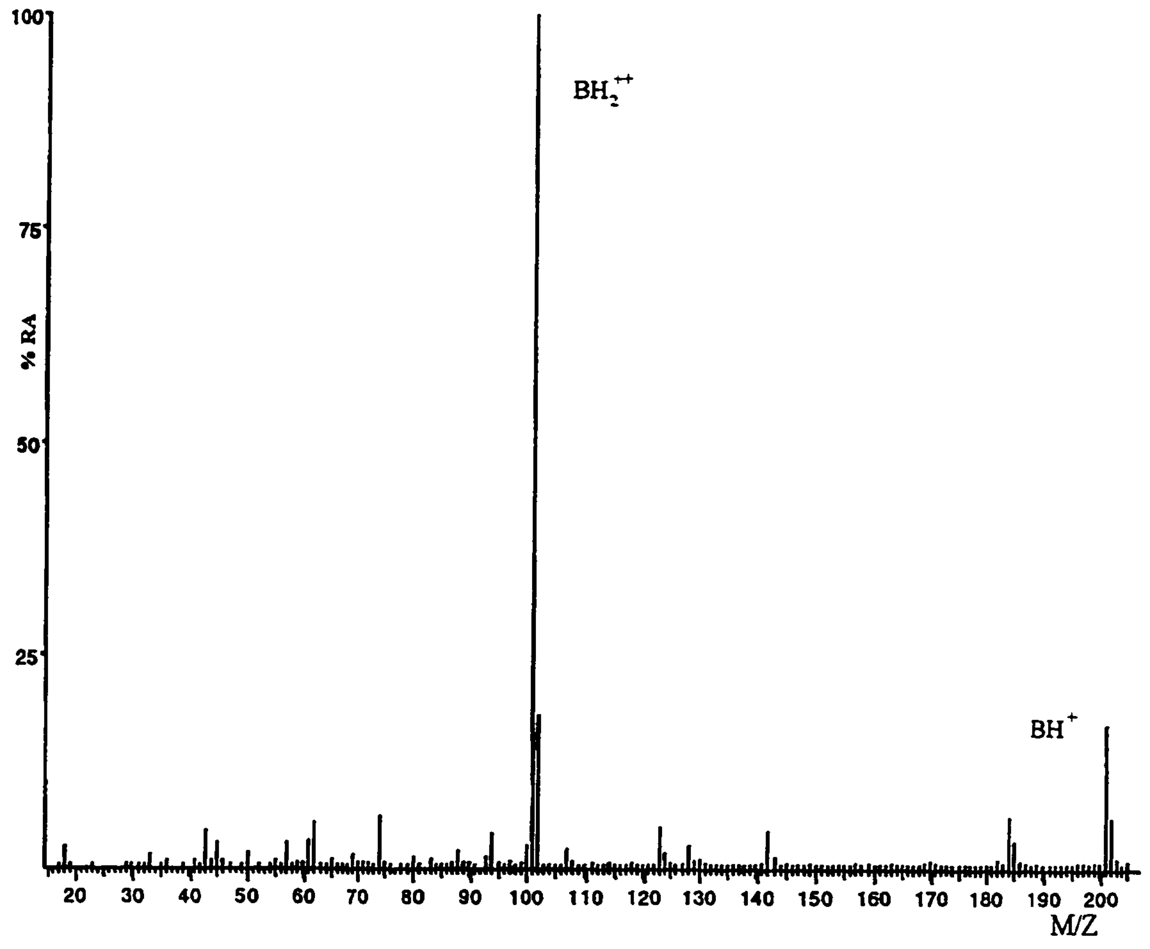

Figure 3.8 Mass spectrum with aligned droplet stream and with observed droplet spray. Solution used: 10^{-4} mol/L dodecyldiamine (abbreviated as B in the figure) in methanol. Ions observed: m/z = 101 of doubly protonated BH_2^{++} base and m/z 201 of singly protonated base BH^+ are the same as would be found in conventional electrospray. Droplet diameter 44 μm. Potential, 2.5 kV. Discharge current, 0.15 μA. Intensity of 100 corresponds to 6 x 10^5 counts/s.

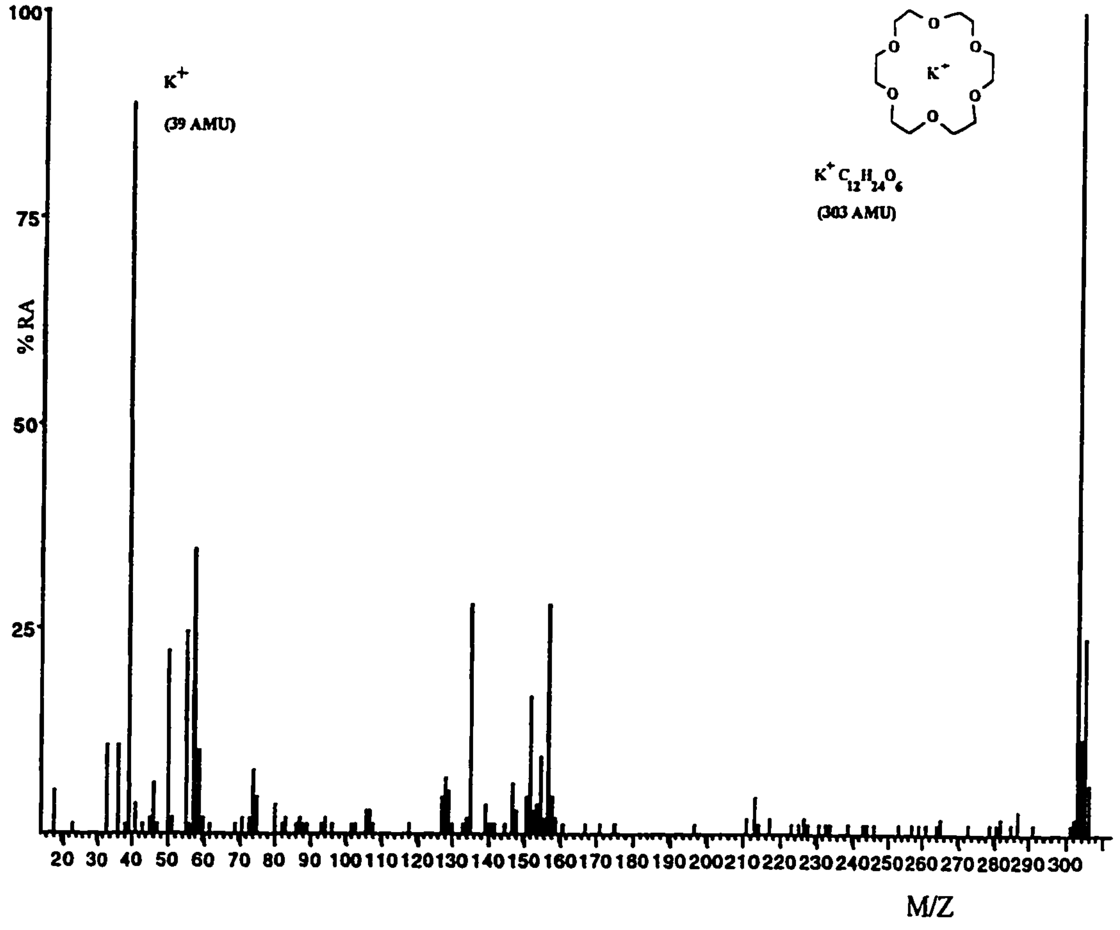

Figure 3.9 Mass spectrum obtained under the same conditions used for Figure 3.8 but with a solution of 10^{-4}M potassium-18-crown-6 in methanol. The major ions are K^+ (18-crown-6) at M/Z 303 and bare K^+ at M/Z 39.

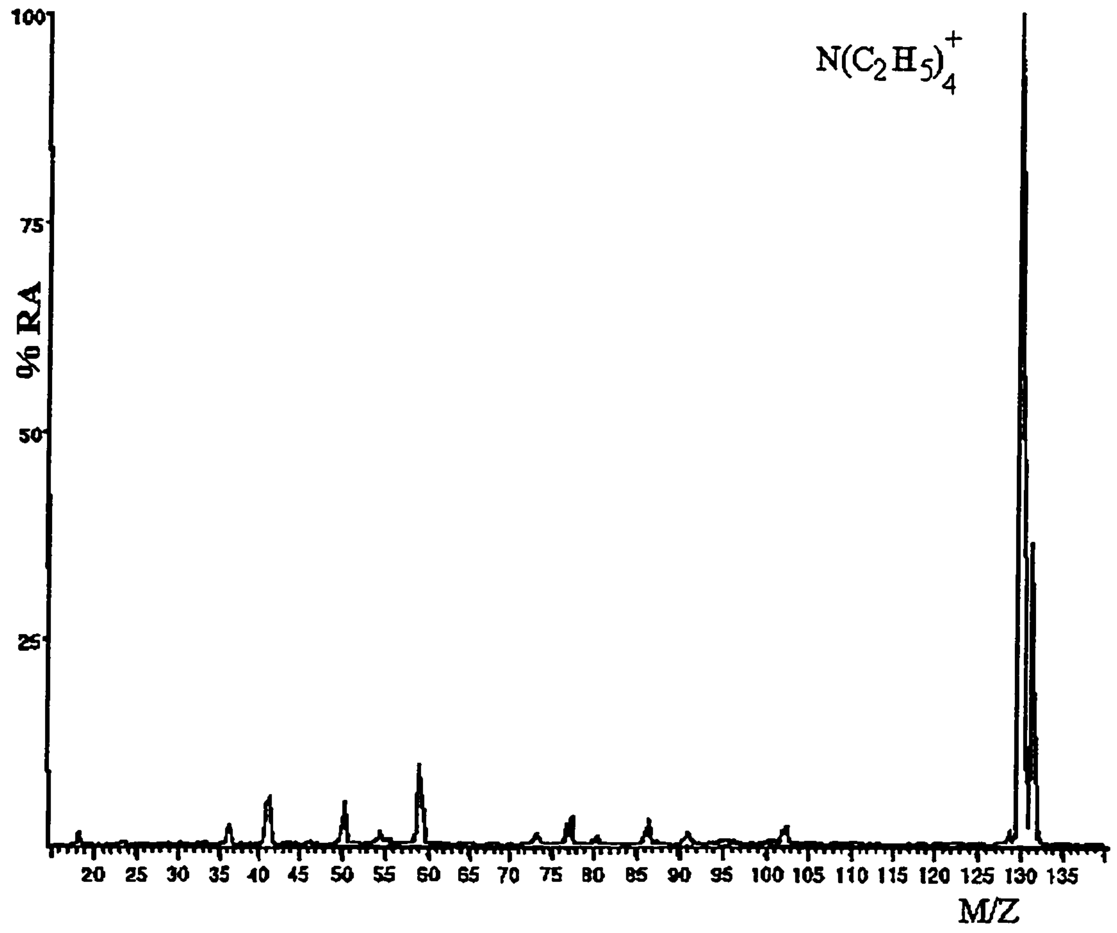

Figure 3.10 Mass spectrum obtained under the same conditions used for Figure 3.8 but with a solution of 10^{-4}M tetraethylammonium bromide in methanol. The major peak in the spectrum appears at the M/Z expected for tetraethylammonium ion, i.e. at 130.

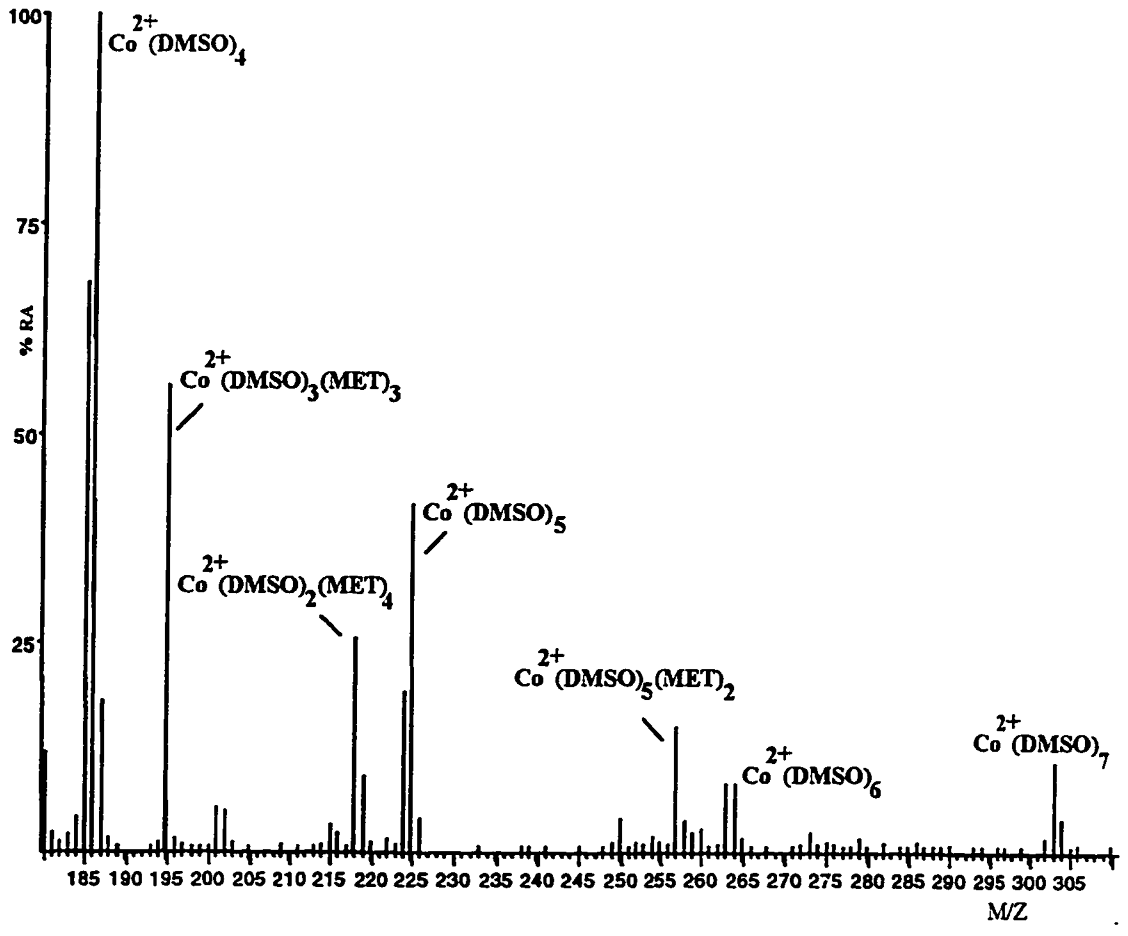

Figure 3.11 Mass spectrum obtained under the same conditions used for Figure 3.8 but with a solution of 10^{-4}M Co^{++} and 0.1% DMSO in methanol. Several different combinations of Co^{++}, DMSO, and methanol were identified and are indicated in the spectrum.

The ion intensity observed with DES relative to conventional ES is lower by a factor of 2 to 3, see for example DES ion intensity in Figure 3.8 that is in the 6×10^5 cps range. Signals above 10×10^5 cps are generally obtained with conventional ES with the same solution and instrument. The solution flow rates used in conventional ES are much less (around a tenth that used in DES) because everything pumped through the system ends up in the electrospray aerosol.

It should be noted that the water and methanol adducts such as $Na^+(H_2O)_n$, observed in the spectra, Figures 3.6 and 3.7, could be converted to the naked ions, *i.e.*, Na^+ by selecting larger voltage gradients on the first electrodes of the mass spectrometer. The spectra shown were obtained with essentially no gradients in this region, *i.e.*, no CID (collision induced dissociation), in order to obtain approximate information on the nature of the ions present in the atmospheric region.

The signal stability observed with DES is shown in Figure 3.12. Intensity traces obtained with ES under the same conditions consist of an essentially straight horizontal line. The DES stability is obviously much poorer. The fluctuations seen are approximately ten times larger than those for conventional ES.

The mass spectrometric DES results described above were obtained without comprehensive attempts to optimize the performance. At this preliminary stage, the intent was to obtain only a qualitative overview. In the present experiments it was assumed essential to be able to observe the occurrence of droplet spray with the microscope. This necessitated keeping the charging electrode tip and the plane of the droplet stream at a relatively large distance (~6 cm) from the ion sampling orifice of the mass spectrometer. The observations with the microscope showed that the stability of the ion intensity was directly related to the stability of the image of the spraying droplet(s). Best results were obtained when the position of the droplet stream and the electrode voltage was adjusted to give the most solid and stationary image of the spraying droplets. Image instabilities may have been caused by several instrumental deficiencies. Airborne ions produced by the corona discharge along with charged solvent droplets would cause a build up of static charges on ungrounded and non-conducting surfaces near the droplet charging region. Variations in the electric field would result from variations in the extent of static charge accumulation and would manifest themselves in slight alterations of droplet trajectory. Several accumulations of solvent were observed clinging to various parts of the apparatus. These accumulations would often form into large drops and, in the case where the accumulation was near the supporting structures of the charging electrode, would periodically fly off creating a momentary disturbance. Another major source of image instability was the flow of laboratory air near the instrumentation, due to incomplete shielding of the equipment. Other, less severe sources of instability were mechanical vibrations, temperature variations, voltage fluctuations, and changes in volumetric flow rate. These sources of image instability along with the correlation of image stability with ion intensity stability suggest that the ion intensity fluctuations observed in Figure 3.12 is not inherent and may be removed by more careful instrumental design.

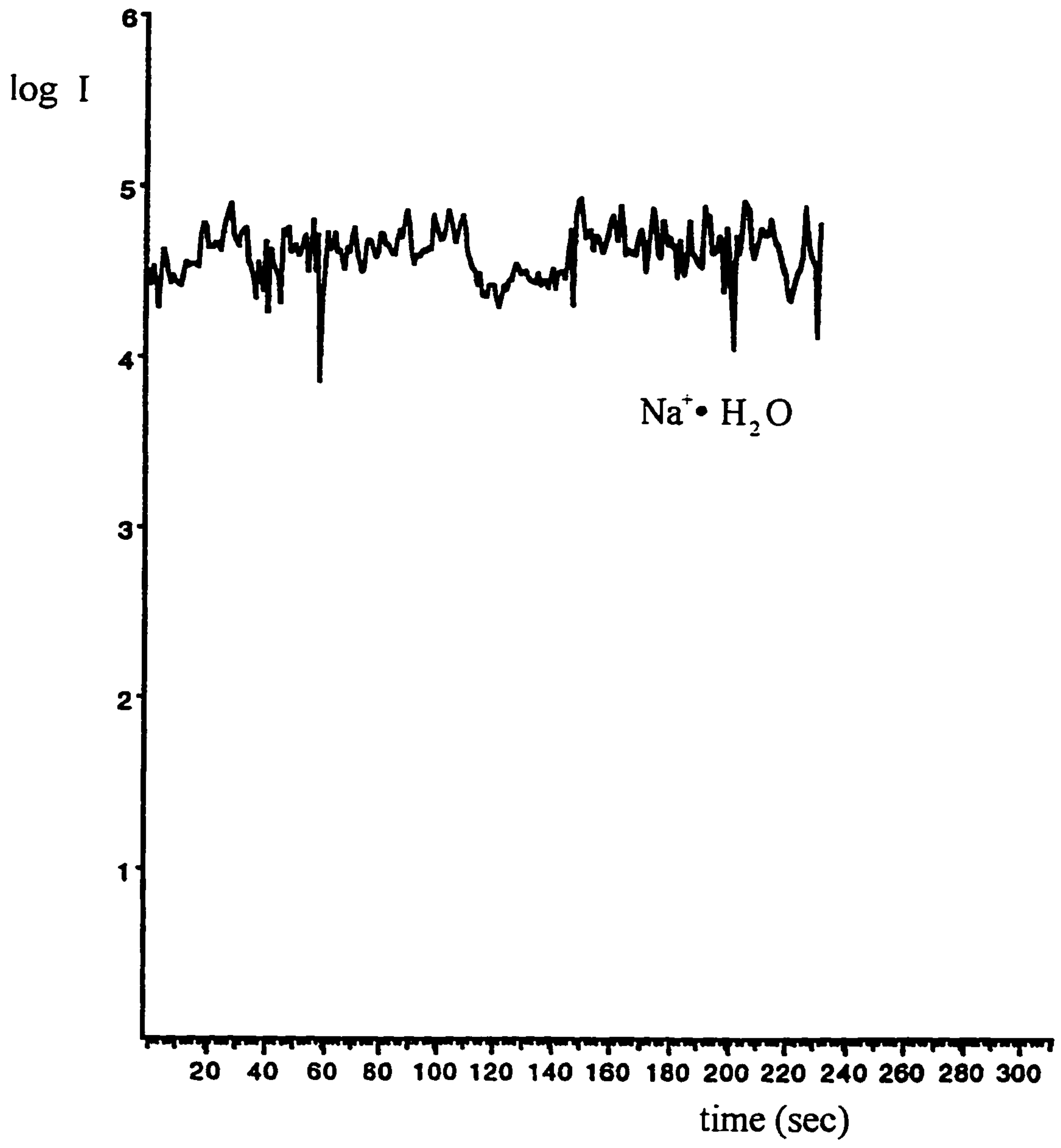

Figure 3.12 Single ion trace of $Na^+(H_2O)_2$ obtained with 10^{-4} mol/L NaCl solution, illustrates relatively noisy signal obtained with present, initial experiments. Solution used 10^{-4} mol/L NaCl in methanol. Intensity, I, in counts/s.

Shorter distances between the ion sampling orifice and the droplet stream / Pt electrode arrangement were expected to lead to higher DES ion intensities. The larger the distance the airborne ions must traverse before entering the mass spectrometer, the more these ions disperse due to coulombic repulsions of the ions, i.e. a space charge effect. Chapter 4 introduces a modification in the apparatus designed to capture the newly formed DES ions and transport them to the mass spectrometer inlet via a capillary. This method, while minimizing the dispersal of the ions and affording improved spatial resolution, provides another loss mechanism as the ions may contact the capillary walls and be discharged.

The effect of droplet size, frequency, and solution flow rate was not studied in detail as these effects did not appear to be large. However, optimization of these parameters could also lead to higher ion intensities.

The present results provide information on some aspects of the mechanism of conventional electrospray. These are discussed next.

3.3.2 Independence of ES mass spectra on mode of droplet charging.

It was shown in the preceding section that the droplet electrospray spectra, where droplet charging was obtained via gas phase ions from an electric discharge, are close to identical to mass spectra observed in conventional electrospray.

The droplet charging in conventional electrospray is due to an electrophoretic process occurring in the solution[6]. When the capillary is the positive pole, the droplets are enriched on positive electrolyte ions and this excess is either due to an electrolytic removal of negative ions at the capillary electrode or an electrolytic gain of positive metal ions from the capillary wall, such as Fe^{++} when the capillary electrode is of stainless steel. As first stated by Kebarle, an electrolytic oxidation reaction occurs at the capillary electrode in each of these two cases[7].

The gas phase positive ions charging the droplets in DES are of a very different nature. They are generally H_3O^+, or $CH_3OH_2^+$ or NH_4^+ ions, see Figure 3.6 and discussion. The similarity between the ES and DES spectra in spite of the difference between the charging ions is easily explained. The number of positive ions required for charging of the droplets is considerably smaller than the number of electrolyte positive ions, charge paired to negative electrolyte ions, in the solution. The relative number of the gas phase ions ultimately produced from the charged droplets depends on the relative number of all the positive ions present in the droplets. Because the electrolyte ions dominate in the droplets, it is these ions that are predominantly expressed in the mass spectrum[6].

The situation is readily examined for DES. The charging of the droplets required to lead to DES is expected to be somewhat below the Rayleigh limit because the droplet fission is partially induced by the electrical field of the electrode and the oscillation of the droplet. For simplicity, we will assume that the Rayleigh limit charge must be supplied by the acquisition of charging ions by the droplet from the charging electrode. Using the Rayleigh equation

$$Q_R^2 = 64\pi^2 \varepsilon_0 \gamma r^3 \qquad (2)$$

where Q_R is the charge, γ the surface tension of the solvent, ε_0 the permittivity of vacuum and r the droplet radius, one can evaluate the ratio of the charge Q_{elc} due to charge paired positive electrolyte ions in the original solution relative to the Rayleigh charge, Q_R acquired from the corona.

$$\frac{Q_{elc}}{Q_R} = \frac{\frac{4}{3}\pi r^3 F(\times 10^3 M)}{(64\pi^2 \varepsilon_0 \gamma r^3)^{1/2}} \qquad (3)$$

$$= 3.6 \times 10^{13} r^{3/2} M \qquad (3a)$$

where F is Faradays constant, M is the concentration of the electrolyte in the solution. This charge ratio corresponds to the ion number ratio for singly charged ions. The numerical constant given in eq. 3a is for methanol solvent, r in meter and M in mol/L. For the mass spectrum shown in Figure 3.7, the droplet radius was 23 μm and the NaCl concentration was 10^{-5} mol/L. These data and eq. 3a lead to an electrolyte-ion to charging-ion ratio of 40, *i.e.*, for every 40 electrolyte ions present in the original uncharged solution, one additional ion needs to be added from an external source for the droplet to reach the Rayleigh limit and explode and therefore less than ~2.5% of the DES spectrum should be due to the gas phase ions required for charging of the droplets. The relative abundance of gas phase type ions in Figure 3.7 is higher, but this is to be expected since with DES some direct sampling of discharge ions from the air should be occurring.

We conclude that the DES spectra are expected to be very similar to the conventional ES spectra as is experimentally observed.

It should be noted that corona discharge has been used previously for droplet charging of aerosols[8], however the intent in that work was the production of aerosols.

The present experiments represent the first use of corona discharge droplet charging for electrospray mass spectrometry.

3.3.3 Significance of droplet electrospray results for Taylor cone ion emission mechanism proposal by Siu et al.

Siu *et al.* have proposed recently[9, 10] that the gas phase ions in ESMS are not produced from very small (less than 0.1 μm diameter) and highly charged droplets as is generally assumed[6, 11-13] but are emitted directly from the Taylor cone tip at the ES capillary.

A brief discussion of the experiments and a critique of the assumptions of Siu *et al.*[9, 10] was included in a recent review article by Kebarle and Tang[6] and therefore we will consider here only the significance of the present results. The electrospray type mass spectra, Figures 3.7 and 3.8, were obtained from charged droplet fission, see Figure 3.4, where a Taylor cone is absent. Therefore, these results refute Siu *et al.'s* assumption.

Considering the above, Siu *et al.'s* assumption would have to be modified to a claim that gas phase ions are emitted directly whenever charge induced dispersion to much smaller droplets occurs. This type of dispersion would include the Taylor cone, and all subsequent uneven fissions of the charged droplets in which a jet of much smaller and more highly charged droplets is emitted. The similarity between the droplet jet emerging from the Taylor cone and the droplet jet emerging from the parent droplet when uneven fission occurs has been pointed out by Gomez and Tang[4] and Willoughby *et al.*[14].

The observations of Gomez and Tang[4] and the present observations, Figure 3.5, involve the uneven fission of relatively large (diameter > 5 μm) parent droplets. However, Sheehan and Willoughby[14b] have obtained scanning electron micrographs of dry solid residues from electrospray droplets that show particles below 0.1 μm and many of these have the characteristic elongated teardrop shape of a parent droplet undergoing uneven droplet fission. Fernandez De La Mora has examined the hypothesis that a more or less permanent Taylor cone may exist on tiny droplets that have a sufficiently high rate of evaporation.[16] In such a case, the droplet may be able to reduce its size quickly enough to remain near the Rayleigh limit and sustain the fission process until it crystallizes. Calculations for aqueous solutions have shown this to be highly unlikely as the actual evaporation rate is lower than that required by several orders of magnitude. Regardless of their cause, the observations of Sheehan *et al.* provide good experimental evidence that the uneven droplet fission continues beyond the 5 μm size to 0.1 μm and smaller droplet sizes. The formation of such small droplets requires a sequence of fissions and each of these must be preceded by shrinkage of the parent droplet by evaporative solvent loss (4, 6). An extension of Siu *et al.'s* (9, 10) assumption to include not only the Taylor cone but also all these droplet fissions will be in contradiction with the interpretation (9, 10) of the experimental results on which the original Taylor cone argument was based. These experiments were interpreted to indicate that no solvent evaporation prior to ion emission occurs[9] and that the large ion intensity decrease from water to ethylene glycol as solvents is due to ion emission from the Taylor cone alone for both water and ethylene glycol[10].

Possible alternative explanations of the experimental evidence (9, 10) have been given in a recent review article by Kebarle and Tang[6]. They point out that even if the Taylor cone is the source of some ion emission, this finding does not require that evaporating charged droplets be ruled out as the major sources of ion production in a conventional electrospray. It is likely that both processes contribute ions to ESMS spectra and that the relative contribution of each ion source will probably depend mainly on the volatility of the solvent.

It was also stressed in the review by Kebarle and Tang that the evaporation rate of very small droplets is determined kinetically as opposed to thermodynamically as was assumed by Siu and co-workers. This type of evaporation leads to solvent vapour compositions that cannot be predicted using equilibrium data. Under these conditions, the accuracy of the calculated pH of the droplets presumed to be the source of the cytochrome *c* ions is questionable. Direct measurements of the pH of these droplets is desirable.

3.3.4 The Role of Deformations in the Fission of Charged Fluids

The droplet spray depicted in Figures 3.3, 3.5 was obtained when the Pt electrode was placed close to the droplet stream and at a position where the liquid filament emerging from the vibrated capillary had separated into droplets. The photograph shown in Figure 3.13 illustrates what happens when the Pt electrode is in contact with the liquid filament at a place where the filament was still complete and barely showed the presence of "beads". The situation in this case closely resembles conventional electrospray since the liquid, as in ES, is in contract with an electrode of high potential. Electrophoretic charging of the surface of the liquid should be occurring for the present experiment as is also the case for conventional electrospray. However in the present experiment the liquid jet carries with it the energy of the elastic vibrations transferred by the buzzer. In Figure 3.13 one observes jets of fine spray that occur at the bulges of the liquid beads. This phenomenon provides a compelling example of the destabilization of the surface and the ejection of a fine stream of droplets, by the collaborative effect of elastic deformation bulges and the presence of surface charge.

The Rayleigh stability limit, eq. 2, holds for spherical droplets. Deformations from spherical geometry, induced for example by the presence of elastic vibrations, destabilize the droplets and lead to uneven fission, *i.e.*, emission of a fine droplet spray from the parent droplet, at droplet charge Q that is lower than Q_R at the Rayleigh limit. Gomez and Tang[4] have provided examples for conventional electrospray where deformations lead to droplet fission below the Rayleigh limit. At high droplet densities near the capillary tip, the repulsive fields of charged neighboring droplets probably induce the deformation, see Figure 5 in reference 4. Deformation induced fission could be observed by these authors also at much lower droplet densities that occur some 3-4 cm downstream from the capillary tip. These deformations are probably produced by aerodynamic forces that cause a flattening of the very small droplets prevailing at such distances[4].

A recent study by Davis and Bridges in which exploding charged droplets were observed by light-scattering confirmed that the Rayleigh limit of charge is not reached even by isolated droplets, at rest in a levitating electric field.[15] Aqueous solutions of sodium dodecylsulfate (SDS) surfactant, whose evaporation rate was limited by the formation of a SDS film on the droplet's surface, exploded at an mean value of 89% of the Rayleigh limit. Dodecanol droplets were found to explode at a mean value of 88%. The droplets studied were on the order of 10 - 20 μm. While Davis and Bridges' study was not designed to follow the aspherical distortions that a drop undergoes in the instant before exploding, they also reported no observable droplet distortion up until one video frame prior to the fission event.

It seems that even though a variety of factors exist that induce deformations that lead to uneven fission below the Rayleigh limit, such factors are present in conventional ES for parent droplets all the way down to very small sizes < 0.1 μm (4, 14). This realization introduces a new aspect in the search for the mechanism of gas phase ion formation from the droplets. Iribarne and Thomson[12] compared their predictions for ion evaporation rates from droplets of 0.1 μm size with droplet fission at the Rayleigh limit and concluded that for such and smaller droplet sizes ion evaporation will be faster. However for uneven fission say at 80% of the Rayleigh limit, the calculations of these authors predict, see Figure 4, ref. (12), a very much reduced domain where ion evaporation will be faster than droplet fission. Whether such a domain exists at all except for very small droplets containing only a few charges, where the distinction between the two mechanisms becomes blurred, remains an open question.

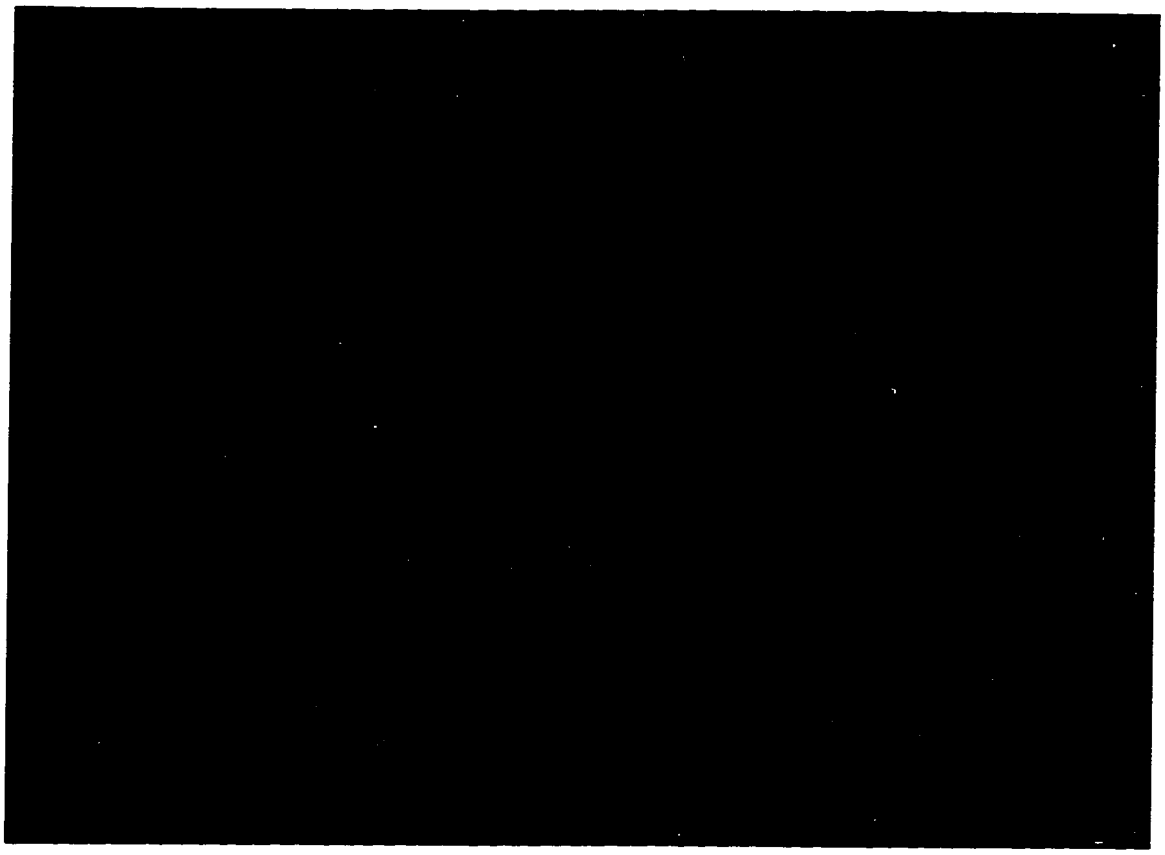

Figure 3.13 Photograph of spray obtained when Pt electrode was in contact with liquid stream at a position where the liquid had not separated into droplets. The spray shown issuing from the bulges along the liquid jet illustrates a compelling example of the destabilization of the surface and the ejection of a fine stream of droplets, by the collaborative effect of elastic deformations and the presence of surface charge.

3.4 REFERENCES

1. Hager, D.B.; Dovichi, N.J. *Anal. Chem.* **66**, 1593 - 1594 (1994).
2. Galley, P.J.; Hieftje, G.M., *Appl. Spectrosc.*, **46**, 1460 - 1463.(1992)
3. Ikonomou, M.G.; Blades, A.T.; Kebarle, P. *Anal. Chem.*, **62**, 957 - 967. (1990)
4. Gomez, A.; Tang, K. *Physics of Fluids*, **6**, 404 - 414. (1994)
5. Sunner, J.; Nicol, G.; Kebarle, P. *Anal. Chem.*, **60**, 1300 - 1307. (1988)
6. Kebarle, P.; Tang, L. *Anal. Chem.*, **65**, 972A - 986A. (1993)
7. Blades, A.T.; Ikonomon, M.G.; Kebarle, P. *Anal. Chem.*, **63**, 2109 - 2114. (1991)
8. Bailey, A.G. *Electrostatic Spraying of Liquids*, John Wiley & Sons, (1988).
9. Siu, K.W.M., Guevremont, R.; Le Blanc, J.C.Y.; O'Brien, R.T.; Berman, S.S. *Org. Mass Spectrom.*, **28**, 579 - 584. (1993)
10. Guevremont, R.; Le Blanc, J.C.Y.; Siu, K.W.M. *Org. Mass Spectrom.* , **28**, 1345 - 1352. (1993)
11. Fenn, J.B.; Mann, M.; Meng, C.K.; Wong, S.F.; Whitehouse, C.M., *Science*, **246**, 64 - 71. (1989)
12. Iribarne, J.V.; Thomson, B.A. *J. Chem. Phys.*, **64**, 2287 - 2294. (1976)
13. Röllgen, F.W.; Bramer-Wegner, E.; Buttering, L. *J. Phys. Colloq.*, **45**, Supplement 12, C9 - 297 (1984); Schmelzeisen-Redeker, G.; Buttering, L.; Röllgen, F.W. *Int. J. Mass Spectrum Ion Processes*, **90**, 139 - 150. (1989)
14. (a) Willoughby R.; Sheehan, E.W.; Jarrell, A.; Marecic, T.; Peddler, R.; Penn, S., *Studies of the Physical Processes of Electrospray*; (b) Sheehan, E.W.;

Willoughby, R., *Photographic Studies of Electrospray Aerosols*. Presented at the Annual Meeting of the American Society for Mass Spectrometry, San Francisco, CA (1993).

15. Davis, E. J.; Bridges, M. A., *J. Aerosol Sci.*, **25**, 1179 - 1199. (1993)

16. Fernandez De La Mora, J., *J. Colloid Interface Sci*, **178**, 209 - 218. (1996)

17. Henion, J. D., *personal correspondence* (1997)

CHAPTER 4:

SPATIAL DISTRIBUTION OF IONS FROM DES

4.1 INTRODUCTION

The apparatus used to study DES has been described in Chapter 3. Droplets were produced by the forced break up of a liquid jet. Images were obtained using bright-field stroboscopic illumination and a video microscope. The video signal was displayed on a 12" TV monitor and recorded onto VHS video tape. The droplets were charged by directing them past a corona discharge at the tip of a fine Pt wire electrode. Droplets acquired sufficient charge to undergo disruption a short distance away from the charging electrode.

In previous chapters, the images of charged parent droplets in the process of producing a short spurt of tiny daughter droplets illustrate how each parent distorts prior to spraying, how the spray begins, the detachment of the spray plume, and the relaxation of the parent droplet to its original spherical shape. These microscopic images enabled the painstaking adjustments of alignment, voltage, frequency, flow rate, etc. necessary for a stable, reproducible spraying droplet event. Once such a system was obtained, clear and detailed views lasting 10 minutes or more were the result. As revealed by stroboscopic imaging, this disruption phenomenon was highly reproducible spatially, temporally and morphologically.

In addition to providing a means for visualization of the droplet disruption process, this technique was found to be a possible alternative to more conventional atmospheric pressure ion sources for mass spectrometry. By using a mass spectrometer, it was found that the spraying droplets produced airborne ions from the ions present in the solution of the droplet. Even when the distance from the spraying droplet and the sampling orifice of the mass spectrometer was greater than the distance typical in conventional ESI-MS, the ion currents obtained were of comparable magnitude.

Due to the stability of the droplet electrospray (DES), another opportunity presented itself; the determination of where in space these ions appeared in relation to the spraying droplet. This determination is the topic of this chapter. Because every droplet produced was of the same composition and size, acquired the same amount of charge, and traveled at the same velocity, all droplets underwent DES at the same point in space. The ionic composition in the vicinity of this spot was what was sought. A small region near the exploding droplet was sampled with a stainless steel capillary through which ions were drawn into a mass spectrometer. The mass spectrometer provided measurements of both the mass/charge ratio and the abundance of the ions detected. The mass/charge measurements provided information on the origin of the ion, *i.e.*, whether they came from the corona discharge or from the solution of the droplet. By making ion mass and ion current measurements at various points around the spraying droplet, maps were obtained that show clearly the distribution of the ions in space. The maps show a dramatic drop-off in corona discharge ion abundance during the droplet's encounter with the corona, followed by a sharply defined zone of sodium ions that came from the tiny jet emitted by the charged droplet.

4.2 EXPERIMENTAL SECTION

Figure 4.1 presents an overview of the apparatus. The solution of interest was pumped through a nozzle comprised of a 34 mm length of 144 μm (o.d.) X 20 μm (i.d.) fused silica capillary, shown at the top of the figure. This nozzle was mounted in a stainless steel union and connected to a high pressure syringe pump (Isco Model 314). A typical flow rate was 0.3 mL/min with small adjustments to the flow being made to eliminate the formation of satellite droplets and maximize stability.

The capillary was vibrated using a piezoelectric buzzer to mechanically generate droplets at regular, precisely timed intervals. Once formed, the droplets were directed past a charging electrode where they were charged by a corona discharge. Airborne ions were collected with a small tube and transported by viscous flow into an evacuated chamber attached to the sampling end of a mass spectrometer.

A photograph of the DES generator is given in Figure 4.2. The base of the device was a precision translation stage of the type used to position optical elements on a laboratory breadboard. A piezoelectric buzzer was used to vibrate the capillary so that the resulting jet of liquid broke up in a regular, controlled manner. A red light emitting diode (LED) was fastened to a length of stiff wire with heat shrinkable tubing and positioned beneath the tip of the Pt charging electrode, pointing upward. The charging electrode, a short (approx. 1 mm) length of 75 μm Pt wire, was soldered to the center conductor of a 12 cm length of RG 8/U coaxial cable. The thick, stiff cable provided structural support and electrical isolation between the high voltage and the mounting structure. The shield was electrically connected to the metal framework

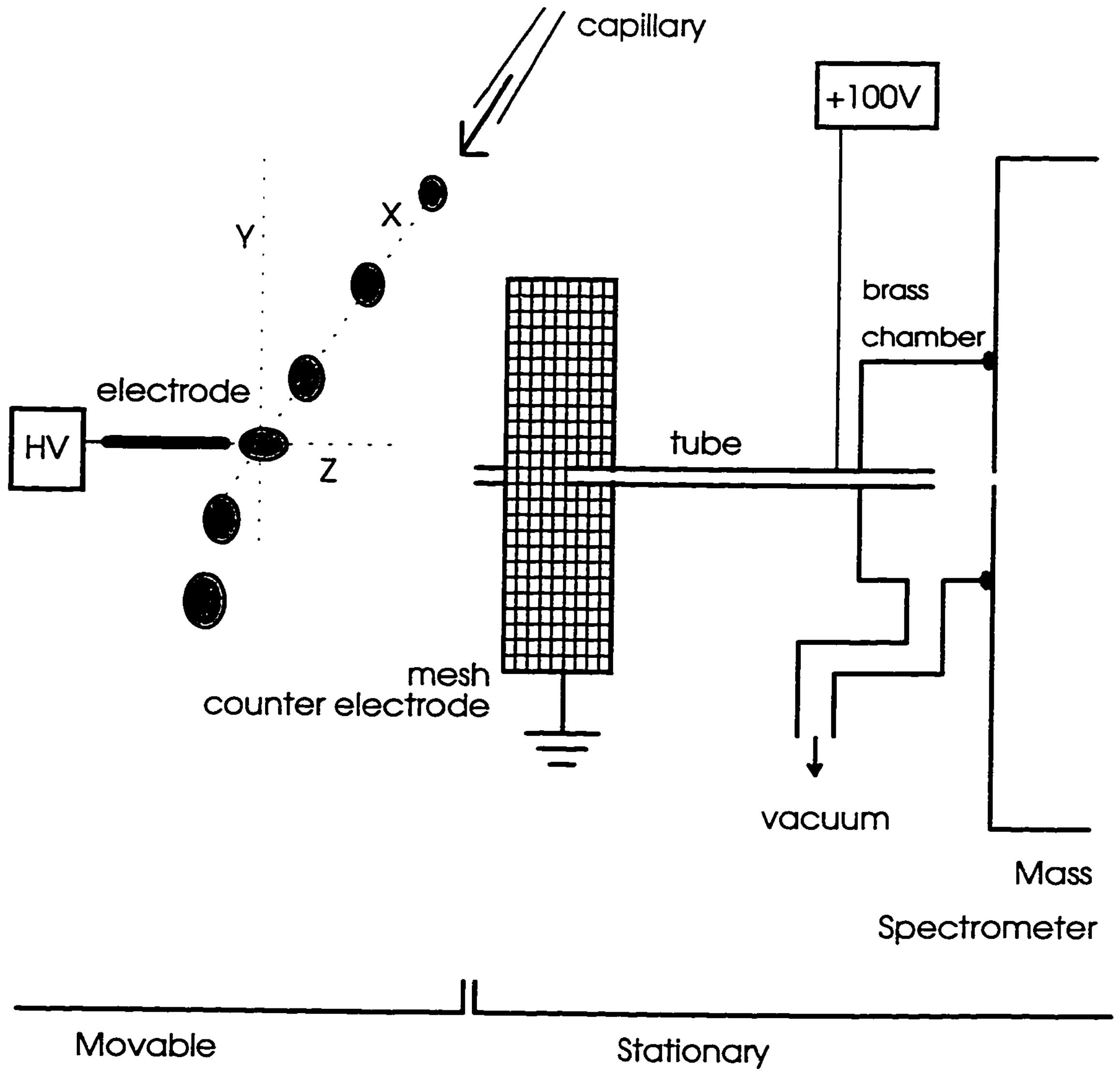

Figure 4.1: Apparatus used for the measurement of ion abundances in the region near the spraying droplet. The droplets were directed past a charging electrode where they acquired sufficient charge to undergo disruption shortly after passing the discharge region. Ions resulting from this event were sampled into a mass spectrometer along with those from the corona. The position of the disrupting droplet was moved about relative to the ion pickup tube. In this way the spatial distribution of the ions present was determined.

of the positioners. The electrode, LED, piezoelectric buzzer, fused silica capillary, and stainless steel union can be seen in the top-view close-up photograph in Figure 4.3.

Heat shrinkable tubing was used to seal the end of the large coaxial cable and the Pt charging electrode can be seen emerging from the tip of this assembly just above the LED in the figure. Also shown is the short piece of Cu wire used to transmit the vibrations of the buzzer to the capillary. This wire was soldered directly to the surface of the buzzer and had a V-shaped notch in the other end to firmly hold the capillary in place.

To avoid plugging the capillary, the inlet end was trimmed off after it was inserted through the graphite ferrule just before the union was assembled. It was then moved into position and both ends of the union were tightened.

Figure 4.4 is another close-up photograph from a different angle. This view better shows the LED support structure and the other end of the stainless steel union. The buzzer was attached to the support with epoxy cement. Figure 4.5 shows the X (horizontal) and Y (vertical) actuators mounted on the rear of the apparatus. These actuators controlled the position of the entire device; the electrode, LED, and droplet generator all moved as one unit relative to the ion pickup tube. The BNC connections for the buzzer and the LED power are shown on the left in the figure and the PTFE-coated high voltage lead is shown on the right. The spring seen in Figure 4.5 was needed to augment the built–in spring in the translation stage used for the base of the apparatus. The built–in spring was not sufficiently strong to ensure accurate motion control, particularly when the direction of motion was from left to right.

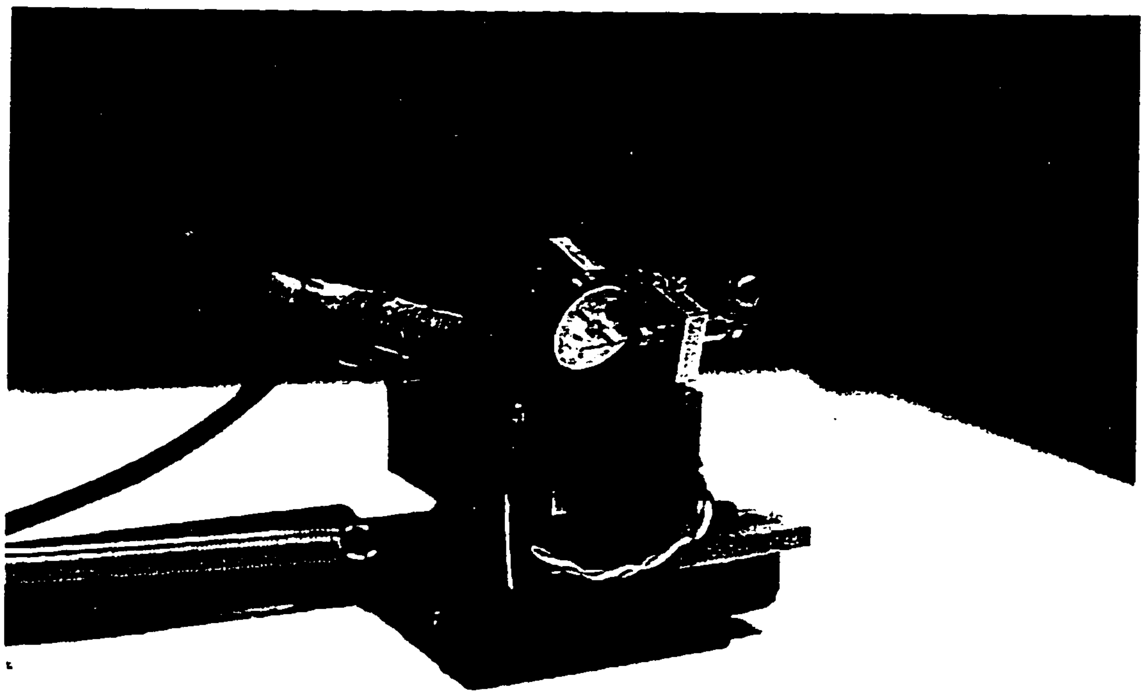

Figure 4.2: Droplet Electrospray Apparatus.

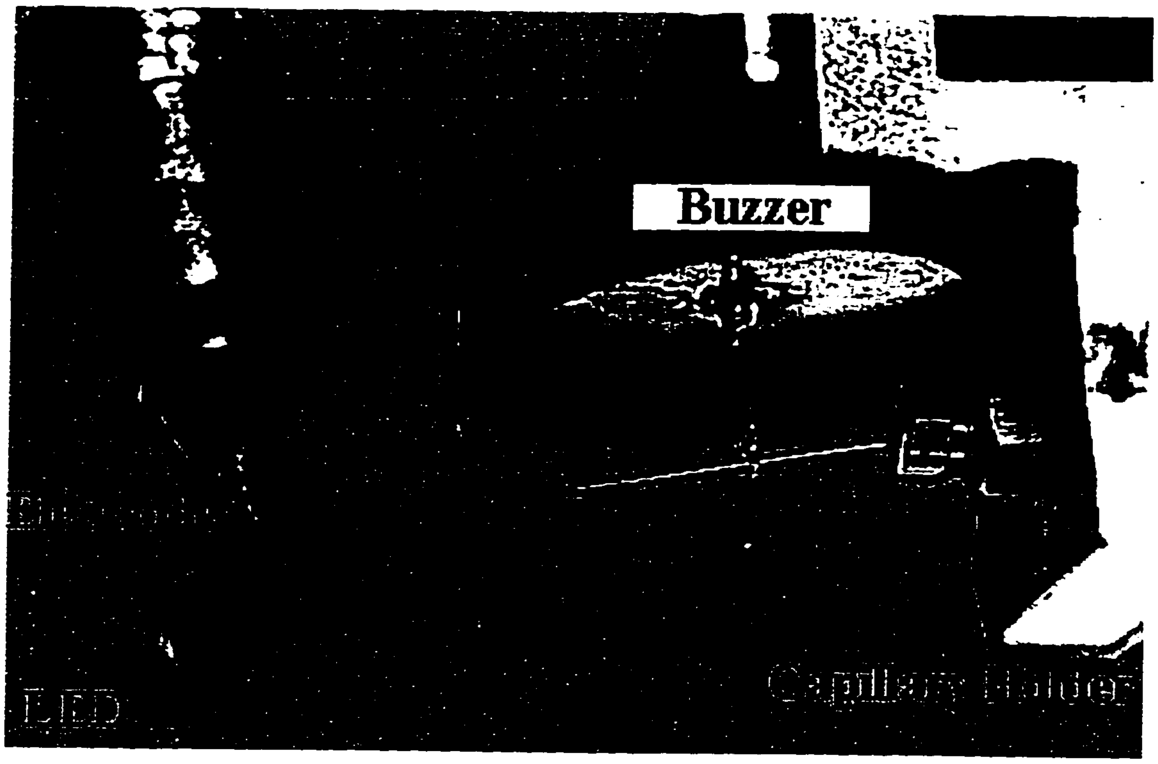

Figure 4.3: Close-up view of the apparatus showing the buzzer, capillary holder, capillary, electrode, and light-emitting diode (LED).

The photograph in Figure 4.6 shows the left side view of the device. In this view the attachment of the vertical translation stage to the base is visible. The electric and liquid connections can also be seen. The apparatus was made using very strong, heavy materials to minimize unwanted vibration and the need for constant fine adjustments.

The top view photograph of Figure 4.7 shows how the apparatus was configured to place the exploding droplet as close as possible to the mass spectrometer inlet. The electrode assembly was mounted on micropositioners to allow fine adjustment of the Y and Z position of the charging electrode relative to the droplet stream. Precise adjustment of the droplet-to-discharge distance (Z) was necessary for a stable spray. By adjusting the vertical position of the charging electrode, it was possible to ensure that the DES jet was emitted from the droplets in the focal plane of the video microscope. The thick cable provided a stiff, solid support for the charging electrode. It was connected to a Spellman CZE1000R high voltage DC power supply.

The stationary section of the experimental system is shown in detail in Figure 4.8. A metal mesh counter electrode was positioned at a distance of 1 cm from the charging electrode and held at ground potential. The mesh electrode was a rectangle 11 cm wide and 9 cm high with 2.5 mm openings and was made of stainless steel. An ion pickup tube was inserted through an opening at the center of the mesh. This transfer line was made from a 94 mm length of 1.6 mm o.d. 380 μm i.d. stainless steel tubing. It protruded 3 mm through the mesh, insulated electrically with a 3 mm piece of heat-shrinkable tubing where the transfer line entered the mesh. The distance

Figure 4.4: Front view close-up showing the LED, electrode, capillary, stainless steel union and buzzer. The droplet production mechanism and charging electrode were mounted together on an X-Y translation stage so that the position of the spraying droplet could be set relative to the stationary ion pickup tube and wire mesh counter electrode.

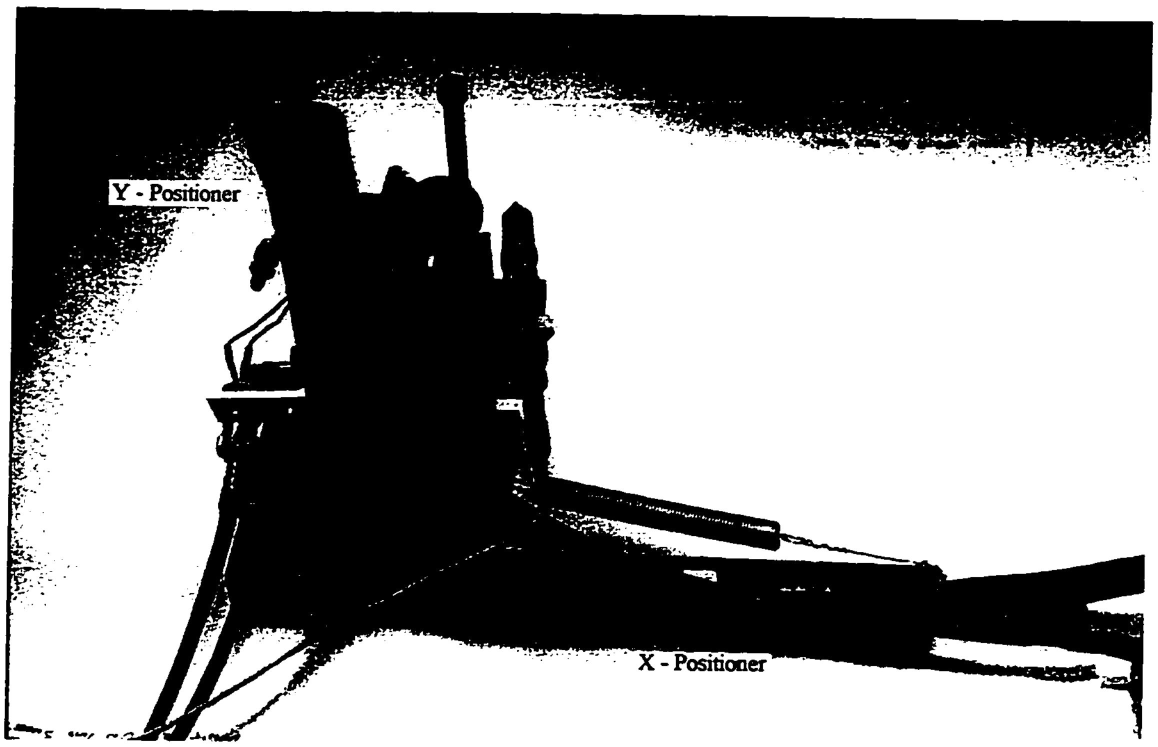

Figure 4.5: Rear view showing the electrical cables for the buzzer, LED, and high voltage. A Newport 855C programmable controller allowed computer control of the X and Y position with a precision of +/- 0.5 μm in both vertical (Y) and left-right (X) directions. The X and Y positioners are indicated in the figure. All measurements were made in this X-Y plane at a fixed distance (Z) from the spraying droplet.

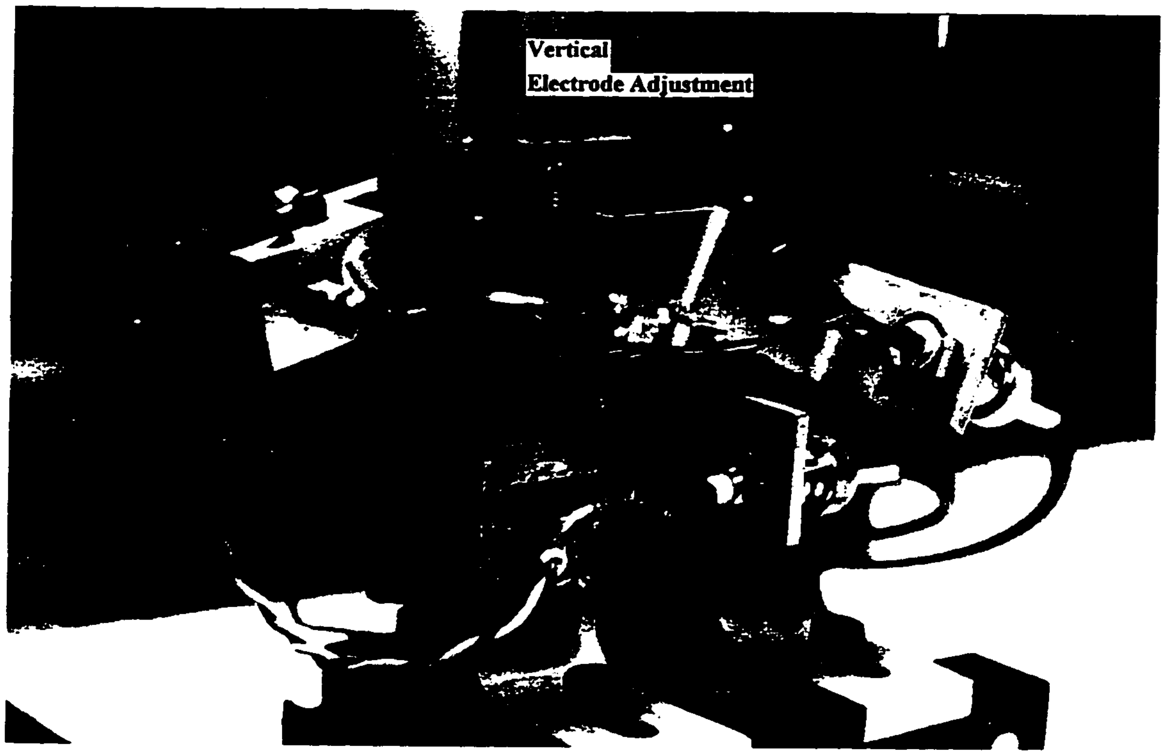

Figure 4.6: Left side view showing the electrical connections, LED support and vertical electrode adjustment.

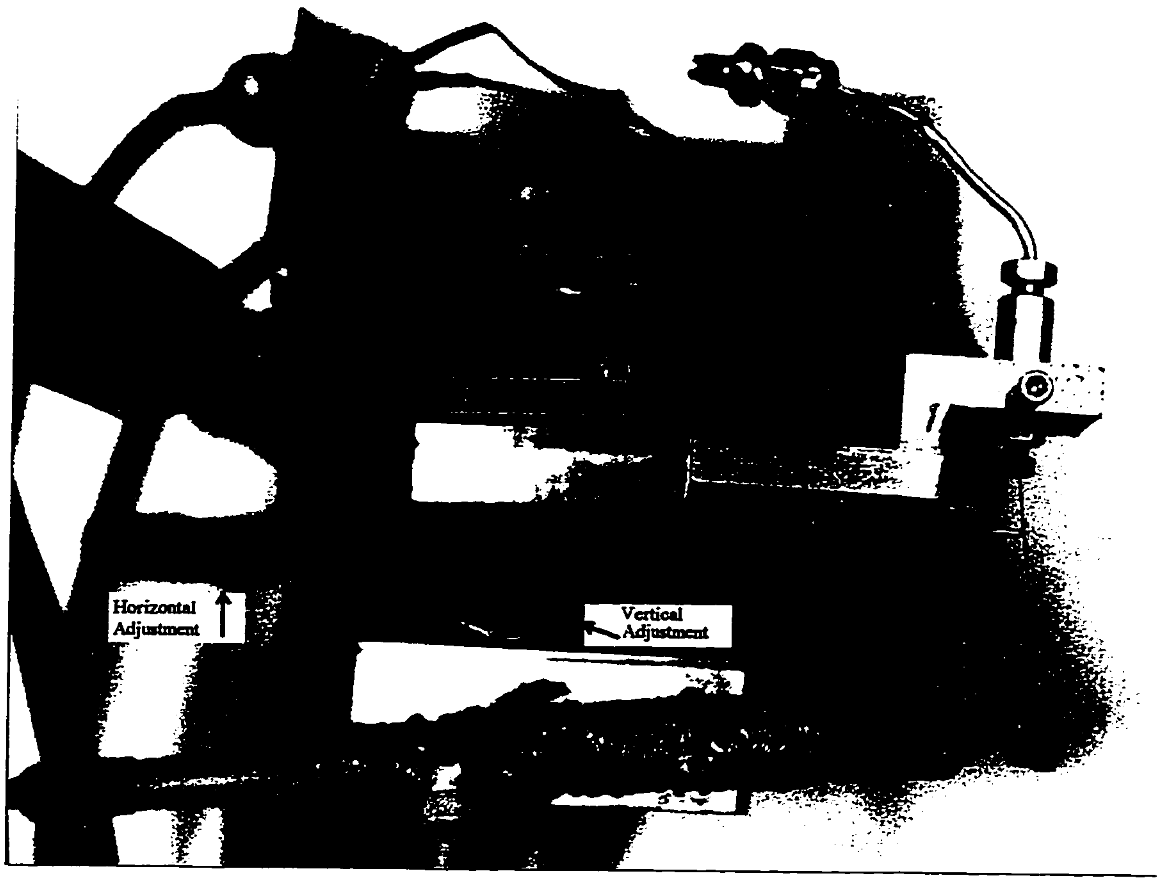

Figure 4.7: Top view. The electrode position adjusters are indicated.

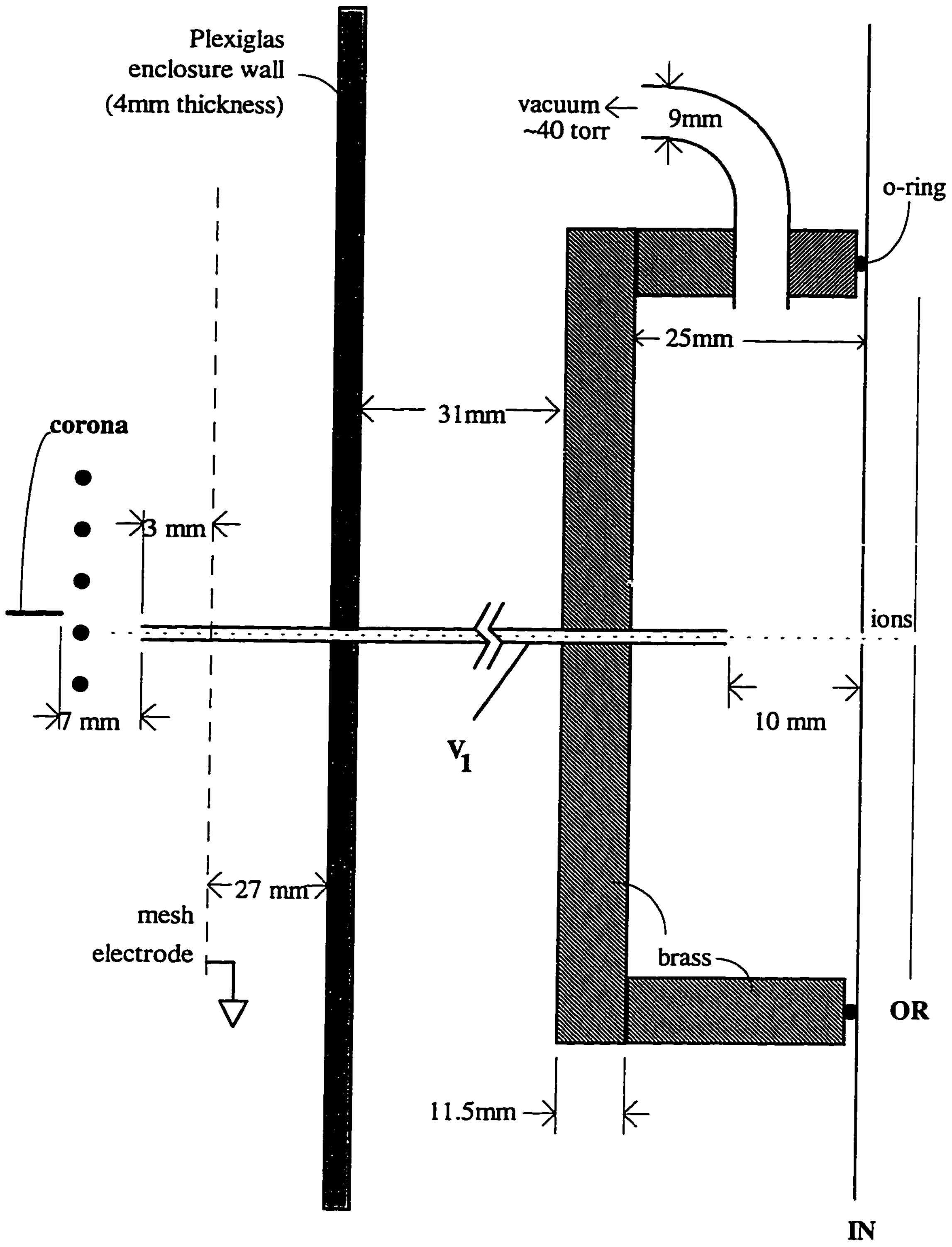

Figure 4.8: The stationary portion of the DES apparatus.

from the spraying droplet to the opening of the transfer line was therefore 7 mm when the droplet was directly opposite the tube opening. The grounded mesh counter electrode served to minimize the changes in the electric field intensity near the discharge as the electrode tip was moved about relative to the transfer line.

The transfer line was connected to a dc power supply and held at a potential, V_1, of +100 V vs. ground. A brass chamber, which had been evacuated to 40 torr, was used to position the transfer line's outlet opposite and 10 mm away from the inlet orifice of the Sciex Taga mass spectrometer. This chamber was 76 mm in diameter and was designed to fit on the end of the mass spectrometer. It was held at the same potential as the transfer line. The use of the transfer line enabled spatially resolved ion abundance measurements by sampling the ions from a small space near its entrance. The transfer line also allowed for modular design of the apparatus by providing a conduit for the transport of ions from near the exploding droplet to the mass spectrometer.[1] The droplet generation, charging, motion control and video microscopy could therefore be carried out in a chamber isolated from the air currents in the laboratory and vibrations from the mass spectrometer.

The pieces labeled **IN** and **OR** in Figure 4.8 refer to the mass spectrometer's inlet and orifice plates respectively. The Viton o-rings sealing the brass chamber to the inlet plate also provided electrical insulation allowing the chamber and its attached transfer line to be held at a different potential than the mass spectrometer inlet plate.

The droplet production mechanism and charging electrode were mounted together on an X-Y translation stage so that the position of the spraying droplet could be set relative to the stationary ion pickup tube and wire mesh counter electrode. A Newport 855C programmable controller allowed computer control of the X and Y position with a precision of +/- 0.5 μm in both vertical (Y) and left-right (X) directions. All measurements were made in this X-Y plane at a fixed distance (Z) from the spraying droplet.

Due to the nature of stroboscopic imaging, any deterioration in the degree of reproducibility of the droplet production step was observed as blurring, ghosting or instability in the normally clear, stationary droplet images. The position of the charging electrode relative to the droplet stream, pumping rate, buzzer frequency, electrode voltage, position of the mesh counter electrode and voltage on the pickup capillary were all adjusted until a sharp, stable stroboscopic image of a spraying droplet was obtained and could be maintained as the DES ion source was moved about. Fine adjustment of the voltage and the droplet-electrode spacing was done to maximize DES ion signal.

The apparatus was housed in a Plexiglas enclosure, primarily to eliminate disturbances due to air currents. Even though the enclosure was not completely airtight, it also helped minimize the fluctuations of other atmospheric conditions, such as humidity, solvent vapor concentration and traces of interfering airborne contaminants.

Solutions of 10^{-4} M NaCl in HPLC grade methanol were used in all measurements. This solution was pumped at 0.46 mL/min. and the buzzer operated at 70 kHz. These conditions produced a stream of droplets with a diameter of 60 μm spaced 321 μm apart and traveling past the charging electrode at 22.5 m/s. The pressure in the fluid system was measured by a mechanical pressure gauge. Typical pressure was about 2000 psig (13.8 MPa). When the pressure began increasing, pluggage of the capillary was indicated.

The mass spectra obtained from the DES experiments were made up of solvated ions and therefore showed a distribution of peaks corresponding to the number of solvent molecules associated with the ion. This mass spectrum was also typical of conventional electrospray. The mass spectrometer was set up to record the ion current for three ions, 129.9, 119.7 and 215.8 AMU. The first ion was produced directly by the discharge and was composed of a cluster of 4 methanol molecules and a proton. The second was a Na(methanol)$_3^+$ cluster and the third was a Na(methanol)$_6^+$ cluster. These ions came from the solution. The actual masses of these three ions were 129, 119 and 215 AMU. The mass discrepancy was attributed to miscalibration of the mass spectrometer. The ion currents, in counts per second (cps), were written to disk by the mass spectrometer computer (Apple Macintosh) at a constant rate of about 1 data point/second. At the same time the entire droplet electrospray (DES) ion source was moved back and forth in the horizontal (X) direction past the pickup tube at the rate of 0.10 mm/sec. Five such scans were done, one at each of five vertical (Y) positions. Many scans were rendered unusable due to drift in the signal during scanning, even though care was taken to minimize such drift before starting the scan. Usually the drift could be traced to a change in the trajectory of the droplet stream. The region was scanned twice, first in one direction followed immediately by the opposite direction. In the absence of drift, this procedure resulted in a symmetrical record of ion currents as the reverse scan retraced the forward scan.

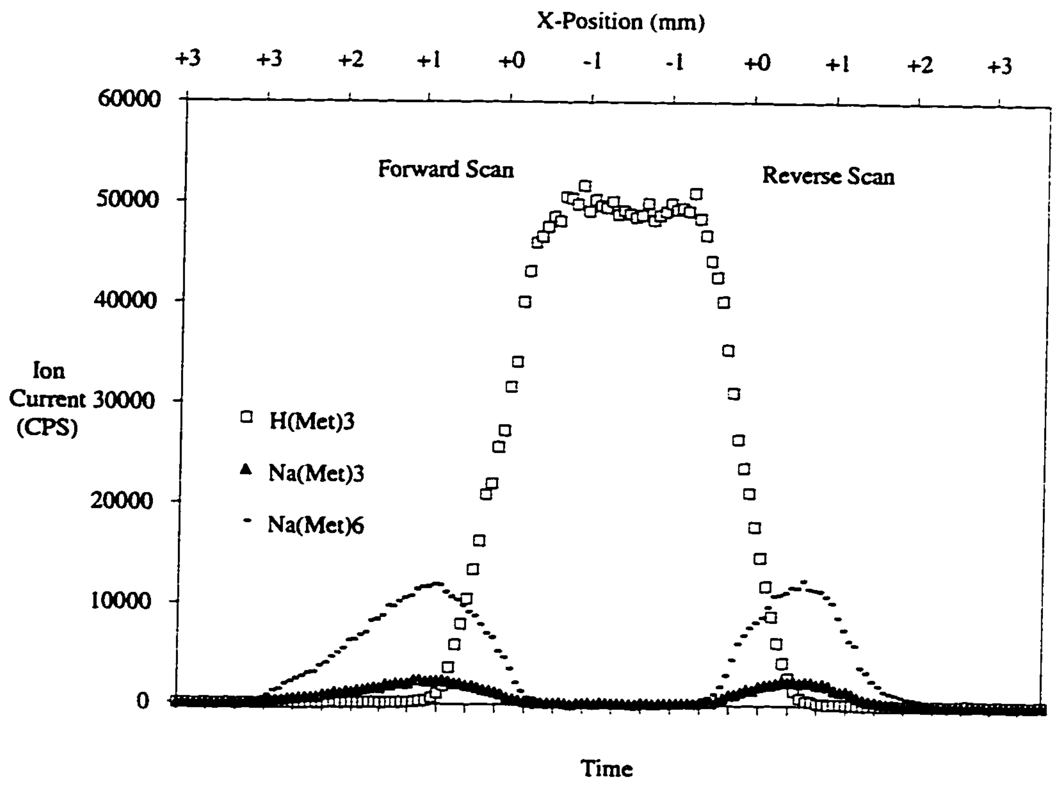

Figure 4.9: Three ions were monitored by the mass spectrometer as the X-position was scanned backward and forward in the direction of motion of the droplets. In this figure, the position of the pickup tube began downstream of the disruption event, traveled past the spraying droplet into the region upstream of the charging electrode, where only discharge ions were found. The motion was then reversed and the path retraced. The time series was converted to signal vs. position data and both directions of the scan averaged. This procedure was repeated for a range of vertical (Y) positions to map the region.

Figure 4.9 shows the raw data collected during a typical bi-directional scan. Data points were recorded at regular intervals while the X position was scanned at a constant rate. Good scans, i.e. those with forward and reverse scan data that were approximate mirror images, were averaged to compensate for any residual drift. The data from the mass-spectrometer computer were converted from Motorola to Intel floating point format and the ion count measurements collected from the two scanning directions averaged and plotted against X-position for each Y-position.

To better visualize the data obtained, all the measurements were used to generate a response grid to facilitate 3D surface plotting. Binary data conversion from Motorola format to Intel format was done using a Pascal program and gridding calculations were done using Microcal Origin, both on an Intel 80486-based microcomputer. Visualization of the ion map was done on an IBM RS/6000 workstation using IBM Visualization Data Explorer.

4.3 RESULTS

Figure 4.10 presents the ion concentration of two ions vs. the position of the ion pickup tube. The base of the figure defines a plane in front of and perpendicular to the spraying droplet and is 4 mm on each side. The height and shading of each surface indicate the abundance of the ion at any point in the measurement plane. The top right of the figure corresponds to the direction from which the droplets were originating. The center of the figure roughly coincides with the position of the charging electrode. The surface that represents the discharge ions is the one, labeled

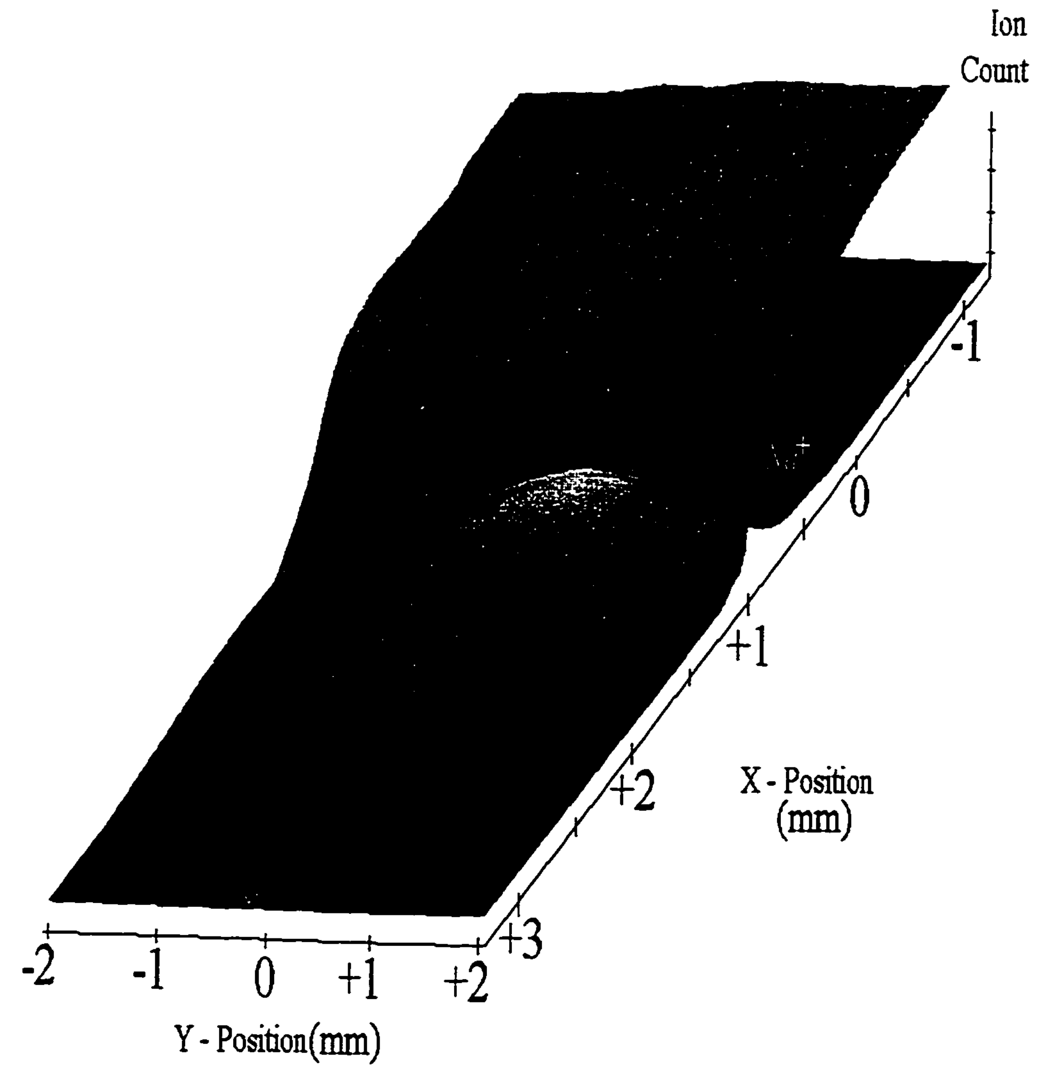

Figure 4.10: This picture shows the ion abundance map for two ions monitored over a planar zone in front of the spraying droplet. The X-direction corresponds to the direction of travel of the droplet stream. The Y-direction is the vertical direction. The region scanned was a plane 10 mm away and perpendicular to the spraying droplet. The ion abundance is represented by the height and shading of the surfaces with higher abundances being represented by more height and lighter shading. The label Na^+ actually represents Na^+ with 6 methanol molecules attached and the H^+ has 3 methanol molecules attached.

H^+, that is higher and more lightly shaded towards the upper right of the figure.

Two sharply defined regions can be seen in the discharge ion population. The region prior to where the droplets enter the corona was rich in discharge ions. As the ion pickup tube passed opposite the corona, discharge ion current quickly dropped to zero. This drop resulted from the droplet blocking the discharge ions.

A small zone was discovered where solvated sodium ions from the spraying droplet were detected. The presence of this zone showed the highly localized nature of the ion beam produced by these precisely oriented and reproducibly charged spraying droplets. It also revealed the distribution of the corona ions and showed that the region where DES ions were most abundant was well separated from the discharge ions.

Figure 4.11 illustrates the procedure used to create the ion distribution maps. A platinum charging electrode, attached to the high voltage power supply is shown at the left of the figure. The three circles represent three droplets that are traveling past the charging electrode from the upper right to the lower left. Gray squares symbolize discharge ions dispersing from the tip of the electrode. Sodium ions in the droplet solution are shown as triangles. As the droplets move past the corona, they acquire positive charge by field charging. Soon after passing near the corona, a disruption occurs where the droplet loses some of its charge and volume. Gas phase ions and ion-solvent clusters are quickly formed from the daughter droplets produced by the parent droplet's uneven fission.

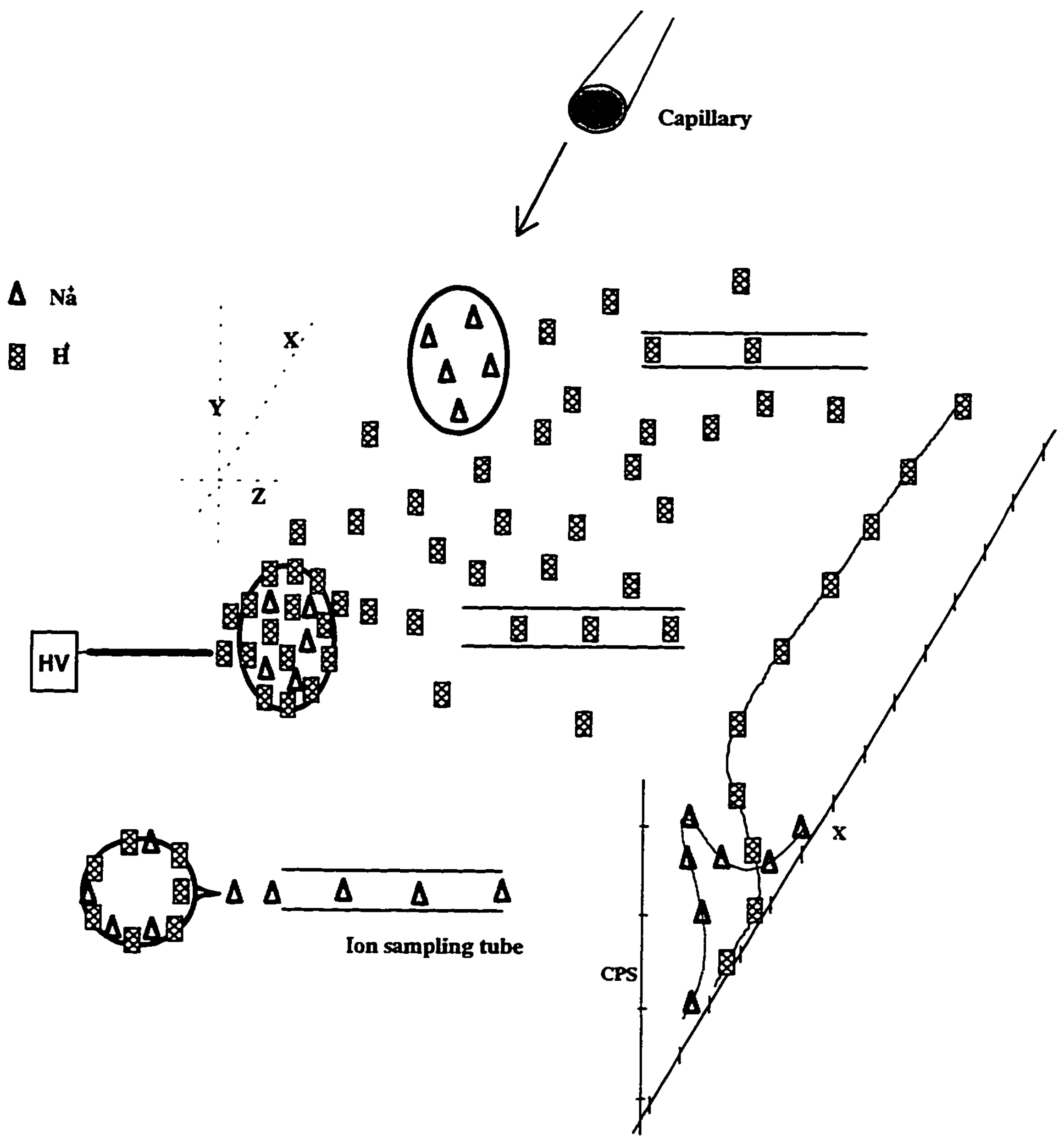

Figure 4.11: Three droplets and three measurement positions. As the droplet travels from upper right to lower left, it encounters the corona discharge, becomes charged and sprays. Triangles represent sodium ions and shaded squares represent discharge ions, H$^+$. The ion sampling tube transfers ions from the zone near its inlet to the mass spectrometer. The figure illustrates how the apparatus was used to map out the ion distributions. As the position of the sampling tube's inlet is moved relative to that of the spraying droplet, the type and abundance of the ions picked up is determined by the mass spectrometer and recorded. The plot at the right corresponds to a slice through the response surface of Figure 4.10. The data from all the slices are used to generate the spatially resolved ion abundance maps.

Samples of gas at various locations near the droplet disruption event are drawn into the ion pickup tube. Three tube positions are drawn in the illustration. The ions picked up at each X position are mass-analyzed and counted by the mass spectrometer. The plot at the right of the figure corresponds to these measurements, and can be thought of as a slice through the ion maps of Figure 4.10 at one Y (vertical) position.

Another set of experiments was done at higher flow rates. In this case, the solution was pumped at 0.80 mL/min. and the buzzer operated at 101 kHz. These conditions produced a stream of droplets with a diameter of 63 μm spaced 380 μm apart and traveling past the charging electrode at 38.5 m/s. The faster moving droplets changed the ion map considerably (Figures 4.15 and 4.16). The zone of highest DES ion abundance was smeared downstream of the charging point. For Figure 4.16, the measurement plane was shifted downward so that the bottom boundary of the shielding region of the corona discharge ions could be seen.

Figures 4.12a-h are a collection of narrow slices through the ion abundance map of Figure 4.10. These slices show how selected sections of the two intersecting ion abundance surfaces look. Figures 4.12a-d show how the discharge ions are blocked by the close approach of the droplet, entering from the negative (top) X-direction. As the discharge ion abundance disappears and the droplet begins to spray, solvated sodium ions appear. Farther along the droplet's path, the DES ion abundance tapers off. The distribution of DES ion abundance along the direction of droplet motion can be seen to be very skewed, having a rapid onset followed by a gradual disappearance.

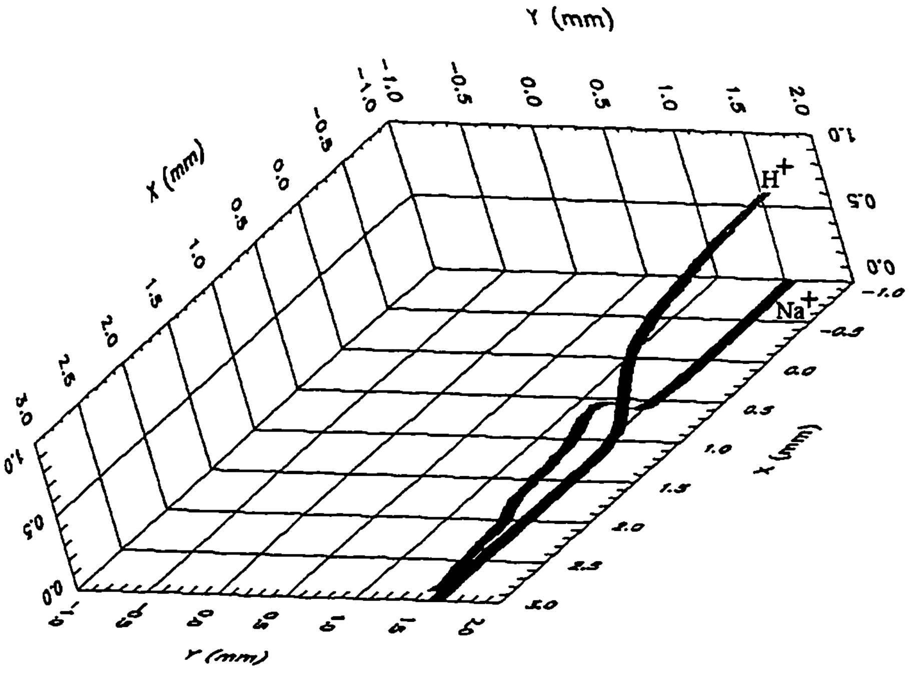

Figure 4.12a: Slice through the ion abundance map of Figure 4.10. Figures 4.12a-d show how the discharge ions are blocked by the close approach of the droplet, entering from the negative (top) X-direction.

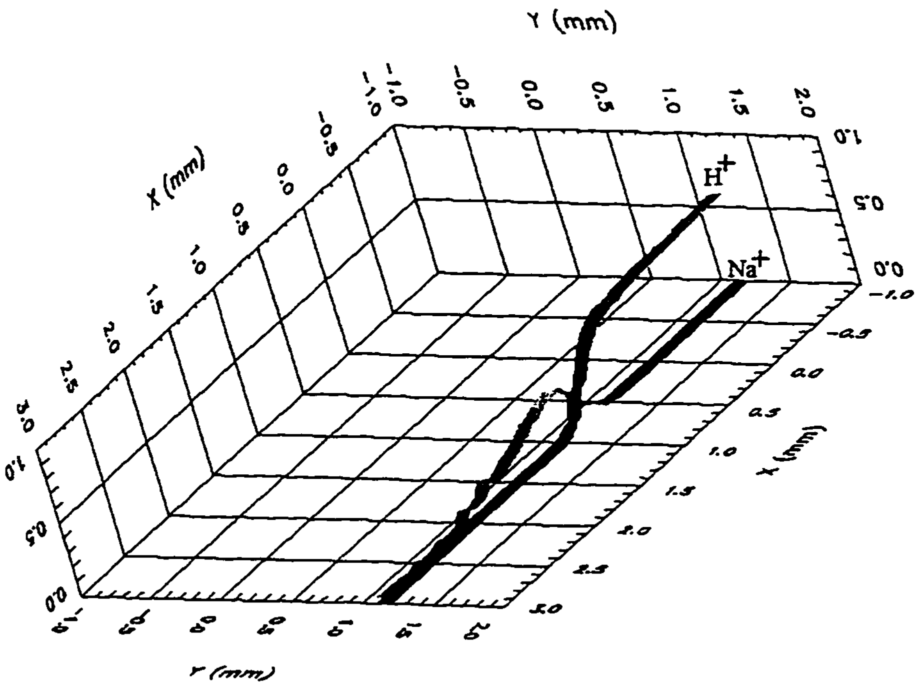

Figure 4.12b. As the discharge ion abundance disappears and the droplet begins to spray, solvated sodium ions appear. Farther along the droplet's path, the DES ion abundance tapers off.

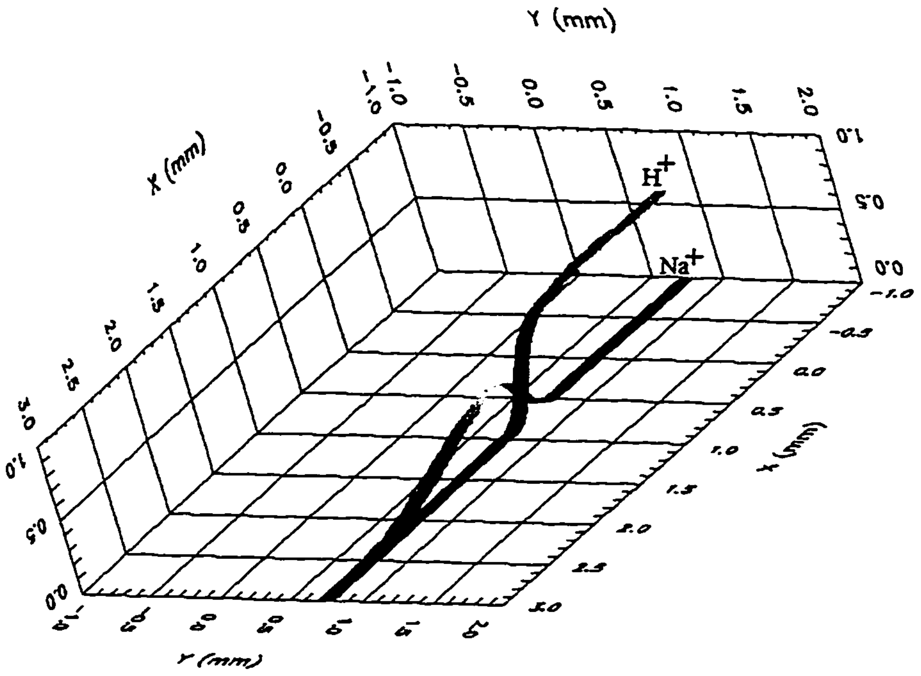

Figure 4.12c The distribution of DES ion abundance along the direction of droplet motion can be seen to be very skewed, having a rapid onset followed by a gradual disappearance.

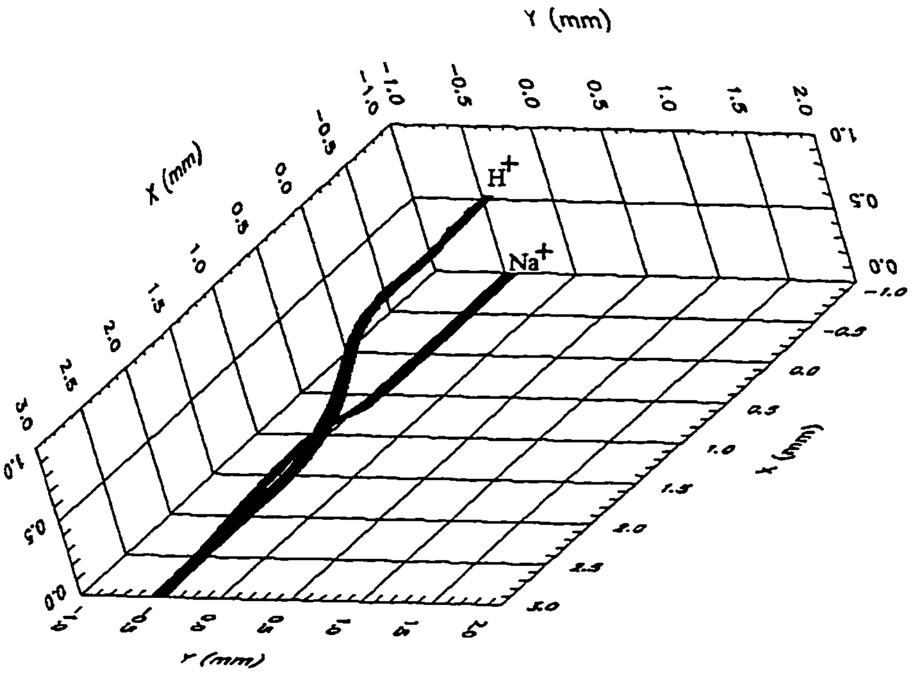

Figure 4.12d Along the side of the droplet stream is a region where the sodium ion abundance disappears, while the discharge ions remain undetected. It is therefore apparent that the ion emission from the droplet is effectively spatially isolated from the discharge ion current, and there exists a distinct region where DES ions may be sampled into a detection system without also including corona discharge ions.

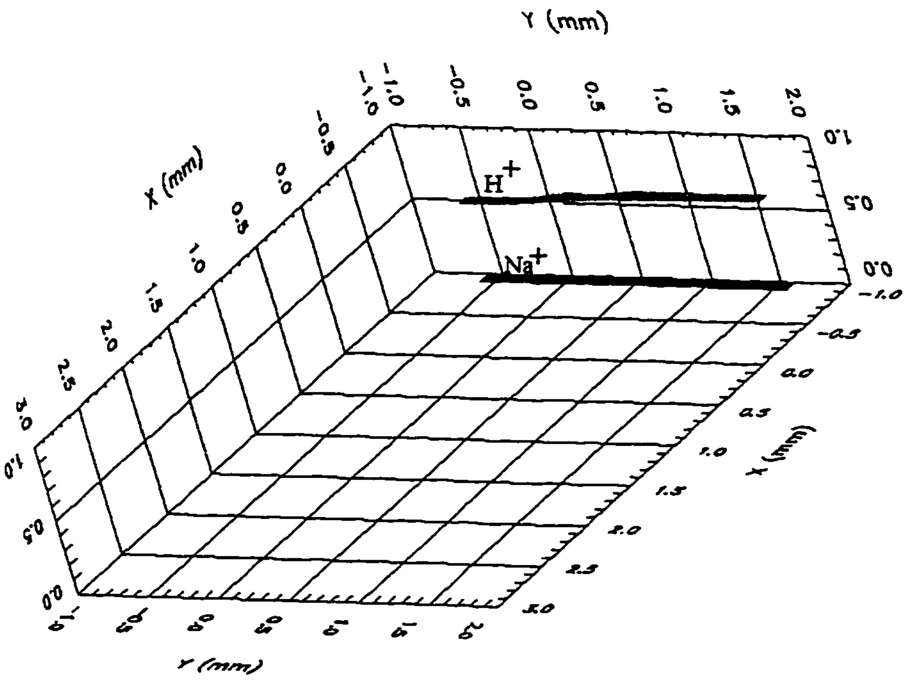

Figure 4.12e Figures 4.12e-h show slices of the ion abundance map across the direction of droplet motion. In Figure 4.12e, we see only discharge ions.

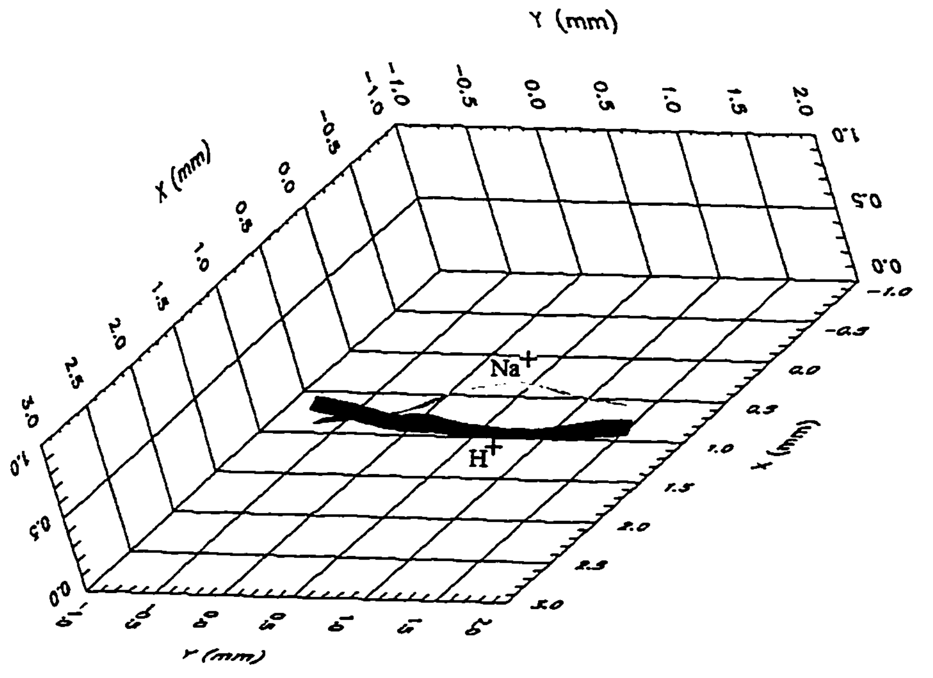

Figure 4.12f In Figure 4.12f, across the point where DES ions first appear, we see a Gaussian distribution of solvated sodium ions existing in a trench of discharge ions, higher at the sides and lowest in the middle.

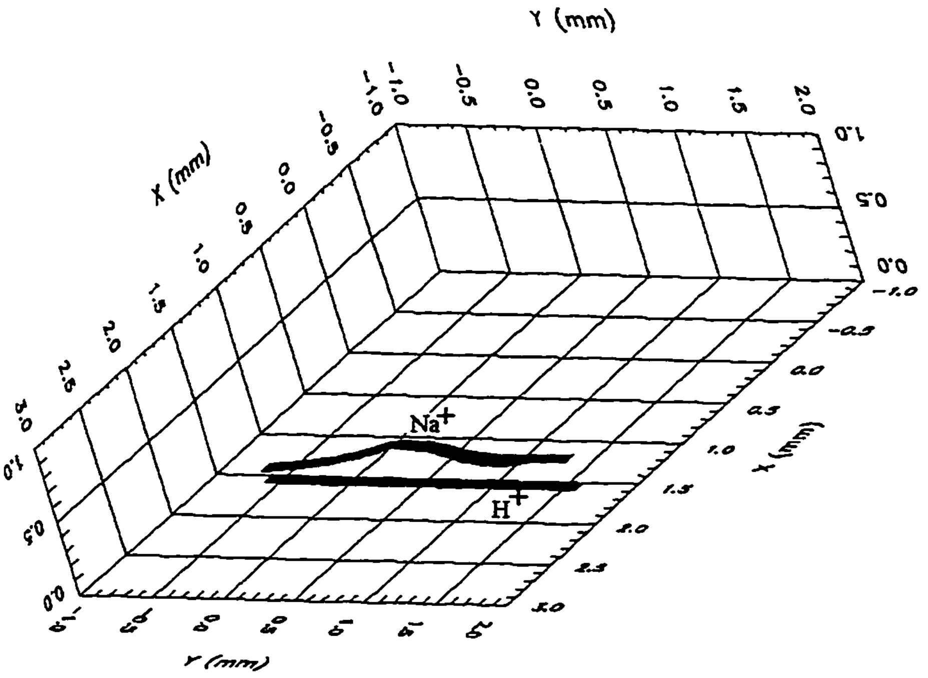

Figure 4.12g 0.5 mm farther downstream, in Figure 4.12g, discharge ions are no longer observed and there is a Gaussian distribution of DES ions.

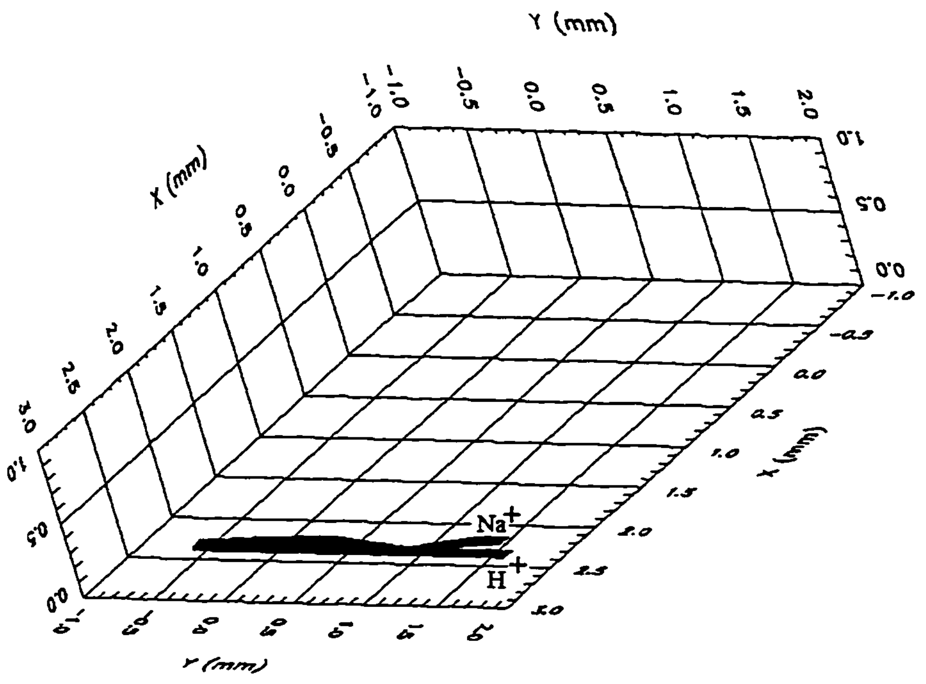

Figure 4.12h The last slice across the ion abundance map, Figure 4.12h, shows essentially no ions.

Along the side of the droplet stream is a region where the sodium ion abundance disappears, while the discharge ions remain undetected. It is therefore apparent that the ion emission from the droplet is effectively spatially isolated from the discharge ion current, and there exists a distinct region where DES ions may be sampled into a detection system without also including corona discharge ions.

Figures 4.12e-h show slices of the ion abundance map across the direction of droplet motion. In Figure 4.12e, we see only discharge ions. In Figure 4.12f, across the point where DES ions first appear, we see a Gaussian distribution of solvated sodium ions existing in a trench of discharge ions, higher at the sides and lowest in the middle. 0.5 mm farther downstream, in Figure 4.12g, discharge ions are no longer observed and there is a Gaussian distribution of DES ions. The last slice across the ion abundance map, Figure 4.12h, shows essentially no ions.

4.4 DISCUSSION

While it was known that the droplets absorbed ions from the corona as they flew by, the abrupt and apparently complete drop-off of discharge ion current as the ion pickup tube passed the point of droplet charging was an unexpected observation. A possible explanation is that the droplets become polarized when in the strong electric field near the charging electrode. This polarization results in the side of the droplet facing the electrode becoming negative with respect to the far side with its surface therefore acting as a sink for the positive ions being ejected from the corona, effectively shielding the pickup tube from those ions. This method of charging of

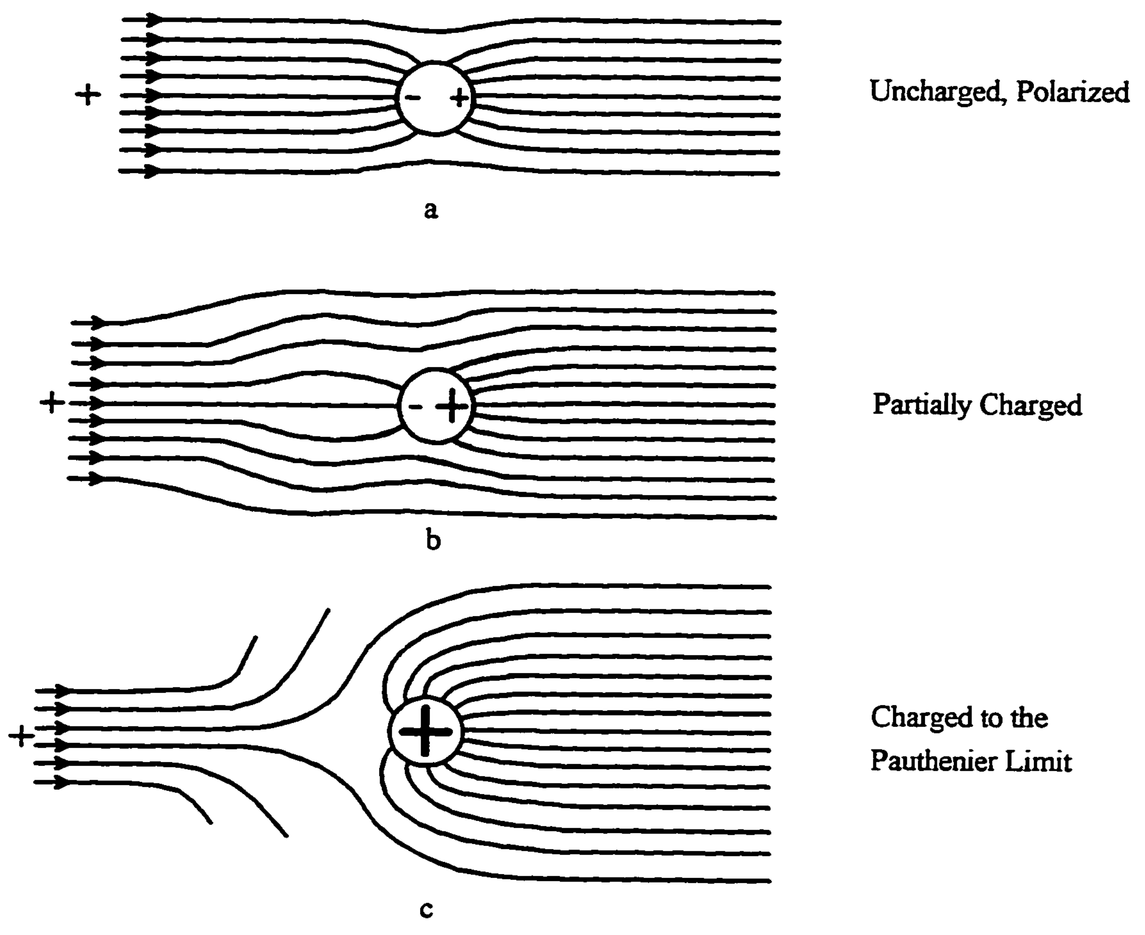

Figure 4.13 Distortion Of An Electric Field During Droplet Charging. The polarization of the uncharged droplet, a, results in a convergence of field lines at its left surface and a divergence of field lines from its right surface. The distortion increases as the droplet acquires charge, until, c, the limiting charge has been reached. In fact, the Rayleigh limit will be reached first.

particles has been studied[2,3] and is considered to be the dominant charging mechanism in corona charging systems.

Figure 4.13 shows how an electric field is perturbed near a droplet with permittivity, ε, greater than unity. The electric field in the figure is oriented from left to right and there is assumed to be a source of positive ions traveling also from left to right. In addition, there is assumed to be no flux of negative ions traveling from right to left. Such a system was used in the DES apparatus. One can see a concentrating effect, i.e. the polarization of the droplet exerts a distorting influence on the electric field resulting in more field lines entering the droplet on the left and exiting the droplet on the right than would be expected based on its size alone. Positive ions streaming away from the corona discharge follow the field lines to the droplet where they attach to the surface.

A consequence of this explanation is that the spraying droplets are doing so without actually reaching the Rayleigh limit of charge. The strong polarizing influence of the electric field would concentrate the positive charge at the far side of the droplet, locally exceeding the surface cohesive force and causing the subsequent Taylor cone formation and jet emission. The high directional reproducibility of the DES emission is consistent with this hypothesis. A corresponding build-up of negative charge on the droplet surface nearest the corona would be offset by attachment of positive ions from the discharge, thus preventing a surface disruption toward the charging electrode.

As the droplet acquires surface charge from the corona, it distorts the electric field nearby until it reaches a maximum charge, the Pauthenier limit,[3] shown in Figure

4.13c. At the Pauthenier limit all positive charges in the gas are repelled from the droplet so that no further charging of the droplet can occur. This limit of charge is given by:

$$Q_0 = 12\pi a^2 \varepsilon_0 E_0 \qquad (1)$$

Where: a = droplet radius, ε_0 = droplet permittivity, E_0 = electric field strength.

Q_0 is independent of ion flux density, in this limiting case. The electric field, E_c, at the tip of a metal capillary of radius r_c held at a potential V_c vs. a planar reference electrode separated from the capillary by a distance d is given by:[4]

$$E_c = 2V_c / r_c \ln(4d / r_c) \qquad (2)$$

For $V_c = 3.2$ kV, $r_c = 37.5$ μm, and $d = 6$ cm, $E_c = 1.5 \text{ X } 10^9$ V/m. Therefore, assuming $E_c \cong E_0$, equation (1) gives $Q_0 = 2.4 \text{ X } 10^{-10}$ coulomb for methanol droplets of 22 μm radius. If course reaching this Pauthenier limiting situation is contingent on also satisfying the Rayleigh limit of charge for the droplet.

The Rayleigh limit of charge, Q_R, is given by:[4]

$$Q_R^2 = 64\pi^2 \varepsilon_0 \gamma a^3 \qquad (3)$$

For methanol droplets of 22 μm radius, Q_R is $1.2 \text{ X } 10^{-12}$ coulomb.

When Q_0 is evaluated for methanol droplets of the sizes under consideration in DES, it is 2 orders of magnitude greater than the Rayleigh limit. This result means that the method of field charging under consideration here as the probable means for charging the droplets in DES is more than up to the task of imparting the Rayleigh limit if charge to the droplets.

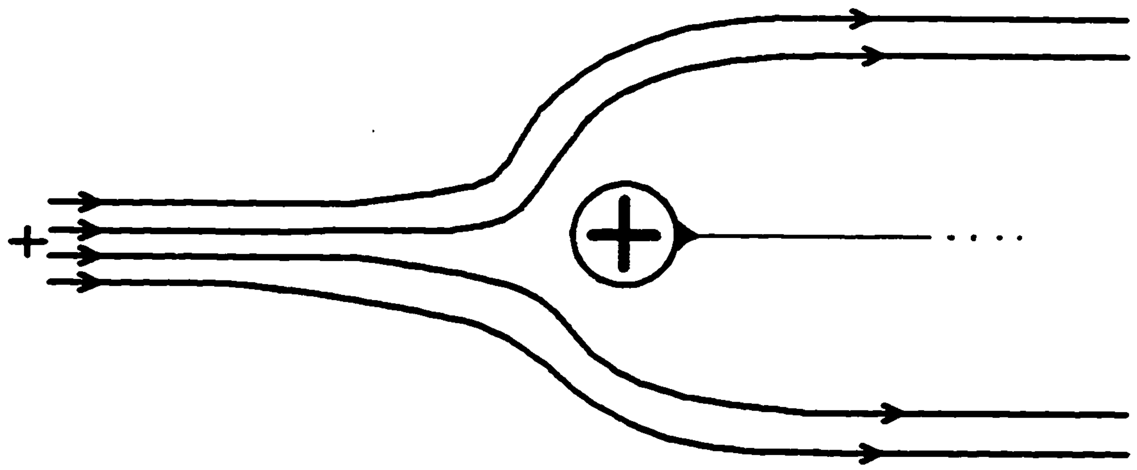

Figure 4.14 Shielding effect of droplet charge. The highly charged droplet, in this case undergoing droplet electrospray, distorts the nearby electric field in such a way as to deflect the positive ions entering from the left of the figure around the droplet and away from the site of the transient Taylor cone forming on the down-field side of the droplet. This shielding effect is thought to be responsible for the apparent spatial isolation between the DES ions and the discharge ions of the corona.

While it is not possible for the droplets to reach the Pauthenier limit, they still distort the electric field such that their presence between the discharge electrode and the ion pickup tube acts as a more effective shield than the uncharged droplet preventing the discharge ions from interfering with the DES ions. Figure 4.14 shows where the spray jet originates in relation to the electric field surrounding a charged droplet undergoing DES.

The ion maps obtained from these measurements suggest that the DES event results in either a narrow beam of ions or a narrow beam of highly charged droplets that give up ions at or very near the entrance to the pickup tube. In either case, the spatial distribution of the sodium ions as measured in this manner was found to be concentrated in a small region that was well separated from the large ion current coming from the corona. The fact that discharge ions were not found in the same region of space as the DES ions, means that the chain of events leading ultimately to ions in the gas phase takes place in the shielded zone behind the parent droplet. This discharge-free zone was apparently quite large as none of the ions characteristic of the corona could be seen across the entire width, in the Y-direction, of the measurement surface downstream of the droplet spray, $X > +1.0$ mm in Figure 4.10 and 4.15. Figure 4.16 shows the discharge ion free region extends out to $Y = +2.0$ mm. It is not known how much further away from the charging electrode, i.e. $Z > 7$ mm, the shielded area persists.

Higher droplet velocities resulted in smearing of the DES ion region (Figure 4.15). They also necessitated higher voltages and a closer approach of the droplet trajectory to the charging electrode, making stability more difficult to achieve and maintain. Too low a velocity caused problems due to the more easily deflected droplets being prone to striking the charging electrode, again resulting in instability. Because the spraying droplet is traveling at a steady velocity, the measurements of ion abundances vs. position relative to the charging electrode can be thought of as measurements of ion abundances vs. time as the droplet begins to spray, reaches a maximum and stops spraying. It thus becomes trivial to measure the length of time the droplet spray event lasts, t_e as pointed out by Fernandez De La Mora[5]. This measurement may be of limited value given the fact that the droplets are almost certainly still acquiring charge as they begin to spray, however there is more to be gained from following the evolution of these tiny ion sources than the time each individual explosion lasts, namely how steady ion production throughout each burst is, whether different ions appear at different times during the burst, and how the ions produced from exploding droplets are dispersed.

The slices in the X-direction through the ion abundance map, shown in Figures 4.12a through 4.12d, indicate a rapid onset of ion production followed by a slow decay. The slices in the Y-direction, shown in Figures 4.12e through 4.12h show a normal distribution of ions around the axis of the droplet's jet. Experiments with multiple ions could be done using this technique to determine if different ions have different rates of ion production from exploding droplets. If different ions are shown to exhibit different ion production rates along the exploding droplet trajectory, it would allow for discrimination between these different ions based on spatially resolved detection. If, on the other hand, it was desired to avoid or minimize

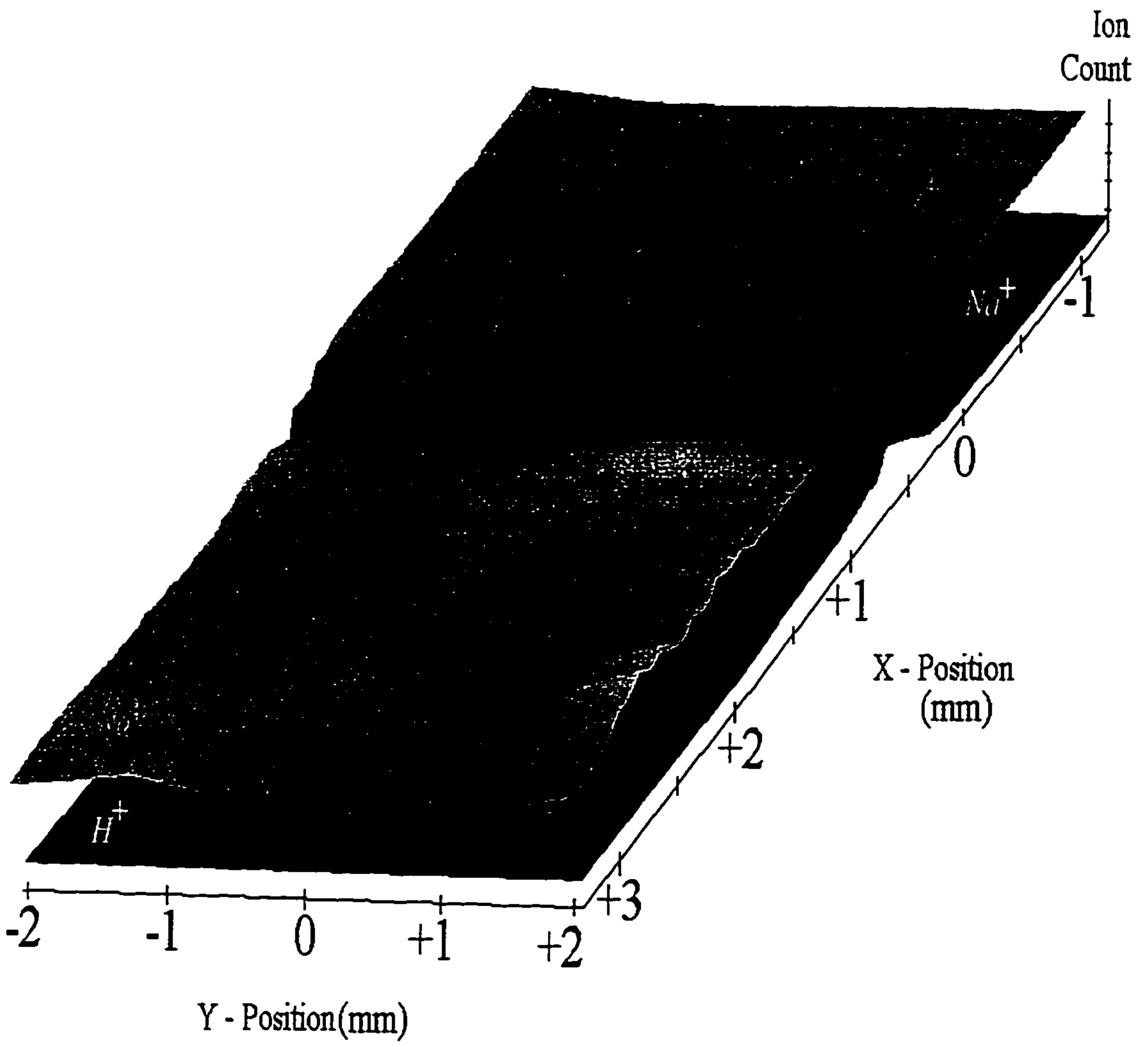

Figure 4.15: The ion abundance response surface resulting from measurements made on faster moving droplets. The presence of sodium ions was detected over a smeared out region downstream of the charging electrode.

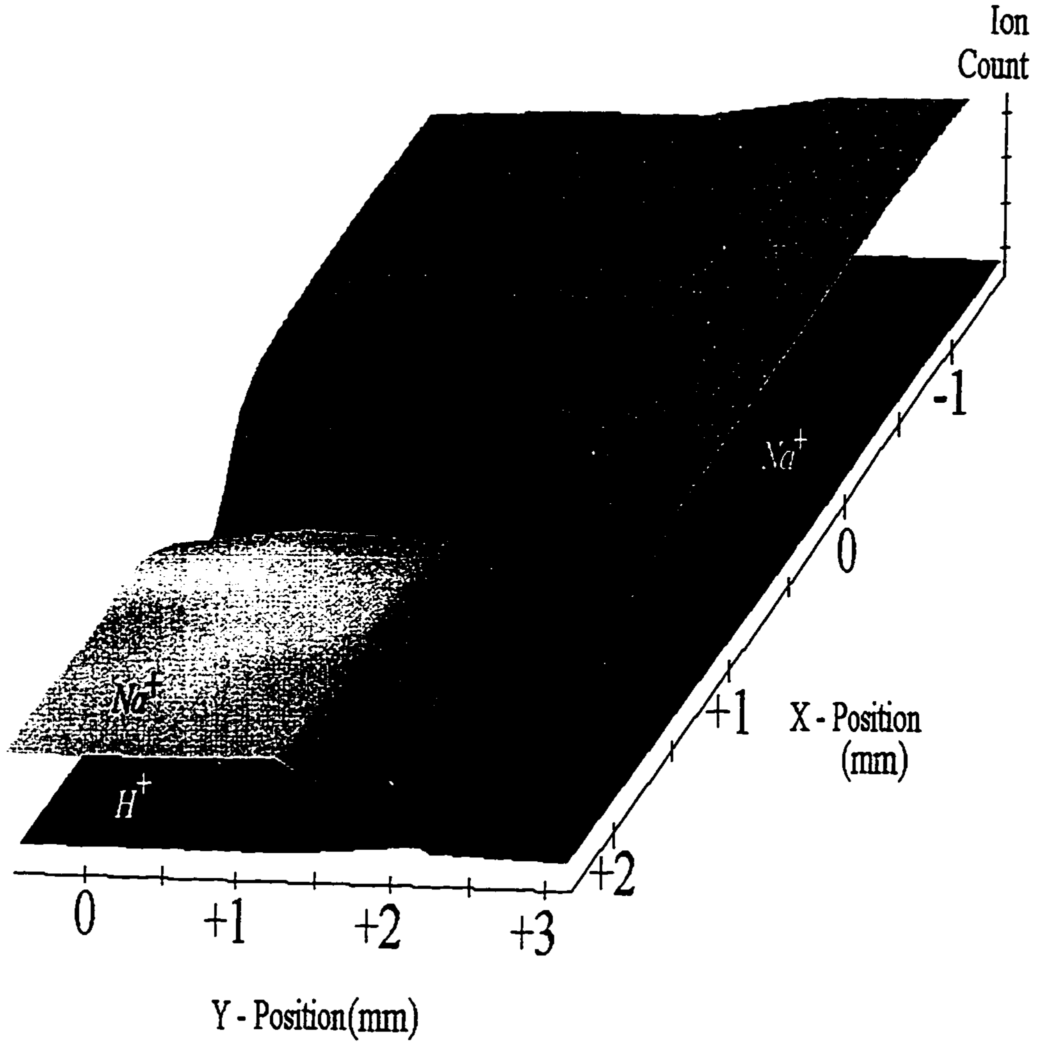

Figure 4.16: With the same experimental conditions as Figure 4.12, but showing the edge of the discharge ion region wrapping around the sodium-rich region.

discrimination, the entire path of ion production would have to be included in designing the apparatus to collect the DES ions. Such a design should be feasible because the zone free of corona discharge ions extends for a long distance along the ion production trajectory.

The spatial distribution data obtained via these experiments have shown the feasibility of constructing a high performance liquid chromatography-mass spectrometry (HPLC-MS) interface based on a DES ion source. The flow rates typical of HPLC are a good match for the flow rates used in the droplet generator. By using the ion transfer line to sample only the region of space near the exploding droplets where the highest concentration of ions is found, sensitivity can be enhanced while interference from corona discharge ions may be avoided.

The parent droplets may also be easily collected and recovered after spraying by directing them into a suitable collector. Because the post-spray parent droplets retain some charge, it should also be possible to electrically select fractions of the chromatographic eluent in real time based on the ionic species detected by mass spectrometry, thus allowing preparative HPLC using DES-MS detection to be carried out.

A pair of electrostatic deflection plates could be used to direct the charged droplet stream into or away from a collection tube, depending on whether the column effluent was to be retained or discarded. The decision to collect or discard the column effluent would be made in real time based on the current mass-spectrometric analysis. Charged droplets can be directed with high precision and high speed using electrostatic or magnetic deflection and this principle is employed in ink-jet printers.[6]

Preliminary data using the apparatus described here as an HPLC-MS interface is the subject of the next chapter.

4.5 REFERENCES:

1. Lin, B.; Sunner, J., *J. Am. Soc. Mass Spectrom.*, **5**, 873 - 85 (1994)
2. Bailey, A.G., *Electrostatic Spraying of Liquids*, John Wiley & Sons, (1988).
3. Pauthenier, M; Moreau-Hanot, M J, *Phys. Rad.*, **3**, 590-613. (1932)
4. Kebarle, P.; Tang, L., *Anal. Chem.*, **65**, 972A - 986A. (1993)
5. Fernandez De La Mora, J., *J. Colloid Interface Sci.*, **178**, 209 - 218. (1996)
6. Brody, H., *PC-Computing*, **3**, #11, 254 - 256 (1990)

CHAPTER 5:

DES BASED HPLC-MS INTERFACE

5.1 INTRODUCTION

This chapter describes experiments that demonstrated the use of droplet electrospray (DES) as an interface between high performance liquid chromatography (HPLC) and mass spectrometry (MS).

In this technique, a sample solution was pumped through the HPLC column where the sample mixture was separated into individual components. The eluent containing the separated sample components was delivered to the DES generator where the solution stream was broken into droplets, the droplets were charged with the corona discharge needle and, when they sprayed, they gave off ions. The ions were directed through a capillary transfer tube into a mass spectrometer where they were mass analyzed and their abundances determined.

One of the advantages of the DES-based HPLC-MS interface is that it is ideally suited to the higher flow rates used in packed column HPLC. HPLC-MS interfaces usually provide some means for minimizing the quantity of solvent vapor introduced into the mass spectrometer, in order to alleviate the load placed on the pumping system. This action is inherent in the DES ion source where the majority of the solvent is carried away by the parent droplets and only the electrospray from these droplets is used in the analyses. By appropriate sizing of the droplet production apparatus, a stream of droplets suitable for DES may be obtained over a range of solvent delivery rates. Very large flows could be handled by means of a splitting device designed to use a small proportion of the total flow in the production of the droplet stream while directing the bulk of the flow to some other detector or to waste.

One can therefore accommodate whatever flow rate the chromatographic separation requires.

The stroboscopic imaging in Chapter 2 confirmed that, once charged by the discharge electrode, each droplet of column eluent sprays off only a small fraction of its volume and does so only once, at the same location in space and in the same direction. This phenomenon is the key behind the ability to use the DES as a reasonably sensitive ion source. Because of the reproducibility inherent in the DES event, a tight spatial distribution (see previous chapter) of analyte ions can be obtained. As a consequence of this tight distribution, a good sensitivity can be achieved, even with flow rates 10 times higher than those used in conventional electrospray.

The bulk of the flow does not end up entering the mass spectrometer and can be put to other uses. The ability to recover most of the HPLC eluent is another advantage of the DES ion source and makes possible high speed fraction collection based on electrostatic droplet deflection.

Another potential advantage of the DES ion source is higher sensitivity. Initial results showed sensitivities comparable to conventional electrospray. It was felt that, with further efforts at optimization, sensitivity could be significantly improved.

Several disadvantages were also found with the DES ion source. One of the more troublesome was its extreme mechanical sensitivity to alignment. As a consequence, anything that has an influence on alignment such as vibration, air currents, flow rate drift, mechanical flexing, etc. will affect the DES ion current. The key to getting each droplet to spray in exactly the same position lies in very reproducible charging of the droplets.

Variations in the amount of charge acquired by the droplets result in dramatic changes in not only when (and consequently where) each droplet will spray, but in whether it will spray at all. The tight spatial distribution of the DES ions imposes very close tolerances on the range through which the spray can occur without affecting the ion current. As observed through the video microscope, the droplets pass within less than a droplet radius of the corona discharge. The electric field in this region is very non-uniform hence the field experienced by a droplet changes rapidly with only a slight change in droplet trajectory. Consequently, all electrical and mechanical components of the apparatus must be as stable as is reasonably affordable. This requirement adds to the expense.

In addition to the requirements of maintaining constant droplet speed and trajectory past the charging electrode are constraints on the properties of the droplets themselves. Specifically, the conductivity and surface tension of the droplets' solution each play roles in both how much charge is acquired and how much charge is required to cause stable droplet fission. Changes in these solution parameters will be immediately reflected in the spraying droplets. To minimize variability in the conductivity and surface tension of the LC effluent, all chromatography was done isocratically and samples were prepared in solutions of identical composition to the mobile phase, with the exception of the addition of analyte.

The alignment sensitivity lies at the heart of the signal's noise. Noise is the enemy of good analytical results. The DES ion source must be aligned, and kept aligned, manually under a microscope. Video microscopy greatly facilitates making the necessary adjustments, but the need to mount a microscope on the apparatus still remains. Allowing for microscopic observation adds to the cost and complexity of the instrumentation and is thus another disadvantage.

It was found to be difficult or impossible to obtain stable DES conditions for purely aqueous solutions. As the voltage applied to the Pt charging electrode was increased, the water droplets were strongly attracted toward this charging electrode. It was impossible to find conditions of voltage and flow rate that allowed the droplets to pass by the electrode and still pick up enough charge to spray. The inability to use aqueous solutions poses an obvious restriction to the types of chromatography with which this technique can currently be used.

Finally, the DES was found to be useful in the positive ion mode only. When operating with negative potentials, the corona appeared at lower voltages than with positive potentials. Therefore lower electric fields were available to polarize the droplets and it was found to be impossible to charge them to the point of disruption. Consequently, DES could not be used to make negative ions.

5.2 EXPERIMENTAL SECTION

The apparatus described in the previous chapter was used directly in this next set of experiments, with one significant modification: the Isco syringe pump was replaced with an HPLC instrument manufactured by Shimadzu. The UV detection cell was removed from the chromatograph and the column effluent directed into the droplet generator through a short length of 1/16" stainless steel tubing.

The column used was an Hamilton PRP-1 part #79425, 150 X 4.1 mm packed with 10 μm styrene-divinylbenzene copolymer beads. The mobile phase was HPLC grade methanol. To align the ion source with the mass spectrometer ion pickup tube, a solution of 10^{-4} M NaCl was pumped through the system. The instrument was then adjusted to maximize the signal of one of the $Na^+(methanol)_x$ clusters. Once aligned, no further changes were made to the position of the DES generator relative to the ion transfer line. It should be noted here that the possibility of discrimination between different analyte ions based on their potentially different rates of ion production along the exploding droplet trajectory had not yet been considered. If, after further investigation, such discrimination was shown to be present, the apparatus should be either constructed to minimize it or optimized for the ionic species of interest. The NaCl solution was then flushed out and injections of several samples were made. It was desired to show that a DES–based HPLC–MS interface was functional and that component peaks could be observed. After that, several injections of a peptide were made to determine the detection limit of the technique.

5.3 RESULTS AND DISCUSSION

5.3.1 The Ion–Solvent Clustering Problem

Figure 5.1 is a typical mass spectrum obtained from the NaCl solution. Obvious from Figure 5.1 is the fact that the sodium ions from the solution were present in several different degrees of solvation. In the figure, the numbers over each peak in the mass spectrum indicate the number of methanol molecules attached to the sodium ion. It was immediately apparent that the high degree of solvent clustering was a major impediment to quantitative work using this interface. In order to quantify sodium, the proportion of sodium in each form would have to remain constant. Otherwise the signals from all the clusters would have to be included, a requirement that would increase the time required for each scan and consequently decrease the number of scans acquired per unit time over the length of the chromatogram. It was clearly preferable to adjust conditions to eliminate all but one form of sodium ion reaching the mass spectrometer.

To this end, various modifications to the way the ions were handled between the pickup tube and the mass spectrometer orifice were tried. Initially, the brass chamber used in the spatial resolution experiments of Chapter 4 was used. This chamber provided the vacuum necessary to create the viscous flow used to transport the ions through the tube. Because a large volume of methanol vapor was introduced through this tube from the droplet aerosol, this brass chamber contained a substantial amount of methanol. Since purging the region of ion production with nitrogen was not an option due the need for an extremely quiet atmosphere there, the brass chamber was modified to allow a stream of nitrogen to be introduced into the reduced pressure region between the exit of the ion transfer line and the inlet of the mass spectrometer, where the purge gas flow could not disrupt the droplet stream. It was hoped that this purge gas would lower the methanol concentration sufficiently to reduce the amount and size of the ion–solvent clusters reaching the mass spectrometer.

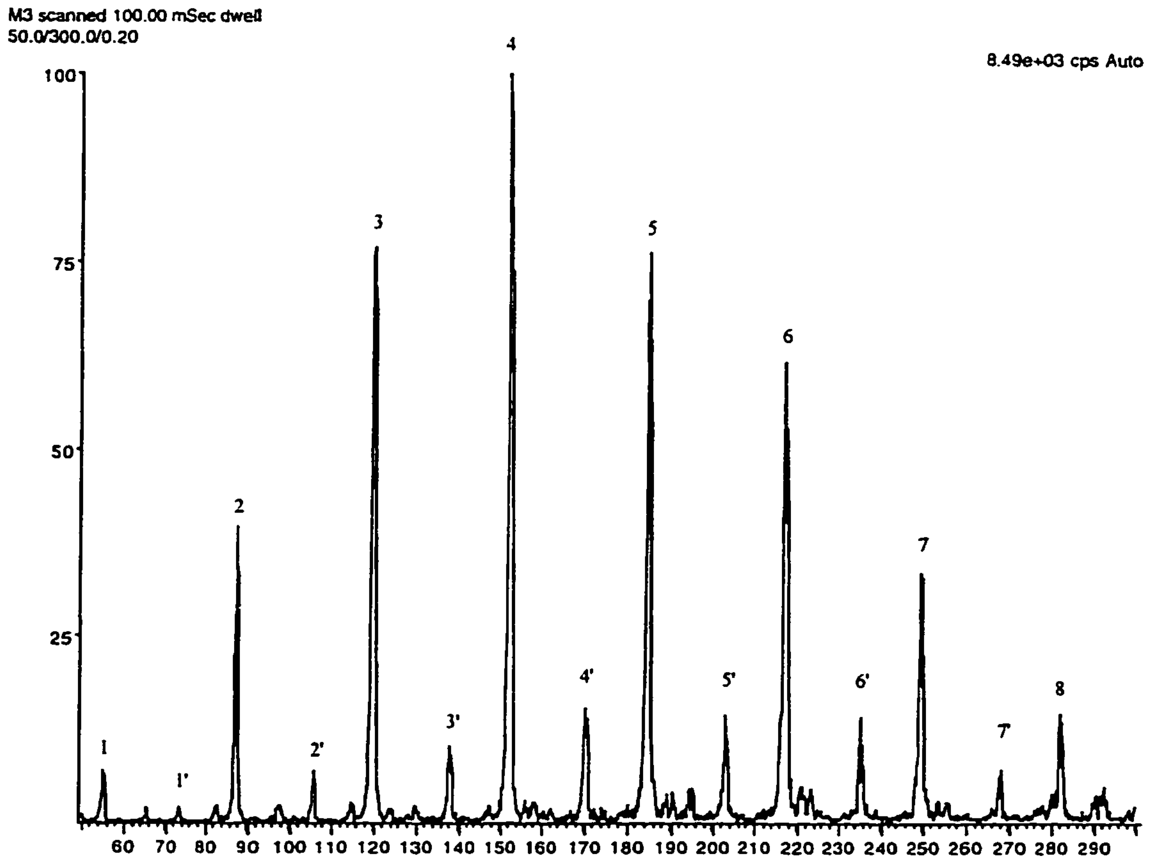

Figure 5.1 Mass spectrum of sodium chloride solution taken prior to instrument modifications. The numbers over each mass peak indicate the number of solvent molecules (methanol) attached to the Na^+ ion. The numbers marked with a ' indicate ions which also contain one attached water molecule.

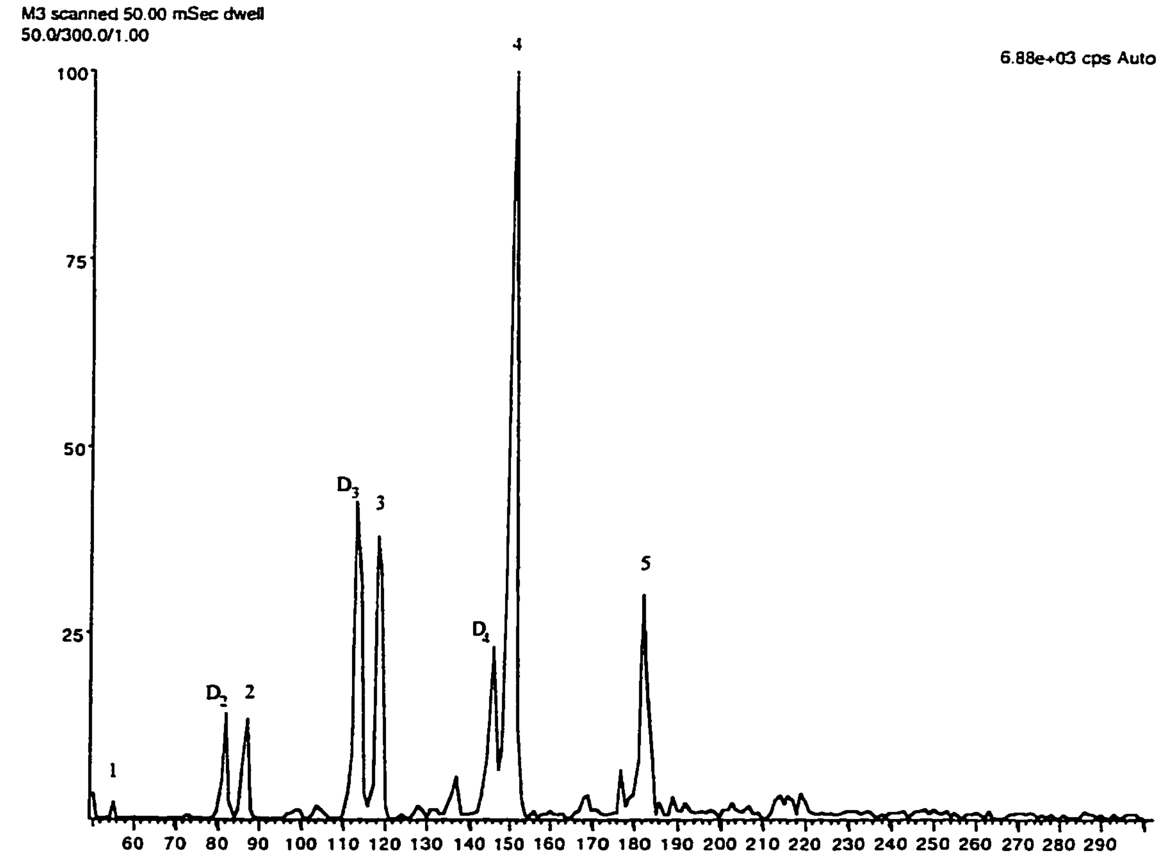

Figure 5.2 Mass spectrum of sodium chloride solution taken after implementation of all the instrument modifications described in the text. The numbers over each mass peak indicate the number of solvent molecules attached to the Na^+ ion. While still not optimal, the degree of solvent clustering was significantly improved by the modifications. The ions with m/z of 82, 114, and 146 amu, labeled D_2, D_3, and D_4 in the figure, are typically found in corona discharges and are $NH_4^+(MeOH)_n$ where n is 2, 3 and 4 respectively.

A slight bend was made in the ion pickup tube so that its exit end was non–concentric with the mass spectrometer inlet. This bend reduced the amount of methanol directed directly at the mass spectrometer inlet and helped reduce the clustering at the expense of a marginally reduced total ion current.

While the clusters were not eliminated, these modifications significantly reduced the signal from Na^+(methanol)$_5$ cluster and eliminated larger sodium–methanol clusters. Figure 5.2 shows the mass spectrum of the 10^{-4} M NaCl solution obtained after all the modifications had been made.

It also became possible to obtain mass spectra containing lysine–phenylalanine ions (LYS–PHE$^+$) with only a single solvent molecule attached as the base peak. Figures 5.3 and 5.4 show the ion introduction chamber used. The stream of dry nitrogen directed across the face of the mass spectrometer inlet orifice was found to reduce the amount of methanol vapor entering the mass spectrometer.

Another modification to the ion pickup tube–mass spectrometer interface was the change of material used on the face of the chamber from brass to Plexiglas. This change allowed different potentials to be applied to the pickup tube, V_1 in Figure 5.3, than that of the rest of the chamber, including the stainless steel nitrogen inlet tube, V_2. Several voltages were tried, however this modification provided only marginal reduction of the clustering problem. The voltages of the various conducting surfaces of the interface were optimized to minimize the extent of solvent clustering. In Figure 5.3, V_1 refers to the potential of the ion transfer line, V_2 refers to the potential of the brass cylinder and the 1/16" stainless steel tubing used to introduce the dry nitrogen gas, and **IN** and **OR** refer to the voltages of the inlet and orifice plates of the mass spectrometer respectively.

The best conditions for minimizing extent of solvent clustering were found to be:

V_1:	300V
V_2:	160V
IN:	140V
OR:	116V
corona:	3kV

These voltage settings were determined with a pressure of ~11 torr in the interface chamber. Had other pressures been used, it is very likely that different optimal voltage settings would have resulted due to changes in the drift velocity of the ions through the chamber atmosphere. More work in this area may provide further improvements in sensitivity and methanol vapor reduction. Doerge and Bajic have described an ingenious device developed at VG–Fisons specifically for this purpose.[1]

Figure 5.4 shows an end view of the interface chamber. The dashed circle indicates the location of the inlet of the mass spectrometer and the solid circle nearby shows the approximate position of the exit end of the ion pickup tube. Not shown is a Plexiglas spacer used as a guide to position the interface at the center of the mass spectrometer's inlet plate. Once in position, the chamber was held tightly against the inlet plate by the vacuum.

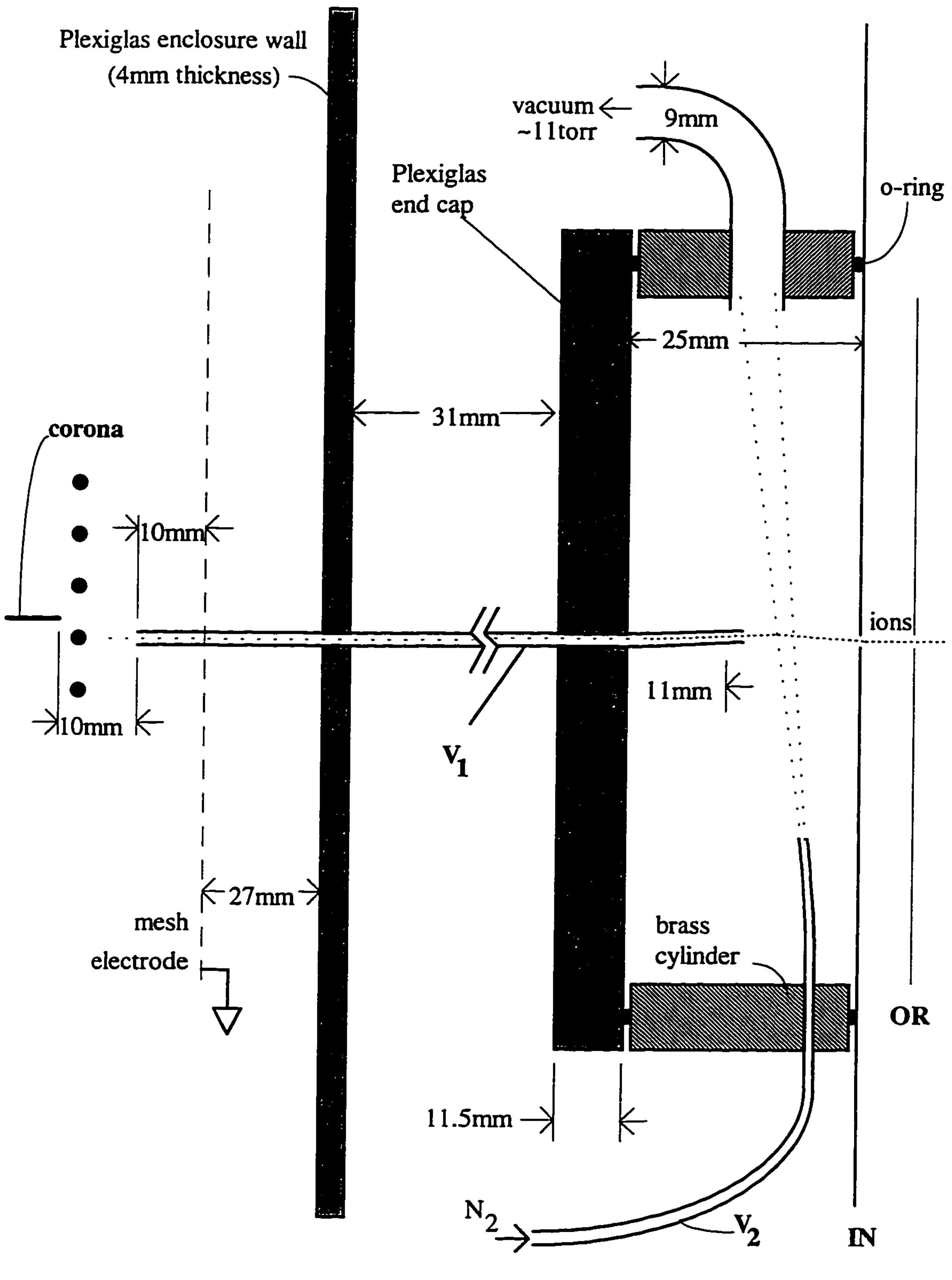

Figure 5.3 Ion introduction interface used (Top View).

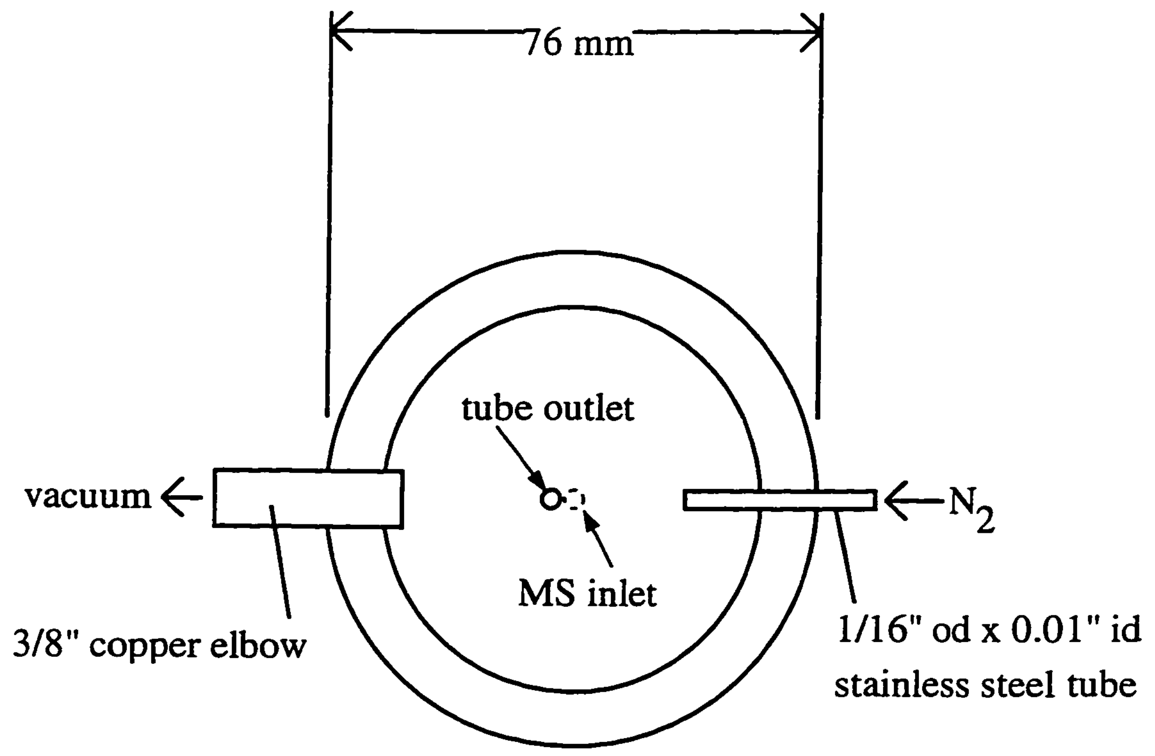

Figure 5.4 Ion introduction interface used (End View). Note the offset between the exit of the ion pickup tube and the inlet to the mass spectrometer. The brass cylinder was sealed to the end of the mass spectrometer and to the Plexiglas end cap using Viton o–rings.

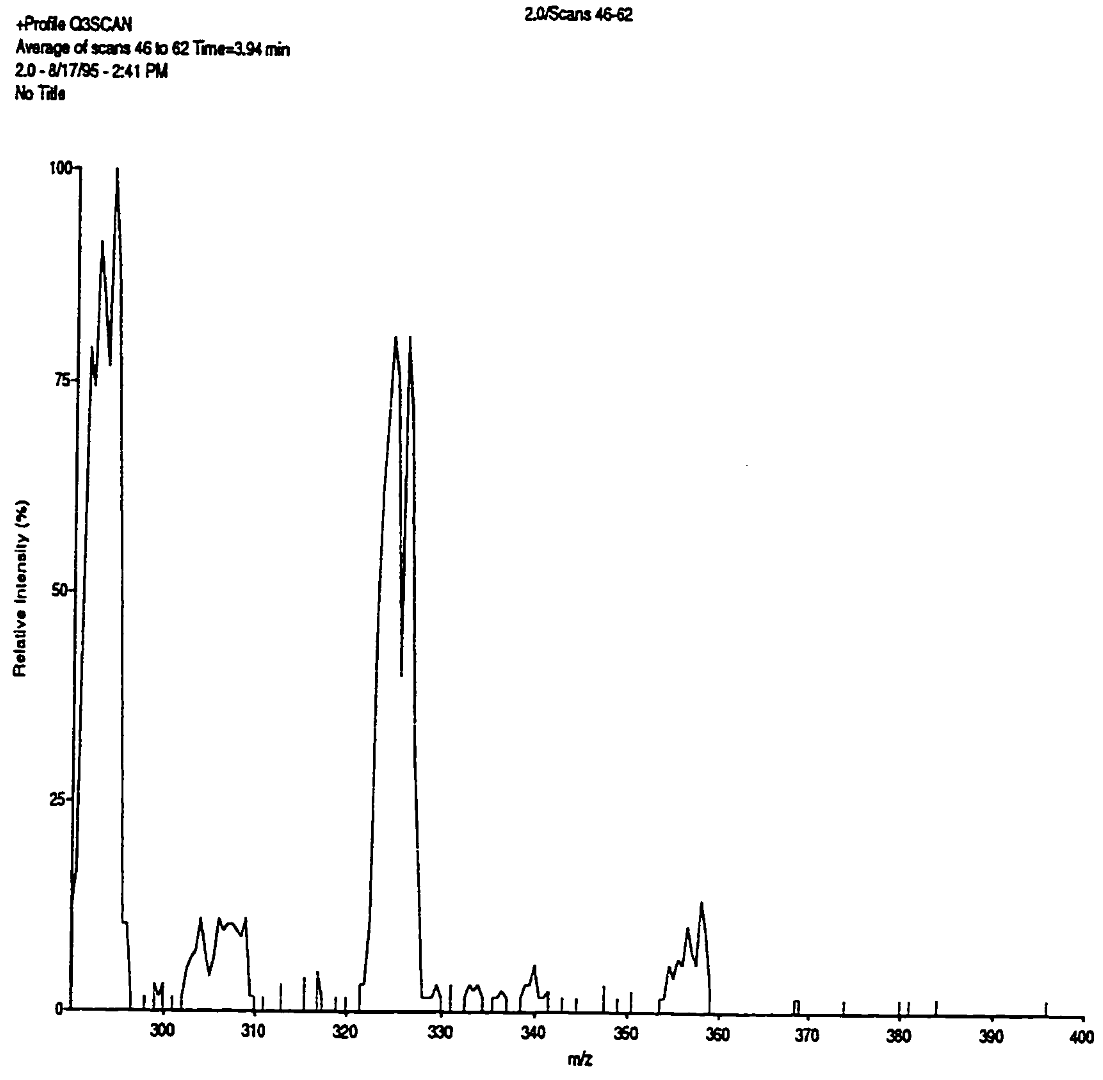

Figure 5.5 Mass Spectrum of 10^{-3} M LYS-PHE. This spectrum was obtained 4 minutes after injection, when the peptide was eluting from the column. LYS–PHE^{+}(MetOH)$_1$ = 325 amu. The split peak shape was due to signal noise.

5.3.2 Quantitation of LYS–PHE$^+$

Figure 5.5 shows the mass spectrum of 10^{-3} M of the dipeptide lysine-phenylalanine (LYS-PHE). The peak detected in the low resolution mass spectrum at m/z 322-327.5 was protonated LYS-PHE with one methanol molecule attached.

The selected ion chromatogram was collected by integrating the low resolution mass spectrum from m/z 322–327.5 amu. Low resolution mode scans provided greater sensitivity and were faster to accumulate than high resolution mode scans. The 5.5 amu mass range provided maximum sensitivity as it included the complete isotope distribution. Larger organic molecules, such as the 15–carbon dipeptide used in these experiments, have a much greater chance of containing one or more C^{13} atoms than do smaller molecules. Therefore, in order to include the signals from LYS–PHE ions with all likely combinations of C^{12} and C^{13} in the selected ion chromatogram, a broader mass range and lower mass resolution was appropriate.

Quantitation was done by using the area under the selected ion chromatographic peak. The ion chromatograms from duplicate injections were averaged. This average is plotted in Figure 5.6.

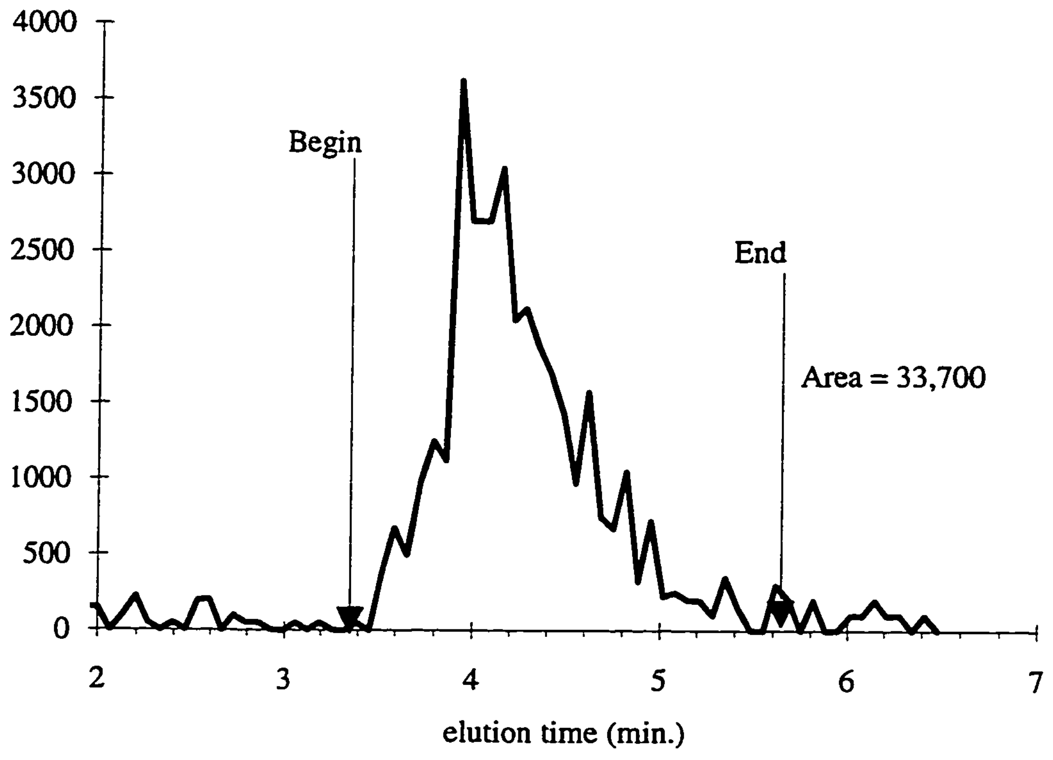

Figure 5.6: Selected Ion Chromatogram of 10^{-3} M LYS-PHE. This chromatogram is a recording of the total ion count in the range of m/z 322 - 327.5 vs. elution time. The area under the peak starting at 3.4 minutes and ending at 5.6 minutes was used for quantitative analysis. Duplicate injections were made, and the average is shown in the figure. The average peak area was 33,700 counts for this concentration.

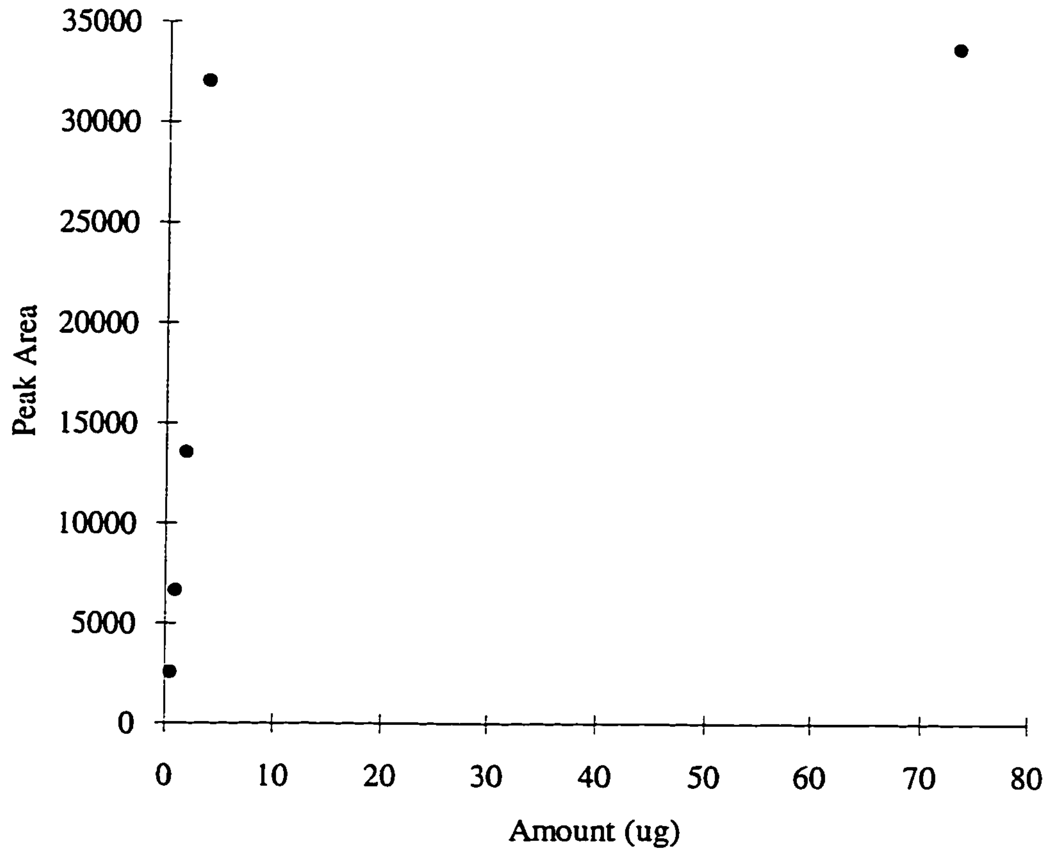

Figure 5.7: Calibration curve for LYS-PHE. Non-linearity is apparent above 4 μg injected.

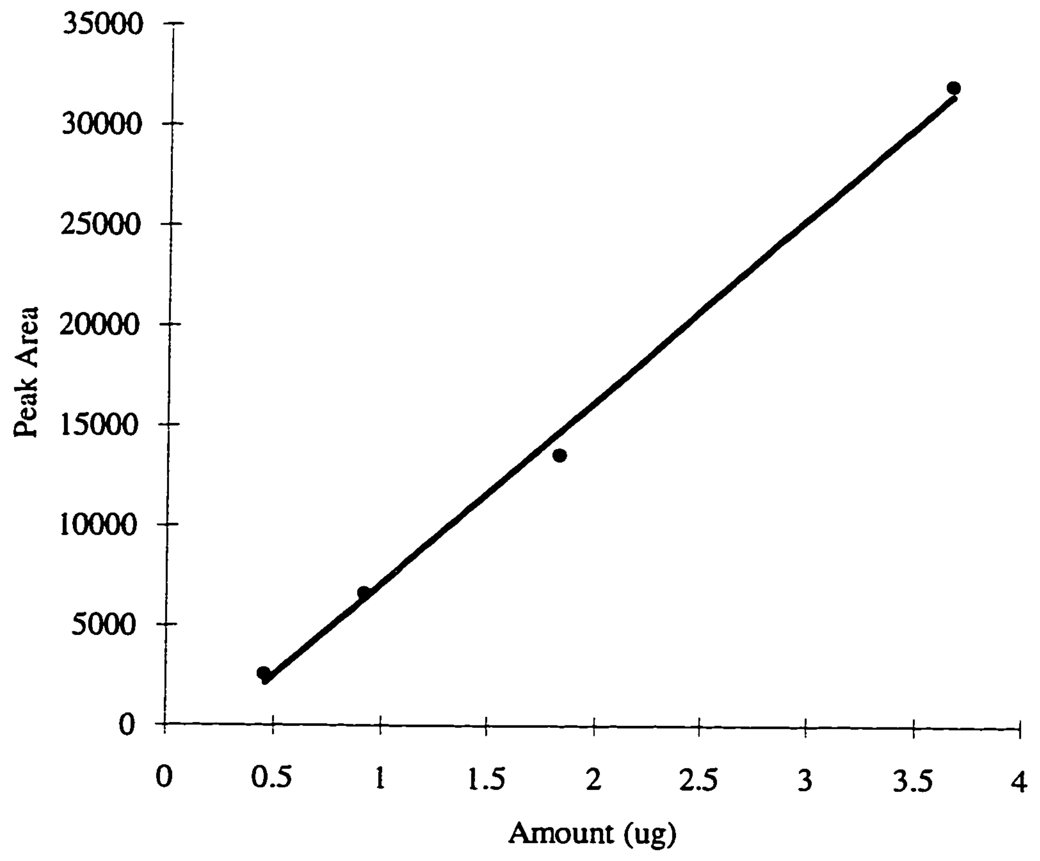

Figure 5.8: Calibration curve for LYS-PHE in the range of linear response. The solid circles represent the experimental data and the solid line indicates the linear regression fit of these data points.

Figures 5.7 and 5.8 show the calibration curve for LYS-PHE. The conditions used were:

Mobile Phase Flow rate: 0.3mL/min.

Droplet production frequency: 56.7kHz

nozzle capillary i.d.: 31μm

charging voltage: 3.0 kV (adjusted +/- a few volts using the fine adjustment to give the most stable DES image)

Figure 5.8 shows the calibration curve in the linear region near the detection limit. The detection limit at the 95% confidence level was that concentration which gave a signal that was larger than the background by twice the standard deviation of the background. The background noise level was determined by integrating the baseline over a length of time equal to that used for an analyte peak near the detection limit. Since the peaks from the 0.5 μg injections were integrated from 2.5 minutes to 4.0 minutes for a total of 1.5 minutes, 1.5 minutes of baseline was integrated in both chromatograms. 0.75 minute sections before and after the sample peak were combined for each injection. Since the signal used was the average of duplicate injections, the background used was the average of two 1.5 minute baseline integrations. The average and standard deviation of the 4 background measurements were then determined and found to be 1920 and 353 counts, respectively.

The signal plus background at the detection limit is therefore 1920 + 2*353, or 2626 counts. The least squares fit of the linear portion of the calibration curve was found to be:

$$A = 9185 * w - 2055 \qquad (1)$$

where A: peak area

w: amount of analyte injected

Solving for w gives:

$$w = \frac{A + 2222}{9185} \qquad (2)$$

The detection limit, in μg, is therefore (2626+2055)/9185 or 0.51 μg.

The quantitative analysis of LYS-PHE using HPLC-DES-MS was not based on the bare, desolvated ion. Figure 5.5 shows a substantial bare ion abundance at m/z 294 along with a trace of doubly solvated ion. Further work in reducing the solvent clustering would improve the detection limit if that reduction could be done while preserving the overall signal level. This improvement would result from an increase in sensitivity arising from having all the analyte ions in the same form, presuming total throughput remained the same. Variability in signal due to changes in the distribution of the various solvent clusters over time would be reduced if all but one form of the analyte ion were eliminated, thus enhancing the precision.

5.3.3 HPLC–DES–MS Analysis of Amines

In another experiment with the HPLC–DES–MS interface, solutions of 2 different amines were injected into the system. A constant concentration of Na^+ was maintained to help maintain a stable spray. Figure 5.9 shows the selected ion chromatogram of three consecutive injections of a solution of heptylamine and methylamine, both 10^{-2} M in methanol. Both the sample and the mobile phase were 10^{-4} M solutions of NaCl in methanol. All three ionic species were monitored throughout the chromatographic separation, the $Na^+(methanol)_6$ ion at m/z 215, the methylamine$^+$(methanol)$_4$ ion at m/z 160, and the heptylamine$^+$(methanol)$_4$ ion at m/z 244.

The observed decrease in sodium ion intensity coincides with the appearance of the positive amine ions in the droplets. This result is consistent with and analogous to those of Agnes and Horlick if one considers the sodium ion present at a constant concentration in the samples and the mobile phase to be an "electrospray stabilizer".[7] As the analyte peaks eluted and concentrations of the ionic amine species in the column effluent became comparable to that of the sodium ion stabilizer, the sodium ion signal was depressed. Compare with Figure 3, reference 7. The chromatographic separation between the two compounds was poor. No attempts were made to optimize the separation as this would have required changes to the mobile phase that would have made DES impossible, at its current level of development.

The DES generator was stable throughout the 20 minute duration of the experiment. Reproducible peak heights and shapes were obtained. It is clear from these very preliminary results that a HPLC–MS interface based on DES is feasible.

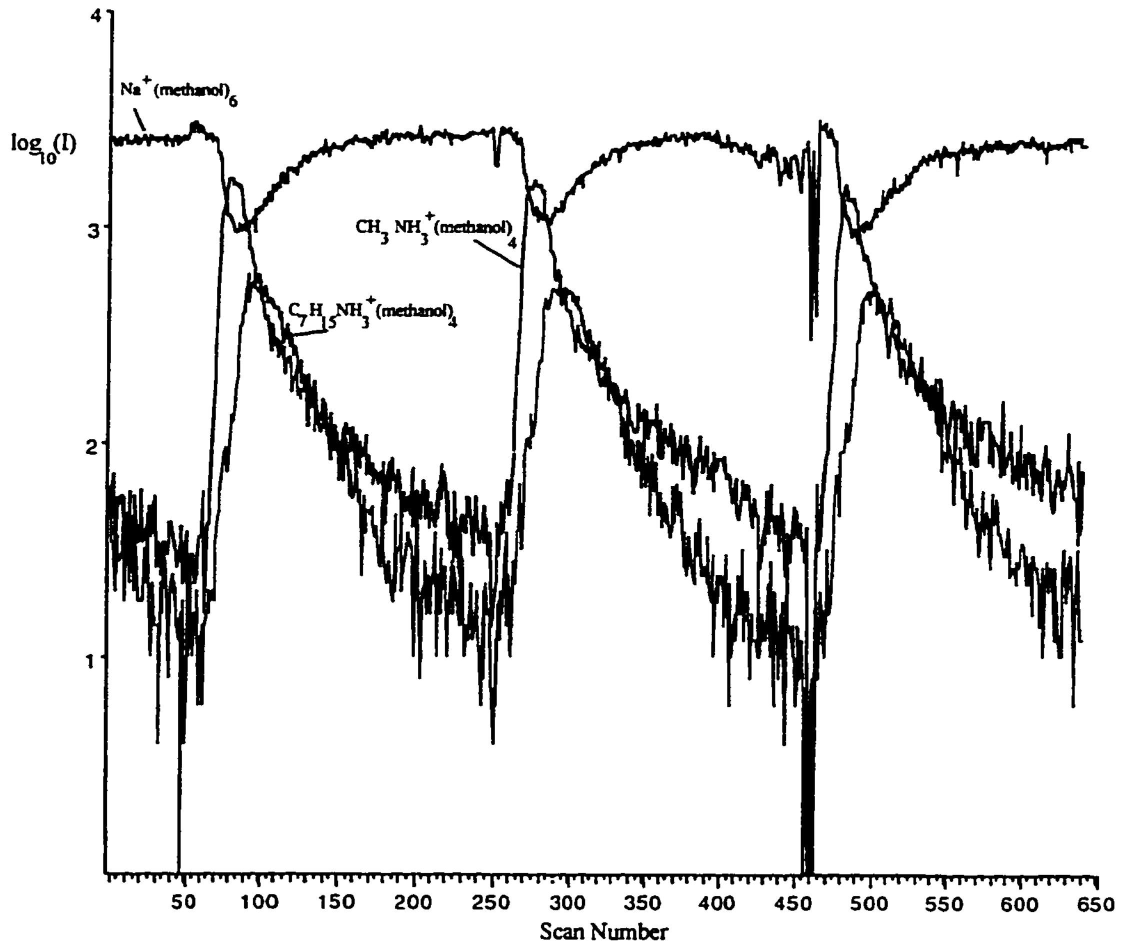

Figure 5.9: Selected Ion Chromatogram of Amines. The figure shows the selected ion chromatogram of three consecutive injections of a solution of heptylamine and methylamine, both 10^{-2} M in methanol. Both the sample and the mobile phase were 10^{-4} M solutions of NaCl in methanol. The sodium ion signal was also monitored during these injections. The three ion abundance signals are plotted on a $\log_{10}$ scale of counts/second.

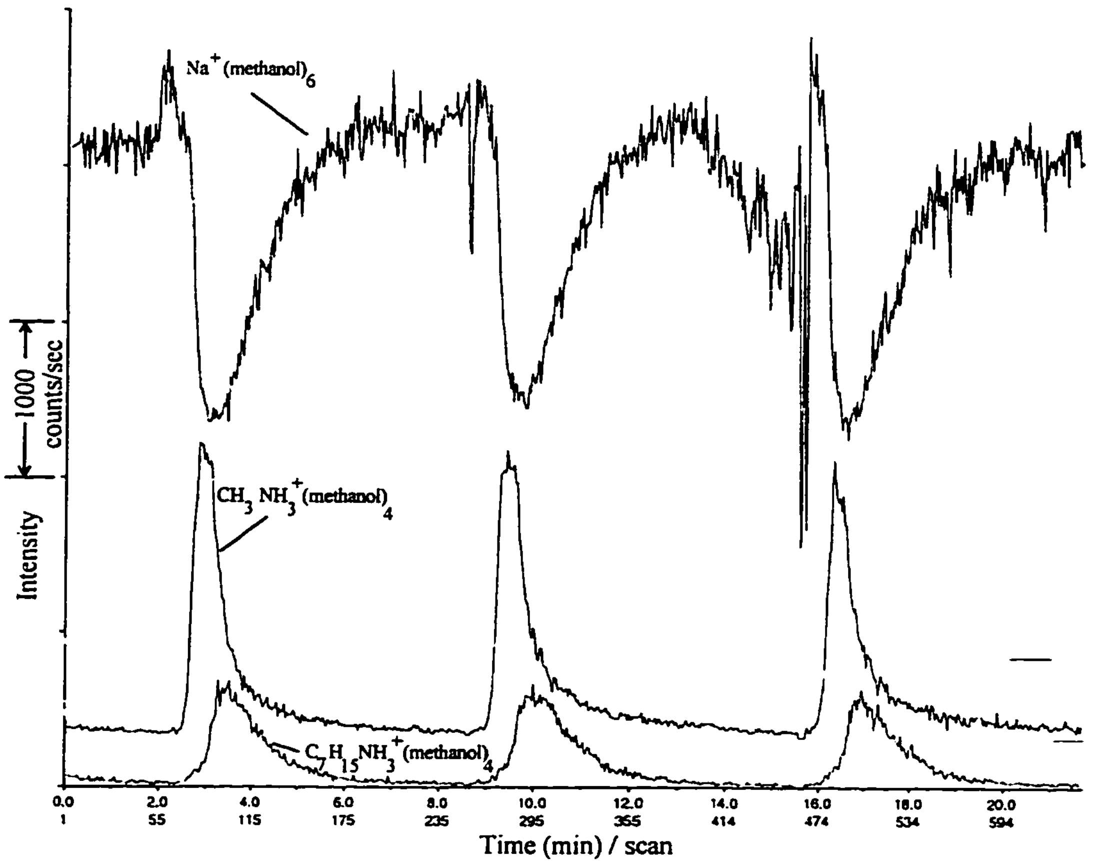

Figure 5.10: Selected Ion Chromatogram of Amines. The three ion signals have been vertically offset for clarity and shown on a linear scale. Data is from the same three injections shown in Figure 5.9. The baseline for the sodium ion chromatogram is at 2530 counts/sec. The scale is 1000 counts/tick for all three signals.

Many techniques are available for coupling HPLC to EI-MS. Where electrospray ionization is the ionization method of choice, several approaches have been reported and, in some manner, all address the problem of accommodating the very low flow rate typically used to supply the electrospray source. In the simplest situation, the chromatographic system is scaled down using open-tubular, or packed microbore columns[2]. Banks *et al.* have developed an ultrasonically assisted electrospray interface that has proven capable of handling mobile phases of high conductivity and high surface tension and having flow rates up to several hundred microliters per minute.[3,4] The DES-based interface can be designed to accommodate much higher flow rates.

In all cases where electrospray or ultrasonically assisted electrospray is employed, the aerosols produced by the electrospray are composed of highly charged droplets whose tendency is to immediately and rapidly disperse. As the aerosol expands, the sample is distributed throughout a relatively large volume. Since it is very likely that ion production does not occur immediately at the electrospray capillary, but requires one or more stages of droplet evaporation and Rayleigh disruption, the rapid expansion of the electrospray aerosol means that ion evaporation will not occur in a small, easily sampled location.[5] Each droplet undergoing fission will eject its daughter droplets in an essentially random direction thus decreasing the ultimate ion density still further. In a recent review article, Niessen discussed the effect of space charge on ion density in an electrospray and how this effect relates to detection sensitivity.[6] He pointed out that the negative impact of space charge on sensitivity is especially important when flow rates are very low and the refreshment rates of the aerosol are reduced. The DES-based ion source, while having several as yet unsolved problems, possesses the unique ability to control the timing and orientation of each droplet disruption such that, potentially, a much larger fraction of the ions produced may be sampled into the mass spectrometer. The resulting improvement in sensitivity would be of great advantage in coupling very low flow techniques such as capillary electrophoresis to a DES-based ion source, provided an appropriately sized DES generator is constructed to accommodate such low flows.

5.4 REFERENCES

1. Doerge, D. R.; Bajic, *S. Rapid Commun. Mass Spectrom.*, **6**, 663. (1992)
2. Henion, J. D.; Wachs, T. W.; Conboy, J. C.; Garcia, F., *J. Chrom. Sci.*, **29**, 357 - 366 (1991)
3. Banks, J. F.; Shen, S.; Whitehouse, C. M.; Fenn, J. B., *Anal. Chem.*, **66**, 406 - 414. (1994)
4. Banks, J. F.; Quinn, J. P.; Whitehouse, C. M., *Anal. Chem.*, **66**, 3688 - 3695. (1994)
5. Wang, G.; Cole, R. B., *Anal. Chem.*, **67**, 2892 - 2900. (1995)
6. Niessen, W. M. A., *J. Chromatogr. Libr.*, **59**, 3 - 70. (1996)
7. Agnes, G. R.; Horlick, G., *Appl. Spectrosc.*, **48**, 649 - 654 (1992)

CHAPTER 6:

SUMMARY, CONCLUSIONS AND FUTURE WORK

6.1 SUMMARY, CONLUSIONS, AND FUTURE DIRECTIONS

The droplet polarization that occurs under the influence of the electric field, combined with the acquisition of charge from its passage through the corona, lead to the highly reproducible production of exploding droplets. By use of regular periodic mechanical breakup of a liquid jet and synchronously pulsed back-lit photography, stroboscopic observation of these explosions became possible. After it became apparent that these tiny exploding droplets gave rise to air-borne ions from solution, the technique was called droplet electrospray.

The fact that each droplet sprays in the same direction implies the electric field is orienting the droplets. This behavior is only possible because the droplets are polarized. A consequence of this polarization is that the droplets must be disrupting at well below the Rayleigh limit of charge. The charge on the droplet surface is concentrated around the point of lowest field strength and locally becomes sufficient to cause the spray to erupt at that point even though the droplet, taken as a whole, possesses insufficient charge to fission.

Another contributing factor in initiating the droplet fission can be distortion of the droplet, either through interaction with the electric field or from residual oscillations arising from the droplet's formation. One may envision the chain of events taking place on a highly charged droplet as it evaporates. The droplet shrinks and the limiting charge it is capable of maintaining decreases, ultimately approaching the Rayleigh limit. When this happens, the droplet becomes highly sensitive to changes in its shape and polarization. Anything that may polarize a droplet in this state, a slight oscillation, passage near another charged droplet, the presence of an external electric field, etc. rapidly gives rise to an uneven fission event. There is good evidence to suggest that this behavior is displayed by droplets of all sizes.

Kebarle has suggested that the emission of a multiply-charged macroion from a sub-micrometer sized droplet could proceed by a mechanism analogous to an uneven fission and may therefore be mediated by oscillations of the droplet.[1] It may be possible to test this hypothesis by exciting droplet deformation through application of ultrasonic energy of a frequency matching that of the oscillation of the smallest droplets.

DES has been shown to produce gas phase ions from ionic species present in the solution comprising the parent droplet. This observation shows that the electrospray process is independent of the means of producing charged droplets. Siu *et al.* have proposed a mechanism of ion production solely from the Taylor cone in the absence of solvent evaporation[2,3]. Their proposal is refuted by elimination of the conventional ESI apparatus, namely the metal capillary, and still observing ion production in the absence of the Taylor cone. Siu *et al.'s* assumption would have to be modified to a claim that gas phase ions are emitted directly whenever charge induced dispersion to much smaller droplets occurs. This type of dispersion would include the Taylor cone, and all subsequent uneven fission of the charged droplets in which a jet of much smaller and more highly charged droplets is emitted. The similarity between the droplet jet emerging from the Taylor cone and the droplet jet emerging from the parent droplet when uneven fission occurs has been pointed out by Gomez and Tang[4] and Willoughby *et al.*[5]. An extension of Siu *et al.'s* assumption to include not only the

Taylor cone but also all these droplet fissions will be in contradiction with the interpretation of the experimental results on which the original Taylor cone argument was based. These experiments were interpreted to indicate that no solvent evaporation prior to ion emission occurs[2] and that the large ion intensity decrease from water to ethylene glycol as solvents is due to ion emission from the Taylor cone alone for both water and ethylene glycol[3].

Methods have been developed that allow the use of the mass spectrometer as a selective, sensitive detector for HPLC. When combined with computer control and data manipulation, rapid quantitative and qualitative analyses of small quantities of complex mixtures can be done with these hybrid instruments. The key to the successful application of HPLC-MS lies with a good interface design. DES has been shown to be capable of handling flow rates in excess of those used in conventional ES while showing comparable sensitivity. This application of DES therefore addresses a central requirement in a successful HPLC-MS interface, namely dealing with the large flow of liquid. In effect, a small proportion of the total liquid flow is sprayed with the rest being left behind in a train of equal sized droplets. Uses may yet be found for these droplets. Rather than just left to evaporate into the laboratory atmosphere, they may be recharged with subsequent electrodes and sprayed again to enhance detection sensitivity or deflected electrostatically and collected in a high-speed fraction collector.

Despite limited success in early DES-based HPLC-MS interfaces, many problems remain unsolved. The signal stability must be improved, especially at low frequencies. The key to this improvement lies in a better apparatus.

Shielding from air currents must be provided. Air currents were found to be the largest source of noise and instability in the DES generator. Miniaturization of the apparatus should make addressing this problem easier.

Ensuring constant droplet speed and trajectory past the charging electrode, while crucial to achieving stability of the DES ion source, is not the only parameter that needs to be addressed. Changes in the conductivity and surface tension of the column effluent will also change the charging and fissioning processes ultimately affecting the production of ions from solution. Achieving a stable, steady stream of exploding droplets requires a delicate balance of mechanical, chemical, and electrical conditions. Therefore, anything causing these solution parameters to change during the chromatogram will need to be compensated for by some combination of changes in voltage, position, and flow rate. Failure to do so will cause drift in the signal or disappearance of the spraying droplets altogether. The examples presented in this work were all done using isocratic elution and steps were taken to closely match the sample solution composition with that of the mobile phase. More often than not, gradient elution is required to achieve the desired chromatographic separation, however. To accommodate solvent gradients, a scheme for adjusting either or both of the charging voltage and the electrode–droplet stream separation needs to be developed.

A device capable of rapid adjustment of the minimum distance between the droplet and the charging electrode should be built. Since the signal was found to be highly sensitive to this distance, the ability to correct for drifts in the droplets' trajectories would provide a means for minimizing this source of noise. An attempt at this was done using a voice-coil actuator driven by a pair of audio amplifiers. When DES was first observed (Chapter 1), measurements of the electric current through the charging electrode indicated that as the droplet stream approached the electrode, the charging current decreased very slightly until the droplet stream made contact with the electrode. This slight dependence of current on droplet position was exploited in an attempt to construct a tracking device designed to maintain a constant distance between the charging electrode and the droplet stream. The electrode was mounted on a voice–coil actuated positioner. A current amplifier was used to control a bank of broad–band, high current audio amplifiers which were in turn used to drive the voice coil. Coarse position adjustment was provided by attaching the voice coil–electrode assembly to a mechanical translation stage. The outputs of the amplifiers were controlled by the current flowing through the corona discharge. An increase in this current caused by an increase in the gap between the discharge electrode and the droplet stream caused the amplifiers to supply more current to the voice coil, moving the electrode closer and restoring the discharge current to its normal operating value. Similarly, as the electrode-droplet stream gap narrowed, the discharge current decreased and the amplifiers drove the voice coil away from the stream again restoring the system to the normal state. This tracking mechanism was designed to be operated in the frequency range of 0-100Hz. Gain and offset controls were used to adjust the circuit so that any deviation in charging current, caused by a change in droplet stream trajectory, was immediately converted into sufficient driving current to move the voice coil to compensate.

The results were disappointing. The amplifiers were unable to sustain the position of the voice coil for longer than a few seconds once it had moved from the resting point. The crude mechanism used was therefore unable to accommodate slow drift in the droplet trajectory. The current/position dependence was not only minuscule but inconsistent. The relationship between discharge current and droplet-to-discharge distance was complicated by effects due to the droplet velocity, electrode voltage, solution ionic strength and other undetermined factors.

Future efforts at designing a tracking mechanism should probably be based on an optical position sensing approach such as that proposed in Figure 6.1. In this device, the droplet stream would be illuminated with light from an LED coupled to an optical fibre. Light scattered from the droplets would be collected by a second optical fibre nearby and detected with a photodiode. The amount of scattered light collected is dependent on the distance this sensor is to the droplet stream. To decrease the interference due to stray light, an optical bandpass filter may be used between the sensing fibre and the photodiode. Since the frequency of the droplets is known, a narrow band electronic filter could be used to further increase the signal to noise ratio of this position sensor. It would be straightforward to construct a voice-coil driver circuit using the output of the optical position sensor as input to the power amplifiers.

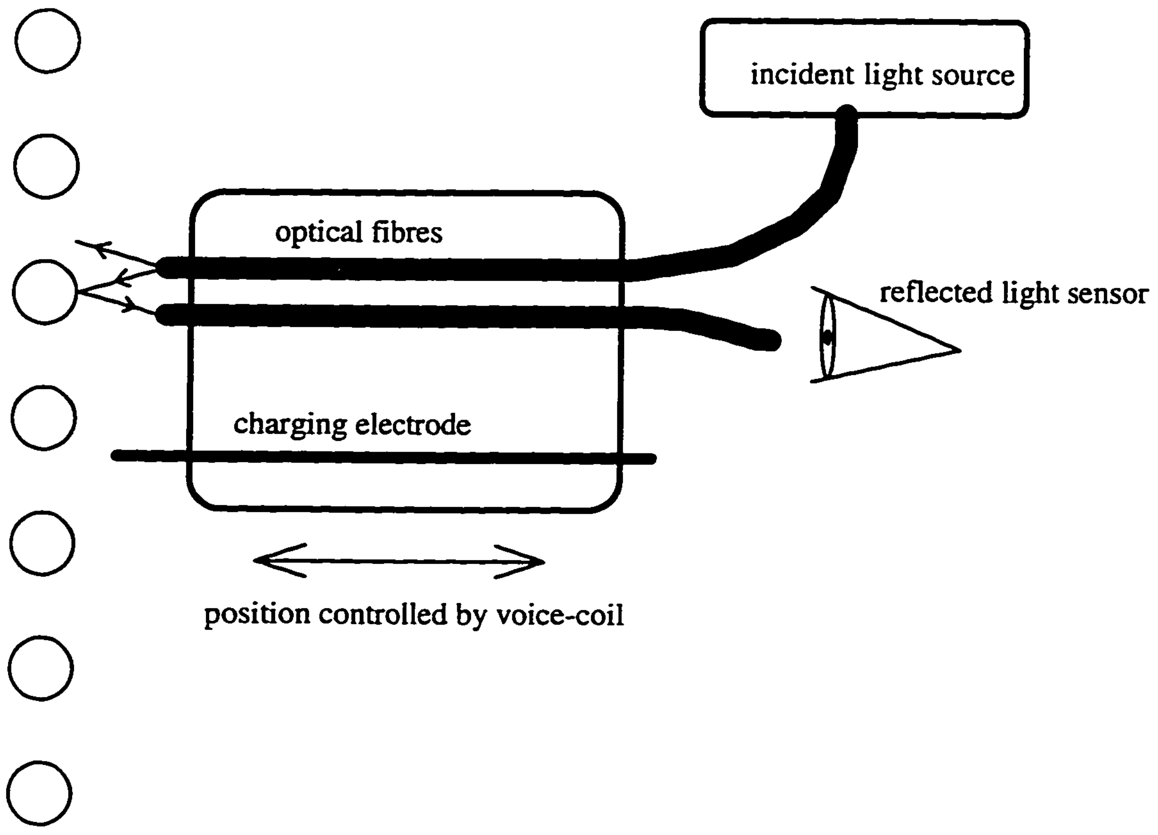

Figure 6.1 Using optical fibre position sensor to adjust the charging electrode position.

Such a scheme would allow for rapid tracking of small deviations of the droplet stream's trajectory and thereby increase the long-term stability of the DES ion source. The non-conducting fibres would not significantly perturb the electrostatic environment near the charging electrode. They would quickly become charged to a steady state value dependent on the charging electrode's voltage and the optimal electrode to droplet distance found through experiment.

Some improvement in DES stability vs. droplet stream trajectory was obtained by operating the high voltage power supply in current regulated mode as opposed to the voltage regulated mode that had previously been used. The droplet's position could be influenced by the electric field near the tip of the charging electrode. In the absence of corona discharge, an increase in voltage caused a corresponding increase in the deflection of the methanol droplets toward the electrode. In the presence of a corona discharge, and especially when the droplets were undergoing DES, increasing the voltage resulted in increased deflection of the droplets *away* from the charging electrode. By operating the high voltage supply in current regulated mode, a slight change in droplet trajectory resulting in a smaller gap between the electrode and the droplet would cause an increase in applied voltage as the supply maintained a constant current to the electrode. The effect of this increased voltage was to push the droplet stream back toward its original location giving some measure of stability enhancement. Being electronic in nature, this enhancement effect could also operate quite rapidly compared to a mechanical tracking device. Future work on droplet tracking schemes may provide the ability to use the DES–based HPLC–MS interface with gradient elution.

Regardless of the means used to improve stability, the ultimate goal should be to achieve sufficient stability to eliminate the need for constant video-microscopic surveillance and manual readjustment.

Banks *et al.* have reported good results in dealing with LC mobile phases that are high in conductivity and high in surface tension with an interface based on ultrasonically assisted electrospray.[6,7] However, their research and that of others described in a recent review article by Niessen has not addressed the problems of electrospraying the high flow regime, i.e. rates > 1 mL/min.[8]

Smith *et al.* have described the value in using electrospray ionization in biochemical mass spectrometry.[9] The ability to analyze single cells has long been sought after.[10] Approaches to date have centered around miniaturized chromatographic or electrophoretic separation techniques with extremely sensitive detectors. Application of the DES–based ion source to streams of cells instead of droplets may provide another means for studying the surface of cells with mass spectrometry.

Wahl *et al.* have described a novel detection scheme for use with a CE–ESI–MS interface.[11] They have found that the ion current measured after expansion of the electrospray into a vacuum provides a means for detecting large ions such as those produced from biomolecules. If this technique could be successfully applied to the DES–based interface, development of an inexpensive biomolecule detector for conventional HPLC could be constructed. The ion current transferred through the ion sampling tube was measured in some preliminary tests prior to attaching the interface to the mass spectrometer. In these tests, which were essentially identical to the experiment described in Chapter 5, section 5.3.3, a Plexiglas plate having a copper plate to which a pico-ammeter was connected replaced the mass spectrometer in Figure 5.3, of Chapter 5. No current change could be measured, however, as the methylamine and heptylamine eluted from the column. Future experiments using large, multiply charged biomolecules may give different results. Replacing the copper plate with an ion multiplier may also prove useful in that experiment. Miniaturization of the DES apparatus should enable it to be used with very low solution flows such as those found in capillary electrophoresis. If so, the DES ion source may show advantages in sensitivity over current CE-ESI-MS techniques. This advantage would come from the inherent control in position and orientation of the exploding droplets that DES provides.

Another problem with using DES as an ion source for mass spectrometry is its inability to generate negative ions. Because electrical breakdown occurs at a much lower voltage when using negative potentials, the electric field strength cannot be made strong enough to produce spraying droplets. As soon as breakdown occurs and a corona is formed, the surface of the corona and not the Pt wire becomes the electrode. The effective size of the charging electrode tip is therefore increased by the size of the corona. Since the voltage at which breakdown begins is much lower with negative potentials, lowering this potential further expands the corona to compensate. Further work in finding conditions for producing negative ions by DES must address this problem.

A possible experiment could use multiple charging electrodes whereby the necessary charge is acquired by the droplets in stages. Such an apparatus would also allow droplets that had sprayed once to be recharged and sprayed again. Mass spectrometry of the ions produced from each fission may then be made.

Another approach could be to employ corona suppressing gas, such as SF_6, to allow higher negative potentials to be used. These gases increase the electric field required to produce a corona discharge by lowering the concentration of free electrons near the electrode tip. Fewer electrons means less chance of initiating an ionization cascade which could lead to the formation of a discharge.

Because droplet deformation lowers the charge requirement for producing a spraying droplet, inducing oscillations in the droplets, possibly with ultrasonic sound waves, would likely assist in coaxing a spray out of otherwise insufficiently charged droplets.

Efforts at addressing the difficulty in spraying aqueous solutions should also be made. This problem is especially important when contemplating using the DES ion source to interface HPLC to a mass spectrometer. Water, in varying concentrations, is a common component in liquid chromatographic mobile phases. Ideally, the interface should function reliably throughout a gradient elution separation. It may be possible to find the right combination of electrode dimensions, voltage, droplet size, velocity, and distance to enable DES with aqueous solutions. An exhaustive search for these conditions was not carried out. As with the difficulty in using negative voltages, a step-wise approach to droplet charging may be appropriate for this problem. Finally pulsing or modulating the charging electrode in synchronization with the droplet production may help. The short pulses of higher voltage may allow for sufficient electric field for Rayleigh charging while the longer periods between pulses of lower voltage could help minimize the amount of droplet deflection.

Further work needs to be done to reduce the formation of solvent clusters entering the mass spectrometer. The clustering problem is linked to the high concentration of methanol vapor in the interface. Besides increasing the size and abundance of ion-solvent clusters, large quantities of methanol place a strain on the mass spectrometer's pumping system. More work in the area of interface design and operation may provide further improvements in sensitivity and methanol vapor reduction.

Finally, the results of the spatial distribution of the ions produced near a spraying droplet indicate that sensitivity is optimized if the ideal ion sampling location is used. In the case where the optimal zone of ion collection is spread out, a more appropriate collection device may be a narrow channel instead of a round capillary.

Further work in this area should determine whether or not this zone of highest abundance occurs in the same location for all ions in droplets containing a mixture of ionic species. This information may be especially useful in the case of fast-moving droplets where the "sweet spot" is spread over an extended region. Differences in the kinetics of ion desorption may appear in this situation. Because the droplets are all traveling at the same uniform velocity, begin spraying at the same point in space and each spraying event proceeds to completion the same way as all the others, the DES apparatus effectively transforms time on a microsecond scale to position on a millimeter scale. Studying these kinetics then becomes a simple matter of scanning the ion pickup tube along the path of ion production and collecting the mass spectra vs. position. Results from measurements such as those could provide insight into how the ions in solution distribute themselves in the transient cone-jet of an exploding droplet.

6.2 REFERENCES

1. Kebarle, P.; Tang, L., *Anal. Chem.*, **65**, 972A - 986A (1993)
2. Siu, K.W.M.; Guevremont, R.; Le Blanc, J.C.Y.; O'Brien, R.T.; Berman, S.S., *Org. Mass Spectrom.*, **28**, 579 - 584. (1993)
3. Guevremont, R.; Le Blanc, J.C.Y.; Siu, K.W.M. *Org. Mass Spectrom.* , **28**, 1345 - 1352. (1993)
4. Gomez, A.; Tang, K., *Physics of Fluids*, **6**, 404 - 414. (1994)
5. (a) Willoughby R.; Sheehan, E.W.; Jarrell, A.; Marecic, T.; Peddler, R.; Penn, S., *Studies of the Physical Processes of Electrospray*; (b) Sheehan, E.W.; Willoughby, R., *Photographic Studies of Electrospray Aerosols.* Presented at the Annual Meeting of the American Society for Mass Spectrometry, San Francisco, CA (1993).
6. Banks, J. F.; Shen, S.; Whitehouse, C. M.; Fenn, J. B., *Anal. Chem.*, **66**, 406 - 414. (1994)
7. Banks, J. F.; Quinn, J. P.; Whitehouse, C. M., *Anal. Chem.*, **66**, 3688 - 3695. (1994)
8. Niessen, W. M. A., *J. Chromatogr. Libr.*, **59**, 3 - 70. (1996)
9. Smith, R. D.; Loo, J. A.; Edmonds, C. G.; Barinaga, C. J.; Udseth, H. R., *Anal. Chem.*, **62**, 882 - 899. (1990)
10. Hofstadler, S. A.; Severs, J. C.; Smith, R. D.; Swanek, F. D.; Ewing, A. G., *Rapid Comm. Mass Spec.*, **10**, 919 - 922. (1996)

11. Wahl, J. H.; Hofstadler, S. A.; Smith, R. D., *Anal. Chem.*, **67**, 462 - 465. (1995)

APPENDIX:

SAMPLE PROGRAM LISTINGS

INTRODUCTION TO SAMPLE PROGRAM LISTINGS

The sample program listings given in this appendix were written in object-oriented Pascal and were designed to run under the Microsoft Windows operating system. The code is compatible with Borland's Turbo Pascal for Windows v1.5 and Delphi compilers. The user interface for each program was designed with the Whitewater Resource Toolkit. Where appropriate, sections of code were broken into separate modules called *UNITS*. Each *UNIT* was compiled separately and contained the code for a logically grouped set of tasks such as those associated with a single piece of instrumentation, for example control of an X-Y positioner or syringe pump. Once compiled and debugged, these code modules were reused by other modules and programs, simplifying development and debugging of new or modified applications.

UNIT EZ488W;

```
{ Support file for IEEE 488 applications.

  Contains:
         ez_out0,
         ez_in0,
         ez_qs0: text files for IEEE communication.
These are primitive logical devices provided by a DOS device driver.
It is through these that we access the BUS.

     PROCEDURE
               IeeeExit: Closes files, shuts down.
}
{$F+}
interface
uses
  Winprocs,Wintypes,Strings,
  MYString,Timer;

VAR

     ez_out0, ez_in0, ez_qs0 : TEXT;
     spoll_byte, qs, error_code : integer;

function IeeeStatus(Sreg:integer):byte;
function QuickStatus:word;
function SRQ:boolean;
function SPOLL(device:integer):byte;

{*******************************************************************}
implementation

VAR
   ExitSave : pointer;    {saves the old exit routine pointer}
   Window : HWnd;
{*******************************************************************}
function IeeeStatus(Sreg:integer):byte;
    { return status register }
var
   StatusByte:byte;
begin
  writeln(ez_out0,'STATUS 7,',Sreg);
  readln(ez_in0,StatusByte);
  IeeeStatus:=StatusByte;
end; {IeeeStatus}
{*******************************************************************}
function QuickStatus:word;
    { Check quick status for error }
var
   qs:word;
  msg:integer;
  error:PChar;
begin
        {$R-}
        readln(ez_qs0, qs);
        {$R+}
        QuickStatus:=qs;
```

```
          if (qs SHR 15)=1 then
               begin
                  writeln(ez_out0, 'GET ERROR');
                  readln(ez_in0, error_code);
                  GetMem(error,12);
                  StrPCopy(error,IToStr(error_code));
                  msg:=MessageBox(Window,error,
                  'EZ 488 interface error code: ',mb_OK);
                   if error<>nil then FreeMem(error,12);
               end; {if}
 end; {QuickStatus}
 {***************************************************************}
 function SRQ:boolean;
 begin
   if (QuickStatus AND 8) = 8 then
   SRQ:=TRUE
   else
   SRQ:=FALSE;
 end;
 {***************************************************************}
 function SPOLL(device:integer):byte;
 var
   SPOLLByte:Byte;
   i:integer;
 begin
     {set timeout to 250 ms}
    writeln(ez_out0,'SET TIMEOUT 7;250');
    writeln(ez_out0,'SPOLL ',IToStr(700+device));
    Pause(100);
    if (quickStatus AND 4096) > 0 then {a timeout occured}
       begin
{         writeln('Device did not respond!'); }
         for i:=1 to 10 do
           begin
             StartSound;
             MessageBeep(0);
           end;
         StopSound;
       end;
    readln(ez_in0,SPOLLByte);
    SPOLL:=SPOLLByte;
{   if device=8 then SPOLLByte:=SPOLL(9); {recurse to ensure device 9
gets polled}
end;{SPOLL}
{***************************************************************}
procedure IeeeExit;
var
  msg:integer;
  Begin
                         { Finish using EZ 488 files}
    ExitProc:=ExitSave;
    writeln(ez_out0, 'CONTROL 7,16;2,13,10'); {Reset to CR,LF & no
EOI terminator}
    Close(ez_out0);
    Close(ez_in0);
    Close(ez_qs0);
{$ifdef debug}
    msg:=MessageBox(Window,'Exit procedure from ez488w has run.',
        'IeeeExit',mb_OK);
{$endif}
{    writeln('EXIT PROCEDURE FROM EZ488W UNIT HAS RUN.');  }
```

```
  End;
{*********************************************************************}

Begin    {Initialization}
    {* Begin communications with EZ 488 *}
        assign(ez_qs0, 'EZQS0');
        reset(ez_qs0);
        assign(ez_out0, 'EZOUT0');
        rewrite(ez_out0);
        assign(ez_in0, 'EZIN0');
        reset(ez_in0);
        writeln(ez_out0, 'RESET 7');
 {       writeln(ez_out0, 'RESUME 7'); }
        ExitSave := ExitProc;
        ExitProc := @IeeeExit;
End.
```

UNIT DATAACQU;

```
 {Provides routines for collecting data from a
  Tek 5223 digital storage oscilloscope.}

 {*****************************************************************}
 interface              {in this case a TEK 5223 storage oscilloscope}
 uses                   {is accessed as device 20 on the IEEE bus}
     windos,winprocs,wintypes,strings,
     ez488w,ldvwglob,DT2827w,MYString;  {My units.}


 const
   ScopeAddr = 'OUTPUT 720;';                {GPIB address of TEK
 5223}
   AcqCompleteLeft = 196;

 var
   spoll_byte, error_code : integer;

 Procedure GetData(var r:IntBuff);
 function GetNewData(var r:IntBuff):boolean;
 procedure SetVectorMode(state:boolean);
 procedure SetSaveMode(state:boolean);
 procedure WaveFormAcquire;

 {*****************************************************************}

 implementation

 VAR
    ExitSave : pointer;    {saves the old exit routine pointer}
    Window : HWnd;
 {*****************************************************************}
 function ScopeStatus:byte;
 var
    garb,i:integer;
    StatusByte:byte;
 begin
    writeln(ez_out0,'SPOLL 720');
    if (quickStatus AND 4096) > 0 then
        begin
 {          writeln('Device did not respond!'); }
          for i:=120 to 480 do
            begin
              garb:=StartSound;
              MessageBeep(0);
            end;
          garb:=StopSound;
        end;
    readln(ez_in0,StatusByte);
    ScopeStatus:=StatusByte;
 end;{ScopeStatus}
 {*****************************************************************}
 procedure ResetScope;
 var
   TekStatus:byte;
 begin
   writeln(ez_out0,'RESET 7');
   TekStatus:=ScopeStatus;
   if TekStatus<>65 then
     begin
       writeln(ez_out0,'CLEAR 7');
       TekStatus:=ScopeStatus;
     end;

     {set timeout to 250 ms}
   writeln(ez_out0,'SET TIMEOUT 7;250');

     {ask for the waveform in binary format}
   writeln(ez_out0,ScopeAddr,'BINARY;ACCESS L1');
   WRITELN(ez_out0,ScopeAddr,'sel disp.l');
 end; {ResetScope}
 {*****************************************************************}
 function WaveFormAcquired:boolean;
 var
    StatusByte:Byte;
    i:integer;
 begin
    i:=0;
    repeat
      StatusByte:=IeeeStatus(1);
      StatusByte:=StatusByte AND 8;
      inc(i);
    until (StatusByte=8) OR (i=10);
    if ScopeStatus=AcqCompleteLeft then
    WaveFormAcquired:=TRUE
    else
    WaveFormAcquired:=FALSE;
 end;{WaveFormAcquired}
 {*****************************************************************}
 procedure SetVectorMode(state:boolean);
 begin
  case state of
  ON : WRITELN(ez_out0,ScopeAddr,'select vector');
  OFF: WRITELN(ez_out0,ScopeAddr,'rel vector');
  end;{case}
 end;
 {*****************************************************************}
 procedure SetSaveMode(state:boolean);
 begin
  case state of
  ON : WRITELN(ez_out0,ScopeAddr,'select save.l');
  OFF: WRITELN(ez_out0,ScopeAddr,'rel save.l');
  end;{case}
 end;
 {*****************************************************************}
 procedure WaveFormAcquire;
 begin
    writeln(ez_out0,ScopeAddr,'ACQ L'); {start an acquisition}
 end;
 {*****************************************************************}
 procedure buffconvert(var outbuff:intbuff);
 var
     j:integer;
 begin
     for j:=0 to 1015 do outbuff[j]:=0;

     for j:=0 to 1015 do
     outbuff[j]:=(ord(PScopeData^[2*j]) shl 8)
```

```
                 + (ord(PScopeData^[2*j+1]) shr 6);
end; {buffconvert}
{*****************************************************************}
Procedure GetData(var r:IntBuff);
begin
   writeln(ez_out0,ScopeAddr,'C?');    {ask for the waveform}
   {get the preamble}
   WRITELN(ez_out0,'ENTER 720 ADDRESS '
               ,TekBuffSeg,':',TekBuffOfs
               ,' COUNT 9');
   {fill the buffer}
   Writeln(ez_out0,'ENTER 720 DMA ADDRESS '
               ,TekBuffSeg,':',TekBuffOfs
               ,' COUNT 2032');
   BuffConvert(r);
end; {GetData}
{*****************************************************************}
Function GetNewData(var r:IntBuff):boolean;
begin
   i:=0;
   repeat
    inc(i);
   until WaveFormAcquired or (i=10);
   if i<10 then
     begin
       StrPCopy(CurrentText,RToStr(DTReadCurrent));
       writeln(ez_out0,ScopeAddr,'C?');    {ask for the waveform}

       {get the preamble}
       WRITELN(ez_out0,'ENTER 720 ADDRESS '
                   ,TekBuffSeg,':',TekBuffOfs
                   ,' COUNT 9');

       {fill the buffer}
       Writeln(ez_out0,'ENTER 720 DMA ADDRESS '
                   ,TekBuffSeg,':',TekBuffOfs
                   ,' COUNT 2032');
       GetNeWData:=TRUE;
       BuffConvert(r);
     end
   else
     begin
       GetNewData:=FALSE;
       ResetScope;
     end;
end; {GetNewData}
{*****************************************************************}
{$F+}
procedure ExitTek;
var
  msg:integer;
  Begin
        { relinquish control of the Scope }
    ExitProc:=ExitSave;
    writeln(ez_out0,ScopeAddr,'rel save.1 vec');
    writeln(ez_out0,'CLEAR 7');
    writeln(ez_out0,'LOCAL 720');
{$ifdef debug}
    msg:=MessageBox(Window,'Exit procedure from DataAcqu has run.',
       'ExitTek',mb_OK);
```

```
{$endif}
{    writeln('EXIT PROCEDURE FROM DataAcqu UNIT HAS RUN.'); }
  End;
{$F-}
{*******************************************************************}
begin {This unit's initialization code. Resets the scope then inserts
        its exit routine into the chain.}
ResetScope;
ExitSave := ExitProc;
ExitProc := @ExitTek;
End.
```

UNIT DATAACQ5;

```
{Provides routines for collecting data from a
 Philips PM3375 digital storage oscilloscope.}

{*****************************************************************}
{$F+}
interface             {in this case a Philips PM3375 storage
oscilloscope}
uses                  {is accessed as device 8 on the IEEE bus}
    windos,winprocs,wintypes,strings,
    ez488w,ldvwglb5,timer,
    DT2827w5,MYString,
    TPWfft;


const
  ScopeDelay = 30;
  ScopeAddr = 'OUTPUT 708;';                {GPIB address of PM3375}
  AcqCompleteLeft = 4;
type
  SixCharFloat = array[0..6] of char;
  time_base_setting = array[0..25] of SixCharFloat;  {0..25}
  vert_sens_setting = array[0..11] of SixCharFloat;
const
  HorTimeBase:time_base_setting = (  {in Seconds}
                                                        '.2E-06',   {0}
                                                       '.5E-06',
                                                       '1E-06',
                                                       '2E-06',
                                                       '5E-06',
                                                       '10E-06',
                                                       '20E-06',
                                                       '50E-06',
                                                       '.1E-03',
                                                       '.2E-03',
                                                       '.5E-03',
                                                       '1E-03',
                                                       '2E-03',    {12 <-
slower than this means 4096 pts}
                                                       '5E-03',
                                                       '10E-03',
                                                       '20E-03',
                                                       '50E-03',
                                                       '.1E+00',
                                                       '.2E+00',
                                                       '.5E+00',
                                                       '1E+00',
                                                       '2E+00',
                                                       '5E+00',
                                                       '10E+00',
                                                       '20E+00',
                                                       '50E+00');  {25}
 VerSens:vert_sens_setting = (  {in volts/div}

     '2E-03',
                                   '5E-03',
                                   '10E-03',
                                   '20E-03',
                                      '50E-03',
                                      '.1E+00',
                                      '.2E+00',
                                      '.5E+00',
                                      '1E+00',
                                      '2E+00',
                                      '5E+00',
                                      '10E+00');

var
  spoll_byte, error_code,
  HTBIndex,VertSensIndex : integer;

procedure FlushScopeBuff;
procedure ResetScope;
Procedure GetData(var r:IntBuff);
procedure SetScopeCoupling;
procedure SetVectorMode(state:boolean);
procedure SetSaveMode(state:boolean);
procedure WaveFormAcquire;
function GetHTBIndex:integer;
function GetVertSensIndex:integer;
procedure SetHTB(index:integer);
procedure SetVertSens(index:integer);

{*******************************************************************}

implementation
VAR
   ExitSave : pointer;    {saves the old exit routine pointer}
   Window : HWnd;
{*******************************************************************}
function ScopeStatus:byte;
begin
   ScopeStatus:=SPOLL(8)
end;{ScopeStatus}
{*******************************************************************}
procedure FlushScopeBuff;
{Fill the binary data buffer with whatever (if anything) remains that
the scope
 wants to send us.}
var
  garb:string;
begin
  Writeln(ez_out0,'ENTER 708');
  Readln(ez_in0,garb);
end;
{*******************************************************************}
procedure ResetScope;
var
  TekStatus:byte;
begin
  writeln(ez_out0,'RESET 7');
  Pause(ScopeDelay);
    {set timeout to 250 ms}
  writeln(ez_out0,'SET TIMEOUT 7;250');
  TekStatus:=ScopeStatus;
  if TekStatus<>0 then
    begin
      writeln(ez_out0,'CLEAR 708');
      Pause(ScopeDelay);
      writeln(ez_out0,'RESUME 7');
      TekStatus:=ScopeStatus;
    end;

    {allow local vertical position control on the scope}
  WRITELN(ez_out0,ScopeAddr,'FRO 0,VER A,POS LOCAL');
    {allow local trigger level control on the scope}
  WRITELN(ez_out0,ScopeAddr,'FRO 0,HOR MTB,LEV LOCAL');
  Pause(ScopeDelay);
  WRITELN(ez_out0,ScopeAddr,'FRO 0,VER A,CPL DC');    {DC-coupled by
default}
  Pause(ScopeDelay);
{  FlushScopeBuff;}
end; {ResetScope}
{*****************************************************************}
procedure SetScopeCoupling;
begin
    if ScopeRecord.local=TRUE then
      writeln(ez_out0,'LOCAL 708')
    else
    begin
      with ScopeRecord do
      begin
        if DC then WRITELN(ez_out0,ScopeAddr,'FRO 0,VER A,CPL DC')
        else
        if AC then WRITELN(ez_out0,ScopeAddr,'FRO 0,VER A,CPL AC')
        else
        WRITELN(ez_out0,ScopeAddr,'FRO 0,VER A,CPL ZERO');
      end;
      Pause(ScopeDelay);
      WRITELN(ez_out0,ScopeAddr,'FRO 0,VER A,POS LOCAL');
      WRITELN(ez_out0,ScopeAddr,'FRO 0,HOR MTB,LEV LOCAL');
    end;
end;{SetScopeCoupling}
{*****************************************************************}
function WaveFormAcquired:boolean;
var
   StatusByte:Byte;
   i:integer;
begin
   i:=0;
   repeat
     StatusByte:=IeeeStatus(1);
     StatusByte:=StatusByte AND 8;
     inc(i);
   until (StatusByte=8) OR (i=10);
   if ScopeStatus=AcqCompleteLeft then
   WaveFormAcquired:=TRUE
   else
   WaveFormAcquired:=FALSE;
end;{WaveFormAcquired}
{*****************************************************************}
procedure SetVectorMode(state:boolean);
begin
 case state of
 ON : WRITELN(ez_out0,ScopeAddr,'FRO 0,MSC AUX,DOT OFF');
 OFF: WRITELN(ez_out0,ScopeAddr,'FRO 0,MSC AUX,DOT ON');
 end;{case}
end;
{*****************************************************************}
procedure SetSaveMode(state:boolean);
```

```
 begin
  case state of
  ON : WRITELN(ez_out0,ScopeAddr,'FRO 0,HOR MTB,TRG SNG');
  OFF: WRITELN(ez_out0,ScopeAddr,'FRO 0,HOR MTB,TRG TRI');
  end;{case}
 end;
{*****************************************************************}
procedure WaveFormAcquire;
begin
   WRITELN(ez_out0,ScopeAddr,'FRO 0,HOR MTB,TRG SNG');
end;
{*****************************************************************}
procedure buffconvert(var outbuff:intbuff;var
Routbuff,Ioutbuff:PRData_array);
var
    j:integer;
begin
  if BigBuff then
  begin
    for j:=0 to 4096 do
    begin
       outbuff[j]:=0;  {array of integers}
       Routbuff^[j+1]:=0;  {array of float}
       Ioutbuff^[j+1]:=0.0;
    end;

    for j:=0 to 4082 do
    begin
       outbuff[j]:=PScopeData^[j+4];  {array of bytes}
       Routbuff^[j+1]:=outbuff[j];
    end;
  end {IF BIGBUFF}
  else       {little buff}
  begin
    for j:=0 to 1024 do
    begin
       outbuff[j]:=0;  {array of integers}
       Routbuff^[j+1]:=0.0;
       Ioutbuff^[j+1]:=0.0;
    end;

    for j:=0 to 1009 do
    begin
       outbuff[j]:=PScopeData^[j+4];  {array of bytes}
       Routbuff^[j+1]:=outbuff[j]-220;   {220 is an arbitrary offset}
    end;

  end {ELSE}

end; {buffconvert}
{*****************************************************************}
Procedure GetData(var r:IntBuff);
begin
   Pause(60);
   writeln(ez_out0,ScopeAddr,
   'REG 0,MSC TRACE,DATA_TYPE BINARY,DAT ?');   {ask for the
waveform}
   {fill the buffer}
   Writeln(ez_out0,'ENTER 708 DMA ADDRESS '
               ,TekBuffSeg,':',TekBuffOfs
               ,' COUNT ',15);
    Writeln(ez_out0,'ENTER 708 DMA ADDRESS '
                ,TekBuffSeg,':',TekBuffOfs
                ,' COUNT ',IntBuffSize+500);

    BuffConvert(r,PRealData,PImagData);
    writeln(ez_out0,'CLEAR 708');
end; {GetData}
{******************************************************************}
function GetHTBIndex:integer;
var
  value,response:PChar;
  strresponse:string;
  index:integer;
begin
  GetMem(response,16);
  GetMem(value,12);
{retrieve the HTB string from the scope}
  writeln(ez_out0,ScopeAddr,'FRO 0,HOR MTB,TIM ?');
  writeln(ez_out0,'ENTER 708');
  readln(ez_in0,strresponse);
  StrPCopy(response,strresponse);
  StrCopy(value,response+4);
  index:=0;
  StrCopy(response,HorTimeBase[index]);
  while (StrComp(response,Value)<>-10)
     and (index<25) do
     begin
       inc(index);
       StrCopy(response,HorTimeBase[index]);
     end;
  GetHTBIndex:=index;
  if index>12 then
    begin
      BigBuff:=TRUE;
      IntBuffSize:=4090;
    end
  else
    begin
      BigBuff:=FALSE;
      IntBuffSize:=1010;
    end;
  if response<>nil then FreeMem(response,16);
  if value<>nil then FreeMem(value,12);
end;{GetHTBIndex}
{******************************************************************}
function GetVertSensIndex:integer;
var
  value,response:PChar;
  strresponse:string;
  index:integer;
begin
  GetMem(response,16);
  GetMem(value,12);
{retrieve the V/div string from the scope}
  writeln(ez_out0,ScopeAddr,'FRO 0,VER A,ATT ?');
  writeln(ez_out0,'ENTER 708');
  readln(ez_in0,strresponse);
  StrPCopy(response,strresponse);
  StrCopy(value,response+4);
  index:=0;
  StrCopy(response,VerSens[index]);
```

```
   while (StrComp(response,value)<>-10)
      and (index<11) do
      begin
        inc(index);
        StrCopy(response,VerSens[index]);
      end;
   GetVertSensIndex:=index;
   if response<>nil then FreeMem(response,16);
   if value<>nil then FreeMem(value,12);
end;{GetVertSensIndex}
{*******************************************************************}
procedure SetHTB(index:integer);
begin
  writeln(ez_out0,ScopeAddr,'FRO 0,HOR MTB,TIM
',HorTimeBase[HTBIndex]);
end;{SetHTB}
{*******************************************************************}
procedure SetVertSens(index:integer);
begin
  writeln(ez_out0,ScopeAddr,'FRO 0,VER A,ATT
',VerSens[VertSensIndex]);
end;{SetVertSens}
{*******************************************************************}
procedure ExitTek;
var
  msg:integer;
  Begin
          { relinquish control of the Scope }
    ExitProc:=ExitSave;
    writeln(ez_out0,'CLEAR 708');
    writeln(ez_out0,'LOCAL 708');
    writeln(ez_out0,'RESUME 7');
{$ifdef debug}
    msg:=MessageBox(Window,'Exit procedure from DataAcq5 has run.',
       'ExitTek',mb_OK);
{$endif}
{    writeln('EXIT PROCEDURE FROM DataAcqu UNIT HAS RUN.'); }
  End;
{*******************************************************************}

begin
ResetScope;
HTBIndex:=GetHTBIndex;
VertSensIndex:=GetVertSensIndex;
ExitSave := ExitProc;
ExitProc := @ExitTek;
End.
```

UNIT HP54542A;

```
{Provides routines for collecting data from a
 Hewlett Packard 54542A digital storage oscilloscope.
 Copied and modified from a similar unit written for
 a Philips PM3375}

{******************************************************************}
interface
uses                  {accessed as device 7 on the IEEE bus}
    winprocs,wintypes,strings,
    Win31,
    ez488w,timer,
    MYString;


const
   MaxTekBuffSize = 16384; {Words}
   WaveformSize:integer = MaxTekBuffSize-1;
   HaveData:boolean = FALSE;
   CrLf = #13#10;
   TAB  = #9;
   CR = #13;
   ON = TRUE;
   OFF= FALSE;
   Bit0  = $1     ; { lsb          }
      Bit1  = $2  ; { Bit 1 value   }
      Bit2  = $4  ; { Bit 2 value   }
      Bit3  = $8  ; { Bit 3 value   }
      Bit4  = $10 ; { Bit 4 value   }
      Bit5  = $20 ; { Bit 5 value   }
      Bit6  = $40 ; { Bit 6 value   }
      Bit7  = $80 ; { Bit 7 value   }
      Bit8  = $100      ; { Bit 8 value   }
      Bit9  = $200      ; { Bit 9 value   }
      Bit10 = $400      ; { Bit 10 value  }
      Bit11 = $800      ; { Bit 11 value  }
      Bit12 = $1000     ; { Bit 12 value  }
      Bit13 = $2000     ; { Bit 13 value  }
      Bit14 = $4000     ; { Bit 14 value  }
      Bit15 = $8000     ; { msb           }


type
  RData_array=array[0..MaxTekBuffSize-1] of BYTE;
  PRData_array=^RData_array;

  intbuff = array [0..MaxTekBuffSize-1] of word;
  Str255  = string[255];
  Text15 = array[0..15] of char;
  time_base_setting = array[1..34] of double;

  ScrollBarTransferRec = record
    LowValue: Integer;
    HighValue: Integer;
    Position: Integer;
    end;

  ScopeXferRecord = record
```

```
    DC,AC,DCfifty,Local:bool;{3 radio buttons + checkbox}
    Vert,Horz:ScrollBarTransferRec;
    end;


const
  HTBsetting:time_base_setting = (  {in Seconds}
      5E-09 ,
      1E-08 ,
      2E-08 ,
      5E-08 ,
      1E-07 ,
      2E-07 ,
      5E-07 ,
      0.000001     ,
      0.000002     ,
      0.000005     ,
      0.000005     ,
      0.00001      ,
      0.00002      ,
      0.00005      ,
      0.0001       ,
      0.0002       ,
      0.0005       ,
      0.001 ,
      0.002 ,
      0.005 ,
      0.005 ,
      0.01  ,
      0.02  ,
      0.05  ,
      0.1   ,
      0.2   ,
      0.5   ,
      1     ,
      2     ,
      5     ,
      5     ,
      10    ,
      20    ,
      50);

  ScopeRecord:ScopeXferRecord = (
    DC:False;
    AC:False;
    DCfifty:True;
    local:False;
    Vert:(LowValue:8;
          HighValue:400;  {8mV - 40V total Range}
          Position:10);
    Horz:(LowValue:1;  {5ns, min total Range}
          HighValue:34;  {50s, max total Range}
          Position:1));


const
  ScopeDelay = 10; {ms}
  ScopeAddr = 'OUTPUT 707;';                {GPIB address of 54542A}
  AcqCompleteLeft = 4;
```

```
type
  SixCharFloat = array[0..6] of char;
VAR
  VerRange,HorTimeRange:DOUBLE;
  spoll_byte, error_code,
  i,msg,HTBIndex,VertSensIndex : integer;
  PScopeData:PRData_array;
  TekBuffSeg,
  TekBuffOfs,            {Real mode address elements}
  TekBuffSelector :word; {Protected mode address}

procedure FlushScopeBuff;
procedure ResetScope;
procedure SetScopeCoupling;
procedure SetVectorMode(state:boolean);
procedure SetSaveMode(state:boolean);
procedure WaveFormAcquire;
function GetHTBRange:double;
function GetVertRange:double;
procedure SetHTB(Range:double);
procedure SetVertSens(Range:double);
procedure Digitize;
procedure SetupWaveform;
Procedure GetData(var r:RData_array);
procedure SCopeRun;
procedure SCopeStop;

{*****************************************************************}

implementation
VAR
   ExitSave : pointer;    {saves the old exit routine pointer}
   Window : HWnd;
{*****************************************************************}
function ScopeStatus:byte;
begin
   ScopeStatus:=SPOLL(7)
end;{ScopeStatus}
{*****************************************************************}
procedure FlushScopeBuff;
{Fill the binary data buffer with whatever (if anything) remains that
the scope
 wants to send us.  Reads until it either times out or there is no
more messages
 in the scope's output queue.}
var
  garb:string;
  qsbyte:word;
begin
  Writeln(ez_out0,'ENTER 707');
  while ((SPOLL(7) and Bit4)=Bit4)      {MAV bit means a response in
the output queue}
        and
        ((QuickStatus and Bit12)<>Bit12)   {bit 12 = Timeout}
  do {MAV (message available) bit is set in the hp status byte}
  Readln(ez_in0,garb);
end;
{*****************************************************************}
procedure ResetScope;
var
  TekStatus:byte;
 begin
   writeln(ez_out0,'RESET 7');
   Pause(ScopeDelay);
     {set timeout to 250 ms}
   writeln(ez_out0,'SET TIMEOUT 7;250');
   Pause(ScopeDelay);
   writeln(ez_out0,ScopeAddr,'*CLS');
   TekStatus:=ScopeStatus;
   if TekStatus<>0 then
     begin
       writeln(ez_out0,'CLEAR 707');
       Pause(ScopeDelay);
       writeln(ez_out0,'RESUME 7');
       TekStatus:=ScopeStatus;
     end;

     {allow local vertical position control on the scope}
     {allow local trigger level control on the scope}
   Pause(ScopeDelay);
   FlushScopeBuff;
 end; {ResetScope}
 {*****************************************************************}
 procedure SetScopeCoupling;
 begin
     if ScopeRecord.local=TRUE then
       writeln(ez_out0,'LOCAL 707')
     else
     begin
       with ScopeRecord do
       begin
         if DC then WRITELN(ez_out0,ScopeAddr,':CHANNEL1:COUPLING DC')
         else
         if AC then WRITELN(ez_out0,ScopeAddr,':CHANNEL1:COUPLING AC')
         else
         WRITELN(ez_out0,ScopeAddr,':CHANNEL1:COUPLING DCfifty');
       end;
     end;
 end;{SetScopeCoupling}
 {*****************************************************************}
 function WaveFormAcquired:boolean;
 var
    StatusByte:Byte;
    i:integer;
 begin
    i:=0;
    repeat
      StatusByte:=IeeeStatus(1);
      StatusByte:=StatusByte AND 8;
      inc(i);
    until (StatusByte=8) OR (i=10);
    if ScopeStatus=AcqCompleteLeft then
    WaveFormAcquired:=TRUE
    else
    WaveFormAcquired:=FALSE;
 end;{WaveFormAcquired}
 {*****************************************************************}
 procedure SetVectorMode(state:boolean);
 begin
  case state of
  ON : WRITELN(ez_out0,ScopeAddr,':DISPLAY:CONNECT OFF');
  OFF: WRITELN(ez_out0,ScopeAddr,':DISPLAY:CONNECT ON');
  end;{case}
 end;
 {*****************************************************************}
 procedure SetSaveMode(state:boolean);
 begin
  case state of
  ON : WRITELN(ez_out0,ScopeAddr,'FRO 0,HOR MTB,TRG SNG');
  OFF: WRITELN(ez_out0,ScopeAddr,'FRO 0,HOR MTB,TRG TRI');
  end;{case}
 end;
 {*****************************************************************}
 procedure WaveFormAcquire;
 begin
    WRITELN(ez_out0,ScopeAddr,'FRO 0,HOR MTB,TRG SNG');
 end;
 {*****************************************************************}
 procedure buffconvert(var outbuff:intbuff;var
 Routbuff,Ioutbuff:PRData_array);
 begin
 end; {buffconvert}
 {*****************************************************************}
 procedure Digitize;
 begin
    writeln(ez_out0,ScopeAddr,':DIGITIZE CHANNEL1');
 end;
 {*****************************************************************}
 procedure SetupWaveform;
 begin
    writeln(ez_out0,ScopeAddr,':ACQUIRE:TYPE NORMAL;COUNT 1;POINTS
 ',Waveformsize);
 {   writeln(ez_out0,ScopeAddr,':ACQUIRE:TYPE RAWDATA,512,10');}
 end;
 {*****************************************************************}
 Procedure GetData(var r:RData_array);
 begin
    {ask for the waveform}
    writeln(ez_out0,ScopeAddr,':WAVEFORM:SOURCE CHANNEL1;FORMAT
 COMP;DATA?');
    {fill the buffer}
    Writeln(ez_out0,'ENTER 707 DMA ADDRESS '
               ,TekBuffSeg,':',TekBuffOfs
               ,' COUNT ',5);
    Writeln(ez_out0,'ENTER 707 DMA ADDRESS '
               ,TekBuffSeg,':',TekBuffOfs
               ,' COUNT ',WaveformSize);
    r[0]:=r[1]; {For some reason the first point is questionable}
 end; {GetData}
 {*****************************************************************}
 function GetHTBRange:double;
 var
   strresponse:PChar;
   tempstr:string;
   temp:double;
   resultcode:integer;
 begin
 {retrieve the HTB string from the scope}
   GetMem(strresponse,32);
   FlushScopeBuff;
   writeln(ez_out0,ScopeAddr,':TIMEBASE:RANGE?');
   writeln(ez_out0,'ENTER 707');
   readln(ez_in0,tempstr);
   strpcopy(strresponse,tempstr);
   stripwhite(strresponse);
   tempstr:=StrPas(strresponse);
   val(tempstr,temp,resultcode);   {Get a GPF if you don't do this!!}
   GetHTBRange:=temp;
   FreeMem(strresponse,32);
   FlushScopeBuff;
 end;{GetHTBRange}
 {*****************************************************************}
 function GetVertRange:double;
 var
   strresponse:PChar;
   tempstr:string;
   temp:double;
   resultcode:integer;
 begin
 {retrieve the V/div string from the scope}
   GetMem(strresponse,32);
   FlushScopeBuff;
   writeln(ez_out0,ScopeAddr,':CHANNEL1:RANGE?');
   writeln(ez_out0,'ENTER 707');
   readln(ez_in0,tempstr);
   strpcopy(strresponse,tempstr);
   stripwhite(strresponse);
   tempstr:=StrPas(strresponse);
   val(tempstr,temp,resultcode);   {Get a GPF if you don't do this!!}
   GetVertRange:=temp;
   FreeMem(strresponse,32);
   FlushScopeBuff;
 end;{GetVertRange}
 {*****************************************************************}
 procedure SetHTB(Range:double);
 begin
   writeln(ez_out0,ScopeAddr,':TIMEBASE:RANGE ',Range);
 end;{SetHTB}
 {*****************************************************************}
 procedure SetVertSens(Range:double);
 begin
   writeln(ez_out0,ScopeAddr,':CHANNEL1:RANGE ',Range);
 end;{SetVertSens}
 {*****************************************************************}
 procedure SCopeRun;
 begin
   writeln(ez_out0,ScopeAddr,':TIMEBASE:MODE TRIGGERED ; :RUN');
 end;
 {*****************************************************************}
 procedure SCopeStop;
 begin
   writeln(ez_out0,ScopeAddr,':STOP');
 end;
 {*****************************************************************}
 procedure ExitScope;
 var
   msg:integer;
 Begin
           { relinquish control of the Scope }
     ExitProc:=ExitSave;
     GlobalDosFree(TekBuffSelector);
     writeln(ez_out0,'CLEAR 707');
     writeln(ez_out0,'LOCAL 707');
     writeln(ez_out0,'RESUME 7');
```

```
{$ifdef debug}
    msg:=MessageBox(Window,'Exit procedure from hp54542A has run.',
        'ExitScope',mb_OK);
{$endif}
{    writeln('EXIT PROCEDURE FROM hp54542A UNIT HAS RUN.'); }
End;
{*************************************************************}
Function GetRealBuff(size:word;var
BuffSeg,BuffOfs,selector:word):pointer;
var
   GDAValue,
   SelectorBase,
   SelectorLimit:longint;
   Para_seg:word;

begin
{now get some REAL MODE memory for a DMA buffer.  Must not cross a
page boundary!}

   GDAValue :=GlobalDosAlloc(size);
   if GDAValue <>0 then
      begin
         GlobalLock(THandle(GDAValue));
         Para_seg := Hiword(GDAValue);   {"paragraph-segment value"}
         selector := Loword(GdaValue);
            SelectorBase:=GetSelectorBase(selector);
            SelectorLimit:=GetSelectorLimit(selector);

            { assign the real-mode address elements for the DMA
driver}
         BuffSeg:=SelectorBase SHR 4;
         BuffOfs:=(SelectorBase AND $0F);
         GetRealBuff:=ptr(Selector,0);
      end
   else  { allocation failed}
      halt(1);
{     writeln('Could not allocate memory for the DMA buffer.');}
end; {GetRealBuff}

{*************************************************************}

begin
ExitSave := ExitProc;
ExitProc := @ExitScope;
{ResetScope;}
writeln(ez_out0, 'CONTROL 7,16;129,10'); {set to LF & EOI terminator}
HorTimeRange:=GetHTBRange;
VerRange:=GetVertRange;
  repeat
        { get a DMA buffer:
           - in the first megabyte of memory
           - with a known REAL MODE (physical) address
           - ensuring it does not cross a page boundary}
    PScopeData:=GetRealBuff(MaxTekBuffSize,TekBuffSeg,TekBuffOfs,
                            TekBuffSelector);

       {if the offset + size crosses a segment boundary, it will wrap
around so...}
    if (TekBuffOfs > word(TekBuffOfs+MaxTekBuffSize)) then
       begin
```

```
          msg:=MessageBox(Window,'Need to try for another DMA
buffer...',
               'LDVWGLOB - init',mb_OK);
          GlobalDosFree(TekBuffSelector);
        end
        else
        begin

{          GetMem(MsgStr,128);
          StrPCopy(MsgStr,IToStr(TekBuffSeg)+':'+IToStr(TekBuffOfs));
          msg:=MessageBox(Window,MsgStr,
              'DMA Buffer location address:',mb_OK);
            FreeMem(MsgStr,128)   }

        end;
  until (TekBuffOfs <= word(TekBuffOfs+MaxTekBuffSize));

{  FreeMem(MsgStr,128);}

for i:=0 to MaxTekBuffSize-1 do PScopeData^[i]:=240; {initialize in a
disctinctive way}
End.
```

UNIT KEITHLEY;

```
{Provides routines for getting measurements from a Keithley 195
 digital multimeter}
{*****************************************************************}
{$X+}
{$F+}
interface

uses
   ez488w,
   winprocs,wintypes,strings,
   timer,Mystring;

const
   DMMAddr = 'OUTPUT 716;';
   DCVolts = 'F0';
   ACVolts = 'F1';
   Ohms    = 'F2';
   DCCurrent='F3';
   ACCurrent='F4';
   AUTORANGE='R0';
   RANGE1   ='R1';
   RANGE2   ='R2';
   RANGE3   ='R3';
   RANGE4   ='R4';
   RANGES5  ='R5';
   RANGE6   ='R6';
   RANGE7   ='R7';
   ZEROOFF  ='Z0';
   ZEROON   ='Z1';

Procedure DMMRemote;
Procedure DMMLocal;
Procedure Want2Measure(what:string);
Procedure SetRange(what:string);
Procedure DMMZero(what:string);
function DMMReadADC:string;

{*****************************************************************}

implementation
{*****************************************************************}
var
   ExitSave : pointer;     {saves the old exit routine pointer}
   Window : HWnd;

Procedure DMMRemote;
begin
    writeln(ez_out0,'REMOTE 716');
end;
{*****************************************************************}

Procedure DMMLocal;
begin
    writeln(ez_out0,'LOCAL 716');
end;
{*****************************************************************}
```

```
Procedure Want2Measure(what:string);
begin
    writeln(ez_out0,DMMAddr,what,'G1X');
    pause(500);
end;
{*****************************************************************}
Procedure SetRange(what:string);
begin
    writeln(ez_out0,DMMAddr,what,'X');
    pause(500);
end;
{*****************************************************************}
Procedure DMMZero(what:string);
begin
    writeln(ez_out0,DMMAddr,what,'X');
end;
{*****************************************************************}
function DMMReadADC:string;
var
  strresponse:string;
begin
    writeln(ez_out0,'ENTER 716;');
    readln(ez_in0,strresponse);
    DMMReadADC:=strresponse;
end;
{*****************************************************************}

Procedure ExitDMM;
var
  msg:integer;
Begin
          { relinquish control of the DMM }
    ExitProc:=ExitSave;
    writeln(ez_out0,'CLEAR 716');
    DMMLocal;
    writeln(ez_out0,'RESUME 7');
{$ifdef debug}
    msg:=MessageBox(Window,'Exit procedure from Keithley has run.',
       'ExitDMM',mb_OK);
{$endif}
End;
{*****************************************************************}
begin
    DMMRemote;
{    writeln(ez_out0,DMMAddr,'D- HELLO- X');
    writeln(ez_out0,DMMAddr,'Q21X');  }
    ExitSave := ExitProc;
    ExitProc := @ExitDMM;
end.
```

UNIT NRC855C5;

```
{Provides routines for controlling the
 Newport 855C Programmable Controller}

{*****************************************************************}
{$F+}
interface
uses
    windos,winprocs,wintypes,strings,
    ez488w,ldvwglb5,timer,
    MYString;


const
  NRCAddr = 'OUTPUT 709;';                {GPIB address of 855C}

var
  i,spoll_byte, error_code : integer;
  NRCStat:byte;

procedure GoToPosition(destination:PosXferRecord);
procedure GoHome;
function CurrentPosition(drive:byte;var response:Text15):boolean;
{*****************************************************************}

implementation

VAR
   ExitSave : pointer;    {saves the old exit routine pointer}
   Window : HWnd;
{*****************************************************************}
procedure ResetNRC;
begin
writeln(ez_out0,'SET TIMEOUT 7;250');
pause(100);
writeln(ez_out0,'RESET 7');
pause(100);
writeln(ez_out0,'CLEAR 709');
pause(100);
writeln(ez_out0,'RESUME 7');
end;
{*****************************************************************}
function NRCStatus:byte;
begin
   if SRQ then NRCStatus:=SPOLL(9)
   else NRCStatus:=0;
end;{NRCStatus}
{*****************************************************************}
procedure FlushNRCBuffer;
var
  junk:string;
begin
  pause(30);
  NRCStat:=NRCStatus;  {bits 6 & 7 set: buffer is empty}
  while ((NRCStat AND 64)=64)
    AND ((NRCStat AND 128)<>128) do
  begin
    writeln(ez_out0,'ENTER 709');
    readln(ez_in0,junk);
    pause(30);
    NRCStat:=NRCStatus;
  end; {while}
end;
{*******************************************************************}
procedure GoToPosition(destination:PosXferRecord);
begin
  with destination do
  begin
    writeln(ez_out0,NRCAddr,'A1',X);
    writeln(ez_out0,NRCAddr,'A2',Y);
  end;
  writeln(ez_out0,NRCAddr,'M0');
  FlushNRCBuffer;
end;
{*******************************************************************}
procedure GoHome;
begin
  writeln(ez_out0,NRCAddr,'H0');
  FlushNRCBuffer;
end;
{*******************************************************************}
function CurrentPosition(drive:byte;var response:Text15):boolean;
var
  tempstr:string;
  j:integer;
begin
  tempstr:='C'+IToStr(Drive);
  FlushNRCBuffer;
  writeln(ez_out0,NRCAddr,tempstr);
  j:=0;
  NRCStat:=NRCStatus;
  while (NRCStat<>64) and (j<10) do
  begin     {wait here until 855C has data ready}
    inc(j);
    pause(30);
    NRCStat:=NRCStatus;
  end;
  writeln(ez_out0,'ENTER 709');
  readln(ez_in0,tempstr);  {get the command echo: 'Cx'}
  if tempstr='' then
     begin
       currentposition:=false;
       exit;
     end;
  j:=0;
  NRCStat:=NRCStatus;
  while (NRCStat<>64) and (j<10) do
  begin     {wait here until 855C has data ready}
    inc(j);
    pause(30);
    NRCStat:=NRCStatus;
  end;
  writeln(ez_out0,'ENTER 709');
  readln(ez_in0,response); {get the position value}
  if response='' then
     begin
       currentposition:=false;
       exit;
     end;
  StripWhite(response);
end;
{*******************************************************************}
procedure ExitNRC;
var
  msg:integer;
Begin
    ExitProc:=ExitSave;
{$ifdef debug}
    msg:=MessageBox(Window,'Exit procedure from NRC855C has run.',
        'ExitNRC',mb_OK);
{$endif}
End;
{*******************************************************************}
begin
ExitSave := ExitProc;
ExitProc := @ExitNRC;
pause(100);
writeln(ez_out0,NRCAddr,'Q2');
FlushNRCBuffer;
writeln(ez_out0,NRCAddr,'V1.4'); {set actuators to maximum vel:
0.4mm/s}
FlushNRCBuffer;
writeln(ez_out0,NRCAddr,'V2.4');
i:=0;
while (i<2) and not CurrentPosition(1,PositionRecord.X) do inc(i);
i:=0;
while (i<2) and not CurrentPosition(2,PositionRecord.Y) do inc(i);
End.
```

PROGRAM KEITHXY;

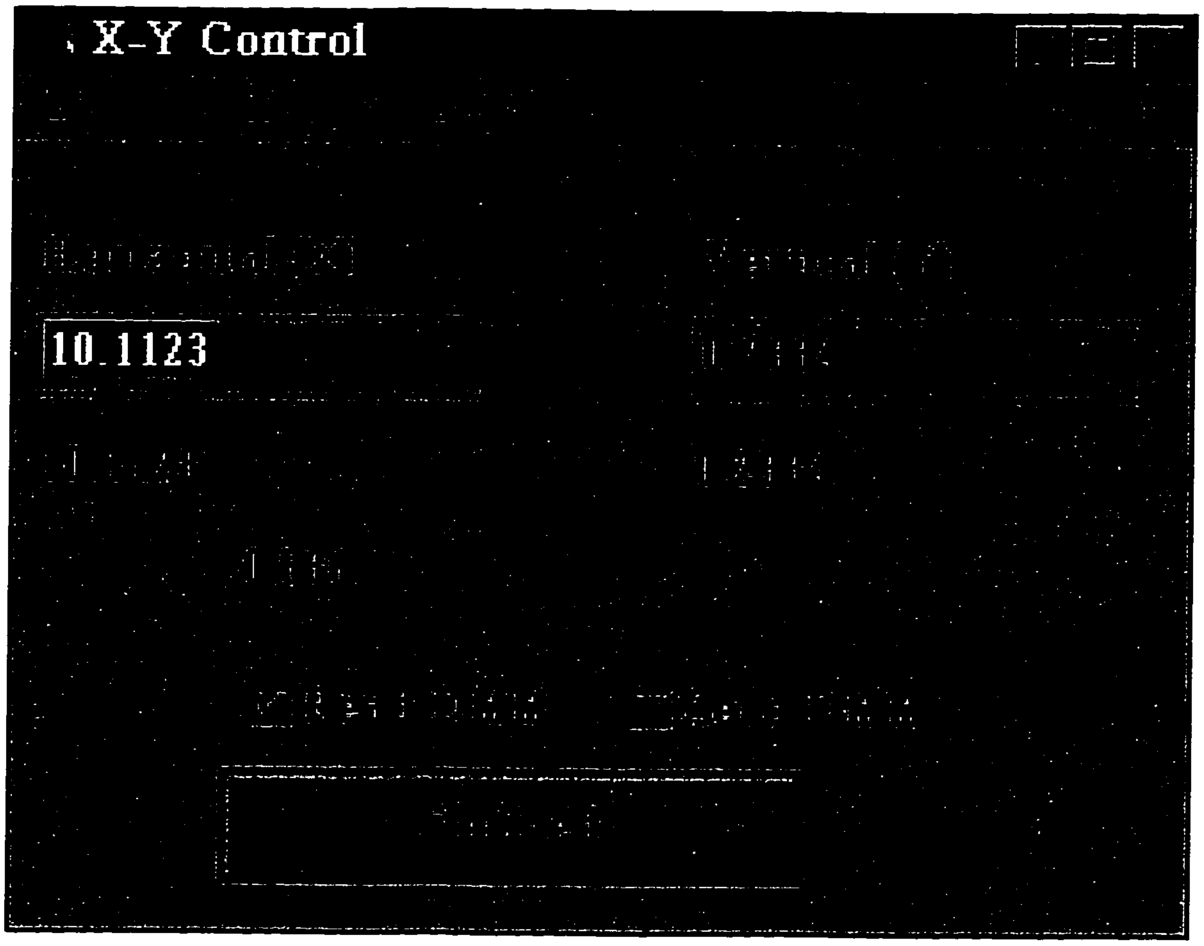

Figure A.1 User Interface for KeithXY Application.

program KeithXY;

```
{$R KeithXY}    {resources}
{$X+}
{$F+}
{$M 16000,20000}

{The purpose of this program is to move a pair of actuators and
report the voltage reading from the Keithley 195 DMM. The positions
are entered into edit boxes and the actual positions and DMM reading
are displayed in text boxes. A menu item copies the data into the
Windows clipboard. Through use of the clipboard and the Windows macro
recorder, the program can be linked to Excel for automated
experiments.}

uses
  WinTypes, WinProcs, Strings, StdWnds,WObjects,
  WinDos,
  ldvwglb5,     {global variable and constant declarations}
  ez488w,timer,
  nrc855c5,    {Handles XY positioner}
  Keithley,    {Keithley 195 GPIB DMM}
  MyString,
  OGL1, OGL2, OGL3,  {Whitewater Object Graphics units}
  StdDlgs;

const
  AppName = 'KeithXY';
  AppMenu = 'KXYmenu';
  AppAccel= 'KXYaccel';  {Used to jump to the X & Y edit fields with
alt-X and alt-Y}
  id_About = 100;     {KXY menu IDs}
  cm_SetPos= 110;     {Set position menu item}
  cm_EditXPos=130;    {Jump to X position edit box}
  cm_EditYPos=140;    {Jump to Y position edit box}
  cm_Copy  = 120;     {Copy data to clipboard menu item}
  id_DMMValue = 113;       {put the reading from the Keithley DMM
here}
  id_DMMEnable= 114;       {check box to enable reading the DMM}
  id_DMMZero=   115;       {Check box to zero the DMM}
  id_XPos=      116;       {Edit box for X position}
  id_YPos=      117;       {Edit box for Y position}
  id_XPosStat=  118;
  id_YPosStat=  119;
  ReadDMM:boolean = FALSE;
  Skip:boolean = False;  {When True, the timer procedure does
nothing.}
  XText:Text15 = '0';
  YText:Text15 = '0';
var
  CurrentText     :Text15;

type
  PKXYDialog = ^KXYDialog;
  KXYDialog = object(TDlgWindow)
    XPosEdit,YPosEdit:PEdit;
    XPosStat,YPosStat,DMMAmps:PStatic;
    DMMEnable,DMMZeroCheck:PCheckbox;
    procedure SetupWindow; virtual;
    constructor Init;
    procedure WMDestroy(var Msg: TMessage);
```

```
        virtual wm_First + wm_Destroy;
      function GetClassName:PChar; virtual;
      procedure GetWindowClass(var AWndClass:TWndClass);virtual;
      procedure getcurrent(var Msg:TMessage);
        virtual wm_first + wm_timer;
      procedure HandleDMMEnableMsg(var Msg: TMessage);
        virtual id_First + id_DMMEnable;
      procedure HandleDMMZeroCheckMsg(var Msg: TMessage);
        virtual id_First + id_DMMZero;
      procedure PositionSet(var Msg: TMessage);
        virtual cm_First + cm_SetPos;
      procedure EditXPosition(var Msg:TMessage);
        virtual cm_First + cm_EditXPos;
      procedure EditYPosition(var Msg:TMessage);
        virtual cm_First + cm_EditYPos;
      procedure FillClipBrd(var Msg: TMessage);
        virtual cm_First + cm_Copy;
      procedure About(var Msg: TMessage);
        virtual cm_First + id_About;
   end;

 KXYApplication = object(TApplication)
     procedure InitMainWindow; virtual;
   end;
 { -----------KXYApplication Method------------------------ }
 procedure KXYApplication.InitMainWindow;
 begin
   MainWindow := New(PKXYDialog, Init);
 end;
 {*****************************************************************}
 {------------------KXYDialog Procedures------------------------}
 Constructor KXYDialog.Init;
 begin
   TDlgWindow.Init(nil,Appname);
   New(DMMEnable,InitResource(@Self,id_DMMEnable));
   New(DMMZeroCheck,InitResource(@Self,id_DMMZero));
   New(DMMAmps,InitResource(@Self,id_DMMValue,15));
   New(XPosEdit, InitResource(@Self, id_XPos,15 ));
   New(YPosEdit, InitResource(@Self, id_YPos,15 ));
   New(XPosStat,InitResource(@Self,id_XPosStat,15));
   New(YPosStat,InitResource(@Self,id_YPosStat,15));
   {LoadAccelerators(HInstance, AppAccel);}
   {TWindowAttr.Menu := LoadMenu(HInstance, AppMenu);}
 end;
 {*****************************************************************}
 procedure KXYDialog.SetupWindow;
 var
   CString:Text15;
   Result,ReturnValue:integer;
 begin
   TDlgWindow.SetupWindow;
   if CurrentPosition(1,PositionRecord.X)
     then XPosEdit^.SetText(PositionRecord.X);
   if CurrentPosition(2,PositionRecord.Y)
     then YPosEdit^.SetText(PositionRecord.Y);
   DMMEnable^.SetCheck(1);
   DMMZeroCheck^.SetCheck(0);
   Want2Measure(DCVolts);
   Pause(800); {used to be 800}
   SetRange(RANGE3);
   Result := IDRetry;
```

```
  while (SetTimer(HWindow, 0, 2000, nil) = 0) and (Result = IDRetry)
do
    Result := MessageBox(GetFocus,'Could not Create Timer',
'SpellmnI',
      mb_RetryCancel);
  if Result = IDCancel then PostQuitMessage(0);
  ReadDMM:=TRUE;
end;
{*****************************************************************}
procedure KXYDialog.GetWindowClass(var AWndClass:TWndClass);
begin
  TDlgWindow.GetWindowClass(AwndClass);
  AWndClass.hIcon:=LoadICon(HInstance,Appname);
{  AWndClass.lpszMenuName:=AppMenu;}
end;
{*****************************************************************}
function KXYDialog.GetClassName:PChar;
begin
 GetClassName:=AppName;
end;
{*****************************************************************}
procedure KXYDialog.HandleDMMEnableMsg(var Msg: TMessage);
begin
   Skip:=True;
   if DMMEnable^.GetCheck=0 then
   begin
     SetRange(RANGE3);  {Can't DMMReadADC anymore if it switches
range on its own}
     Want2Measure(DCVolts{Current});
     Pause(800); {used to be 800}
     DMMEnable^.SetCheck(1);
     ReadDMM:=TRUE;  {as soon as this happens, the Keithley will
start being read}
   end
   else
   begin
     ReadDMM:=FALSE;
     DMMLocal;
     DMMEnable^.SetCheck(0);
   end;
   Skip:=False;
end;
{*****************************************************************}
procedure KXYDialog.HandleDMMZeroCheckMsg(var Msg: TMessage);
begin
   Skip:=True;
   if ReadDMM then   {only need to do something if the DMM is
connected}
      if DMMZeroCheck^.GetCheck=0 then
      begin
       DMMZeroCheck^.SetCheck(1);
       DMMZero(ZEROON);
      end
      else
      begin
       DMMZeroCheck^.SetCheck(0);
       DMMZero(ZEROOFF);
      end;
   Skip:=False;
end;
{*****************************************************************}
```

```
procedure KXYDialog.getcurrent(var Msg:TMessage);
    {timer procedure for main window}
begin
  if (Msg.wParam=0) and not skip then
  begin
    if CurrentPosition(1,PositionRecord.X)
    then XPosStat^.SetText(PositionRecord.X);
    if CurrentPosition(2,PositionRecord.Y)
    then YPosStat^.SetText(PositionRecord.Y);
    if ReadDMM then
      begin
        StrPcopy(CurrentText,DMMReadADC);
        DMMAmps^.SetText(CurrentText);
      end;
  end;
end;{getcurrent}
{*****************************************************************}
procedure KXYDialog.WMDestroy(var Msg: TMessage);
begin
  KillTimer(HWindow, 0);
  TDlgWindow.WMDestroy(Msg);
end;
{*****************************************************************}
procedure KXYDialog.FillClipBrd(var Msg:Tmessage);
var
  GlobalHandle:THandle;
  i:integer;
  temp:string;
  tmpstr:array[0..25] of char;
  Pstr:PChar;
  XSetpoint,YSetpoint:double;
  TimerProcMsg:TMessage;
begin
  i:=0;
  TimerProcMsg:=Msg;
  TimerProcMsg.wParam:=0;
  XPosEdit^.GetText(tmpstr,15);
  XSetpoint:=PCharToR(tmpstr);
  YPosEdit^.GetText(tmpstr,15);
  YSetpoint:=PCharToR(tmpstr);  {Grab the setpoints and convert to
doubles.}
  GetCurrent(TimerProcMsg);
  while (i<10)
   and ( Abs( PCharToR(PositionRecord.X) - XSetpoint ) >0.0005 )
   and ( Abs( PCharToR(PositionRecord.Y) - YSetpoint ) >0.0005 )
   do
     begin
       Skip:=False;
       GetCurrent(TimerProcMsg);
       Skip:=True;
       pause(2000);{wait up to 20 seconds for the actuators to reach
their setpoints before copying}
       inc(i);
     end;
  Skip:=True;
  GlobalHandle:=GlobalAlloc(gmem_Moveable,200);
  if GlobalHandle<>0 then
    begin
      Pstr:=GlobalLock(GlobalHandle);
      if PStr <> nil then
      begin
```

```
          XPosStat^.GetText(Pstr,15);
          StrCat(Pstr,TAB);
          YPosStat^.GetText(tmpstr,15);
          StrCat(Pstr,tmpstr);
          StrCat(Pstr,TAB);
          DMMAmps^.GetText(tmpstr,15);
          StrCat(Pstr,tmpstr);
          GlobalUnlock(GlobalHandle);
          if OpenClipBoard(HWindow) then
            begin
             EmptyClipBoard;
             SetClipBoardData(cf_Text,GlobalHandle);
             CloseClipBoard;
            end
          else GlobalFree(GlobalHandle);
        end
        else GlobalFree(GlobalHandle);
     end;{if got a GlobalHandle}
     Skip:=False;
end;  {FillClipBrd}
{****************************************************************}
procedure KXYDialog.EditXPosition(var Msg:TMessage);
begin
 Setfocus(GetItemHandle(ID_XPos));
 XPosEdit^.SetSelection(0,15);
end;
{****************************************************************}
procedure KXYDialog.EditYPosition(var Msg:TMessage);
begin
 Setfocus(GetItemHandle(ID_YPos));
 YPosEdit^.SetSelection(0,15);
end;
{****************************************************************}
procedure KXYDialog.PositionSet(var Msg:TMessage);
begin
  {Send the actuators to the values in the edit boxes.}
  Skip:=True;
  XPosEdit^.GetText(PositionRecord.X,15);
  YPosEdit^.GetText(PositionRecord.Y,15);
  GoToPosition(PositionRecord);   {set the positioners}
  Skip:=False;
end;
{****************************************************************}
procedure KXYDialog.About(var Msg:TMessage);
begin
  Application^.ExecDialog(New(PDialog,Init(@Self, 'ABOUTBOX')));
end;
{****************************************************************}

var
  KXYApp : KXYApplication;

begin
  KXYApp.Init(AppName);
  KXYApp.Run;
  KXYApp.Done;
end.
```

UNIT DT2827W5;

```
{$F+}
interface
uses
   WinProcs,WinTypes,
   timer,
   strings;


    { ************** GLOBAL CONSTANT DECLARATIONS **************** }


CONST

   outport = 1;   { port for digital outputs }
   inport  = 0;   { port for digital inputs }
   PowerSupplyVDAC = 0; {channel 0 to voltage input of power supply}
   PowerSupplyIDAC = 1; {channel 1 to current input of power supply}

   DT2827BASE = $240;
   NumADChan  = 4;
   Bit0  = $1      ; { lsb           }
      Bit1  = $2  ; { Bit 1 value    }
      Bit2  = $4  ; { Bit 2 value    }
      Bit3  = $8  ; { Bit 3 value    }
      Bit4  = $10 ; { Bit 4 value    }
      Bit5  = $20 ; { Bit 5 value    }
      Bit6  = $40 ; { Bit 6 value    }
      Bit7  = $80 ; { Bit 7 value    }
      Bit8  = $100       ; { Bit 8 value    }
      Bit9  = $200       ; { Bit 9 value    }
      Bit10 = $400       ; { Bit 10 value   }
      Bit11 = $800       ; { Bit 11 value   }
      Bit12 = $1000      ; { Bit 12 value   }
      Bit13 = $2000      ; { Bit 13 value   }
      Bit14 = $4000      ; { Bit 14 value   }
      Bit15 = $8000      ; { msb            }
   HBOE = 2;        { digital high byte output enable }
   LBOE = 1;        { digital low byte output enable }

      { DMA register constants }

      PageReg5 = $8B     ; { DMA Page Select, channel 5 }
      PageReg6 = $89     ; { DMA Page Select, channel 6 }
      PageReg7 = $8A     ; { DMA Page Select, channel 7 }
      BaseReg5 = $C4   ; { Base Address, channel 5     }
      CountReg5 = $C6 ; { Word Count, channel 5       }
      BaseReg6 = $C8   ; { Base Address, channel 6     }
      CountReg6 = $CA ; { Word Count, channel 6       }
      BaseReg7 = $CC   ; { Base Address, channel 7     }
      CountReg7 = $CE ; { Word Count, channel 7 }
      MaskReg     = $D4 ; { Mask Register       }
      ModeReg = $D6      ; { Mode Register       }
      FlipFlop = $D8   ; { Byte Pointer Flip/Flop      }


{ *********** GLOBAL TYPED CONSTANT DECLARATIONS ************ }
```

```
 { These constants are initialized to a value, but can be re-  }
 { defined in the program to other values.             }

       { Interrupt definitions pre-defined for interrupt level 15 (FC)
 }

    DTInt                      = $77;   { hardware IRQ15 }
    IntMaskReg       : integer = $A1;   { 8259 Mask Register address }
       IRQ_line            : integer = $80; { IRQ bit value of Mask
 }

       { Interrupt register constants }

       IntCommandMaster = $20 ; { Command register address for master
 8259 }
       IntCommandSlave = $A0 ;  { Command register address for slave
 8259  }
    IntEOI = $20 ; { non-specific end of interrupt command }

      ADCSR : integer = $240; {  A/D control register R/W  (BASE)
            }
    CHANCSR : integer = $242; {  A/D channel list control register
 (BASE + 2)}
      ADDAT : integer = $244; {  A/D data register RO (BASE + 4)
 }
      DACSR : integer = $246; {  D/A and DIO control register R/W
 (BASE + 6) }
      DADAT : integer = $248; {  D/A data register WO (BASE + 8)
 }
     DIODAT : integer = $24A; {  DIO data register R/W (BASE + A)
 }
     SUPCSR : integer = $24C; {  DMA control register R/W (BASE + C)
            }
     TMRCTR : integer = $24E; {  Timer/Counter register R/W (BASE + E)
            }

       DMAPage1 : longint = 0; {the page in memory where the DMA
Buffer1 starts}
       DMAPage2 : longint = 0; {tha high 4 bits of the 20 bit address}
       DMABase1 : word    = 0; {the low 16 bits }
       DMABase2 : word    = 0;

DA_ErrorFlag  : Boolean = FALSE; { flag set if DA error occurs during
DMA }
AD_ErrorFlag  : Boolean = FALSE; { flag set if AD error occurs during
DMA }
DTdatasize = 1024;
type
  DTBckgnd_array = array[1..DTDatasize]of word;
  DTData_array = array[1..DTDatasize]of double;
  Text15 = array[0..15] of char;
{ **************** GLOBAL VARIABLE DECLARATIONS ****************** }
VAR
DTBckGrndPtr : ^DTBckgnd_array;
MaskStatus              { holds the status of the interrupt mask
register }
            : integer;
SmplFreq:double;
DTDMAInProgress,DTDMADone:boolean;
{****************************************************************}
procedure DTReset;
procedure DTSetClock(var interval:double); {interval is in us}
procedure DTSetIrqLine;
procedure DTReSetIrqLine;
function  DTvoltage(ADV:integer):double;
function  DTdig_read(dio_port,outbyte:BYTE):boolean;
procedure DTdig_set(dio_port,outbyte:BYTE);
procedure DTdig_clr(dio_port,outbyte:BYTE);
procedure DTdig_flip(dio_port,outbyte:BYTE);
procedure DTDAC_Set(ch:byte;voltage:double);
procedure HV_Set(var value:Text15);
procedure I_Set(var value:Text15);
procedure DT_DMA_ISR; interrupt;
procedure DTProgram_Dma5(Start:pointer;Length:word);
function  DTReadAdi(channel:byte):integer;
function  DTReadTemperature:double;
function  DTReadCurrent:double;
function  DTReadSpellCurrent:double;
function  DTReadSpellVoltage:double;
procedure DTCopyDTData(source:DTBckgnd_array;var
dest:DTdata_array;size:integer);
procedure DTGetMoreDTData;
function  DTBuffull:boolean;


implementation
VAR
   ExitSave : pointer;    {saves the old exit routine pointer}
   Window:HWnd;
{************  useful procedures to have around *********}
procedure DTReset;
begin
   portw[SUPCSR]:=1;
end;
{***************************************************************}
procedure DTSetIrqLine;
begin
  inline($fa);   { cli  disable all interrupts}
  MaskStatus:=port[IntMaskReg];
  port[IntMaskReg]:=MaskStatus AND (NOT IRQ_line); {enable IRQ line}
  inline($fb);   { sti  enable all interrupts}
end;
{***************************************************************}
procedure DTReSetIrqLine;
begin
  inline($fa);   { cli  disable all interrupts}
  MaskStatus:=port[IntMaskReg];
  port[IntMaskReg]:=(MaskStatus OR IRQ_line); {enable IRQ line}
  inline($fb);   { sti  enable all interrupts}
end;
{***************************************************************}
function DTWaitForMUX(waittime:integer):boolean;
var
   endtime:double;
   muxready:boolean;
begin
   endtime:=(Time)+waittime;
   repeat
     muxready:=(portw[ADCSR] and bit8) = 0;
   until muxready OR (Time>endtime);
   if muxready then DTWaitForMUX:=TRUE
   else DTWaitForMUX:=FALSE;
```

```
end;
{*****************************************************************}
procedure DTSetClock(var interval:double); {interval is in us}
var
   prescaler,counter:longint;
   outword:word;
{$R-}
begin
if interval>10.0 then
  begin
    prescaler:=0;
    counter:=-1;
    while counter<0 do
    begin
      counter:=255-round((4.0*interval) / (1 shl prescaler));
      if counter<0 then prescaler:=prescaler+2;
    end; {while}
    if prescaler>14 then
      begin
{       writeln('ERROR prescaler too big. Can not run that slow!');}
      prescaler:=14;
      counter:=1;
      end;
     outword:=(counter and $ff) + ((prescaler and $0f) shl 8);
     portw[TMRCTR]:=outword;
    Smplfreq:=((4E6 / (1 SHL prescaler)) / (255 - counter)); {Hz}
    interval:=1000000.0/Smplfreq;  {usec}
   end;
{$R+}
end; { DTSetClock }
{*****************************************************************}
function DTvoltage(ADV:integer):double;
{ ADV -> A/D converter digital value }
const
     pfs = 10.0000;
     nfs = -10.0000;
     noc = 65536; {16 bits of precision}
{R-}
begin
   DTvoltage:=ADV*(pfs-nfs)/noc;
{R+}
end;
{*****************************************************************}
function DTdig_read(dio_port,outbyte:BYTE):boolean;
begin
  CASE dio_port of
       0 : DTdig_read:=outbyte=(portw[DIODAT] and outbyte);
       1 : DTdig_read:=outbyte=(portw[DIODAT] and (outbyte shl 8));
      end; {case}
end;  {DTdig_read}
{*****************************************************************}
procedure DTdig_set(dio_port,outbyte:BYTE);
begin
  CASE dio_port of
       0 : begin
              portw[DIODAT]:=portw[DIODAT] or outbyte;
              portw[DACSR]:=LBOE;
           end;
       1 : begin
              portw[DIODAT]:=portw[DIODAT] or (outbyte shl 8);
              portw[DACSR]:=HBOE;
```

```
             end;
        end; {case}
 end;  {DTdig_set}
{*******************************************************************}
 procedure DTdig_clr(dio_port,outbyte:BYTE);
 begin
   CASE dio_port of
        0 : begin
              portw[DIODAT]:=portw[DIODAT] and not outbyte;
              portw[DACSR]:=LBOE;
            end;
        1 : begin
              portw[DIODAT]:=portw[DIODAT] and not (outbyte shl 8);
              portw[DACSR]:=HBOE;
            end;
        end; {case}
 end;  {DTdig_clr}
{*******************************************************************}
 procedure DTdig_flip(dio_port,outbyte:BYTE);
 begin
   CASE dio_port of
        0 : begin
              portw[DIODAT]:=portw[DIODAT] xor outbyte;
              portw[DACSR]:=LBOE;
            end;
        1 : begin
              portw[DIODAT]:=portw[DIODAT] xor (outbyte shl 8);
              portw[DACSR]:=HBOE;
            end;
        end; {case}
 end;   {DTdig_flip}
{*******************************************************************}
procedure DTDAC_Set(ch:byte;voltage:double);
const
     pfs = 10.000;
     nfs = -10.000;
     noc = 4095; {12 bits of precision}
var
  DACDataWord:word;
begin
{ Volts -> D/A converter digital value }
{$R-}
  DACDataWord:=2048+trunc(voltage*noc/(pfs-nfs));
{$R+}
  portw[SUPCSR]:=portw[SUPCSR] or Bit5;  {init the DAC}
  portw[DACSR]:=portw[DACSR] or Bit8; {Set bit 8 for dual channel
mode}
  if ch=1 then
    portw[DACSR]:=portw[DACSR] or Bit9    {channel1 sets bit 9
channel0 clears it}
  else
    portw[DACSR]:=portw[DACSR] and $FDFF;    {bit 9 = 0}
  portw[DADAT]:=DACDataWord;
  portw[SUPCSR]:=Bit14 or Bit7;
end;{DTDAC_Set}
{*******************************************************************}
procedure HV_Set(var value:Text15);
var
  volts:double;
  code:integer;
begin
```

```
   VAL(Value,volts,code);
   volts:=volts/3000;
   if volts>10.0 then
     begin
       volts:=10.0;
       strcopy(value,'30000');
     end
   else if volts<-10.0 then
     begin
       volts:=-10.0;
       strcopy(Value,'-30000');
     end;
   DTDAC_Set(PowerSupplyVDAC,volts);
   DTDAC_Set(PowerSupplyIDAC,10);       {Sets current limit to
maximum}
  DTDIG_Set(outport,Bit1);{operates the relay connecting DAC to HV
supply}
end;
{*******************************************************************}
procedure I_Set(var value:Text15);
var
  uamps:double;
  code:integer;
begin
  VAL(Value,uamps,code);
  uamps:=uamps/30;
  if uamps>10.0 then
    begin
      uamps:=10.0;
      strcopy(value,'300');
    end
  else if uamps<-10.0 then      {unfortunately, however, negative
control voltage}
    begin                        {won't give you negative polarity in
the Spellman}
      uamps:=-10.0;
      strcopy(Value,'-300');
    end;
  DTDAC_Set(PowerSupplyIDAC,uamps);
  DTDAC_Set(PowerSupplyVDAC,10);       {Sets voltage limit to
maximum}
  DTDIG_Set(outport,Bit1);{operates the relay connecting DAC to HV
supply}
end;
{*******************************************************************}
procedure DT_DMA_ISR; {interrupt; }
begin
{$R-}
    DTDMAinProgress := FALSE;
    DTDMADone:=((portw[SUPCSR] AND Bit15) = Bit15);;

    AD_ErrorFlag:=((portw[ADCSR] AND Bit15) = Bit15);
    if AD_ErrorFlag then portw[SUPCSR] := portw[SUPCSR] or Bit6;

    DA_ErrorFlag:=((portw[DACSR] AND Bit15) = Bit15);
    if DA_ErrorFlag then portw[SUPCSR] := portw[SUPCSR] or Bit5;

  port[IntCommandMaster] := IntEOI; { issue the EOI to the 8259's }
  port[IntCommandSlave]  := IntEOI;

{$R+}
```

```
end; {DT_DMA_ISR}
{*******************************************************************}
procedure DTProgram_Dma5(Start:pointer;Length:word);
{ The 80486's DMA controller is programmed to receive
  'length' number of WORDs and stuff them into memory
  starting at the 'START' pointer.}
var
   NumWords:word;
begin
   NumWords:=Length-1;  { DMA counts down to 0 inclusive }
   DMAPage1 := SEG(Start^);
   DMAPage1 := ((DMAPage1 SHL 4) + OFS(Start^)) SHR 1;
   DMABase1 := (DMAPage1 AND $FFFF); { the lower 16 bits };
   DMAPage1 := DMAPage1 SHR 16;

   Port[ModeReg]  := $45;{set mode for Write, channel 5, with no
autoinitialize}
   Port[FlipFlop] := 0;           { reset }
   Port[BaseReg5] := LO(DMABase1); { write low byte to base register
}
   Port[BaseReg5] := HI(DMABase1); { write high byte to base register
}
   Port[CountReg5] := LO(NumWords); { write low byte to count
register }
   Port[CountReg5] := HI(NumWords); { write high byte. 1023 = $03ff}
   Port[PageReg5] := DMAPage1 SHL 1;    { select page }
   Port[MaskReg]  := 1;            { enable DMA channel 5 }

end;{DTProgram_Dma5}
{*******************************************************************}
function DTReadAdi(channel:byte):integer;
var
   donetime:double;
   RegCheck:word;
   ADdone  :boolean;
begin
{$R-}
  channel:=channel AND $03; {mask off the high 6 bits (only 4
channels)}

  portw[supcsr]:=(bit13 or bit6);

   { Load list enable, first 4 bits = #chan/gain entries-1 (1-16)}
  portw[CHANCSR]:=Bit15;           { $8000, 1000 0000 0000 0000 }

   { With Load List enabled bits 0-1 -> channel 0-3 (gain is fixed)
     Bit9 enables A/D conversions}
  portw[ADCSR]:=channel OR Bit9;

   { load list disabled (Bit15) }
  portw[CHANCSR]:= 0;

   { preload MUX }
  portw[SUPCSR]:=Bit4;

  while not DTWaitForMUX(1) do
   { begin
      gotoxy(20,20);
      write('Mux still busy.');
    end};
    { issue a software trigger to get the ADC going }
   portw[SUPCSR]:=Bit3;              { $0008, 0000 0000 0000 1000 }

    { now wait for A/D DONE }

    { set Timer }
    DoneTime:=Time+4.0; {allow 4 second;our clock may be set slow}
    repeat
      RegCheck := Portw[ADCSR];
      ADdone := (RegCheck AND Bit7) = Bit7;
      if (RegCheck AND Bit15) <> 0 then
         begin
{           Writeln;
           Writeln('  ERROR BIT IS SET !!! ');}
           Portw[SUPCSR] := Bit6; { clear error }
          end
      else
        if Time > DoneTime then
           begin
{             writeln(' A/D Done Bit will not set in ReadAdi !!! ');}
             DoneTime:=Time+4.0; {allow another 4 seconds}
           end;
  until ADdone;

{ now read the data in the ADDATA register and return it }
  DTReadAdI := Portw[ADDAT];

{$R+}
end;  { DTReadAdi }
{*******************************************************************}
function DTReadCurrent:double;
const
   readings = 16;
var
   i:integer;
   temp:extended;
begin
    { get the ion collector current from analog input channel 0, in
uA }
   temp:=0.0;
   for i:=1 to readings do temp:=temp+DTReadAdi(0);
   DTReadCurrent := 0.99*DTvoltage(round(temp/readings));
end;
{*******************************************************************}
function DTReadSpellCurrent:double;
const
   readings = 32;
var                           {This function returns uA}
   i:integer;
   temp:extended;
begin                                     {MAY BE SOMETHING WRONG WITH
CHANNEL 1}
    { get the total current from analog input channel 1, in uA (green
wire)}
   temp:=0.0;
   for i:=1 to readings do temp:=temp+DTReadAdi(1);
     {0-10V => 0-300uA}
   DTReadSpellCurrent :=  41.58*DTvoltage(round(temp/readings)) +
154.79; {zero adjust!}
end;
{*******************************************************************}
```

```
function DTReadSpellVoltage:double;
const
   readings = 16;
var
   i:integer;
   temp:extended;
begin                                        {MAY BE SOMETHING WRONG
WITH CHANNEL 2}
    { get the output voltage from analog input channel 3, in uA
(white wire)}
   temp:=0.0;
   for i:=1 to readings do temp:=temp+DTVoltage(DTReadAdi(3));
   DTReadSpellVoltage := 3000.0*temp/readings;
end;
{*****************************************************************}
function DTReadTemperature:double;
const
   readings = 50;
var
   i:integer;
   temp:extended;
begin
    { get the pressure from analog input channel 3, in degrees C }
   temp:=0.0;
   for i:=1 to readings do temp:=temp+DTReadAdi(3);
   DTReadTemperature :=( 45.985 * DTvoltage(round(temp/readings))) -
49.252;
end;
{*****************************************************************}
procedure DTCopyDTData(source:DTBckgnd_array;var
dest:DTdata_array;size:integer);
{ procedure to copy the two's complement 16 bit words into the
  REAL half of the COMPLEX array and zero the IMAGINARY half.
  This complex array can then be processed while the background
  array is refilled by DMA. }
var
   i:integer;
begin
    for i:=1 to size do
     begin
                          {do this to allow correct conversion of}
      dest[i]:=source[i]*2.0;  {two's complement words to doubles}
     end;
end;
{*****************************************************************}
procedure DTGetMoreDTData;
{ procedure initiates DMA acquisition of the voltage on input
  channel 0.  'DTDatasize' 16 bit A/D conversions are done by the
  DT2827 at the rate determined by the on-board timer set by a
  call to SetDTClock and the results appear in the array pointed
  to by BckGrndPtr. }
{$R-}
begin
  DTDMADone := false;
  DTProgram_DMA5(DTBckGrndPtr,DTDatasize);
   { Clear Dma Done , A/D Init. }
  portw[SUPCSR]:=$2240;   { $2240, 0010 0010 0100 0000 }
   { Load list enable, first 4 bits = #chan/gain entries-1 (1-16)}
  portw[CHANCSR]:=Bit15;           { $8000, 1000 0000 0000 0000 }
   { With Load List enabled , enable A/D clock, interrupt on A/D
done,
       and bits 0-1 -> channel 0-3 (gain is fixed)}
    portw[ADCSR]:=(Bit9 OR Bit6);      { $0240, 0000 0010 0100 0000 }
     { load list disabled (Bit15) }
    portw[CHANCSR]:= 0;
     { enable interrupt on error, preload MUX }
    portw[SUPCSR]:=$4410;              { $4410, 0100 0100 0001 0000 }
    while not DTWaitForMUX(1) do
 {     begin
        gotoxy(20,20);
        write('Mux still busy.');
      end};
     { enable interrupts on error, select DMA mode 1 and
       issue a software trigger to get the ball rolling again}
    portw[SUPCSR]:=$4408;      {portw[SUPCSR] OR Bit3;}
    DTDMAinProgress:=TRUE;
    {$R+}
 end;  { DTGetMoreDTData }
 {****************************************************************}
 function DTBuffull:boolean;
 begin
    DTBuffull:=DTDMADone and not DTDMAinProgress;
    if not DTDMADone and not DTDMAinProgress then
              { In this case the buffer is not full and is not being
 filled, }
               { so a data grab is started. }
       DTGetMoreDTData;
 end;  {DTBuffull}
 {****************************************************************}
 Procedure ExitDT;
 var
   msg:integer;
   zero:Text15;
 begin
   ExitProc:=ExitSave;
   StrCopy(zero,'0');
   {shut down the HV supply and disconnect it}
   DTDAC_Set(PowerSupplyVDAC,0);       {Sets voltage limit to minimum}
   DTDAC_Set(PowerSupplyIDAC,0);       {Sets current limit to minimum}
 {  DTDIG_Clr(outport,Bit1);{opens the relay connecting DAC to HV
 supply}
 {$ifdef debug}
   msg:=MessageBox(Window,'Exit procedure from dt2827W has run.',
        'EXITDT',mb_OK);
 {$endif}
 end;
 {****************************************************************}
 var
   zero:Text15;
 begin
   ExitSave:=ExitProc;
   ExitProc:=@ExitDT;
   DTDIG_Clr(outport,$ff);{start out with all outputs cleared}
   StrCopy(zero,'0');
   DTDAC_Set(PowerSupplyVDAC,0);       {Sets voltage limit to minimum}
   DTDAC_Set(PowerSupplyIDAC,0);       {Sets current limit to minimum}
 end.
```

PROGRAM DTINIT;

```
uses
   DT2827D;
var
  zero:text15;
begin
  {The program does nothing except allow the initialization routine
   of the DT2827D unit to run.  In it, the power supply DAC gets set
   to zero volts.}
zero:='0';
if paramcount=0 then
     begin
       hv_set(zero);
       dtdig_clr(outport,Bit1);
     end
else
     if paramstr(1) = 'on' then
        begin
         if paramcount>1 then
           begin
             zero:=paramstr(2);
             hv_set(zero);
           end;
         dtdig_set(outport,Bit1);
        end
     else if paramstr(1) = 'off' then
     begin
       dtdig_clr(outport,Bit1);
     end
end.
```

PROGRAM SPELLMAN;

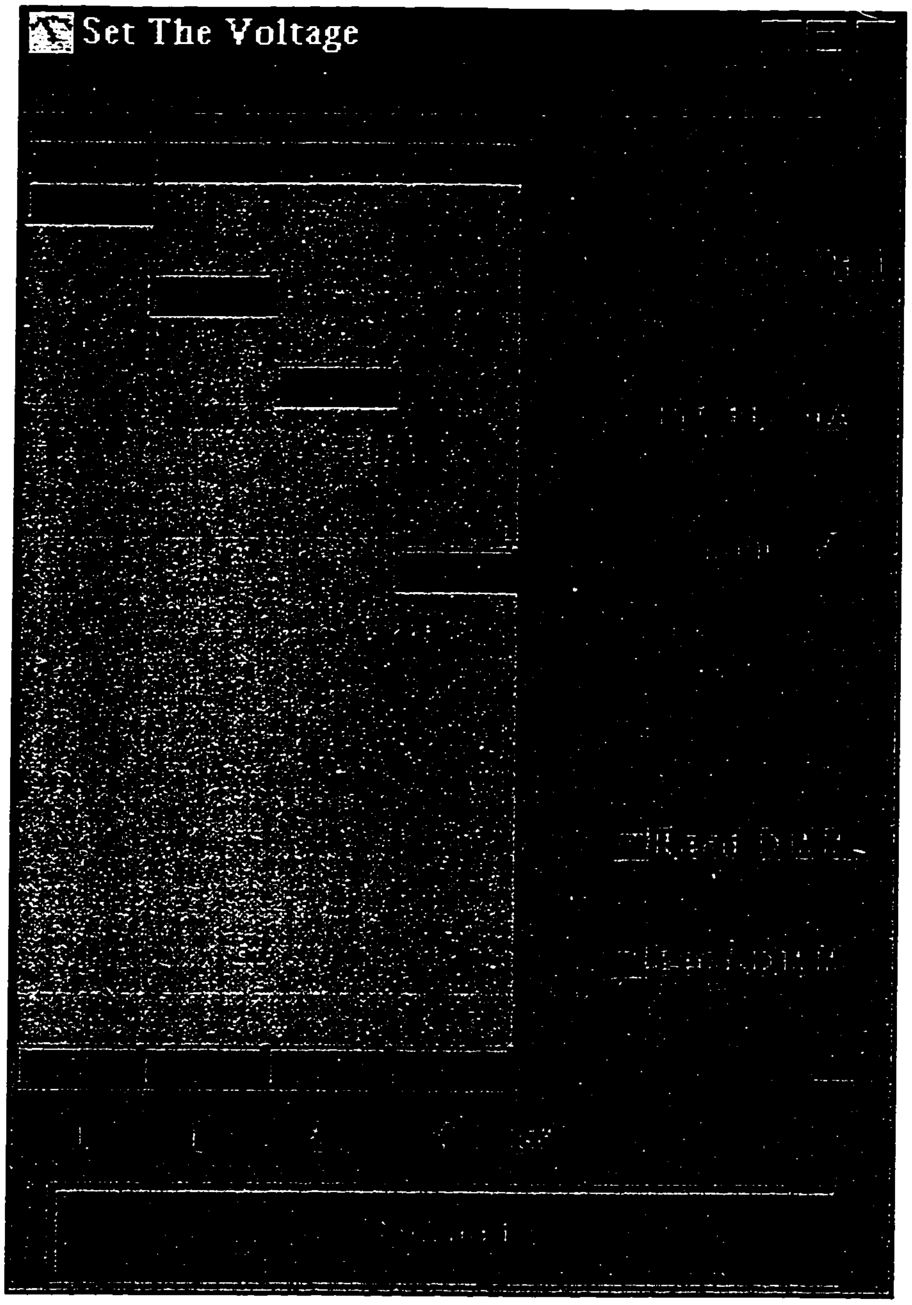

Figure A.2 User Interface for Spellman Application.

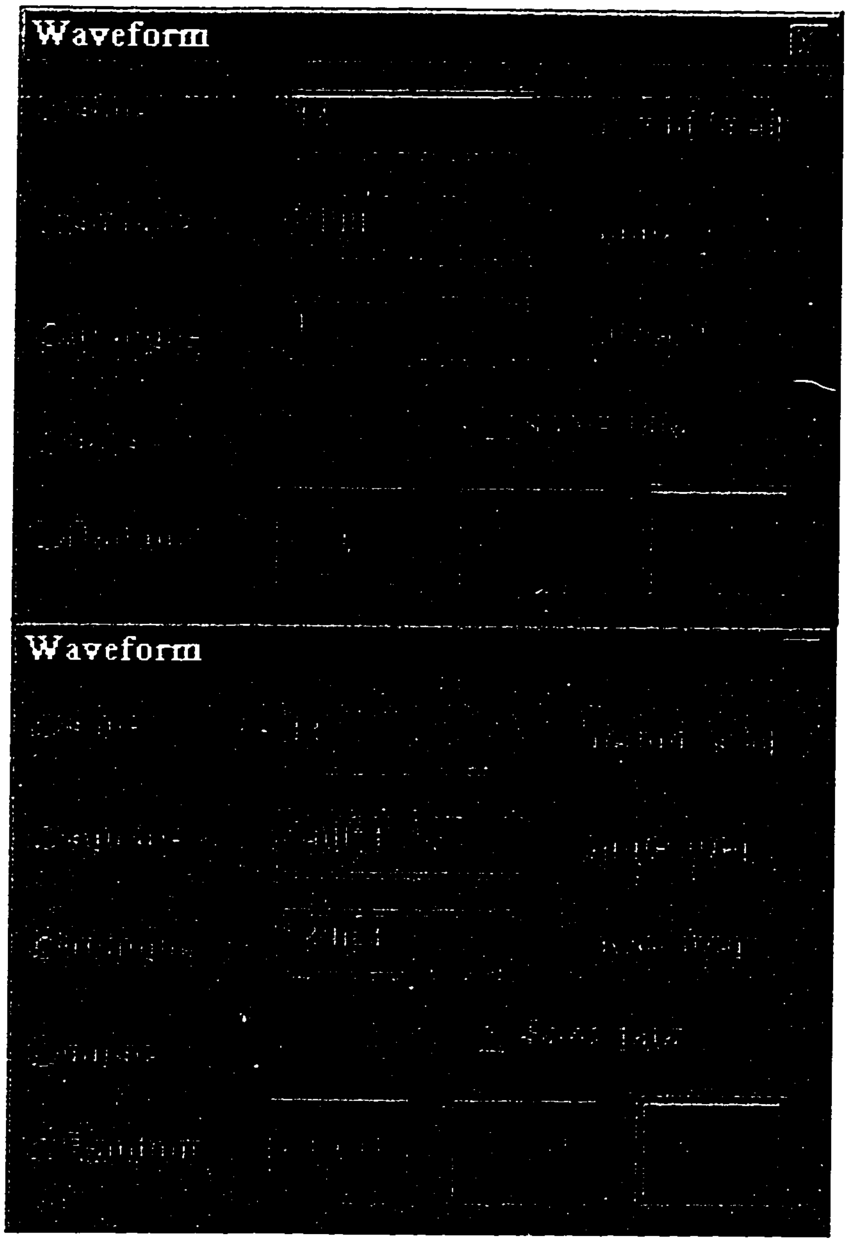

Figure A.3 Spellman application voltage and frequency dialogs.

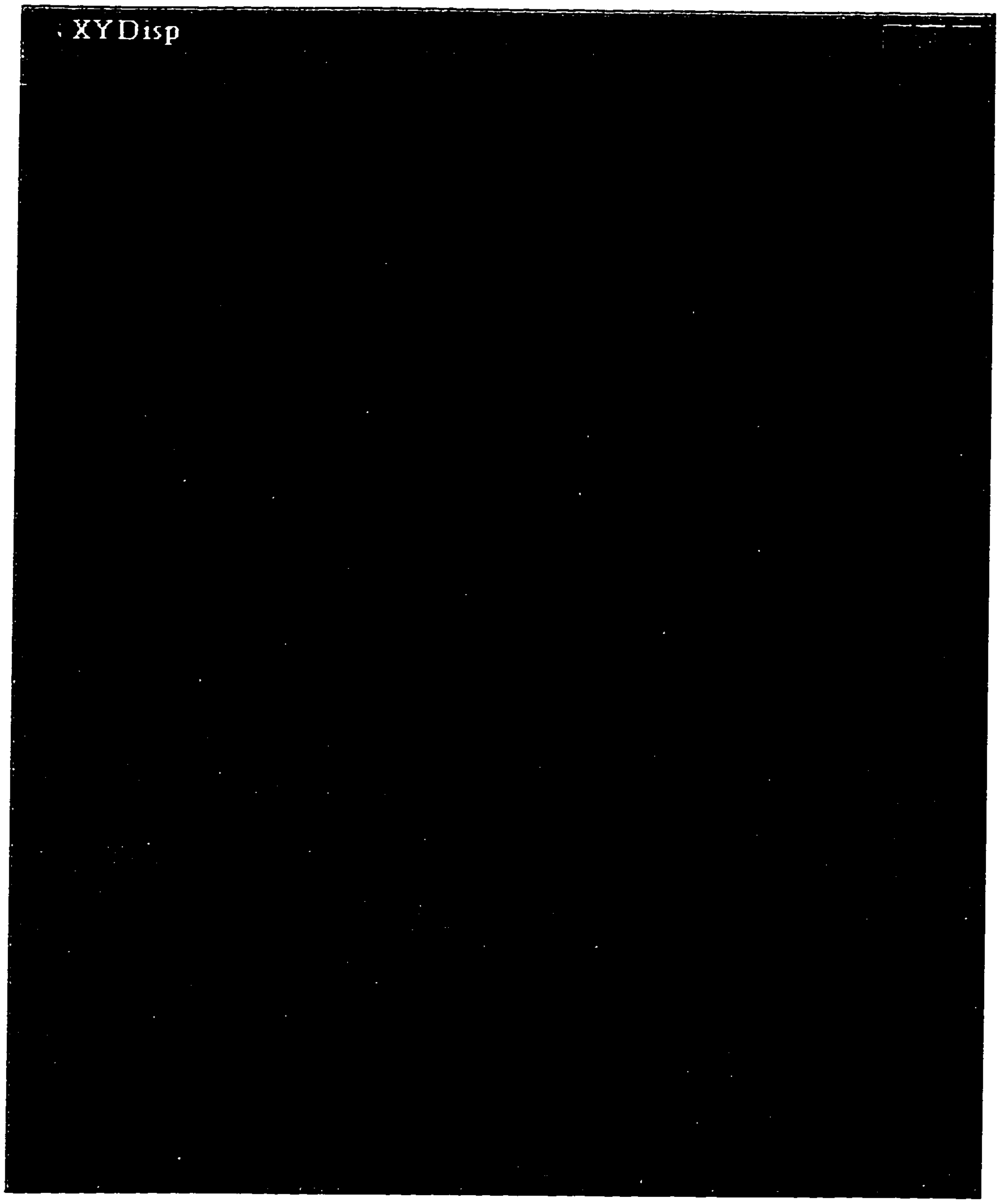

Figure A.4 Spellman application, waveform display. As the voltage or frequency changed according to the waveform chosen, the current data was displayed and saved to disk. The display area could be zoomed by highlighting an area with the mouse.

program Spellman;

```
{ This application can control the high voltage power supply, digital
function generator and Data Translation card. Designed to transfer
data and setpoints via the Windows clipboard.}
{$R Spellman}    {resources}
{$X+}
{$F+}
{$M 16000 16000}
uses
  WinTypes, WinProcs, Strings, StdWnds,WObjects,
  WinDos,
  Timer,
  DT2827w5,     {Data Translation routines}
  Keithley,     {Keithley digital multimeter}
  PM5193,       {Philips model PM5193 digital synthesizer/function
generator}
  MyString,
  OGL1, OGL2, OGL3,  {Whitewater Object Graphics units}
  StdDlgs;

const
  TAB  = #9;
  AppName:PChar = 'Spellman';  {Ensure that these are specified in
the resource file}
  AppMenu = 'voltmenu';        {in the appropriate boxes for the main
dialog window}
  id_uAmps   = 110;   {VoltDialog IDs}
  id_SpelluAmps = 111;
  id_SpellVolts  = 112;
  id_DMMValue = 113;      {put the reading from the Keithley DMM
here}
  id_DMMEnable= 114;      {check box to enable reading the DMM}
  id_DMMZero=   115;      {Check box to zero the DMM}
  id_VScroll3 = 206;
  id_VScroll2 = 207;
  id_VScroll1 = 208;
  id_VScroll0 = 209;
  id_VStat3 = 210;
  id_VStat2 = 211;
  id_VStat1 = 212;
  id_VStat0 = 213;
                 {Voltage waveform dialog IDs}
  cm_Waveform=214;
  id_sine=101;
  id_square=102;
  id_triangle=103;
  id_pulse=104;
  id_random=114;
  id_period=105;
  id_upperV=106;
  id_lowerV=107;
  id_start=111;
  id_stop=113;
  id_saveData=115;
                 {Waveform dialog IDs for frequency}
  cm_FreqWave=215;

type
  PVoltDialog = ^VoltDialog;
  VoltDialog = object(TDlgWindow)
```

```
    periodEdit,upperVEdit,lowerVEdit:PEdit;

SineButt,SquareButt,TriangleButt,PulseButt,RandomButt:PRadioButton;
    TenThouScroll,ThouScroll,HundScroll,TenScroll : PScrollBar;

uAmps,SpelluAmps,Volts,DMMAmps,StatTenThou,StatThou,StatHund,StatTen:
PStatic;
    DMMEnable,DMMZeroCheck,SaveData:PCheckbox;
    procedure SetupWindow; virtual;
    constructor Init;
    procedure WMDestroy(var Msg: TMessage);
      virtual wm_First + wm_Destroy;
    function GetClassName:PChar; virtual;
    procedure GetWindowClass(var AWndClass:TWndClass);virtual;
    procedure UpdateVoltage;
    procedure HandleTenThousScrollMsg(var Msg: TMessage);
      virtual id_First + id_VScroll3;
    procedure HandleThousScrollMsg(var Msg: TMessage);
      virtual id_First + id_VScroll2;
    procedure HandleHundScrollMsg(var Msg: TMessage);
      virtual id_First + id_VScroll1;
    procedure HandleTenScrollMsg(var Msg: TMessage);
      virtual id_First + id_VScroll0;
    procedure getcurrent(var Msg:TMessage);
      virtual wm_first + wm_timer;
    procedure HandleDMMEnableMsg(var Msg: TMessage);
      virtual id_First + id_DMMEnable;
    procedure HandleDMMZeroCheckMsg(var Msg: TMessage);
      virtual id_First + id_DMMZero;
    procedure WaveDLG(var Msg: TMessage);
      virtual cm_First + cm_Waveform;
    procedure FreqWaveDLG(var Msg: TMessage);
      virtual cm_First + cm_FreqWave;
  end;

  waveformtype=(sine,square,triangle,pulse,rand);
  WaveXferRecord = record
    sinebutt,squarebutt,trianglebutt,pulsebutt,randombutt:bool;
{radio buttons}
    periodtxt,uppervtxt,lowervtxt:Text15;{edit boxes}
    SaveData:bool; {Checkbox}
    end;

  PWaveDialog = ^WaveDialog;
  WaveDialog = object(TDlgWindow)
    Waveshape:waveformtype;
    lowerV,upperV:double;
    WavePeriodMS:integer;
    FileName: array[0..fsPathName] of Char;
    constructor Init(AParent: PWindowsObject; ATitle: PChar);
    destructor Done; virtual;
    procedure HandleDataSaveCheck(var Msg: TMessage);
     virtual id_First + id_SaveData;
    procedure Start(var Msg:TMessage);
     virtual id_First + id_start;
    procedure Stop(var Msg:TMessage);
     virtual id_First + id_stop;
    procedure WaveGenerate(var Msg:TMessage);
     virtual wm_first + wm_timer;
    end;
```

```
   PFreqWaveDialog = ^FreqWaveDialog;
   FreqWaveDialog = object(TDlgWindow)
      Waveshape:waveformtype;
      lowerV,upperV:double;
      WavePeriodMS:integer;
      FileName: array[0..fsPathName] of Char;
      constructor Init(AParent: PWindowsObject; ATitle: PChar);
      destructor Done; virtual;
      procedure HandleDataSaveCheck(var Msg: TMessage);
       virtual id_First + id_SaveData;
      procedure Start(var Msg:TMessage);
       virtual id_First + id_start;
      procedure Stop(var Msg:TMessage);
       virtual id_First + id_stop;
      procedure WaveGenerate(var Msg:TMessage);
       virtual wm_first + wm_timer;
      end;

   PXYDispWindow = ^XYDispWindow;
   XYDispWindow = object(TGWindow)  {OGL-Descendent of TWindow}
     ThePlot: PPolyLine;
     Circle: PEllipse;
     TheVertices:PPointColl;
     APoint:PGPoint;
     Fence:PChooser;

     constructor Init(AParent: PWindowsObject; ATitle: PChar);
     destructor Done; virtual;

     procedure BeginDrag(MousePt: PGPoint; KeyStates: Word); virtual;
     procedure Drag(MousePt: PGPoint; KeyStates: Word); virtual;
     procedure EndDrag(MousePt: PGPoint; KeyStates: Word); virtual;

{    procedure CMColor(var Msg: TMessage);virtual cm_First +
cm_Color;
     procedure CMWidth(var Msg: TMessage);virtual cm_First + cm_Width;
}

     procedure Update; virtual;
     procedure SetupWindow; virtual;
   end;{XYDispWindow}


   CTRLApplication = object(TApplication)
     procedure InitMainWindow; virtual;
   end;

const
  NumWaveSamples=32;       {data points in one waveform}
  ReadDMM:boolean = FALSE;
  VoltText:Text15 = '0';

  WaveRecord:WaveXferRecord = (
    sinebutt:False;
    squarebutt:False;
    trianglebutt:True;
    pulsebutt:False;
    randombutt:False;
    periodtxt:'32';      {seconds for one waveform}
    uppervtxt:'1000';
    lowervtxt:'0';
```

```
      SaveData:False);

    FreqWaveRecord:WaveXferRecord = (
      sinebutt:False;
      squarebutt:False;
      trianglebutt:True;
      pulsebutt:False;
      randombutt:False;
      periodtxt:'32';       {seconds for one waveform}
      uppervtxt:'150E3';
      lowervtxt:'120E3';
      SaveData:False);

 var
   waveformvolts,waveformfreq:array[1..NumWaveSamples] of double;
   waveindex:integer;
   XScale,Xoffset,AScale,AOffset,lower,upper:double;
   waveform:waveformtype;
   Rawfile:text;
   PXYDisplay:PXYDispWindow;
   XYDisplayOpen:BOOLEAN;
 { -----------CTRLApplication Method----------------------- }
 procedure CTRLApplication.InitMainWindow;
 begin
   MainWindow := New(PVoltDialog, Init);
 end;
 {*****************************************************************}
 {------------------VoltDialog Procedures------------------------}
 Constructor VoltDialog.Init;
 begin
   TDlgWindow.Init(nil,Appname);
   New(DMMEnable,InitResource(@Self,id_DMMEnable));
   New(DMMZeroCheck,InitResource(@Self,id_DMMZero));
   New(TenThouScroll,InitResource(@Self,id_VScroll3));
   New(ThouScroll,InitResource(@Self,id_VScroll2));
   New(HundScroll,InitResource(@Self,id_VScroll1));
   New(TenScroll,InitResource(@Self,id_VScroll0));
   New(StatTenThou,InitResource(@Self,id_VStat3,2));
   New(StatThou,InitResource(@Self,id_VStat2,2));
   New(StatHund,InitResource(@Self,id_VStat1,2));
   New(StatTen,InitResource(@Self,id_VStat0,2));
   New(uAmps,InitResource(@Self,id_uAmps,8));
   New(SpelluAmps,InitResource(@Self,id_SpelluAmps,8));
   New(Volts,InitResource(@Self,id_SpellVolts,8));
   New(DMMAmps,InitResource(@Self,id_DMMValue,8));
   {TWindowAttr.Menu := LoadMenu(HInstance, AppMenu);}
 end;

 {*****************************************************************}
 procedure VoltDialog.SetupWindow;
 var
   CString:Text15;
   Result,VoltValue,TenThous,thous,hunds,tens,ReturnValue:integer;
 begin
   TDlgWindow.SetupWindow;
   Result := IDRetry;
   while (SetTimer(HWindow, 0, 250, nil) = 0) and (Result = IDRetry)
 do
     Result := MessageBox(GetFocus,'Could not Create Timer',
 'Spellman',
       mb_RetryCancel);
    if Result = IDCancel then PostQuitMessage(0);
    val(VoltText,VoltValue,ReturnValue);
    Tenthous:=VoltValue div 10000;
    Thous:=(VoltValue-10000*Tenthous) div 1000;
    Hunds:=(VoltValue-10000*Tenthous-1000*Thous) div 100;
    tens:=(VoltValue-10000*Tenthous-1000*thous-100*Hunds) div 10;
    TenThouScroll^.SetRange(0,3);
    ThouScroll^.SetRange(0,9);
    HundScroll^.SetRange(0,9);
    TenScroll^.SetRange(0,9);
    TenThouScroll^.PageMagnitude:=1;
    ThouScroll^.PageMagnitude:=1;
    HundScroll^.PageMagnitude:=1;
    TenScroll^.PageMagnitude:=1;
    TenThouScroll^.SetPosition(tenthous);
    ThouScroll^.SetPosition(thous);
    HundScroll^.SetPosition(hunds);
    TenScroll^.SetPosition(tens);
    Str(TenThous:1,CString);
    StatTenThou^.SetText(CString);
    Str(thous:1,CString);
    StatThou^.SetText(CString);
    Str(Hunds:1,CString);
    StatHund^.SetText(CString);
    Str(Tens:1,CString);
    StatTen^.SetText(CString);

{   DMMEnable^.SetCheck(0);
    DMMZeroCheck^.SetCheck(0);
    SetRange(AUTORANGE);
    Want2Measure(DCCurrent);}
end;
{*****************************************************************}
procedure VoltDialog.GetWindowClass(var AWndClass:TWndClass);
begin
  TDlgWindow.GetWindowClass(AwndClass);
  AWndClass.hIcon:=LoadICon(HInstance,Appname);
{   AWndClass.lpszMenuName:=AppMenu;}
end;
{*****************************************************************}
function VoltDialog.GetClassName:PChar;
begin
 GetClassName:=AppName;
end;
{*****************************************************************}
procedure VoltDialog.UpdateVoltage;
var
  CString: Text15;
  VoltInt:integer;
begin
  VoltInt:=10000*TenThouScroll^.GetPosition
          +1000*ThouScroll^.GetPosition
          +100*HundScroll^.GetPosition
          +10*TenScroll^.GetPosition;
  Str(VoltInt,CString);
  HV_Set(CString);
  StrCopy(VoltText,CString);
end; {UpdateVoltage}
{*****************************************************************}
procedure VoltDialog.HandleTenThousScrollMsg(var Msg: TMessage);
var
```

```
  CString: Text15;
begin
  Str(TenThouScroll^.GetPosition:1, cString);
  StatTenThou^.SetText(CString);
  UpdateVoltage;
end;

{*****************************************************************}
procedure VoltDialog.HandleThousScrollMsg(var Msg: TMessage);
var
  CString: Text15;
begin
  Str(ThouScroll^.GetPosition:1, cString);
  StatThou^.SetText(CString);
  UpdateVoltage;
end;

{*****************************************************************}
procedure VoltDialog.HandleHundScrollMsg(var Msg: TMessage);
var
  CString: Text15;
begin
  Str(HundScroll^.GetPosition:1, cString);
  StatHund^.SetText(CString);
  UpdateVoltage;
end;

{*****************************************************************}
procedure VoltDialog.HandleTenScrollMsg(var Msg: TMessage);
var
  CString: Text15;
begin
  Str(TenScroll^.GetPosition:1, cString);
  StatTen^.SetText(CString);
  UpdateVoltage;
end;
{*****************************************************************}
procedure VoltDialog.HandleDMMEnableMsg(var Msg: TMessage);
begin
   if DMMEnable^.GetCheck=0 then
   begin
     SetRange(RANGE3);  {Can't DMMReadADC anymore if it switches
range on its own}
     Want2Measure(DCVolts{Current});
     Pause(800); {used to be 800}
     DMMEnable^.SetCheck(1);
     ReadDMM:=TRUE;  {as soon as this happens, the Keithley will
start being read}
   end
   else
   begin
     ReadDMM:=FALSE;
     DMMLocal;
     DMMEnable^.SetCheck(0);
   end;
end;
{*****************************************************************}
procedure VoltDialog.HandleDMMZeroCheckMsg(var Msg: TMessage);
begin
   if ReadDMM then   {only need to do something if the DMM is
connected}
```

```
        if DMMZeroCheck^.GetCheck=0 then
        begin
         DMMZeroCheck^.SetCheck(1);
         DMMZero(ZEROON);
        end
        else
        begin
         DMMZeroCheck^.SetCheck(0);
         DMMZero(ZEROOFF);
        end;
 end;
 {****************************************************************}
 procedure VoltDialog.getcurrent(var Msg:TMessage);
    {timer procedure for main window}
 var
   CurrentText     :Text15;
 begin
   if Msg.wParam=0 then
   begin
     StrPCopy(CurrentText,RToStr(DTReadSpellCurrent));
     SpelluAmps^.SetText(CurrentText);
     StrPCopy(CurrentText,RToStr(DTReadCurrent));
     uAmps^.SetText(CurrentText);
     StrPCopy(CurrentText,RToStr(DTReadSpellVoltage));
     Volts^.SetText(CurrentText);
     if ReadDMM then
       begin
         StrPcopy(CurrentText,DMMReadADC);
         DMMAmps^.SetText(CurrentText);
       end; {if checked}
   end;
 end;{getcurrent}
 {****************************************************************}
 procedure VoltDialog.WMDestroy(var Msg: TMessage);
 begin
   KillTimer(HWindow, 0);
   TDlgWindow.WMDestroy(Msg);
 end;
 {****************************************************************}
 procedure VoltDialog.WaveDLG(var Msg:TMessage);
 var
   P:PWaveDialog;
   ReturnValue:integer;
 begin
   {open dialogBox}
   P:=New(PWaveDialog, Init(@Self,'Waveform Dialog'));
   New(SineButt, InitResource(P,id_sine));
   New(SquareButt, InitResource(P,id_square));
   New(TriangleButt, InitResource(P,id_triangle));
   New(PulseButt, InitResource(P,id_pulse));
   New(RandomButt, InitResource(P,id_random));
   New(periodEdit, InitResource(P, id_period,
SizeOf(WaveRecord.periodtxt) ));
   New(upperVEdit, InitResource(P, id_upperV,
SizeOf(WaveRecord.uppervtxt) ));
   New(lowerVEdit, InitResource(P, id_lowerV,
SizeOf(WaveRecord.lowervtxt) ));
   New(SaveData, InitResource(P, id_SaveData) );
   P^.TransferBuffer:=@WaveRecord;
   ReturnValue:=Application^.ExecDialog(P);
   UpDateVoltage;
```

```
end;
{*****************************************************************}
procedure VoltDialog.FreqWaveDLG(var Msg:TMessage);
var
  P:PFreqWaveDialog;
  ReturnValue:integer;
begin
  {open dialogBox}
  P:=New(PFreqWaveDialog, Init(@Self,'Frequency Waveform Dialog'));
  New(SineButt, InitResource(P,id_sine));
  New(SquareButt, InitResource(P,id_square));
  New(TriangleButt, InitResource(P,id_triangle));
  New(PulseButt, InitResource(P,id_pulse));
  New(RandomButt, InitResource(P,id_random));
  New(periodEdit, InitResource(P, id_period,
SizeOf(FreqWaveRecord.periodtxt) ));
  New(upperVEdit, InitResource(P, id_upperV,
SizeOf(FreqWaveRecord.uppervtxt) ));
  New(lowerVEdit, InitResource(P, id_lowerV,
SizeOf(FreqWaveRecord.lowervtxt) ));
  New(SaveData, InitResource(P, id_SaveData) );
  P^.TransferBuffer:=@FreqWaveRecord;
  ReturnValue:=Application^.ExecDialog(P);

end;
{*****************************************************************}
procedure WaveDialog.WaveGenerate(var Msg:TMessage);
var
   DMMReading:extended;
   SpellVoltage,SpellCurrent:double;
   i:integer;
begin
if Msg.wParam=1 then
  begin
   if ReadDMM then
     begin
      val(DMMReadADC,DMMReading,i);
      SpellVoltage:=DTReadSpellVoltage;
      SpellCurrent:=DTReadSpellCurrent;
      if WaveRecord.SaveData then   {write out volts, Spellman
(total) current, Keithley current}

writeln(rawfile,SpellVoltage:7:2,TAB,SpellCurrent:7:2,TAB,{1.0E3*}DMM
Reading:8:4);
      {Draw a dot in the XYDispWindow at the coordinates of
X=SpellVoltage Y=DMMReading}
      PXYDisplay^.Circle:=New(PEllipse,
                          Init(round((SpellVoltage-
XOffset)*XScale),560-round((DMMReading-AOffset)*AScale),
                          round((SpellVoltage-XOffset)*XScale)+2,558-
round((DMMReading-AOffset)*AScale),
                          tc_Tools));
      PXYDisplay^.Port^.Associate(PXYDisplay);
      PXYDisplay^.Picture^.Add(PXYDisplay^.Circle^.Copy);
      PXYDisplay^.Circle^.FastDraw(PXYDisplay^.Port);
      PXYDisplay^.Port^.Dissociate;
     end; {of what to do if reading DMM is enabled}
   inc(waveindex);
   if waveindex>NumWaveSamples then
     begin
```

```
        waveindex:=1;
        if waveform=rand then
           for i:=1 to NumWaveSamples do
                waveformvolts[i]:=random(1024)*abs(upper-
lower)/1024.0+lower;
      end;
    DTDAC_Set(PowersupplyVDAC,waveformvolts[waveindex]);
   end;
end;
{*****************************************************************}
procedure WaveDialog.HandleDataSaveCheck(var Msg: TMessage);
begin
   Transferdata(tf_GetData);
   if ReadDMM then  {only need to save anything when the DMM is
connected}
   if NOT WaveRecord.SaveData then
     begin  {now get a file name to save the data into as text}
       StrCopy(FileName, '*.xls');
       if Application^.ExecDialog(New(PFileDialog,
          Init(@Self, PChar(sd_FileOpen), FileName))) = id_Ok then
         begin  {Got a file name}
           assign(RawFile,Filename);
              {$I-}
           Reset(Rawfile);
           {$I+}
           if IOResult = 0 then         {now append if it exists,
rewrite if not}
              append(Rawfile)
           else
              begin
                rewrite(Rawfile);

writeln(Rawfile,'Volts:',TAB,'Spellman:',TAB,'Keithley:');
              end;
           WaveRecord.SaveData:=TRUE;
         end;  {if got a file name.  If no filename was selected, do
nothing.}
      end {if checked before}
   else
     begin
        close(rawfile);
        WaveRecord.SaveData:=FALSE;
     end; {if NOT checked before}

   Transferdata(tf_SetData);    {Make the dialog controls match the
transfer record}
end;
{*****************************************************************}
procedure WaveDialog.Start(var Msg:TMessage);
var
   volts:double;
   i,waveperiod:integer;
begin
  {ideally, we would set up a timer which would execute
WaveDialog.WaveGenerate}

  waveindex:=1;
  Transferdata(tf_GetData);
  lower:=PCharToR(WaveRecord.lowervtxt)/3000.0;
  upper:=PCharToR(WaveRecord.uppervtxt)/3000.0;
```

```
   with waverecord do
     if sinebutt then waveform:=sine
     else if squarebutt then waveform:=square
     else if trianglebutt then waveform:=triangle
     else if pulsebutt then waveform:=pulse
     else if randombutt then waveform:=rand;

   case waveform of
       sine:for i:=1 to NumWaveSamples do
             waveformvolts[i]:=abs(upper-
lower)*0.5*sin(i*2*pi/numWaveSamples)+abs(upper+lower)*0.5;
     square:begin
             for i:=1 to NumWaveSamples div 2 do
waveformvolts[i]:=upper;
             for i:=(NumWaveSamples div 2) + 1 to NumWaveSamples do
waveformvolts[i]:=lower;
             end;
   triangle:begin
                for i:=1 to NumWaveSamples div 2 do {ramp up}
                    waveformvolts[i]:=(2*i*(upper-
lower)/NumWaveSamples)+lower;
                for i:=(NumWaveSamples div 2) + 1 to NumWaveSamples do
{ramp down}
                    waveformvolts[i]:=waveformvolts[NumWaveSamples + 1
- i];
                end;
      pulse:begin
             for i:=1 to NumWaveSamples div 8 do
waveformvolts[i]:=upper;
             for i:=(NumWaveSamples div 8) + 1 to NumWaveSamples do
waveformvolts[i]:=lower;
             end;
     rand:for i:=1 to NumWaveSamples do
                waveformvolts[i]:=random(1024)*abs(upper-
lower)/1024.0+lower;

  end;{case}
  DTDAC_Set(PowersupplyVDAC,waveformvolts[waveindex]);
  waveperiod:=Round(PCharToR(WaveRecord.periodtxt)*1000.0 /
NumWaveSamples)-32;
  if waveperiod<50 then
    begin
      waveperiod:=(50+32)*NumWaveSamples div 1000;
      Str(waveperiod,Waverecord.periodtxt);
      Transferdata(tf_SetData);
    end;
  if SetTimer(HWindow,1,
              waveperiod,
              nil) = 0 then
     begin
         MessageBox(HWindow, 'Problem getting a timer!', 'Error',
mb_Ok);
         Halt(1);
     end;
end;
{******************************************************************}
procedure WaveDialog.Stop(var Msg:TMessage);
begin
  KillTimer(HWindow,1);
end;
{******************************************************************}
```

```
constructor WaveDialog.Init(AParent: PWindowsObject; ATitle: PChar);
begin
  TDlgWindow.Init(AParent, 'WAVEDLG');
  if ReadDMM then
  begin
          {pop up Spectral Display window}
          PXYDisplay:=PXYDispWindow(
            Application^.MakeWindow(
              new(
              PXYDispWindow, Init(nil, 'XYDisp')
              )
            )
          );
    {initial mapping parameters:}
    XScale:=475.0/10000.0;   {factor changing volts to X-coordinate}
    XOffset:=0.0;                  {pixels}
    AScale:=560.0/10{e-6};    {factor changing amps to Y-coordinate}
    AOffset:=0.0;                {pixels}
  end;
end;
{****************************************************************}
destructor WaveDialog.Done;
begin
  TDlgWindow.Done;
  if ReadDmm then PXYDisplay^.Done;
  if WaveRecord.SaveData then
    begin
      Close(rawfile);
      WaveRecord.SaveData:=FALSE;
    end;
end;
{================================================================
========================}
procedure FreqWaveDialog.WaveGenerate(var Msg:TMessage);
var
   DMMReading,SpellVoltage,SpellCurrent:double;
   i:integer;
begin
if Msg.wParam=1 then
  begin
   if ReadDMM then
     begin
      val(DMMReadADC,DMMReading,i);

{      DMMReading:=DMMReading*1e-6;         {scale the DMMReading,
usually commented out.}

      SpellVoltage:=DTReadSpellVoltage;
      SpellCurrent:=DTReadSpellCurrent;
      if FreqWaveRecord.SaveData then   {write out volts, Spellman
(total) current, Keithley current}
            writeln(rawfile,waveformfreq[waveindex],TAB,
                   SpellVoltage:7:2,TAB,
                   SpellCurrent:7:2,TAB,
                   1.0E6*DMMReading:8:4);
      {Draw a dot in the XYDispWindow at the coordinates of
X=SpellVoltage Y=DMMReading}
      PXYDisplay^.Circle:=New(PEllipse,
                            Init(round((waveformfreq[waveindex]-
XOffset)*XScale),560-round((DMMReading-AOffset)*AScale),
                        round((waveformfreq[waveindex]-
XOffset)*XScale)+2,558-round((DMMReading-AOffset)*AScale),
                        tc_Tools));
      PXYDisplay^.Port^.Associate(PXYDisplay);
      PXYDisplay^.Picture^.Add(PXYDisplay^.Circle^.Copy);
      PXYDisplay^.Circle^.FastDraw(PXYDisplay^.Port);
      PXYDisplay^.Port^.Dissociate;
     end; {of what to do if DMM is turned connected}
    inc(waveindex);
    if waveindex>NumWaveSamples then
     begin
       waveindex:=1;
       if waveform=rand then
         begin
          randomize;
          for i:=1 to NumWaveSamples do
            waveformfreq[i]:=random(1024)*abs(upper-
lower)/1024.0+lower;
         end;
     end;
   FreqSet(waveformfreq[waveindex]);
  end;
end;
{*****************************************************************}
procedure FreqWaveDialog.HandleDataSaveCheck(var Msg: TMessage);
begin
   Transferdata(tf_GetData);
   if ReadDMM then  {only need to save anything when the DMM is
connected}
   if NOT FreqWaveRecord.SaveData then
     begin  {now get a file name to save the data into as text}
       StrCopy(FileName, '*.xls');
       if Application^.ExecDialog(New(PFileDialog,
          Init(@Self, PChar(sd_FileOpen), FileName))) = id_Ok then
         begin  {Got a file name}
           assign(RawFile,Filename);
              {$I-}
           Reset(Rawfile);
           {$I+}
           if IOResult = 0 then         {now append if it exists,
rewrite if not}
              append(Rawfile)
           else
              begin
                rewrite(Rawfile);

writeln(Rawfile,'Hz:',TAB,'Volts:',TAB,'Spellman:',TAB,'Keithley:');
              end;
           FreqWaveRecord.SaveData:=TRUE;
         end;  {if got a file name.  If no filename was selected, do
nothing.}
      end {if checked before}
   else
     begin
       close(rawfile);
       FreqWaveRecord.SaveData:=FALSE;
     end; {if NOT checked before}

   Transferdata(tf_SetData);    {Make the dialog controls match the
transfer record}
end;
```

```
{*****************************************************************}
procedure FreqWaveDialog.Start(var Msg:TMessage);
var
    volts:double;
    i,waveperiod:integer;
begin
   {ideally, we would set up a timer which would execute
FreqWaveDialog.WaveGenerate}

   waveindex:=1;
   Transferdata(tf_GetData);
   lower:=PCharToR(FreqWaveRecord.lowervtxt);
   upper:=PCharToR(FreqWaveRecord.uppervtxt);

   with FreqWaveRecord do
     if sinebutt then waveform:=sine
     else if squarebutt then waveform:=square
     else if trianglebutt then waveform:=triangle
     else if pulsebutt then waveform:=pulse
     else if randombutt then waveform:=rand;

   case waveform of
       sine:for i:=1 to NumWaveSamples do
             waveformfreq[i]:=abs(upper-
lower)*0.5*sin(i*2*pi/numWaveSamples)+abs(upper+lower)*0.5;
     square:begin
             for i:=1 to NumWaveSamples div 2 do
waveformfreq[i]:=upper;
             for i:=(NumWaveSamples div 2) + 1 to NumWaveSamples do
waveformfreq[i]:=lower;
             end;
  triangle:begin
               for i:=1 to NumWaveSamples div 2 do {ramp up}
                   waveformfreq[i]:=(2*i*(upper-
lower)/NumWaveSamples)+lower;
               for i:=(NumWaveSamples div 2) + 1 to NumWaveSamples do
{ramp down}
                   waveformfreq[i]:=waveformfreq[NumWaveSamples + 1 -
i];
               end;
      pulse:begin
              for i:=1 to NumWaveSamples div 8 do
waveformfreq[i]:=upper;
              for i:=(NumWaveSamples div 8) + 1 to NumWaveSamples do
waveformfreq[i]:=lower;
             end;
       rand:begin
              randomize;
              for i:=1 to NumWaveSamples do
                waveformfreq[i]:=random(1024)*abs(upper-
lower)/1024.0+lower;
             end;
  end;{case}
  FreqSet(waveformfreq[waveindex]);
  waveperiod:=Round(PCharToR(FreqWaveRecord.periodtxt)*1000.0 /
NumWaveSamples)-32;
  if waveperiod<50 then
    begin
      waveperiod:=(50+32)*NumWaveSamples div 1000;
      Str(waveperiod,FreqWaverecord.periodtxt);
      Transferdata(tf_SetData);
      end;
    if SetTimer(HWindow,1,
               waveperiod,
               nil) = 0 then
      begin
          MessageBox(HWindow, 'Problem getting a timer!', 'Error',
mb_Ok);
          Halt(1);
      end;
end;
{*****************************************************************}
procedure FreqWaveDialog.Stop(var Msg:TMessage);
begin
  KillTimer(HWindow,1);
end;
{*****************************************************************}
constructor FreqWaveDialog.Init(AParent: PWindowsObject; ATitle:
PChar);
begin
  TDlgWindow.Init(AParent, 'FREQWAVEDLG');
  SynthRemote;
  if ReadDMM then
  begin
          {pop up Spectral Display window}
          PXYDisplay:=PXYDispWindow(
            Application^.MakeWindow(
              new(
              PXYDispWindow, Init(nil, 'XYDisp')
              )
            )
          );
    {initial mapping parameters:}
    XScale:=475.0/300000.0;   {factor changing volts to X-coordinate}
    XOffset:=0.0;                   {pixels}
    AScale:=560.0/10;{0e-6;    {factor changing amps to Y-coordinate}
    AOffset:=0.0;                  {pixels}
  end;
end;
{*****************************************************************}
destructor FreqWaveDialog.Done;
begin
  TDlgWindow.Done;
  SynthLocal;
  if ReadDmm then PXYDisplay^.Done;
  if FreqWaveRecord.SaveData then
    begin
      Close(rawfile);
      FreqWaveRecord.SaveData:=FALSE;
    end;
end;
{*****************************************************************}

{ ---------------------------------XYDispDialog methods-------------
------------ }
constructor XYDispWindow.init(AParent: PWindowsObject;
  ATitle: PChar);
var
   window:HWnd;
   i:integer;
begin
    TGWindow.Init(AParent, ATitle);
```

```
     Brush := New(PBrush, InitDefault);
     Brush^.Pattern := bp_Solid;
     Brush^.Color^.SetRGB(128,128,128);

     Attr.Style := ws_Popup or ws_OverlappedWindow or ws_Visible;
     Attr.X:=325;
     Attr.Y:=0;
     Attr.W:=475;
     Attr.H:=600;
{     Attr.Menu := LoadMenu(HInstance, 'DISP_MENU');}
     Fence := New(PChooser, Init(0, 0));      {Contains PGPOINT fields
ORIGIN and CORNER}
     XYDisplayOpen:=TRUE;
end;
{*****************************************************************}
destructor XYDispWindow.Done;
begin
    Dispose(Fence,Done);
    TGWindow.Done;
    XYDisplayOpen:=FALSE;
end;
{*****************************************************************}
procedure XYDispWindow.Update;
    {redraws the spectral frequency display}
var
   i:integer;
begin
   Port^.Associate(@Self);

   {Really doesn't doo anything, yet.}

   port^.Disscciate;
end;{Update}
{*****************************************************************}
procedure XYDispWindow.SetupWindow;
begin
  TGWindow.SetupWindow;
  SetGCursor(cs_Graphic);
end;
{*****************************************************************}
procedure XYDispWindow.BeginDrag(MousePt: PGPoint; KeyStates: Word);
begin
  Port^.Associate(@Self);
  Fence^.SetOrigin(MousePt);
  Fence^.SetCorner(MousePt);
  Fence^.Draw(Port);
end;
{*****************************************************************}
procedure XYDispWindow.Drag(MousePt: PGPoint; KeyStates: Word);
begin
  Fence^.FastDraw(Port);
  Fence^.SetCorner(MousePt);
  Fence^.FastDraw(Port);
end;
{*****************************************************************}
procedure XYDispWindow.EndDrag(MousePt: PGPoint; KeyStates: Word);
var
  i,PkPos,XStart,XEnd,YStart,Yend:integer;
  txtmessage:PChar;
begin
  Fence^.Draw(Port);
```

```
   XStart:=Fence^.Origin^.X;
   if XStart<1 then XStart:=1;
   XEnd:=Fence^.Corner^.X;
   YStart:=Fence^.Origin^.Y;
   if YStart<1 then YStart:=1;
   YEnd:=Fence^.Corner^.Y;
   if (abs(XStart-XEnd)>10) and (abs(YStart-YEnd)>10) then
   begin   {only recalculate if the selection box is bigger than 10X10
pixels}
     {recalculate the mapping parameters}
     if XStart<=XEnd then XOffset:=XStart/XScale+XOffset
     else  XOffset:=XEnd/XScale+XOffset;
     XScale:=475.0*XScale/ABS(XEnd-XStart);   {factor changing volts
or freq. to X-coordinate}

     if YStart>=YEnd then AOffset:=(560-YStart)/AScale+AOffset
{lowest current=highest Y value}
     else  AOffset:=(560-YEnd)/AScale+AOffset;     {sets AOffset to
actual amps}
     AScale:=560.0*AScale/ABS(YEnd-YStart);   {factor changing amps to
Y-coordinate}

     {clear the screen}
     Picture^.FreeAll;
     Invalidate;
     port^.Dissociate;
     GetMem(txtmessage,128);
     FreeMem(txtmessage,128);
   end;
end;
{*********************************************************************}
var
  LDVApp : CTRLApplication;

begin
  LDVApp.Init(AppName);
  LDVApp.Run;
  LDVApp.Done;
end.
```

UNIT LDVWGLB5;

```
{ contains the global declarations for LDVW }
{$F+}
interface
uses
  ez488w
  ,WinTypes,WinProcs,Strings,
  MYString,
  Win31;

const
  TekBuffSize = 8192; {bytes}
  HaveData:boolean = FALSE;
  CrLf = #13#10;
  TAB  = #9;
  CR = #13;
  ON = TRUE;
  Forward=TRUE;
  OFF= FALSE;
  Reverse=FALSE;


type
  RData_array=array[1..(TekBuffSize div 2)+1] of double;
  PRData_array=^RData_array;

  intbuff = array [0..TekBuffSize-1] of integer;
  tekbuff = array[0..TekBuffSize-1] of byte;
  Str255  = string[255];
  Text15 = array[0..15] of char;
  PumpXferRecord = record
    running,stopped,forward,reverse:bool; {radio buttons}
    Flow:Text15;
    end;
  PosXferRecord = record
    x,y:text15;
    end;
  ScopeXferRecord = record
    DC,AC,Grnd,Local:bool;{3 radio buttons + checkbox}
    end;
  FFTXferRecord = record
    single,multi:bool;{2 radio buttons}
    avgNum:Text15;
    end;
const
  VoltText:Text15 = '0';
  PumpRecord:PumpXferRecord = (
    running:False;
    stopped:True;
    Forward:True;
    Reverse:False;
    Flow:'2.00');
  PositionRecord:PosXferRecord = (
    x:'0.0000';
    y:'0.0000');
  ScopeRecord:ScopeXferRecord = (
    DC:False;
    AC:True;
```

```
     Grnd:False;
     local:False);
   SpectrumRecord:FFTXferRecord = (
     single:False;
     multi:TRUE;
     avgNum:'16');
   SpecDisplayOpen:bool = FALSE;

var
   devicedata,Response,Reading: Str255;
   i,msg            :INTEGER;
   PScopeData              :^TekBuff;
   TekBuffSeg,
   TekBuffOfs,             {Real mode address elements}
   TekBuffSelector :word; {Protected mode address}
   IntBuffSize      :integer; {points from the Scope}
   DataArray        :intbuff;
   PRealData,PImagData,PAvgSpectrum:PRData_array;
   FlowRate         :single; {in uL/min}
   PosX,PosY        :double; {in um}
   CurrentText      :Text15;
   BigBuff                 :Boolean; {used to indicate whether to
collect 1024 or 4096 pts.}
   Window           :HWnd;

{*******************************************************************}

Implementation

VAR
   ExitSave : pointer;    {saves the old exit routine pointer}
{*******************************************************************}
Function GetRealBuff(size:word;var
BuffSeg,BuffOfs,selector:word):pointer;
var
   GDAValue,
   SelectorBase,
   SelectorLimit:longint;
   Para_seg:word;

begin
{now get some REAL MODE memory for a DMA buffer.  Must not cross a
page boundary!}

   GDAValue :=GlobalDosAlloc(size);
   if GDAValue <>0 then
      begin
         GlobalLock(THandle(GDAValue));
         Para_seg := Hiword(GDAValue);   {"paragraph-segment value"}
         selector := Loword(GdaValue);
            SelectorBase:=GetSelectorBase(selector);
            SelectorLimit:=GetSelectorLimit(selector);

            { assign the real-mode address elements for the DMA
driver}
         BuffSeg:=SelectorBase SHR 4;
         BuffOfs:=(SelectorBase AND $0F);
         GetRealBuff:=ptr(Selector,0);
      end
   else  { allocation failed}
      halt(1);
```

```
{     writeln('Could not allocate memory for the DMA buffer.');}
end; {GetRealBuff}

{*****************************************************************
}
Procedure ExitGlob;
begin
  ExitProc:=ExitSave;
  dispose(PRealData);dispose(PImagData);
  GlobalDosFree(TekBuffSelector);
{$ifdef debug}
  msg:=MessageBox(Window,'Exit procedure from LDVWGlob has run.',
       'EXITGLOB',mb_OK);
{$endif}
end;
{*****************************************************************
}

var
   MsgStr:PChar;
begin
  repeat
        { get a DMA buffer:
            - in the first megabyte of memory
            - with a known REAL MODE (physical) address
            - ensuring it does not cross a page boundary}
     PScopeData:=GetRealBuff(TekBuffSize+500,TekBuffSeg,TekBuffOfs,
                             TekBuffSelector);

        {if the offset + size crosses a segment boundary, it will wrap
around so...}
     if (TekBuffOfs > word(TekBuffOfs+TekBuffSize)) then
        begin
          msg:=MessageBox(Window,'Need to try for another DMA
buffer...',
               'LDVWGLOB - init',mb_OK);
          GlobalDosFree(TekBuffSelector);
        end
        else
        begin

{          GetMem(MsgStr,128);
          StrPCopy(MsgStr,IToStr(TekBuffSeg)+':'+IToStr(TekBuffOfs));
          msg:=MessageBox(Window,MsgStr,
              'DMA Buffer location address:',mb_OK);
            FreeMem(MsgStr,128)   }

        end;
  until (TekBuffOfs <= word(TekBuffOfs+TekBuffSize));

{  FreeMem(MsgStr,128);}

  for i:=0 to TekBuffSize-1 do PScopeData^[i]:=$F0; {initialize in a
disctinctive way}
  new(PRealData);new(PImagData);new(PAvgSpectrum);
  ExitSave:=ExitProc;
  ExitProc:=@ExitGlob;
End.
```

PROGRAM LDVW5;

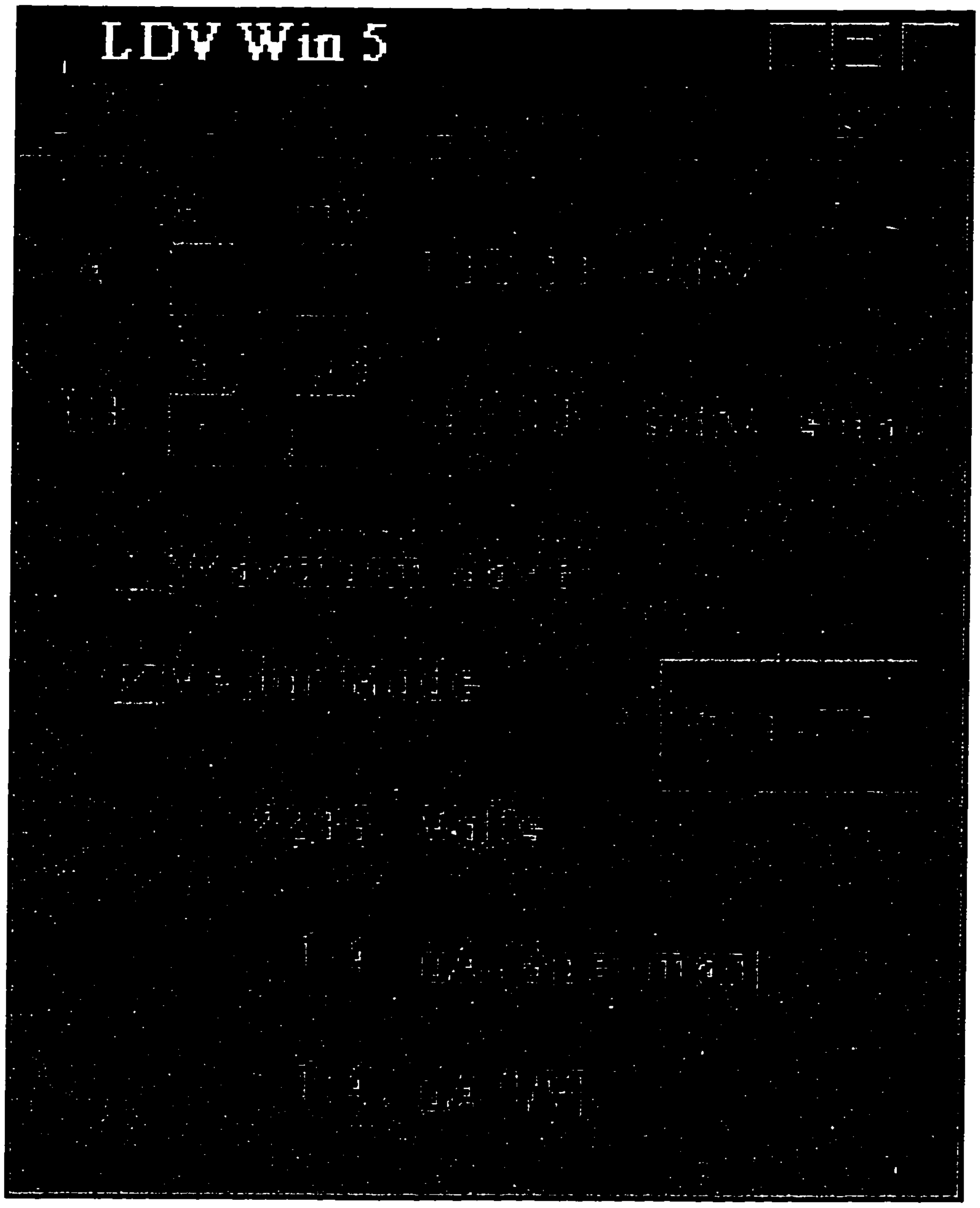

Figure A.5 User Interface for LDVW5 Application.

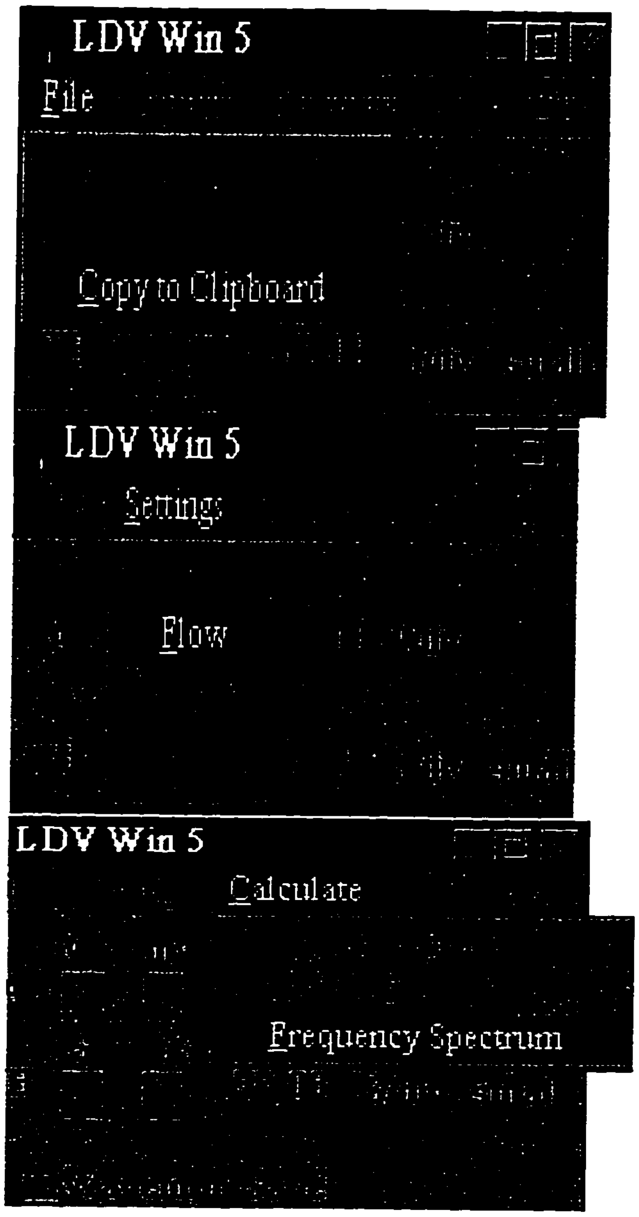

Figure A.6 LDVW5 Application Menus.

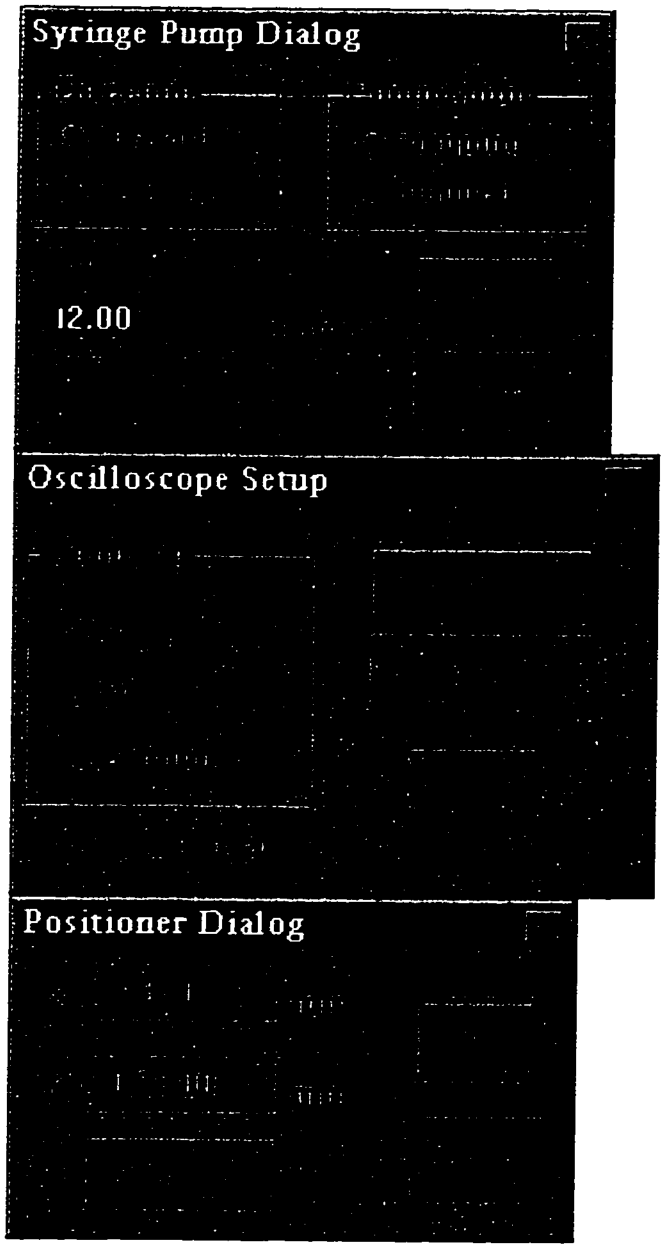

Figure A.7 LDVW5 application dialogs for controlling the pump, oscilloscope and positioners.

program LDVW5;

```
{This application controls the X and Y position, syringe pump, Data
Translation card, and Philips digitizing oscilloscope. Parameters can
be read from the clipboard and set into the instruments. Waveforms
and other measured values can be placed onto the clipboard.}
{$R ldvwin5}    {resources}
{$X+}
{$F+}
{$M 16000,20000}
uses
  WinTypes, WinProcs, Strings, StdWnds,WObjects,
  WinDos,
  ldvwglb5,    {global variable and constant declarations}
  ez488w,timer,
  DataAcq5,
  TPWFFT,
  nrc855c5,   {Handles XY positioner}
  PumpHnd5,   {Communications routines for the syringe pump}
  DT2827w5,    {Data Translation routines}
  MyString,
  OGL1, OGL2, OGL3,  {Whitewater Object Graphics units}
  StdDlgs;

const
  AppName = 'ldvwin5';
  AppMenu = 'ldvmenu';
  AppAccel= 'ldvaccel';
  id_About = 100;    {ldv menu IDs}
  id_TSave = 101;
  id_BSave = 102;
  id_Copy  = 103;
  id_Volts = 104;
  id_pump  = 105;
  id_position = 106;
  id_avg  = 107;
  id_ch3  = 108;
  id_Scope= 109;
  id_FSpec= 130;

  DispMenu= 'disp_menu';
  id_specSave=201;    {Spectrum Display menu IDs}
  id_specCopy=202;
                      {main dialog IDs}
  id_TekSave = 110;
  id_TekVec  = 111;
  id_TekAcq  = 112;
  id_msdiv   = 113;
  id_uAmps   = 114;
  id_VDiv    = 115;
  id_SpelluAmps = 116;
  id_SpellVolts   = 117;
  id_decrVDiv= 118;
  id_incrVDiv= 119;
  id_decrSDiv= 120;
  id_incrSDiv= 121;
  id_BuffSize= 211;

  id_PumpOn  = 105;  {FlowDlg IDs}
  id_PumpOff  = 106;
  id_PumpForward = 107;
```

```
   id_PumpReverse = 108;
   id_PumpFlow  = 109;
   id_PumpGrp1  = 110;
   id_PumpGrp2  = 111;
   id_PositionX = 120; {PosDlg IDs}
   id_PositionY = 121;
   id_Home = 3;
   id_AC = 130; {Scopsetup IDs}
   id_DC = 131;
   id_GRND=132;
   id_CouplGrp=133;
   id_local=134;
   id_Reset = 3;
   id_snglEvent=101;  {FFTSetup IDs}
   id_avgEvents=102;
   id_avgNumber=103;
   id_VScroll3 = 206;  {VoltDialog IDs}
   id_VScroll2 = 207;
   id_VScroll1 = 208;
   id_VScroll0 = 209;
   id_VStat3 = 210;
   id_VStat2 = 211;
   id_VStat1 = 212;
   id_VStat0 = 213;

var
  Hour0,Hour1,Hour2,Hour3,Hour4,Hour5,Hour6,Hour7,Hour8:^HCursor;
  Waitcursor,newcursor:hcursor;

type
  PldvWindow = ^LDVWindow;
  LDVWindow = object(TDlgWindow)
    TEKSave,TekVec: PCheckBox;
    Abutt: PButton;
    uAmps,SpelluAmps,Volts,misc,VDiv,msdiv,BuffSize:PStatic;
    FileName: array[0..fsPathName] of Char;
    procedure SetupWindow; virtual;
    constructor Init;
    procedure GetWindowClass(var AWndClass:TWndClass);virtual;
    function GetClassName:PChar; virtual;
    procedure WMDestroy(var Msg: TMessage);
      virtual wm_First + wm_Destroy;
    procedure HandleTekAcqMsg(var Msg: TMessage);
      virtual id_First + id_TekAcq;
    procedure HandleTEKSaveMsg(var Msg: TMessage);
      virtual id_First + id_TEKSave;
    procedure HandleTEKVecMsg(var Msg: TMessage);
      virtual id_First + id_TEKVec;
    procedure FillClipBrd(var Msg: TMessage);
      virtual cm_First + id_Copy;
    procedure TSaveData(var Msg: TMessage);
      virtual cm_First + id_TSave;
    procedure BSaveData(var Msg: TMessage);
      virtual cm_First + id_BSave;
    procedure SetVoltage(var Msg: TMessage);
      virtual cm_First + id_Volts;
    procedure SetFlow(var Msg: TMessage);
      virtual cm_First + id_Pump;
    procedure SetPosition(var Msg: TMessage);
      virtual cm_First + id_Position;
    procedure SetScope(var Msg: TMessage);
```

```
        virtual cm_First + id_Scope;
      procedure AverageSignal(var Msg: TMessage);
        virtual cm_First + id_avg;
      procedure Ch3Signal(var Msg: TMessage);
        virtual cm_First + id_ch3;
      procedure About(var Msg: TMessage);
        virtual cm_First + id_About;
      procedure incrHTB(var Msg: TMessage);
        virtual id_First + id_incrSDiv;
      procedure decrHTB(var Msg: TMessage);
        virtual id_First + id_decrSDiv;
      procedure incrVDiv(var Msg: TMessage);
        virtual id_First + id_incrVDiv;
      procedure decrVDiv(var Msg: TMessage);
        virtual id_First + id_decrVDiv;
      procedure getcurrent(var Msg:TMessage);
        virtual wm_first + wm_timer;
      procedure FFTSetup(var Msg:TMessage);
        virtual cm_first + id_FSpec;
  end; {LDVWindow}

  PFTDispWindow = ^FTDispWindow;
  FTDispWindow = object(TGWindow)  {OGL-Descendent of TWindow}
    FileName: array[0..fsPathName] of Char;
    ThePlot: PPolyLine;
    TheVertices:PPointColl;
    APoint:PGPoint;
    Fence:PChooser;

    constructor Init(AParent: PWindowsObject; ATitle: PChar);
    destructor Done; virtual;

    procedure BeginDrag(MousePt: PGPoint; KeyStates: Word); virtual;
    procedure Drag(MousePt: PGPoint; KeyStates: Word); virtual;
    procedure EndDrag(MousePt: PGPoint; KeyStates: Word); virtual;

{    procedure CMColor(var Msg: TMessage);virtual cm_First +
cm_Color;
    procedure CMWidth(var Msg: TMessage);virtual cm_First + cm_Width;
}

    procedure Update; virtual;
    procedure SetupWindow; virtual;
    procedure FTSave(var Msg: TMessage); virtual cm_First +
id_SpecSave;
    procedure FTCopy(var Msg: TMessage); virtual cm_First +
id_SpecCopy;
  end;{FTDispWindow}

var
  PFTDisplay:PFTDispWindow;

type
  PPosDialog = ^PosDialog;
  PosDialog = object(TDialog)
    procedure Home(var Msg:TMsg); virtual id_first + id_Home;
  end;

  PScopeDialog = ^ScopeDialog;
  ScopeDialog = object(TDialog)
    procedure Reset(var Msg:TMsg); virtual id_first + id_Reset;
```

```
   end;

   PVoltDialog = ^VoltDialog;
   VoltDialog = object(TDlgWindow)
     TenThouScroll,ThouScroll,HundScroll,TenScroll : PScrollBar;
     StatTenThou,StatThou,StatHund,StatTen : PStatic;
     procedure SetupWindow; virtual;
     constructor Init;
     procedure GetWindowClass(var AWndClass:TWndClass);virtual;
     function GetClassName:PChar; virtual;
     procedure UpdateVoltage;
     procedure HandleTenThousScrollMsg(var Msg: TMessage);
       virtual id_First + id_VScroll3;
     procedure HandleThousScrollMsg(var Msg: TMessage);
       virtual id_First + id_VScroll2;
     procedure HandleHundScrollMsg(var Msg: TMessage);
       virtual id_First + id_VScroll1;
     procedure HandleTenScrollMsg(var Msg: TMessage);
       virtual id_First + id_VScroll0;
   end;

   LDVApplication = object(TApplication)
     procedure InitMainWindow; virtual;
     procedure InitInstance; virtual;
     destructor Done; virtual;
   end;

{ -----------LDVApplication Method----------------------- }
procedure LDVApplication.InitMainWindow;
begin
  MainWindow := New(PldvWindow, Init);
  New(Hour0);
  New(Hour1);
  New(Hour2);
  New(Hour3);
  New(Hour4);
  New(Hour5);
  New(Hour6);
  New(Hour7);
  New(Hour8);
  Hour0^:=LoadCursor(Hinstance,'HOUR0');
  Hour1^:=LoadCursor(Hinstance,'HOUR1');
  Hour2^:=LoadCursor(Hinstance,'HOUR2');
  Hour3^:=LoadCursor(Hinstance,'HOUR3');
  Hour4^:=LoadCursor(Hinstance,'HOUR4');
  Hour5^:=LoadCursor(Hinstance,'HOUR5');
  Hour6^:=LoadCursor(Hinstance,'HOUR6');
  Hour7^:=LoadCursor(Hinstance,'HOUR7');
  Hour8^:=LoadCursor(Hinstance,'HOUR8');
end;
{************************************************************}
procedure LDVApplication.InitInstance;
begin
 TApplication.InitInstance;
 HAccTable := LoadAccelerators(HInstance, AppAccel);
end;
{************************************************************}
destructor LDVApplication.Done;
begin
  {shut off the syringe pump}
  TurnPump(OFF,PumpRecord.Forward);
```

```
    Dispose(Hour0);
    Dispose(Hour1);
    Dispose(Hour2);
    Dispose(Hour3);
    Dispose(Hour4);
    Dispose(Hour5);
    Dispose(Hour6);
    Dispose(Hour7);
    Dispose(Hour8);
 end;

 { ---------------------------------PPosDialog method---------------
 -- }
 procedure PosDialog.Home(var Msg:TMsg);
 begin
   if TDialog.CanClose then
   begin
     TDialog.EndDlg(id_Home);
     TDialog.TransferData(tf_GetData);
   end;
 end;
 { ---------------------------------PScopeDialog method-------------
 -- }
 procedure ScopeDialog.Reset(var Msg:TMsg);
   {pressing the RESET button closes the dialog and returns id_Reset}
 begin
     TDialog.EndDlg(id_Reset);
 end;


 { ---------------------------------FTDispDialog methods------------
 -- }
 constructor FTDispWindow.init(AParent: PWindowsObject;
   ATitle: PChar);
 var
    window:HWnd;
    i:integer;
 begin
     TGWindow.Init(AParent, ATitle);
     Attr.Style := ws_Popup or ws_OverlappedWindow or ws_Visible;
     Attr.X:=250;
     Attr.Y:=0;
     Attr.W:=550;
     Attr.H:=600;
     Attr.Menu := LoadMenu(HInstance, 'DISP_MENU');
     Fence := New(PChooser, Init(0, 0));      {Contains PGPOINT fields
ORIGIN and CORNER}
     ThePlot := New(PPolyline, Init(tc_Tools));
     SpecDisplayOpen:=TRUE;
     StrCopy(FileName, '*.dat');
 end;
{***************************************************************}
destructor FTDispWindow.Done;
begin
    Dispose(ThePlot,Done);
    Dispose(Fence,Done);
    TGWindow.Done;
    SpecDisplayOpen:=FALSE;
end;
{***************************************************************}
procedure FTDispWindow.Update;
```

```
     {redraws the spectral frequency display}
 var
    i:integer;
 begin
    Port^.Associate(@Self);

    {Clear the old picture}
    Picture^.FreeAll;

    {plot the frequency spectrum}
    TheVertices:=New(PPointColl, Init(10,10));
    APoint:=New(PGPoint, InitDefault);
    for i:=1 to 512 do
       begin
         APoint^.Build(i+16,round(540-PAvgSpectrum^[i]/32));
         TheVertices^.Insert(APoint^.Copy)
       end;
    Dispose(APoint, Done);
    ThePlot^.SetVertices(TheVertices);
    ThePlot^.Draw(Port);
    Dispose(TheVertices, Done);
    Picture^.Add(ThePlot^.Copy);
    Invalidate;
    port^.Dissociate;
end;{Update}
{*****************************************************************}
procedure FTDispWindow.SetupWindow;
begin
  TGWindow.SetupWindow;
  SetGCursor(cs_Graphic);
end;
{*****************************************************************}
procedure FTDispWindow.BeginDrag(MousePt: PGPoint; KeyStates: Word);
begin
  Port^.Associate(@Self);
  Fence^.SetOrigin(MousePt);
  Fence^.SetCorner(MousePt);
  Fence^.Draw(Port);
end;
{*****************************************************************}
procedure FTDispWindow.Drag(MousePt: PGPoint; KeyStates: Word);
begin
  Fence^.FastDraw(Port);
  Fence^.SetCorner(MousePt);
  Fence^.FastDraw(Port);
end;
{*****************************************************************}
procedure FTDispWindow.EndDrag(MousePt: PGPoint; KeyStates: Word);
var
  i,PkPos,XStart,XEnd:integer;
  PkHt:double;
  Freq:PChar;
begin
  Fence^.Draw(Port);
  XStart:=Fence^.Origin^.X-15;
  if XStart<1 then XStart:=1;
  XEnd:=Fence^.Corner^.X-15;
  if XEnd>512 then XEnd:=512;
  PkHt:=0;
  For i:=XStart to XEnd do
    if PAvgSpectrum^[i]>PkHt then
```

```
     begin
       PkHt:=PAvgSpectrum^[i];
       PkPos:=i;
     end;
   port^.Dissociate;
   GetMem(Freq,128);          {N                   T
 P}

 StrPCopy(Freq,RToStr(FTFreq(1024,PCharToR(HorTimeBase[HTBIndex])/0.10
 24,PkPos))+' kHz');
   MessageBox(Window,Freq,'Selected Peak Frequency',mb_OK);
   FreeMem(Freq,128);
 end;
 {***************************************************************}
 procedure FTDispWindow.FTSave(var Msg: TMessage);
 var
   Rawfile:text;
   GlobalHandle:THandle;
   i:integer;
   temp:string;
   tmpstr:array[0..25] of char;
 begin
   {now get a file name and save the spectral data out as text}
   if Application^.ExecDialog(New(PFileDialog,
       Init(@Self, PChar(sd_FileSave), FileName))) = id_Ok then
 {sd_FileSave from StdDlgs}
    begin
      SetCursor(Hour0^);
      assign(RawFile,Filename);
      rewrite(rawfile);

      SetCursor(Hour1^);

 {     msdiv^.GetText(tmpstr,15);
      writeln(rawfile,tmpstr,' ms/div');
      Volts^.GetText(tmpstr,15);
      writeln(rawfile,tmpstr,' V');
      SetCursor(Hour2^);
      uAmps^.GetText(tmpstr,15);
      writeln(rawfile,tmpstr,' uA');

      writeln(rawfile,PumpRecord.Flow,' uL/min');   }

      SetCursor(Hour3^);
      writeln(rawfile,'X-position: ',TAB,PositionRecord.X,TAB,' mm');
      writeln(rawfile,'Y-position: ',TAB,PositionRecord.Y,TAB,' mm');

 {     Vdiv^.GetText(tmpstr,15);
      writeln(rawfile,tmpstr,' V/div');  }

      writeln(Rawfile,'kHz',TAB,'Intensity');
      SetCursor(Hour8^);
      for i:=1 to 512 do writeln(rawfile,

FTFreq(1024,PCharToR(HorTimeBase[HTBIndex])/0.1024,i):9,
                 TAB,
                 PAvgSpectrum^[i]:8);
     SetCursor(Hour8^);
     close(rawfile);
     SetCursor(NewCursor);
   end;
 end;


 {****************************************************************}
 procedure FTDispWindow.FTCopy(var Msg: TMessage);
 begin
   MessageBox(Window,'Not yet implemented.','APOLOGY',mb_OK);
 end;
 {****************************************************************}

 { ---------------------------------LDVWindow methods----------------
 -- }
 constructor LDVWindow.Init;
 begin
   TDlgWindow.Init(nil, AppName);
   TekSave:=New(PCheckBox,InitResource(@Self,id_TekSave));
   TekVec :=New(PCheckBox,InitResource(@Self,id_TekVec));
   msDiv  :=New(PStatic,InitResource(@Self,id_msDiv,6));
   VDiv   :=New(PStatic,InitResource(@Self,id_VDiv,6));
   AButt  :=New(PButton,InitResource(@Self,id_TekAcq));
   uAmps  :=New(PStatic,InitResource(@Self,id_uAmps,8));
   SpelluAmps  :=New(PStatic,InitResource(@Self,id_SpelluAmps,8));
   Volts  :=New(PStatic,InitResource(@Self,id_SpellVolts,8));
   BuffSize    :=New(PStatic,InitResource(@Self,id_BuffSize,8));
 end;
 {****************************************************************}
 procedure LDVWindow.GetWindowClass(var AWndClass:TWndClass);
 begin
   TDlgWindow.GetWindowClass(AWndClass);
   NewCursor:=LoadCursor(HInstance,'CURSOR');
   AWndClass.hCursor:=NewCursor;
   AWndClass.hIcon:=LoadICon(HInstance,'LDVICON');
 end;
 {****************************************************************}
 function LDVWindow.GetClassName:PChar;
 begin
   GetClassName:='LDVWindow';
 end;
 {****************************************************************}
 procedure LDVWindow.SetupWindow;
 var
   Result: Integer;
 begin
   TDlgWindow.SetupWindow;
   Result := IDRetry;
   while (SetTimer(HWindow, 0, 500, nil) = 0) and (Result = IDRetry)
 do
     Result := MessageBox(GetFocus,'Could not Create Timer', 'LDVW3',
       mb_RetryCancel);
   if Result = IDCancel then PostQuitMessage(0);
   MSdiv^.SetText(HorTimeBase[HTBIndex]);
   VDiv^.SetText(VerSens[VertSensIndex]);
   if BigBuff then BuffSize^.SetText('BIG')
   else BuffSize^.SetText('small');
   SetVectorMode(ON);
   TekVec^.SetCheck(1);
 end;
 {****************************************************************}
 procedure LDVWindow.WMDestroy(var Msg: TMessage);
 begin
```

```
  KillTimer(HWindow, 0);
  TDlgWindow.WMDestroy(Msg);
end;
{****************************************************************}
procedure LDVWindow.HandleTekAcqMsg(var Msg:TMessage);
var
  N:integer;
begin
{   FFTXferRecord = record
    single,multi:bool;     <----- 2 radio buttons
    avgNum:Text15;         <----- edit field
    end;          }
if SpectrumRecord.single then
  begin
    HaveData:=FALSE;
    WaveFormAcquire;
    TekSave^.SetCheck(1);
  end
else
  begin {Acquire SpectrumRecord.avgNum spectra}
    TekSave^.SetCheck(1);
    if SpecDisplayOpen then SetFocus(PFTDisplay^.HWindow);
    SpecClear(PAvgSpectrum,512);
    N:=PCharToI(SpectrumRecord.avgNum);
    for i:=1 to N do
      begin
        WaveFormAcquire;
        GetData(DataArray);
        Bfft(PRealData,PImagData,512);
        Amplitude(PRealData,PImagData,512);
        CoAdd(PRealData,PAvgSpectrum,512,i);
        if SpecDisplayOpen then
            PFTDisplay^.Update;
      end;
  end;
end;
{****************************************************************}
procedure LDVWindow.HandleTekSaveMsg(var Msg:TMessage);
begin
HaveData:=FALSE;
if TekSave^.GetCheck = 0 then
  begin
    SetSaveMode(ON);
    TekSave^.SetCheck(1);
  end
else
  begin
    SetSaveMode(OFF);
    TekSave^.SetCheck(0);
  end;
end;
{****************************************************************}
procedure LDVWindow.HandleTekVecMsg(var Msg:TMessage);
begin
if TekVec^.GetCheck = 0 then
  begin
    SetVectorMode(ON);
    TekVec^.SetCheck(1);
  end
else
  begin
```

```
      SetVectorMode(OFF);
      TekVec^.SetCheck(0);
    end;
  end;
 {*****************************************************************}
 procedure LDVWindow.incrHTB(var Msg: TMessage);
 begin
   if HTBIndex<25 then
   begin
     inc(HTBIndex);
     if HTBindex>12 then
     begin
       BigBuff:=TRUE;
       IntBuffSize:=4090;
     end
     else
     begin
       BigBuff:=FALSE;
       IntBuffSize:=1010;
     end;
     SetHTB(HTBIndex);
     MSdiv^.SetText(HorTimeBase[HTBIndex]);
   end;
   if BigBuff then BuffSize^.SetText('BIG')
   else BuffSize^.SetText('small');
 end;
 {*****************************************************************}
 procedure LDVWindow.decrHTB(var Msg: TMessage);
 begin
   if HTBIndex>0 then
   begin
     dec(HTBIndex);
     if HTBindex>12 then
     begin
       BigBuff:=TRUE;
       IntBuffSize:=4090;
     end
     else
     begin
       BigBuff:=FALSE;
       IntBuffSize:=1010;
     end;
     SetHTB(HTBIndex);
     MSdiv^.SetText(HorTimeBase[HTBIndex]);
   end;
   if BigBuff then BuffSize^.SetText('BIG')
   else BuffSize^.SetText('small');
 end;
 {*****************************************************************}
 procedure LDVWindow.incrVDiv(var Msg: TMessage);
 begin
    if VertSensIndex<11 then
    begin
      inc(VertSensIndex);
      SetVertSens(VertSensIndex);
      VDiv^.SetText(VerSens[VertSensIndex]);
    end;
 end;
 {*****************************************************************}
 procedure LDVWindow.decrVDiv(var Msg: TMessage);
 begin
```

```
    if VertSensIndex>0 then
    begin
      dec(VertSensIndex);
      SetVertSens(VertSensIndex);
      VDiv^.SetText(VerSens[VertSensIndex]);
    end;
end;
{*****************************************************************}

procedure LDVWindow.FillClipBrd(var Msg:Tmessage);
var
  GlobalHandle:THandle;
  i,Step1,Step2,Step3,Step4:integer;
  temp:string;
  tmpstr:array[0..25] of char;
  Pstr:PChar;
begin
  SetCursor(Hour0^);
  GetData(DataArray);
  HaveData:=TRUE;
  SetCursor(Hour1^);
  if BigBuff then
     GlobalHandle:=GlobalAlloc(gmem_Moveable,84000)
  else
     GlobalHandle:=GlobalAlloc(gmem_Moveable,44000);
  if GlobalHandle<>0 then
    begin
      Pstr:=GlobalLock(GlobalHandle);
      SetCursor(Hour2^);
      if PStr <> nil then
      begin
        msdiv^.GetText(Pstr,15);
        StrCat(Pstr,CrLf);
        Volts^.GetText(tmpstr,15);
        StrCat(Pstr,tmpstr);
        StrCat(Pstr,CrLf);
        uAmps^.GetText(tmpstr,15);
        StrCat(Pstr,tmpstr);
        StrCat(Pstr,CrLf);
        StrCat(Pstr,PumpRecord.Flow);
        StrCat(Pstr,CrLf);
        StrCat(Pstr,PositionRecord.X);
        StrCat(Pstr,CrLf);
        StrCat(Pstr,PositionRecord.Y);
        StrCat(Pstr,CrLf);
        Vdiv^.GetText(tmpstr,15);
        StrCat(Pstr,tmpstr);
        StrCat(Pstr,CrLf);
        strcopy(tmpstr,'');
        SetCursor(Hour3^);
        Step1:=IntBuffSize Div 5;
        Step2:=IntBuffSize Div 4;
        Step3:=IntBuffSize Div 3;
        Step4:=IntBuffSize Div 2;
        for i:=0 to IntBuffSize-1 do
         begin
          if i = Step4 then SetCursor(Hour7^)
           else if i=Step3 then SetCursor(Hour6^)
            else if i=Step2 then SetCursor(Hour5^)
             else if i=Step1 then SetCursor(Hour4^);
          Str(DataArray[i],temp);
```

```
              StrPCopy(tmpstr,temp+TAB);
              StrCat(Pstr,tmpstr);

              Str(PRealData^[i+1],temp);
              StrPCopy(tmpstr,temp+CrLf);
              StrCat(Pstr,tmpstr);
             end;
            GlobalUnlock(GlobalHandle);
            SetCursor(Hour8^);
            if OpenClipBoard(HWindow) then
              begin
               EmptyClipBoard;
               SetClipBoardData(cf_Text,GlobalHandle);
               CloseClipBoard;
              end
            else GlobalFree(GlobalHandle);
          end
          else GlobalFree(GlobalHandle);
       end;{if got a GlobalHandle}
       SetCursor(NewCursor);
end;  {FillClipBrd}
{*****************************************************************}
procedure LDVWindow.TSaveData(var Msg:TMessage);
var
  Rawfile:text;
begin
{  if NOT HaveData then}
  begin
    GetData(DataArray);
    HaveData:=TRUE;
  end;
  {now get a file name and save the data out as text}
  StrCopy(FileName, '*.txt');
  if Application^.ExecDialog(New(PFileDialog,
      Init(@Self, PChar(sd_FileSave), FileName))) = id_Ok then
   begin
     assign(RawFile,Filename);
     rewrite(rawfile);
     writeln(rawfile);
     for i:=0 to IntBuffSize-1 do writeln(rawfile,DataArray[i]:5);
     close(rawfile);
   end;
end;
{*****************************************************************}
procedure LDVWindow.BSaveData(var Msg:TMessage);
var
  Rawfile:file of integer;
begin
{  if NOT HaveData then  }
  begin
    GetData(DataArray);
    HaveData:=TRUE;
  end;
  {now get a file name and save the data out as text}
  StrCopy(FileName, '*.bin');
  if Application^.ExecDialog(New(PFileDialog,
      Init(@Self, PChar(sd_FileSave), FileName))) = id_Ok then
   begin
     assign(RawFile,Filename);
     rewrite(rawfile);
     for i:=0 to IntBuffSize-1 do write(rawfile,DataArray[i]);
```

```
      close(rawfile);
    end;
end;
{*****************************************************************}
procedure LDVWindow.SetVoltage(var Msg:TMessage);
var
  V:PVoltDialog;
  ReturnValue:integer;
begin
  {open dialogBox}
  V:=New(PVoltDialog, Init);
  ReturnValue:=Application^.ExecDialog(V);
{  if ReturnValue=id_OK then
  begin
  end;                        }
end;
{*****************************************************************}
procedure LDVWindow.SetFlow(var Msg:TMessage);
var
  P:PDialog;
  FlowEdit:PEdit;
  GBox:PGroupBox;
  RButt:PRadioButton;
  ReturnValue:integer;
begin
  {open dialogBox}
  P:=New(PDialog, Init(@Self, 'FLOWDLG'));
  New(GBox, InitResource(P,id_PumpGrp1));
  New(GBox, InitResource(P,id_PumpGrp2));
  New(RButt, InitResource(P,id_PumpOn));
  New(RButt, InitResource(P,id_PumpOff));
  New(RButt, InitResource(P,id_PumpForward));
  New(RButt, InitResource(P,id_PumpReverse));
  New(FlowEdit, InitResource(P, id_PumpFlow, SizeOf(PumpRecord.Flow)
));
  P^.TransferBuffer:=@PumpRecord;
  ReturnValue:=Application^.ExecDialog(P);
  if ReturnValue=id_OK then  {set the pump}
      with PumpRecord do
      begin
        TurnPump(Running,Forward);
        PumpFlow(Flow);
      end;
end;
{*****************************************************************}
procedure LDVWindow.SetScope(var Msg:TMessage);
var
  P:PScopeDialog;
  GBox:PGroupBox;
  RButt:PRadioButton;
  local:PCheckbox;
  ReturnValue:integer;
begin
  {open dialogBox}
  P:=New(PScopeDialog, Init(@Self, 'SCOPSETUP'));
  New(GBox, InitResource(P,id_CouplGrp));
  New(RButt, InitResource(P,id_DC));
  New(RButt, InitResource(P,id_AC));
  New(RButt, InitResource(P,id_Grnd));
  New(local,InitResource(P,id_local));
  P^.TransferBuffer:=@ScopeRecord;
```

```
    ReturnValue:=Application^.ExecDialog(P);
    Case ReturnValue of
       id_OK:SetScopeCoupling;  {set the Scope}
       id_Reset:begin
                   ResetScope;
                   HTBIndex:=GetHTBIndex;
                   MSdiv^.SetText(HorTimeBase[HTBIndex]);
                   VertSensIndex:=GetVertSensIndex;
                   VDiv^.SetText(VerSens[VertSensIndex]);
                 end;
   end;{case}
 end;
{*****************************************************************}
procedure LDVWindow.SetPosition(var Msg:TMessage);
var
  S:PPosDialog;
  PosEdit:PEdit;
  ReturnValue:integer;
begin
  {open dialogBox}
  S:=New(PPosDialog, Init(@Self, 'POSDLG'));
  New(PosEdit, InitResource(S, id_PositionX, SizeOf(PositionRecord.X)
));
  New(PosEdit, InitResource(S, id_PositionY, SizeOf(PositionRecord.Y)
));
  S^.TransferBuffer:=@PositionRecord;
  ReturnValue:=Application^.ExecDialog(S);
  case ReturnValue of
   id_OK : GoToPosition(PositionRecord);   {set the positioners}
   id_Home:begin {go to home position}
             StrCopy(PositionRecord.X,'0.0000');
             StrCopy(PositionRecord.Y,'0.0000');
             GoHome;
           end;
  end;{case}
end;
{*****************************************************************}
procedure LDVWindow.AverageSignal(var Msg: TMessage);
{procedure to put the current coordinates and the average
 scope signal into the clipboard.  Scope should be DC
 coupled for this to mean anything.}
var
  GlobalHandle:THandle;
  i:integer;
  sum,ave:double;
  tmpstr:array[0..25] of char;
  Pstr:PChar;
begin
  GetData(DataArray);
  GlobalHandle:=GlobalAlloc(gmem_Moveable,5000);
  if GlobalHandle<>0 then
    begin
      Pstr:=GlobalLock(GlobalHandle);
      if PStr <> nil then
      begin
        StrCopy(Pstr,PositionRecord.X);
        StrCat(Pstr,TAB);
        StrCat(Pstr,PositionRecord.Y);
        StrCat(Pstr,TAB);
        Vdiv^.GetText(tmpstr,15);
        StrCat(Pstr,tmpstr);
```

```
          StrCat(Pstr,TAB);
          strcopy(tmpstr,'');
          sum:=0;
          for i:=0 to IntBuffSize-1 do
             sum:=sum+DataArray[i];
          ave:=sum/IntBuffSize;
          StrPCopy(tmpstr,RToStr(Ave));
          StrCat(Pstr,tmpstr);
          GlobalUnlock(GlobalHandle);
          if OpenClipBoard(HWindow) then
            begin
             EmptyClipBoard;
             SetClipBoardData(cf_Text,GlobalHandle);
             CloseClipBoard;
            end
          else GlobalFree(GlobalHandle);
       end
       else GlobalFree(GlobalHandle);
     end;{if got a GlobalHandle}
 end;  {AverageSignal}
 {*****************************************************************}
 procedure LDVWindow.Ch3Signal(var Msg: TMessage);
 {procedure to put the current coordinates and the average
  signal on A/D channel #3 into the clipboard. }
 var
   GlobalHandle:THandle;
   ch3volts:double;
   tmpstr:array[0..25] of char;
   Pstr:PChar;
 begin
   GlobalHandle:=GlobalAlloc(gmem_Moveable,200);
   if GlobalHandle<>0 then
     begin
       Pstr:=GlobalLock(GlobalHandle);
       if PStr <> nil then
       begin
         StrCopy(Pstr,PositionRecord.X); {start with the X position in
 the clipboard string}
         StrCat(Pstr,TAB); {tack on a TAB}
         StrCat(Pstr,PositionRecord.Y);  {tack on the Y position,}
         StrCat(Pstr,TAB);
         strcopy(tmpstr,'');
         ch3volts:=DTVoltage(DTReadADI(3));
         StrPCopy(tmpstr,RToStr(Ch3Volts));    {finish with the
 voltage on Ch.3}
         StrCat(Pstr,tmpstr);
         GlobalUnlock(GlobalHandle);
         if OpenClipBoard(HWindow) then
           begin
            EmptyClipBoard;
            SetClipBoardData(cf_Text,GlobalHandle);
            CloseClipBoard;
           end
         else GlobalFree(GlobalHandle);
      end
      else GlobalFree(GlobalHandle);
    end;{if got a GlobalHandle}
 end;{channel 3 signal}
 {*****************************************************************}
 procedure LDVWindow.About(var Msg:TMessage);
 begin
```

```
   Application^.ExecDialog(New(PDialog,Init(@Self, 'ABOUTBOX')));
 end;
 {**************************************************************}
 procedure LDVWindow.getcurrent(var Msg:TMessage);
 begin
   StrPCopy(CurrentText,RToStr(DTReadSpellCurrent));
   SpelluAmps^.SetText(CurrentText);
   StrPCopy(CurrentText,RToStr(DTReadCurrent));
   uAmps^.SetText(CurrentText);
   StrPCopy(CurrentText,RToStr(DTReadSpellVoltage));
   Volts^.SetText(CurrentText);
 end;{getcurrent}
 {**************************************************************}
 procedure LDVWindow.FFTSetup(var Msg:TMessage);
 var
   P:PDialog;
   AVGNumEdit:PEdit;
   RButt:PRadioButton;
   ReturnValue:integer;
 begin
   {open dialogBox}
   P:=New(PDialog, Init(@Self, 'FFTSETUP'));
 {  New(GBox, InitResource(P,id_XXXXXX));  <----- IF NEEDED!! }
   New(RButt, InitResource(P,id_snglEvent));
   New(RButt, InitResource(P,id_avgEvents));
   New(AVGNumEdit, InitResource(P, id_avgNumber,
 SizeOf(SpectrumRecord.avgNum) ));
   P^.TransferBuffer:=@SpectrumRecord;
   ReturnValue:=Application^.ExecDialog(P);
   if ReturnValue=id_OK then  {}
     begin
       if NOT SpecDisplayOpen then {create one}
         begin
           {pop up Spectral Display window}
           PFTDisplay:=PFTDispWindow(
             Application^.MakeWindow(
               new(
               PFTDispWindow, Init(nil, 'FTDisp')
               )
             )
           );
         SetFocus(HWindow);
         end;
     end;
 end;{FFTSetup}
 {**************************************************************}
 {------------------VoltDialog Procedures------------------------}
 Constructor VoltDialog.Init;
 begin
   TDlgWindow.Init(Application^.MainWindow,'VOLTDLG');
   TenThouScroll:=New(PScrollBar,InitResource(@Self,id_VScroll3));
   ThouScroll:=New(PScrollBar,InitResource(@Self,id_VScroll2));
   HundScroll:=New(PScrollBar,InitResource(@Self,id_VScroll1));
   TenScroll:=New(PScrollBar,InitResource(@Self,id_VScroll0));
   StatTenThou:=New(PStatic,InitResource(@Self,id_VStat3,2));
   StatThou:=New(PStatic,InitResource(@Self,id_VStat2,2));
   StatHund:=New(PStatic,InitResource(@Self,id_VStat1,2));
   StatTen:=New(PStatic,InitResource(@Self,id_VStat0,2));
 end;
 {**************************************************************}
 procedure VoltDialog.SetupWindow;
 var
   CString:Text15;
   VoltValue,TenThous,thous,hunds,tens,ReturnValue:integer;
 begin
   TDlgWindow.SetupWindow;
   val(VoltText,VoltValue,ReturnValue);
   Tenthous:=VoltValue div 10000;
   Thous:=(VoltValue-10000*Tenthous) div 1000;
   Hunds:=(VoltValue-10000*Tenthous-1000*Thous) div 100;
   tens:=(VoltValue-10000*Tenthous-1000*thous-100*Hunds) div 10;
   TenThouScroll^.SetRange(0,3);
   ThouScroll^.SetRange(0,9);
   HundScroll^.SetRange(0,9);
   TenScroll^.SetRange(0,9);
   TenThouScroll^.PageMagnitude:=1;
   ThouScroll^.PageMagnitude:=1;
   HundScroll^.PageMagnitude:=1;
   TenScroll^.PageMagnitude:=1;
   TenThouScroll^.SetPosition(tenthous);
   ThouScroll^.SetPosition(thous);
   HundScroll^.SetPosition(hunds);
   TenScroll^.SetPosition(tens);
   Str(TenThous:1,CString);
   StatTenThou^.SetText(CString);
   Str(thous:1,CString);
   StatThou^.SetText(CString);
   Str(Hunds:1,CString);
   StatHund^.SetText(CString);
   Str(Tens:1,CString);
   StatTen^.SetText(CString);
 end;
 {****************************************************************}
 procedure VoltDialog.GetWindowClass(var AWndClass:TWndClass);
 begin
  TDlgWindow.GetWindowClass(AwndClass);
 end;
 {****************************************************************}
 function VoltDialog.GetClassName:PChar;
 begin
  GetClassName:='VoltDlgWindow';
 end;
 {****************************************************************}
 procedure VoltDialog.UpdateVoltage;
 var
   CString: Text15;
   VoltInt:integer;
 begin
   VoltInt:=10000*TenThouScroll^.GetPosition
           +1000*ThouScroll^.GetPosition
           +100*HundScroll^.GetPosition
           +10*TenScroll^.GetPosition;
   Str(VoltInt,CString);
   HV_Set(CString);
   StrCopy(VoltText,CString);
 end; {UpdateVoltage}
 {****************************************************************}
 procedure VoltDialog.HandleTenThousScrollMsg(var Msg: TMessage);
 var
   CString: Text15;
 begin
   Str(TenThouScroll^.GetPosition:1, cString);
  StatTenThou^.SetText(CString);
  UpdateVoltage;
end;

{****************************************************************}
procedure VoltDialog.HandleThousScrollMsg(var Msg: TMessage);
var
  CString: Text15;
begin
  Str(ThouScroll^.GetPosition:1, cString);
  StatThou^.SetText(CString);
  UpdateVoltage;
end;

{****************************************************************}
procedure VoltDialog.HandleHundScrollMsg(var Msg: TMessage);
var
  CString: Text15;
begin
  Str(HundScroll^.GetPosition:1, cString);
  StatHund^.SetText(CString);
  UpdateVoltage;
end;

{****************************************************************}
procedure VoltDialog.HandleTenScrollMsg(var Msg: TMessage);
var
  CString: Text15;
begin
  Str(TenScroll^.GetPosition:1, cString);
  StatTen^.SetText(CString);
  UpdateVoltage;
end;

{****************************************************************}
var
  LDVApp : LDVApplication;

begin
  LDVApp.Init(AppName);
  LDVApp.Run;
  LDVApp.Done;
end.
```