VIBRATIONAL PREDISSOCIATION SPECTROSCOPY OF MASS-SELECTED IONIC CLUSTERS

Thesis by

Yi-Bin Cao

In Partial Fulfillment of the Requirements
for the Degree of
Doctor of Philosophy

California Institute of Technology
Pasadena, California
1994
(submitted May 9, 1994)

I dedicate this thesis to my late grandmother Ying-Bao Zheng, who was the first teacher in my life.

ACKNOWLEDGMENTS

This thesis would not have been possible without the contribution of many others. First of all, I would like to thank my research advisor, Professor Mitchio Okumura. Mitchio is a first-rate scientist and an excellent teacher. He taught me many experimental skills as well as the proper way to approach a scientific problem. Whenever I encountered difficulties with my research, he was always there to offer me direction and encouragement. I am equally grateful for the help he gave me when I was in tough situations. And last but not least, his financial support to me over the last five years is deeply appreciated. In a word, I feel very fortunate to have Mitchio as my advisor, and I wish him the best with his scientific career.

Members of the Okumura group help each other and work as a team. Jong-Ho Choi is my main collaborator, and I enjoyed working with him. Some of the experiments would not have been successful without his persistent efforts. Dr. Bermi Haas was instrumental in getting the OPO to work. Matt Johnson played a key role in the development of the discharge source and was always available to lend a hand or listen to my (mostly stupid) ideas. Keith Kuwata performed *ab initio* calculations to help interpret the experimental data. He is now taking over my "spot," and I wish him the best with the experiments. Other current and former members of our group, Alan Frankel, Christine Nelson, Teresa Moore, and Jim Spotts were also very helpful to me.

Before joining the Okumura group, I worked in applied physics department with Professor Marc Nicolet for half a year. Members of the Nicolet group got me started in doing scientific research. Dr. Elzbieta Kolawa taught me many techniques of semiconductor device fabrication and analysis. I also enjoyed working with Tom

Workman, Gang Bai and En Ma.

Many campus facilities provided vital support services to our research. Tony Stark, Ray Garcia and Guy Duremberg of the instrument shop made numerous mechanical devices for us and took the time to teach me the art of machining on many occasions. With the rising unemployment rate for scientists, the machining skills they taught me might come in handy someday. Tom Dunn of the electronics shop and Ya-Yun Liu of the EE stockroom gave me much invaluable advice on electronics (as well as many free electronic components). Jess Miller and Daryl Riley, who used to work in the machine shop, were always friendly and helpful.

My fellow Chinese students here made my Caltech experience an enjoyable one. I had a very pleasant experience rooming with Yuhai Tu. Xiangdong Fang always informed me when he set up some sort of party on the weekends. I would also like to thank Jiangtao Hong, Bing Chen, Jun Xu, Daqi Lu, Ming Lu, Guanhua Gao, Mei Zhu, Xin Sun and Shujuan Ji. In addition, I would miss the "generals" of the Friday night six-war games. Special thanks goes to Tim Sun, who took me under his wing when I first came here, for his kindness and gracious help.

Throughout my life, I have received the help and guidance from some great teachers. My elementary teachers taught me to be independent as a person and creative in thoughts. My high school math teacher Liang-Xiu Wang believed in me and constantly challenged me to do my best. I could not have come so far without him. I am also grateful to my college class-master Guo-Hua Ye, for guiding me through a very difficult freshman year at the University of Science and Technology of China.

Finally, I would like to thank my family for their support and understanding. To my father, who continued his research in a rural pharmaceutical plant during the Cultural Revolution, and my mother, who as a high school teacher, struggled to educate hundreds of young people in that lawless and benighted era, this Ph.D. degree is also for you. To my sister Jianwen, who often called to remind me that I cannot be a graduate student forever, thanks for the advice.

ABSTRACT

This thesis presents structural investigations of molecular ions and ionic clusters using vibrational predissociation spectroscopy. Experimentally, a pulsed beam of the mass-selected ion is crossed by a tunable infrared laser beam generated by a Nd:YAG pumped $LiNbO_3$ optical parametric oscillator. The resulting fragment ion is mass-analyzed and detected, with its intensity as a function of the laser wavelength being the "action" spectrum of the parent ion. In the case of SiH_7^+, we observed a vibrational band centered at 3866 cm^{-1} with clear P, Q, R branches, which is assigned as a perturbed H_2 stretch. The absence of a second H_2 band suggests that the ion forms a symmetric complex with a structure $H_2{\bullet}SiH_3^+{\bullet}H_2$, in contrast to the species CH_7^+, which has the structure $CH_5^+{\bullet}H_2$. The infrared spectra of $NO_2^+(H_2O)_n$ clusters exhibit a marked change with cluster size, indicating that an intracluster reaction occurs with sufficient solvation. Specifically, in $NO_2^+(H_2O)_n$ clusters where $n \leq 3$, H_2O binds to a nitronium ion core; but at $n=4$ the NO_2^+ reacts, transforming the cluster to a new structure of $H_3O^+{\bullet}(H_2O)_{n-2}{\bullet}HNO_3$. For protonated chlorine nitrate, we have observed two distinct isomers previously predicted by *ab initio* calculations: $NO_2^+{\bullet}(HOCl)$, the lowest energy isomer, and $(ClO)(HO)NO^+$, a covalently bonded isomer about 20 kcal/mol higher in energy. Both isomers decompose to NO_2^+ and HOCl upon photo-excitation. These results for $HClONO_2^+$ lend strong support to the involvement of an ionic mechanism in the reaction of $ClONO_2$ on polar stratospheric cloud surfaces, a critical step in the dramatic springtime depletion of ozone over Antarctica. Current research activities on halide-solvent clusters and metal-ligand complexes as well as technological improvements of the apparatus are also discussed.

TABLE OF CONTENTS

Chapter 1. Introduction

Infrared spectroscopy has been well established as a successful tool for the structural determination of molecules. However, extending this technique to ions has been a challenge. The high reactivity of the ions makes it difficult to generate sufficient ion concentrations for the traditional gas-phase absorption spectroscopy.[1]

In the 1980s, with the advent of high-power, narrow-linewidth tunable infrared lasers (color center laser, difference frequency laser, diode laser, etc), the sensitivity of infrared spectroscopy has been greatly improved. With this improvement, Oka was able to observe the ν_2 band of the H_3^+ ion in a DC discharge of hydrogen.[2] This was followed by the development of the velocity modulation technique by Saykally and co-workers,[3] with which the absorption lines of the charged species are detected while those of the neutral species are removed from the spectra. Using this method, high resolution infrared spectra have been observed for a number of molecular ions, and many of those, such as H_3O^+, $C_2H_3^+$, NH_4^+ and OH^-, are of great chemical and astrophysical importance.[4-7] However, molecular ions produced in the discharge are limited to fundamental ions very small in size, and with the co-existence of several ion species in the discharge, interpretation and analysis of the spectra can be very time-consuming.

The use of an ion beam in infrared spectroscopy was pioneered by Wing *et al.*[8] In their experiments, a fast beam of HD^+ was coincident with a beam from a CO laser. Doppler tuning was used to bring the HD^+ vibrational transitions into resonance with a nearby laser line. Since the ion concentration in a beam is much lower than that in a discharge, direct absorption measurement is not feasible. Instead, "action spectra" were measured, *i.e.*, vibrational spectra were obtained by detecting the effects of infrared excitation on the ions. (Specifically, the change in charge transfer efficiency were measured to obtain the vibrational spectra.) However, these ion beams contained highly excited "hot" ions, hence the resulting spectra consisted of thousands of transition lines, making data analysis extremely difficult. So far, their technique has only been applied successfully in a small number of cases such as HD^+, H_3^+, HNO^+, NH^+.[8-9] Direct absorption spectroscopy of ion beam by Saykally, e.g., HF^+, H_3O^+, have only seen strongly bound species also observed in discharges.[10]

A scientific field that has undergone explosive growth in the last two decades is the study of clusters. Perhaps one overriding reason for this extensive activity derives from the promise that these studies will serve to bridge the gap between an understanding of isolated molecular systems on the one hand, and a microscopic understanding of condensed matter on the other. In fact, work in this field has begun to reveal some molecular details of the collective effects in phase transitions, surface evolution, and solvation. The studies on nanometer materials have also attracted attention from the semiconductor industry, as it tries to increase device density on a chip even further.

The studies of ionic clusters,[11] which can be mass-analyzed for size-dependent studies, were initially concentrated on the gas phase thermochemistry of clustering reactions. Using high-pressure mass spectrometry, Kebarle and others obtained thermochemical data on numerous ion-molecule association and clustering reactions.[12] Their work was followed by the generation of even larger clusters using the supersonic expansion technique, as well as studies on cluster stability, structure, reaction, and dynamics. Among them, the discovery of "magic numbers" (cluster sizes at which an abnormally large intensity are observed in the mass spectrometry) illustrates a interesting relationship between the stability of a cluster and its structure. The most well-known examples are C_{60}^+, which has a soccer ball structure,[13] and $H^+(H_2O)_{21}$, whose proposed structure is a dodecahedron with a H_3O^+ ion enclosed in its cavity.[14]

Nevertheless, the best way to study the structure of ionic clusters is still spectroscopy. By coupling an ionized supersonic expansion with a mass spectrometer to generate a mass-selected ion beam, Lineberger and co-workers pioneered visible photodissociation spectroscopy of cluster ions.[15] Their studies of $(CO_2)_n^+$, Ar_n^+, $Br^-(CO_2)_n$ have identified the central ion core of the cluster, and revealed the effect of the surrounding molecules on the photodissociation process.[16] But this technique is most successful for clusters such as Mg^+Ar, $Mg^+(H_2O)$, $V^+(CO_2)$ and $Cr^+(N_2)$ where the metal cations are strong chromophores, giving rise to intense vibronic cluster spectra.[17] Approximate rotational constants might also be extracted from some of these spectra, as demonstrated by Ducan *et al.* in the case of $Mg^+(H_2O)$.

Infrared spectroscopy can directly provide vibrational frequencies and rotational constants, especially of systems without bound-to-bound electronic transitions in the visible and near ultra-violet. However, only a few studies have been carried out in this region, partly due to the signal reduction caused by a smaller cross section of vibrational transitions. In the late 1980s, Lee and co-workers obtained the first infrared spectra of ionic clusters using vibrational predissociation spectroscopy.[18] Structures of H_{2n+1}^+, $H_3O^+(H_2O)_n$ and $NH_4^+(NH_3)_n$ were established with their spectroscopic data, aided by *ab initio* calculations. Infrared spectroscopy was also used by Lisy *et al.* to study the solvation of $Cs^+(CH_3OH)_n$ clusters.[19]

We have built an apparatus which combines Lee's infrared spectroscopic method with the design of Lineberger's photodissociation spectrometer. This thesis will present my work, in collaboration with several co-workers, on the construction of this apparatus, and some results we obtained with this apparatus.

1.1 OUTLINE

The experimental technique will be described in chapter 2. It is hoped that this chapter can serve as a useful lab guide for future students. Therefore, brief discussions of the underlying principles will be provided along with descriptions of various components of the apparatus built during my time at Caltech. The pulsed discharge source that I helped develop, as well as my original contributions on the improvement of TOF spectrometer, will be described in detail. The operation of the $LiNbO_3$ optical parametric oscillator, which I built, will also be extensively discussed.

Chapters 3 through 5 present my work on three separate ionic systems using our apparatus. The SiH_7^+ ion, the topic of chapter 3, was first detected in our apparatus. The reason for a dramatic structural difference between SiH_7^+ and CH_7^+ will also be discussed. In chapter 4, I will present the first spectroscopic evidence of intracluster reactions provided by our study of the $NO_2^+(H_2O)_n$ clusters. The infrared spectra of $HClONO_2^+$ will be given in chapter 5. Our data on $HClONO_2^+$, together with an earlier study from our group, have provided important clues on the mechanism of heterogeneous reactions that occur on polar stratospheric clouds. These reactions play a critical role in Antarctic ozone loss.

In chapter 6, I will describe some research projects that are currently unfinished. These projects are infrared spectroscopic studies of $Cl^-(H_2O)_n$ and $F^-(H_2O)_n$ clusters, and metal ion-ligand complexes. They are currently being pursued by my co-workers Jong-

Ho Choi and Keith Kuwata. In the last section of the chapter, my thoughts and efforts on improving the performance and extending the capability of our apparatus, particularly by reducing the metastable background and achieving rotational resolution, will be presented.

The source code for *OPODATA*, the computer program which controls the scanning of our $LiNbO_3$ OPO, will be furnished in the appendix.

1.2 REFERENCES

(1) Sears, T. J. *J. Chem. Soc. Faraday Trans. 2* **1987**, *83*, 111.

(2) Oka, T. *Phys. Rev. Lett.* **1980**, *45*, 531.

(3) Gudeman, C. S.; Saykally, R. J. *Annu. Rev. Phys. Chem.* **1984**, *35*, 387.

(4) Begemann, M. H.; Gudeman, C. S.; Pfaff, J.; Saykally, R. J. *Phys. Rev. Lett.* **1983**, *51*, 554. Begemann, M. H.; Saykally, R. J. *J. Chem. Phys.* **1985**, *82*, 3570. Liu, D. J.; Oka, T. *Phys. Rev. Lett.* **1985**, *54*, 1787.

(5) Crofton, M. W.; Jagod, M. F.; Rehfuss, B. D.; Oka, T. *J. Chem. Phys.* **1989**, *91*, 5139.

(6) Crofton, M. W.; Oka, T. *J. Chem. Phys.* **1983**, *79*, 3157. Schafer, E.; Begemann, M. H.; Gudeman, C. S.; Saykally, R. J. *J. Chem. Phys.* **1983**, *79*, 3159. Schafer, E.; Saykally, R. J.; Robiette, A. G. *J. Chem. Phys.* **1984**, *80*, 3969.

(7) Owrutsky, J. C.; Rosenbaum, N. H.; Tack, L. M.; Saykally, R. J. *J. Chem. Phys.* **1985**, *83*, 2426. Liu, D. J.; Oka, T. *J. Chem. Phys.* **1986**, *84*, 2426.

(8) Wing, W. H.; Ruff, G. A.; Lamb, W. E.; Spezeski, J. J. *Phys. Rev. Lett.* **1976**, *36*, 1488.

(9) Carrington, A.; McNab, I. R.; Montgomerie, C. A. *Chem. Phys. Lett.* **1988**, *151*, 258. Miller, H. C.; Hardwick, J. L.; Farley, J. W. *J. Molec. Spec.* **1989**, *134*, 329. Miller, H. C.; Alzaal, M.; Farley, J. W. *Phys. Rev. Lett.* **1987**, *58*, 2031.

(10) Coe, J. V.; Owrutsky, J. C.; Keim, E. R.; Agman, N. V.; Horde, C. V.; Saykally, R. J. *J. Chem. Phys.* 1989, 90, 3893.

(11) Castleman Jr., A. W.; Keesee, R. G. *Chem. Rev.* **1986**, *86*, 589.

(12) A complete compilation of ion-molecule clustering dynamics can be found in:

Keesee, R. G.; Castleman Jr., A. W. *J. Phys. Chem. Ref. Data*, **1986**, *15*, 1011.

(13) Kroto, H. W.; Heath, J. R.; Obrien, S. C.;Curl, R. F.; Smalley, R. E. *Nature* **1985**, *318*, 162.

(14) Holland, P. M.; Castleman Jr., A. W. *J. Chem. Phys.* **1980**, *72*, 5984.

(15) Johnson, M. A.; Lineberger, W. C. In *Techniques for the Study of Ion-Molecule Reactions*; Farrar, J. M., Saunders Jr., W. H., Eds.; Wiley-Interscience: New York, **1988**.

(16) Johnson, M. A.; Alexander, M. L.; Lineberger, W. C. *Chem. Phys. Lett.* **1984**, *112*, 285. Levinger, N. E.; Ray, D. Murray, K. K.; Mullin, A. S.; Schulz, C. P; Lineberger, W. C. *J. Chem. Phys.* **1988**, *89*, 71. Alexander, M. L.; Levinger, N. E.; Johnson, M. A.; Ray, D.; Lineberger, W. C. *J. Chem. Phys.* **1988**, *88*, 6200.

(17) Pilgrim, J. S.; Yeh, C. S.; Duncan, M. A. *Chem. Phys. Lett.* **1993**, *210*, 322. Willey, K. F.; Yeh, C. S.; Robbins, D. L.; Pilgrim, J. S.; Duncan, M. A. *J. Chem. Phys.* **1992**, *97*, 8886. Lessen, D. E.; Asher, R. L.; Brucat, P. J. *J. Chem. Phys.* **1991**, *95*, 1414. Lessen, D. E.; Asher, R. L.; Brucat, P. J. *Chem. Phys. Lett.* **1991**, *177*, 380.

(18) Okumura, M.; Yeh, L. I.; Lee, Y. T. *J. Chem. Phys.* **1988**, *88*, 79. Yeh, L. I.; Okumura, M.; Myers, J. D.; Price, J. M.; Lee, Y. T. *J. Chem. Phys.* **1989**, *91*, 7319. Price, J. M.; Crofton, M. W.; Lee, Y. T. *J. Chem. Phys.* **1991**, *95*, 2182.

(19) Liu, W. L.; Lisy, J. M. *J. Chem. Phys.* **1988**, *89*, 605.

Chapter 2. Experimental

2.1 INTRODUCTION

All of the experiments that are described in this thesis were performed on an ion infrared spectrometer, utilizing the technique of vibrational predissociation spectroscopy. A schematic diagram of this machine is presented in figure 1. In short, the chemical precursor species is seeded in a pulsed molecular beam. Various ionization techniques are employed to generate cold molecular ions and ionic clusters inside the molecular beam. These resulting ions are then separated by a time-of-flight (TOF) mass spectrometer, and only ions of a selected mass are intersected by a pulsed tunable infrared laser beam. Ions which are pumped to vibrationally excited states can predissociate (*i.e.*, are prepared in zeroth-order bound vibrational states which are weakly coupled to the dissociation continuum) if their internal energy exceeds the binding energy. The resulting photofragment ions are mass-analyzed, and the infrared spectra of the selected ions are obtained by measuring the photofragment intensity as a function of the laser wavelength. For certain ionic species, more than one dissociation channel could be open; in these cases, the fragmentation pattern of the parent ion provides additional information about the ion structure.

Before describing its design and operation in detail, let us first take a look at the capability as well as the promise of our apparatus. First of all, since the ions are mass-analyzed by the TOF spectrometer and only the ions of a certain mass are selected for spectroscopy, there exists a clear one-to-one correspondence between the spectrum and the ion species from which the spectrum arises. Hence, we have eliminated one of the major problems in studying ions in a discharge, which is identifying the ionic species to which each individual absorption peak belongs. Secondly, due to the huge gain of the multi-channel plate ion detector (~10^6, achieved by electron cascading), even a single ion event generates a signal output that is large enough to be detected. Therefore, our apparatus has the promise to be an instrument with very high sensitivity. Lastly, our infrared laser beam intersects the ion beam perpendicularly, making high-resolution spectroscopy a distinct possibility since the first-order Doppler effect is eliminated.

The rest of this chapter is organized as follows. In section 2.2, various components of the ion beam apparatus will be described. The $LiNbO_3$ optical parametric oscillator (OPO), which is the source of the infrared radiation, will be discussed in section 2.3. The operation of the system as a whole will be presented in section 2.4.

2.2 THE ION BEAM APPARATUS

The ion beam apparatus is essentially a tandem TOF mass spectrometer coupled with a pulsed ion source. This apparatus, shown in figure 1, consists of three sections. The first section (source section) is a 20" × 20" × 30" chamber where the ions are generated. A skimmer with a 3 mm diameter orifice collimates the ion beam into the second section (TOF section) which consists of a 6" diameter, 45" long tube. Ions are quickly accelerated to a very high kinetic energy (~2.6 keV) and then travel field-free the rest of the way into the third section (detection section). This produces a separation of the ions according to their mass. The detection region of the apparatus is a 26" diameter bell-jar which contains optical windows, a reflectron, and ion detectors. The laser beam enters the vacuum chamber through an optical window and is timed to intersect the ions of selected mass. The resulting photofragment ions are mass-analyzed by the reflectron and detected by the multi-channel plate detector (MCP).

The source section of the apparatus is pumped by a 5000 l/s Varian VHS-10 diffusion pump backed by two 8 l/s Sargent-Welch mechanical pumps. The other two sections are differentially pumped, with sections I and II separated by the 3 mm skimmer, while sections II and III are separated by a 1" diameter hole. The TOF section is pumped by a 2400 l/s VHS-6 and a 1200 l/s VHS-4 diffusion pump. All diffusion pumps are baffled by freon cold traps. To eliminate the vapor of diffusion pump oil that could contaminate the surface of the MCP detector, the detection section is pumped by a 500 l/s 6" turbomolecular pump. The pressures in each section are pumped down to $\sim 3 \times 10^{-7}$ Torr before the experiment. They rise to steady-state values of $\sim 10^{-4}$ torr, $\sim 2 \times 10^{-6}$ Torr, and $\sim 5 \times 10^{-7}$ Torr respectively during the experiment in which the pulsed valve is operated at 10 Hz.

The following is a detailed discussion of the ion beam apparatus in terms of ion generation, the TOF mass spectrometer, and fragment ion detection.

2.2.1 Ion Generation

The ion source is composed of a pulsed gas valve and an accompanying ionization device. A small amount of the "active" species is mixed with a buffer gas such as H_2, He, or Ar. The gas mixture is delivered to the stagnation chamber of the pulsed valve through the gas inlet line. The mixture is then pulsed out through the valve nozzle, followed by ionization. The primary ionization products undergo collisional cooling and charge exchange, forming cold, stable ions and ionic clusters. The resulting plasma is skimmed into the TOF section.

2.2.1.1 Gas Inlet System

A gas inlet system equipped with MKS flow controllers is used to produce the proper gas mixture in the pulse valve stagnation chamber (figure 2). To eliminate the residual water vapor that could significantly change the make-up of the ionic species formed by the source, a clean, leak-tight gas inlet system is required. Therefore, the gas inlet line is composed of stainless steel tubing connected with VCR fittings and is pumped out by a Edwards Diffstak-63 diffusion pump. The MKS flow controllers, driven by home-made electronics, regulate flow speed through the channel by using the feedback loop $F = g_i\ (P_{set} - P_{act})$. (In this formula, P_{set} and P_{act} are the pre-set and the actual pressures in the stagnation chamber, respectively.) Therefore, by putting different gas species through different MKS channels and setting the flow speed ratio g_i / g_j between these channels, a constant gas mixture can be maintained during the entire experiment.

In some cases, the chemical species of interest is in the liquid phase at room temperature. An alternative method is then used to create the gas mixture. We install a glass tube containing the liquid and pass the buffer gas over the liquid surface. The mixture ratio between the vapor of the active species and that of the buffer gas is controlled by immersing the glass tube in a liquid nitrogen or dry-ice based solvent bath and varying the temperature of the bath.

2.2.1.2 Pulsed Molecular Beam

The use of a pulsed valve is essential in generating intense molecular beams without raising the background pressure of the vacuum system excessively. The pulsed valve we use in our experiments is a piezo-driven device designed by Trickl.[1] The piezo element is a 1.5" disc made by Physik Instrumente (model P.910.450). Since the piezo element requires at least 200 V in order to produce significant motion and is highly capacitative, a high-voltage, high-current pulse is needed to drive the pulsed valve. This is provided by a home-made 0-1000 V, 2 A pulse driver (figure 3), whose key elements are high-power metal-oxide-semiconductor field-effect transistors (MOSFETs). Driven by the circuit output, the pulse valve is able to open for a duration as short as 150 μs with a maximum repetition rate of ~200 pulses per second. This piezo-driven valve is superior to the commercially available pulsed valve (General Valve, Inc.), which can only deliver gas pulses of 1 ms or longer.

Because of the mechanical complexity of pulsed beam sources and because the optimization of the conditions is highly empirical, it is extremely useful to have some independent means to examine the time profile of the pulse. The device we constructed for this purpose is a fast ionization gauge (FIG),[2] which responds directly to molecular density on a microsecond time scale (figure 4). Similar to a Bayard-Alpert gauge, it consists of a central ion collector wire surrounded by a helical grid, with the electron-emitting filament at the outside of the grid. The grid is made 3 mm in diameter so that the flight time of the molecules through the grid volume does not impose a serious limitation on time resolution of the detector. As in a conventional ion gauge, the current in the FIG is proportional to the gas density $n(t)$ inside the grid. The fast response of the FIG is achieved by connecting the ion collector wire directly to the negative input of an operational amplifier to minimize the input capacitance. This is accomplished by attaching the amplifier circuit directly to the gauge head.

Under typical operation conditions, the pulsed valve has a stagnation pressure P_0 of ~1000 Torr. The piezo element is triggered to open the valve for a duration of ~200 μs, allowing gas to escape from the 0.5 mm diameter nozzle. This amounts to a gas load of ~40 Torr cm^3 per pulse. The gas pulse expands supersonically, since the pressure ratio

P_0/P_b exceeds the critical value of ~2.1.[3] The most important consequence of the expansion's being supersonic is the supersonic cooling. A fully expanded molecular beam can attain a translational temperature of ~10 K as well as greatly reduced rotational and vibrational temperatures.[3] Along with the expansion is a drastic decrease of the gas density. Under our experimental conditions, the number density of the gas is ~3 × 10^{17} cm^{-3} at 2 mm from the nozzle and ~10^{15} cm^{-3} at 1 cm from the nozzle.

2.2.1.3 Ionization

Three different ionization techniques are employed in our laboratory: electron-beam (e-beam) ionization,[4] pulsed discharge ionization, and the method of ionic pickup exchange.[5] The first two techniques will be described here.

In e-beam ionization, ions are generated by crossing the free jet expansion perpendicularly with an electron beam. A home-made electron gun, mounted on the side of the pulsed valve, produces a continuous 600 eV electron beam.[6] The electron beam produces a current of 0.1-0.4 mA, as measured by a Faraday cup located on the other side of the valve. As will be explained later, the distribution of the cluster ions generated by this technique depends critically on the focusing condition of the e-beam as well as the location of the jet-beam intersection. From a visual inspection of the fluorescence created by crossing the electron beam with an argon free jet, we estimate a properly focused e-beam to be ~1 mm in diameter.

Ions can also be produced by a pulsed high-pressure glow discharge source which was developed in our laboratory. The source, shown in Figure 5, consists of a 2 mm long, 0.5 mm diameter channel and a 10 mm long, 1 mm diameter discharge cell, and is attached directly to the nozzle of the pulsed valve. Two sharp nickel electrodes extend into the cell and are separated by less than 1 mm. After a short delay from the opening of the pulsed valve, the pressure inside the cell reaches the maximum (typically ~100 Torr). At that moment, a high voltage pulse (1.5 - 3 kV) is applied across the electrodes to initiate the glow discharge. This high voltage pulse is produced by a thyratron, since it is much less susceptible to damage than its silicon-based counterparts. The source conditions are optimized to produce a high-pressure glow discharge without initiating an arc discharge. Ions formed in the discharge undergo many thermal collisions as the gas flows out of the discharge cell and expands into the vacuum, leading to the generation of a cold cluster ion beam.

Successful use of these ionization techniques requires some basic knowledge of the mechanism of cluster ion formation. For the positive ions, clusters are formed by sequential association of neutral molecules onto an ionic core. The ion-neutral bimolecular collision is governed by the long-range ion-dipole interaction, and in the free jet, its rate is two orders of magnitude higher than that of collision between neutrals. Due to this enhanced collision rate, significant clustering growth of ions is possible in a jet expansion. However, the rate of collision is strongly dependent on the gas density, which decreases drastically along the expansion axis of a free jet. To ensure cluster growth, ionization should occur near the nozzle, where the gas pressure is high.

In the e-beam technique, high-pressure ionization is achieved by focusing the electron beam tightly onto the free expansion at ~2 mm downstream from the nozzle. In the pulsed-discharge technique, it is accomplished by placing the electrodes very close (~3 mm) to the nozzle and using a 1 mm diameter channel to maintain a high gas pressure after the ionization. A longer channel can be used to make larger clusters at the expense of ion intensity.

The distance between the ion source and the skimmer is another important experimental parameter. Putting the ion source far away from the skimmer causes a reduction of the ion signal. On the other hand, if the distance is so short that the skimmer lies deep inside the "zone of silence" of the supersonic expansion, shock waves caused by the skimmer can disturb the expansion significantly.[3] In addition, a higher gas load to the TOF section would also give rise to more background metastable fragments. To facilitate a convenient adjustment of the source-skimmer distance, our whole ion source assembly is mounted on a rail system coupled to a motion feedthrough. The optimum value for the source-skimmer distance lies between 4 - 7 cm in most experiments.

2.2.2 TOF Mass Spectrometer

Most of the components that make up the TOF mass spectrometer are depicted in figure 1. This mass spectrometer consists of a pulsed extraction region, two acceleration regions, three sets of einzel lenses, and two sets of deflection plates (not shown in figure 1), with the rest being regions where ions travel field-free. The TOF mass spectrometer is operated so that the maximum mass resolution is achieved at "the laser interaction spot," 175 cm downstream from the last acceleration plate where the ion beam and the laser beam intersect.

The operation of the TOF mass spectrometer will be described for the case of positive ions (conversion to a negative ion mass spectrometer can be accomplished by simply reversing the polarity of the potentials on the ion optics). The first acceleration plate is held at a DC positive potential (~1 kV) and the last acceleration plate is held at ground. The extraction plates are also held at ground potential initially. After passing through the skimmer, the ionized jet expansion reaches the pulsed-extraction region. Since the first acceleration plate is held at a positive voltage, a retarding electrical field exists between the last extraction plate and the first acceleration plate, preventing the positive ions from penetrating beyond the extraction region. When a maximum concentration of positive ions is reached in the extraction region, a high voltage pulse is applied to the extraction plates, raising the front plate to ~2.5 kV and the back plate to ~2.2 kV. (The high-voltage pulse circuit is shown in figure 6). The ions within the pulsed-extraction region have therefore immediately acquired a potential energy of 2.2 - 2.5 keV, depending on their position in the extraction region. This high voltage pulse also creates two acceleration fields, one between the first and the last extraction plates (2.5 kV → 2.2 kV) and the other between the last extraction plate and the first acceleration plate (2.2 kV → 1 kV). These two fields, along with the DC electrical field between the first and last acceleration plates, push the positive ions downstream, converting their potential energy into the kinetic energy.

In their subsequent field-free journey, the ions start to separate from one another according to their mass. This is because for ions with same kinetic energy, those with smaller mass have larger velocity and vise versa. (This is the basic principle of TOF mass spectrometry.) But since the kinetic energy of each ion is not exactly the same (ranging from 2.2 to 2.5 keV, depending on the initial longitudinal position of the ion), the packets for ions of the same mass would eventually start to diverge. Our TOF mass spectrometer is designed so that the mass resolution (the highest mass at which two neighboring mass peaks can still be separated) is at its maximum at the laser interaction spot. Along the trajectory, the ion beams are collimated by the einzel lenses and corrected by the deflection plates.

Our TOF mass spectrometer differs from those of conventional design. To maximize the collection volume of the mass spectrometer, we pulse-extract the ions along the expansion axis, rather than employing the usual scheme of accelerating ions perpendicularly. In our configuration, ions fills the entire extraction region instead of being confined in a smaller region longitudinally, thus reducing the mass resolution of our TOF spectrometer. To compensate for this loss of mass resolution, we have employed a new design for the acceleration electrical fields in our TOF spectrometer.

A majority of TOF mass spectrometers employ the Wiley-McLaren design,[7] in which the extraction field is followed by only one acceleration field. Specifically, if the extraction region has a length of s_0 and an electrical field E_s, the acceleration region has a length of d and an electrical field E_d, the field-free flight length is D, for an ion that is located in the extraction region with a distance of s from the final extraction plate with an initial kinetic energy of U_0 at the time of pulse-extraction, the time-of-flight formula is

$$T(U_0,s)=\sqrt{2m}\left(\frac{\sqrt{U_0+qsE_s}-\sqrt{U_0}}{qE_s}+\frac{\sqrt{U_0+qsE_s+qdE_d}-\sqrt{U_0+qsE_s}}{qE_d}+\frac{D}{\sqrt{U_0+qsE_s+qdE_d}}\right).$$

By imposing dT/ds = 0 at $s = s_0/2$ (midpoint of the extraction region) and $U_0 = 0$ (ions having no initial velocity), we can find an optimum value of, say, E_d, such that the first-order spacial focusing is achieved. The choice of $U_0 = 0$ is justified, since the kinetic energy of a supersonic expansion is ~1 eV, much smaller than the final ion energy of 2.5

KeV. However, second-order focusing ($d^2T/ds^2 = 0$) cannot be accomplished with the Wiley-McLaren configuration with parameter values that are experimentally realistic. This severely limits the mass resolution achievable if the Wiley-McLaren spectrometer is used in our apparatus. Specifically, the (spatial) mass resolution of a Wiley-McLaren spectrometer can be approximated as follows

$$M_s \approx 16\ (\ 1 + 2\ d\ E_d\ /\ s_0\ E_s)\ (s_0/2\Delta s)^2\ \ .$$

In our case where the ions are spread out, $s_0/2\Delta s = 1$, giving rise to a theoretical mass resolution of ~300 amu. Actual resolution will be much lower when the spread of initial ion energies is taken into account.

In our apparatus, an additional acceleration field E_x of length x is added to the Wiley-McLaren arrangement (figure 7). The time-of-flight formula then becomes

$$T(U_0,s) = \sqrt{2m}\Big(\frac{\sqrt{U_0+qsE_s}-\sqrt{U_0}}{qE_s} + \frac{\sqrt{U_0+qsE_s+qdE_d}-\sqrt{U_0+qsE_s}}{qE_d} + \frac{\sqrt{U_0+qsE_s+qdE_d+qxE_x}-\sqrt{U_0+qsE_s+qdE_d}}{qE_x} + \frac{D}{\sqrt{U_0+qsE_s+qdE_d+qxE_x}}\Big).$$

Analytical and numerical calculations reveal that simultaneous first and second order spatial focusing can be achieved with a set of parameter values that are experimentally realistic. For our apparatus, the calculations yield an estimate of the spatial mass resolution of our TOF spectrometer at ~5000 amu. Experimentally, we are able to improve the mass resolution of our apparatus to ~350 amu. The disparity between the theory and experiment is due to the spread in the initial kinetic energy of the ions, which can be reduced with more careful design of the ion source. The improvement in mass resolution with the new triple-field design was also discovered by De Heer *et al.*,[8] who have recently presented a detailed analysis for the special case in which the second electric field is zero.

Mass Gate

Following the inspection of the mass spectrum and the optimization of the intensity of the ion species that we intend to study, a device is activated for parent ion selection. This device, called the mass gate, is located ~5 cm before the laser interaction spot. It consists of two deflector plates 1 cm in length and 1.5 cm apart from each other, as well as two shielding plates with pinholes in the center (figure 8). The ion beam passes through the 6 mm diameter pinhole of the front shielding plate, making it the beam-defining pinhole of our apparatus. When the mass gate controller circuit is turned off, both deflector plates are held at ground potential, causing no deflection of the ion beam. When the circuit is turned on, a high voltage pulse is applied between the plates. This pulse is normally at 300 V, deflecting the ions significantly and causing them to miss the ion detector; only when the ions of the selected mass arrive, is it switched to 0 V. Consequently, an ion beam of a single species is produced (barring the rare situation in which two different species have the same mass), and the subsequent infrared spectroscopy will be performed on this ionic species.

2.2.3 Fragment Ion Detection

At the laser interaction spot, the ions of selected mass are intersected by a tunable infrared laser beam. The resulting fragment ions are mass-analyzed by a reflectron spectrometer, which consists of a 14 cm long retarding electrical field (tilted 4° with respect to the beam axis) and a 30 cm long field-free region, and detected by a MCP detector. The signal from the MCP is amplified by a Comlinear CLC-100 preamplifier and recorded by a LeCroy 8818/6010 waveform digitizer/signal averager.

Because the focused ions at the laser interaction spot have slightly different kinetic energies, the ion beam will diverge, with the higher energy ions moving in front. However, higher energy ions of a given mass spend longer time in the reflectron field than lower energy ions of the same mass. Therefore, at some distance the effects listed above cancel each other, and if a detector is placed at that point, a refocused ion beam will be observed. Specifically, if E_r is the reflectron electrical field tilted at angle ϕ with respect to the axis of the ion beam, L the length of the drift region, E_B the beam energy, then the arrival time of a parent ion with mass m_p is

$$T(m_p,E_r)=\frac{2m\sec\phi}{eE_r}\sqrt{\frac{2E_B}{m_p}}+\frac{L\sec2\phi}{\sqrt{\frac{2E_B}{m_p}}}\ .$$

By choosing an electric field E_r so that $\mathrm{d}T/\mathrm{d}E_B = 0$, the arrival time of the parent ion is, to first order, independent of the initial energy. This energy-focusing ability has made the reflectron a popular device in mass spectrometry.

In our experiment, the energy-focusing ability of the reflectron is combined with its capability to perform fragment analysis. The principle is discussed here. Since the fragment ion (mass m_f) and its parent ion (mass m_p) have the same velocity, they behave identically in the field-free region. But they would act differently inside the reflectron field, due to their different masses (m_f vs. m_p) and energies ($(m_f/m_p)\ E_B$ vs. E_B). Fortunately, it can easily be found that the fragment ion inside a reflectron field (m_f/m_p) E_r will behave identically as the parent ion inside the original reflectron field E_r, i.e.,

$T(m_f, (m_f/m_p)\ E_r) = T(m_p, E_r)$. Therefore, under the new reflectron field of $(m_f/m_p)\ E_r$, the fragment ions will be refocused at the detector and have the same arrival time as the parent ions in the *original* field. Furthermore, from the ratio of the two reflectron fields, the mass of the fragment ion is also deduced.

2.3 OPTICAL PARAMETRIC OSCILLATOR

The pulsed tunable infrared laser we use to excite the ions is a $LiNbO_3$ optical parametric oscillator (OPO).[9] An OPO is based on the parametric interaction of a strong pump beam with molecules in a crystal which has a sufficiently large nonlinear susceptibility. This interaction can be described as an inelastic scattering of a pump photon ω_p by a molecule where the pump photon is absorbed and two new photons ω_s ("signal") and ω_i ("idler") are generated. Because of energy conservation, the frequencies ω_s and ω_i are related to the pump frequency ω_p by

$$\omega_p = \omega_s + \omega_i \quad .$$

The most efficient OPO generation is achieved in collinear phase matching where all three beams are parallel to each other. In those cases, the phase-matching condition (*i.e.* momentum conservation) is given by

$$n_p\ \omega_p = n_s\ \omega_s + n_i\ \omega_i \quad .$$

$LiNbO_3$ is an uniaxial crystal. In a $LiNbO_3$ OPO, the pump beam is an extraordinary wave and the signal and idler beams are both ordinary waves (i.e. $e \rightarrow o + o$, type I phase-matching). The preceding phase-matching equation can therefore be rewritten as follows

$$n_e(\theta, \omega_p)\ \omega_p = n_o(\omega_s)\ \omega_s + n_o(\omega_i)\ \omega_i \quad .$$

An OPO can be tuned by changing either the orientation or the temperature of its crystal. Compared to temperature tuning, angle tuning is faster and easier to implement. Figure 9 shows the angular tuning curve of $LiNbO_3$ when pumped by Nd:YAG laser, in which a 4° change in the crystal orientation covers a wide tuning range from 1.4 μm to 4 μm (1.4-2.1μm for the signal and 2.1-4 μm for the idler). In most of our experiments, the low frequency component of the OPO output (i.e. the idler) is used to excite the ions.

Our OPO follows the design of McDonald *et al.*[10] Its schematic diagram is shown in figure 10. The optical cavity of the OPO is singly resonant, with the high frequency wave ("signal") being the resonant wave. It is made as short as possible. (The optical length of the cavity is 20 cm.) The main components of the OPO are a 5 cm long, 1.0 cm^2 cross section $LiNbO_3$ crystal (Crystal Technology), a set of ZnSe prisms (Janos) as beam expander, a 600 line/mm blazing-angle grating (Bausch and Lomb) as the rear mirror as well as a linewidth-narrowing element, a 2 mm thick Infrasil etalon (CVI) of either 60% or 93% reflectance for further linewidth reducing, and a partial reflector ($R \approx 60\%$ at 1.5 μm, CVI) as the output coupler. The crystal, grating, and etalon are all set in rotational mounts with one degree of freedom, and their angular motions are controlled by DC stepping motors with an accuracy of $\sim 2 \times 10^{-4}$ degree per motor step. The output coupler is mounted on a piezoelectric translational stage (Burleigh Model PZS-50), which has a moving range of 35 μm with nanometer resolution. The $LiNbO_3$ crystal is placed in a brass housing which can be temperature stabilized actively to ~10 mK precision. All the optical components of the OPO are mounted on a 1 cm thick board made of Super-Invar, an iron-nickel alloy with an extremely small thermal expansion coefficient ($\sim 3 \times 10^{-7}$ K^{-1}).

The OPO is pumped by a 10 Hz Spectra-Physics GCR-12s Nd:YAG laser which can generate 200 - 300 mJ, 10 - 15 ns long pulses at 1.06 μm. The YAG laser contains an injection seeder, and when properly seeded, the laser produces a linewidth of ~0.001 cm^{-1} by lasing on a single longitudinal mode. The pump beam is multi-passed 4 meters to achieve a nominally Gaussian intensity profile with a 5 mm spot size. Because of the low damage threshold of $LiNbO_3$ (~500 MW cm^{-2}),[11] multi-passing is necessary to remove the "hot spots" in the pump beam. The multi-passed YAG beam is then coupled into the oscillator cavity by a thin optical element which is coated with a standard 45^0 YAG reflection coating. As indicated by the tuning curve, the OPO is continuously tunable from 2500 cm^{-1} to 6900 cm^{-1}. Lasing action of the OPO beyond that range is prevented by a strong absorption of $LiNbO_3$ at ~2200 cm^{-1}.

The operation of the OPO is controlled by a IBM-PC computer nicknamed "LASER" (figure 11). The two dual-channel stepping motor controller boards (Keithley MSTEP-5) and an integrated data acquisition and control board (Keithley DAS-16) are installed in the computer as the interfacing devices. The MSTEP-5 boards control the rotation of the crystal, grating, and etalon by sending signals to their stepping motor drivers. The DAS-16 board contains an external trigger input, a programmable timer, and 12-bit analog-to-digital ports. The timer is activated upon the reception of a TTL-high at the trigger input, and after a programmable delay, the analog-to-digital ports are ready for data acquisition. These ports are used to characterize the OPO. One port measures the pulse energy of the OPO beam by digitizing the output from a Molectron J25LP joulemeter. Another port measures the photoacoustic signal whose spectrum is used to calibrate the wavenumber of the OPO beam. The DAS-16 board also has a 12-bit digital-to-analog port, which is used to adjust the OPO cavity length by controlling the piezo-stage of the output coupler through its output voltage signal.

The software we developed for the OPO is called *OPODATA*. Through the interface boards, *OPODATA* controls and keeps track of the angular positions of the crystal, grating and etalon as well as the OPO cavity length. *OPODATA* facilitates different scanning algorithms for the three different configurations of the OPO (low resolution, medium resolution, and high resolution configurations). The details of each OPO configuration and its scanning algorithm will be described in the following paragraphs.

2.3.1 Low Resolution Configuration

Since the crystal bandwidth $\Delta\nu_c$ of $LiNbO_3$ varies from 10 cm^{-1} to 200 cm^{-1}, linewidth reduction is necessary for the $LiNbO_3$ OPO. The principles of linewidth control in a singly resonant OPO can be found in a 1979 paper by Brosnan and Byer.[11] In the low resolution (~1 cm^{-1}) configuration of the OPO, the grating is the only linewidth control element. The intra-cavity etalon is removed and the controller for the output coupler piezo-stage is turned off. The OPO linewidth $\Delta\nu$ is the result of the crystal and grating component lineshape functions after consideration for multiple passes in the optical cavity. Using a Gaussian approximation for line shapes, we may write the OPO linewidth $\Delta\nu$ as follows

$$\frac{1}{\Delta\nu}=\sqrt{\frac{1}{[\Delta\nu_c/\sqrt{p}]^2}+\frac{1}{[\Delta\nu_g(p)]^2}}$$

where p is the number of cavity transits by the signal wave. The linewidth due to the grating, $\Delta\nu_g(p)$, which can be calculated with a Gaussian approximation for the transverse profile of the pump beam; it is found to be ~1.5 cm^{-1} for our OPO. Therefore, the total linewidth of the OPO is around 1 cm^{-1}. One of the reasons for the narrow linewidth is the ZnSe prism set, which expands the beam by a factor of ~5.7, significantly increasing the resolution of the grating.

In order to be resonant in the OPO cavity, the signal wave must be reflected back 180° when it encounters the grating. Therefore, the wavelength λ_s of a possible signal wave is primarily determined by the grating angle through the equation

$$m\,\lambda_s = 2d\sin\theta_g\ .$$

In our case, d = 1.67 μm and m = 1. However, lasing will not occur unless the crystal angle is tuned so that the wavelength of its signal wave from the tuning curve is very close to λ_s (within ~5 cm^{-1}). This is because only at those crystal angles can the signal wave experience sufficient gain from the crystal. To sum up, there is an optimum crystal position for lasing at each grating position, and every grating position corresponds to a signal wavenumber.

Experimentally, the relationship between the grating position and its corresponding crystal position is obtained by first recording a number of these position pairs that give rise to a maximum OPO output, then fitting these crystal positions to a cubic polynomial function of the grating position. We then go on to calibrate the OPO with a photoacoustic cell (the details of which will be given in the next section), thus obtaining many sets of grating positions and their corresponding signal or idler wavenumbers. These grating positions are subsequently fitted to a cubic spline function of the idler wavenumber.

With the parameters of both functions stored in the computer, scanning of the low resolution OPO can be achieved readily with *OPODATA*. To make the OPO lase at a certain idler wavenumber, the computer simply calculates its corresponding grating and crystal positions and moves the stepping motors to these positions. With this algorithm, the OPO is able to scan continuously over a range as wide as 400 cm^{-1}. Typical output energies of the low resolution OPO are 4-7 mJ for the idler and 6-12 mJ for the signal.

2.3.2 Medium Resolution Configuration

To resolve the rotational structure of the infrared spectra for most ions, an OPO resolution of 0.1 cm^{-1} or higher is necessary. To achieve the medium resolution, a 2 mm thick Infrasil etalon of ~60% reflectance is inserted into the (low resolution) OPO cavity. This etalon has a free spectral range of ~1.6 cm^{-1} and a finesse of ~7. Since the linewidth of the low resolution OPO (~1 cm^{-1}) is smaller than the separation between neighboring etalon modes (~1.6 cm^{-1}), the OPO will lase on a single etalon mode. The reasonably high etalon finesse reduces the OPO linewidth from ~1 cm^{-1} to ~0.1 cm^{-1}.

The relationship between the etalon angle θ_e and the signal wavelength is

$$\sin\theta_e = n\sin[\arccos(\frac{N\lambda_s}{2nd})]$$

where N is the interference order, while d and n are the thickness and the index of refraction of the etalon, respectively. The computer uses this relationship to move the etalon to the correct orientation after locating the crystal and grating to their corresponding positions for a certain idler wavenumber. This procedure allows the OPO to lase at that particular wavenumber. However, the scanning of the medium resolution OPO is complicated by the change of the interference order N during the scan, thus *limiting* the continuous scanning range to ~30 cm^{-1}. Typical output energies of the medium resolution OPO are 2-3 mJ for the idler and 4-6 mJ for the signal.

2.3.3 High Resolution Configuration

To achieve even higher resolution, the etalon of 60% reflectance is replaced by a etalon of 93% reflectance.[10] The high resolution configuration of the OPO has not yet been tested extensively. It should be pointed out that to ensure the proper operation of the high resolution OPO, the optical length of the cavity must be kept stable. The super-Invar board and the temperature stabilizer for the crystal are used for this purpose.

With the extremely high finesse of the etalon, the OPO is able to lase on a single longitudinal mode of the cavity. However, the OPO threshold is also significantly increased, making the precise match of the etalon angle with the cavity length very critical for the OPO lasing. Therefore, high resolution scanning of the OPO is very difficult, and the continuous scanning range is only ~2 cm^{-1}. The high resolution OPO could produce a output with a intensity of ~1 mJ and a Fourier-transform limited linewidth of ~0.001 cm^{-1}.

2.3.4 Separation of the Signal, Idler and Pump Beams

In our experiment, instead of using the collinear configuration where the pump, signal, and idler beams are exactly parallel, we choose a *near-collinear* phase-matching configuration. Shown in figure 12, the pump beam is tilted slightly upwards (by ~0.5°) while the OPO cavity is horizontal. The phase-matching equation for the collinear configuration still holds very well in the near-collinear case. However, with our configuration, the three laser beams have slightly different propagation directions: the idler tilts upwards the most, the pump tilts slightly upwards, while the signal beam travels horizontally. This is very advantageous because the laser beams can be easily separated at a distance.

2.3.5 Wavenumber Calibration

The wavenumber calibration for the OPO is accomplished by using photoacoustic spectroscopy.[12] A brief description is given here:

An OPO beam is sent through a gas cell. If the wavenumber of the OPO beam is tuned to a rovibrational transition of the gas in the cell, some of the molecules that are located along the optical path are excited. By collisions with other molecules, these excited molecules transfer their excitation energy into translational, rotational, or vibrational energy of the collision partners. The redistribution of excitation energy into thermal energy causes a local increase of thermal temperature, which produces an acoustic wave that is detected by a sensitive microphone placed inside the cell. Therefore, the wavenumber of the OPO beam is calibrated by measuring the photoacoustic spectrum of a simple molecule and comparing its peak positions with the known locations of the molecule's transitions.

Our photoacoustic cell is a 10 cm long brass bar with a 1 cm diameter hole drilled through the entire length. CaF_2 windows are glued to seal the cell while providing passage for the OPO beam. The acoustic cell is filled with 5 - 50 Torr of the sample gas. The photoacoustic signal is detected by a Knowles EK-3031 microphone and amplified by a OP-27 operational amplifier before being measured by the 12-bit analog-to-digital port on the DAS-16 board. A typical photoacoustic spectrum is shown in figure 13.

By using different gas species and measuring the idler or the signal (utilizing the relation $\omega_p = \omega_s + \omega_i$), most of the OPO tuning range is accurately calibrated with the photoacoustic cell. Specifically, the idler tuning range is calibrated as follows:[13] 2650-3050 cm^{-1} by measuring the fundamental band of HCl with the idler, 2950-3200 cm^{-1} by measuring the ν_3 band of CH_4 with the idler, 3250-3550 cm^{-1} by measuring the $2\nu_3$ band of CH_4 with the *signal*, 3550-3950 cm^{-1} by measuring the first overtone band of HCl with the *signal*, 4000-4150 cm^{-1} by measuring the $\nu_1+\nu_5$ band of C_2H_2 with the idler, and 4200-4500 cm^{-1} by measuring the first overtone band of HBr with the idler.

2.4 SYSTEM OPERATION

To do an ion spectroscopy experiment, we carry out the following procedures. First, the operations of the ion apparatus and the OPO are optimized independently. Then, both the pump laser of the OPO and the Stanford Research System digital delay generator (model DG-535), which controls the pulse sequence of the ion apparatus, are switched to external triggering by an EG&G digital delay generator (model 9650). Finally, the delay time between the pulse sequences of the ion apparatus and the OPO is adjusted so that the selected ions and the OPO beam arrive at the laser interaction spot simultaneously. Only then are we ready for an infrared spectroscopic scan of the selected ion.

2.4.1 Ion Apparatus Operation

The pulse sequence of the ion apparatus is controlled by the Stanford Research System digital delay generator. A TTL-pulse from delay channel A triggers the pulsed valve driver. If the pulsed discharge ionization is used for the experiment, channel B goes to TTL-high ~200 μs later, triggering the thyratron, which initiates a discharge across the Ni electrodes. If the electron beam ionization is used, channel B is not used since the e-beam is continuous. About 400 μs after the pulse from channel A, channel C goes to TTL-high, triggering the TOF pulsed repeller circuit, marking the starting-time for the flight of the ions in the mass spectrometer. (Channel C is also connected to the trigger input of the mass gate controller, which has an internal delay circuit.) The resulting TOF mass spectrum is collected by the LeCroy waveform digitizer, which starts its data acquisition upon reception of a triggering pulse from channel D (usually set at 20 μs after the channel C pulse).

A TOF waveform trace is collected by the LeCroy with a single pulsing event. But to obtain a mass spectrum with a good signal-to-noise ratio, we need to collect the data for a large number of pulsing events and signal-average the data. This is accomplished by using an IBM-PC computer (nicknamed "TOFU"), which runs a computer program named *TOFVIEW2*. To collect data, "TOFU" activates the LeCroy waveform digitizer and sets its accumulative event counter N_s through a GPIB interface bus. (Usually N_s is set to 100, as a much larger number may cause data overflow in the memory buffer). The LeCroy accumulates the digitized TOF waveform trace into its memory buffer until its scan counter reaches zero, after which it stops and waits for the data transfer command from the "TOFU" computer. Upon the reception of that command, the TOF waveform data are retrieved to the computer memory and subsequently displayed on the monitor by *TOFVIEW2*. The LeCroy is then reactivated for the next N_s events. The preceding procedure repeats N/N_s times until the total event number N is reached. The resulting final TOF mass spectrum can be saved, displayed, and printed out using *TOFVIEW2*. The intensity of each individual mass peak can also be calculated (integrated) automatically by *TOFVIEW2*.

2.4.2 The Pulse Sequence of the OPO

The Nd:YAG laser which pumps the OPO can be triggered by its internal pulse circuit, or externally by the EG&G delay generator. When the laser is externally triggered, channel A of the EG&G triggers the flashlamp, and channel B triggers the Q-switch firing of the laser. Typical delay-times involved are: A − T_0 = 1000 μs, B − A = 120-160 μs. A laser beam at 1.06 μm emerges from the cavity of the YAG laser a few nanoseconds after the Q-switch firing; and after an additional delay of ~5 ns, the OPO is lasing.

2.4.3 Synchronization Between the Ion Apparatus and the OPO

Unless both the parent ion beam and the OPO beam arrive at the laser interaction spot simultaneously, vibrational predissociation will never occur. The synchronization of these two beams is achieved by using a master delay generator for the whole experiment, and finding the correct delay-time.

In our full experiment, the EG&G is the master delay generator. Besides triggering the pump laser of the OPO, it can also control the pulsing of the ion apparatus by connecting the output of its channel T_0 to the external-trigger input of the Stanford Research System delay generator. The SRS thus becomes a slave delay generator which is externally triggered by the EG&G master, and the adjustable delay-time which is responsible for the beam synchronization is the delay between channel A of the SRS and the SRS external-trigger input.

To get the correct delay-time at channel A that leads to beam synchronization, we carry out the following procedure. We first position a MCP ion detector, which is mounted on a motion feedthrough, at the laser interaction spot. With an oscilloscope, we observe a signal of the parent ion arriving at this position. Then we detect the laser pulse with a photodiode. The correct delay-time is found by adjusting the delay so that these two signals appear at the same time on the oscilloscope screen.

2.4.4 The Complete Experiment

The flowchart for the whole experiment is shown in figure 14. The ion infrared spectrum is recorded by stepping the wavenumber of the OPO beam and averaging the photofragment intensity for 400 pulsing events at each OPO wavenumber. Background fragment ion intensity is subtracted by firing the laser beam alternately at the parent ion arrival time (IN SYNC) and 10 μs before the ion arrival (OUT OF SYNC), and taking the difference. The fragment intensity data are sent from "TOFU", through a serial port connection, to the "LASER" computer where they are normalized with respect to the OPO laser intensity.

The procedure described above generates a vibrational predissociation spectrum of the parent ions. Usually, the signal-to-noise ratio of one spectrum is not satisfactory. To achieve a better signal-to-noise ratio, we repeat the experiment for about 10 times and average the resulting spectra.

2.5 REFERENCES

(1) Proch, D.; Trickl, T. *Rev. Sci. Instrum.* **1989**, *60*, 173.

(2) Gentry, W. R.; Giese, C. F. *Rev. Sci. Instrum.* **1978**, *49*, 595.

(3) For detailed information about pulsed supersonic molecular beam technique, see: *Atomic and Molecular Beam Methods, Volume 1*; Scoles, G., Eds.; Oxford University Press: New York, **1988**.

(4) Johnson, M. A.; Lineberger, W. C. In *Techniques for the Study of Ion-Molecule Reactions*; Farrar, J. M.; Saunders Jr., W. H., Eds.; Wiley-Interscience: New York, **1988**.

(5) Levander, D. J.; McCombie, J.; Pursel, R.; Scoles, G. *J. Chem. Phys.* **1987**, *86*, 7239. Gough, T. E.; Mengel, M.; Rowntree, P. A.; Scoles, G. *J. Chem. Phys.* **1985**, *83*, 4953.

(6) Erdman, P. W.; Zipf, E. C. *Rev. Sci. Instrum.* **1982**, *53*, 225.

(7) Wiley, W. C.; McLaren, I. H. *Rev. Sci. Instrum.* **1955**, *26*, 1150.

(8) De Heer, W. A.; Milani, P. *Rev. Sci. Instrum.* **1991**, *62*, 670.

(9) For a general introduction on optical parametric oscillator, see: *Optical Electronics*; Yariv, A.; Holt, Rinehart and Winston: New York, **1985**.

(10) Minton, T. K.; Reid, S. A.; Kim, H. L.; McDonald, J. D. *Optics Communications* **1989**, *69*, 289.

(11) Brosnan, S. J.; Byer, R. L. *IEEE J. Quantum Electronics* **1979**, *15*, 415.

(12) A general review on photoacoustic spectroscopy can be found in: West, G. A.; Barrett, J. J.; Siebert, D. R.; Reddy, K. V. *Rev. Sci. Instrum.* **1983**, *54*, 797.

(13) Calibrated molecular spectra can be found in: *Tables of Wavenumbers for the Calibration of Infrared Spectrometers*; the IUPAC Commission on Molecular Structure and Spectroscopy; Butterworth, Washington, D. C.; **1961**.

2.6 FIGURE CAPTIONS

Figure 1. Schematics of the experimental apparatus.

Figure 2. The schematic diagram of the gas inlet system.

Figure 3. The electronic circuit of the pulsed valve driver. The two power MOSFETs are in the push-pull configuration.

Figure 4. The schematic diagram of the fast ionization gauge.

Figure 5. Schematics of the pulsed high-pressure glow discharge source. As the gas pulse enters the discharge cell, 1.5-3.0 kV is applied between the nickel electrodes to initiate the discharge.

Figure 6. The pulse circuit for the TOF repeller. The 1 kV MOSFETs are stacked together to produce a pulse of higher voltage (up to 6 kV).

Figure 7. A simplified diagram of our TOF mass spectrometer. The exact dimensions are s_0 = 4.4 cm, d = 3.0 cm, x = 4.8 cm, and D = 175 cm.

Figure 8. The schematic diagram of the mass gate.

Figure 9. Angle tuned $LiNbO_3$ OPO tuning curve. The crystal is at 25 °C and is pumped

by a Nd:YAG laser (λ_p = 1.06 μm).

Figure 10. Schematic diagram of the optical cavity of the OPO. The etalon and grating tilt in the plane of the figure, and the $LiNbO_3$ crystal tilts out of the plane. The output coupler is mounted on a piezo translational stage.

Figure 11. A diagram illustrating the computer control of the OPO.

Figure 12. A vector diagram illustrating the separation of the signal, idler and pump beams. The pump beam is introduced into the horizonal OPO cavity with a slight upward tilt. The signal beam is horizontal since it is resonant in the OPO cavity. The idler beam, whose direction results from momentum conservation, tilts upward the most.

Figure 13. A photoacoustic spectrum of the low resolution OPO. The spectrum is the result of exciting the first overtone of HCl with the signal beam. The pump beam is at 9394 cm^{-1}. The ordinate of the spectrum is the idler wavenumber from the computer, and the numbers above the peaks are the idler wavenumbers of the corresponding HCl rovibrational transitions. Comparison of these numbers with the ordinate reveals that the wavenumbers from the computer are offset from the experimental values by 0 - 2 cm^{-1}.

Figure 14. The flowchart for the ion spectroscopy experiment.

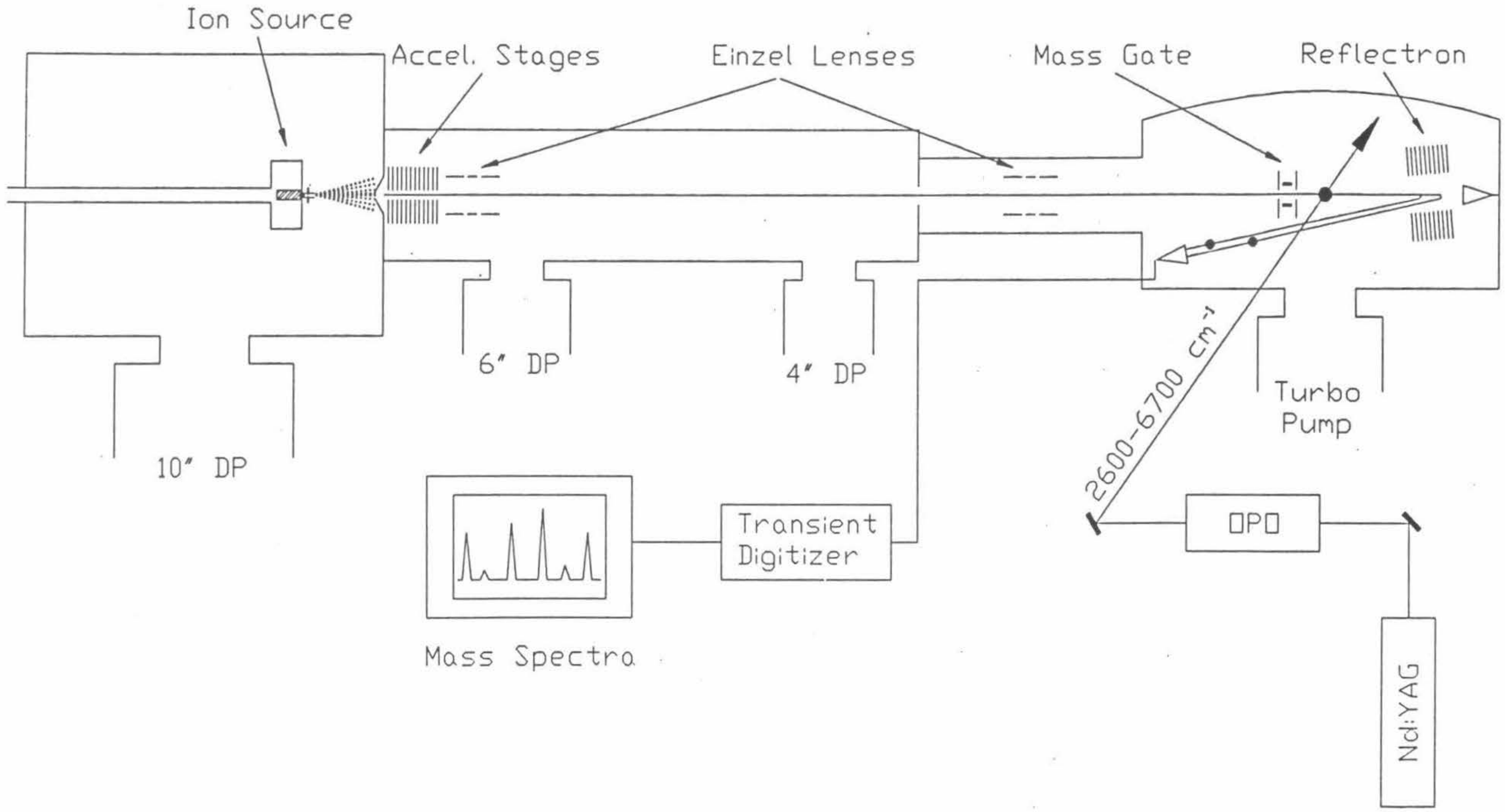

Ion Source
Accel. Stages
Einzel Lenses
Mass Gate
Reflectron
10″ DP
6″ DP
4″ DP
Turbo Pump
2600-6700 cm-1
Mass Spectra
Transient Digitizer
OPO
Nd:YAG

Figure 1.

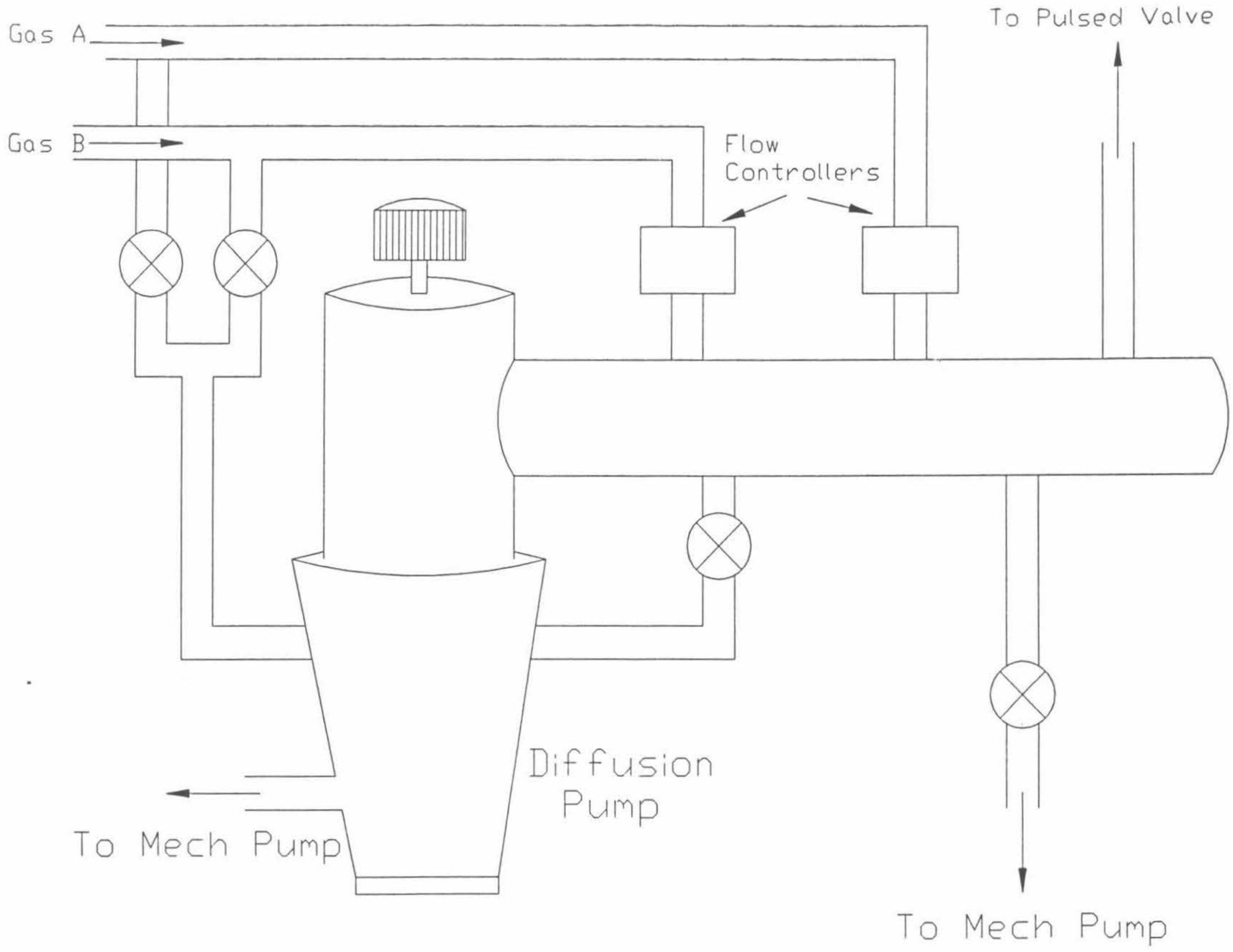

Gas A
Gas B
Flow Controllers
To Pulsed Valve
Diffusion Pump
To Mech Pump
To Mech Pump

Figure 2.

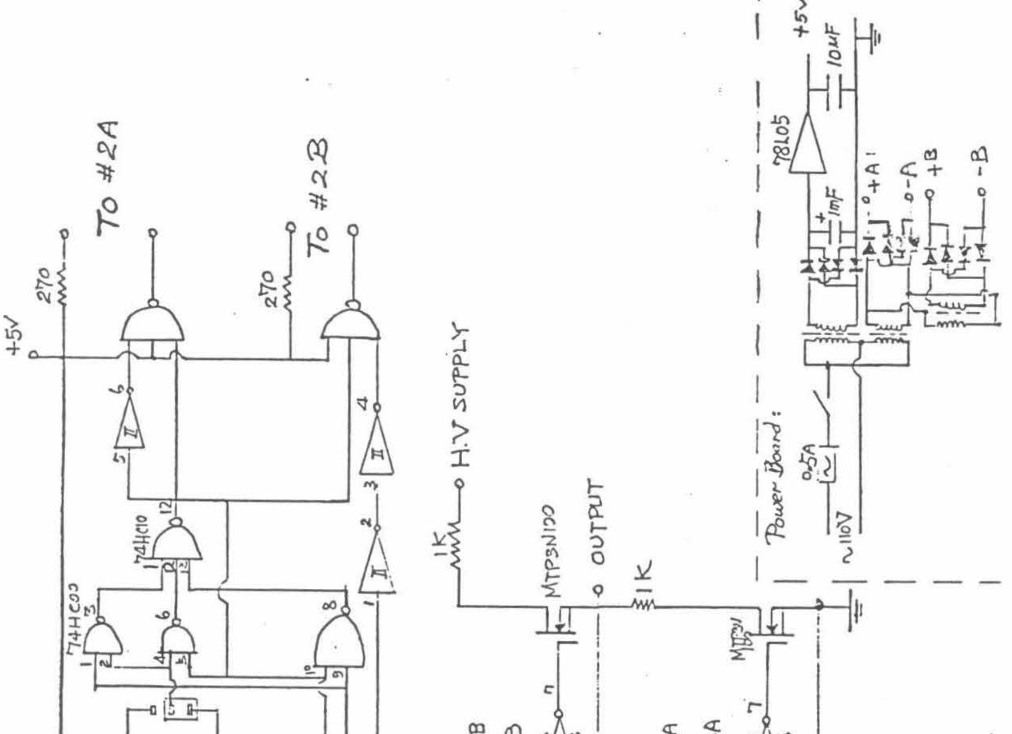
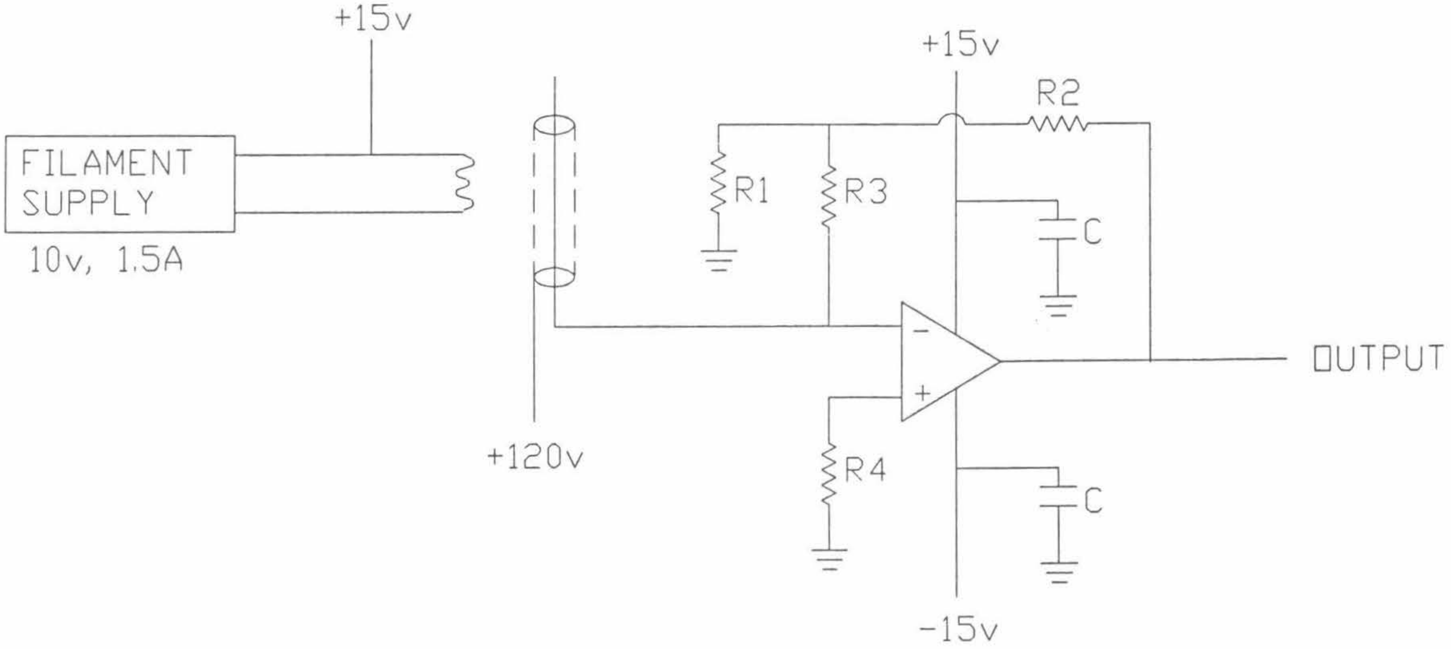

+5V
270
To #2A
270
To #2B
74HC00
74HC10
74HC14
SN74LS123
100k
7nF
27K
27K
1.8K
INPUT
H.V SUPPLY
1K
MTP3N120
OUTPUT
1K
78L05
+B
-B
1428
HCPL2601
2A
78L05
+A
-A
1428
HCPL2601
2B
Power Board:
0.5A
~110V
78L05
+5v
10uF
1mF
+A
-A
+B
-B

Figure 3.



R1= 5.1K; R2 = 225K; R3 = R4 = 820K; C = 2200 nF

Figure 4.

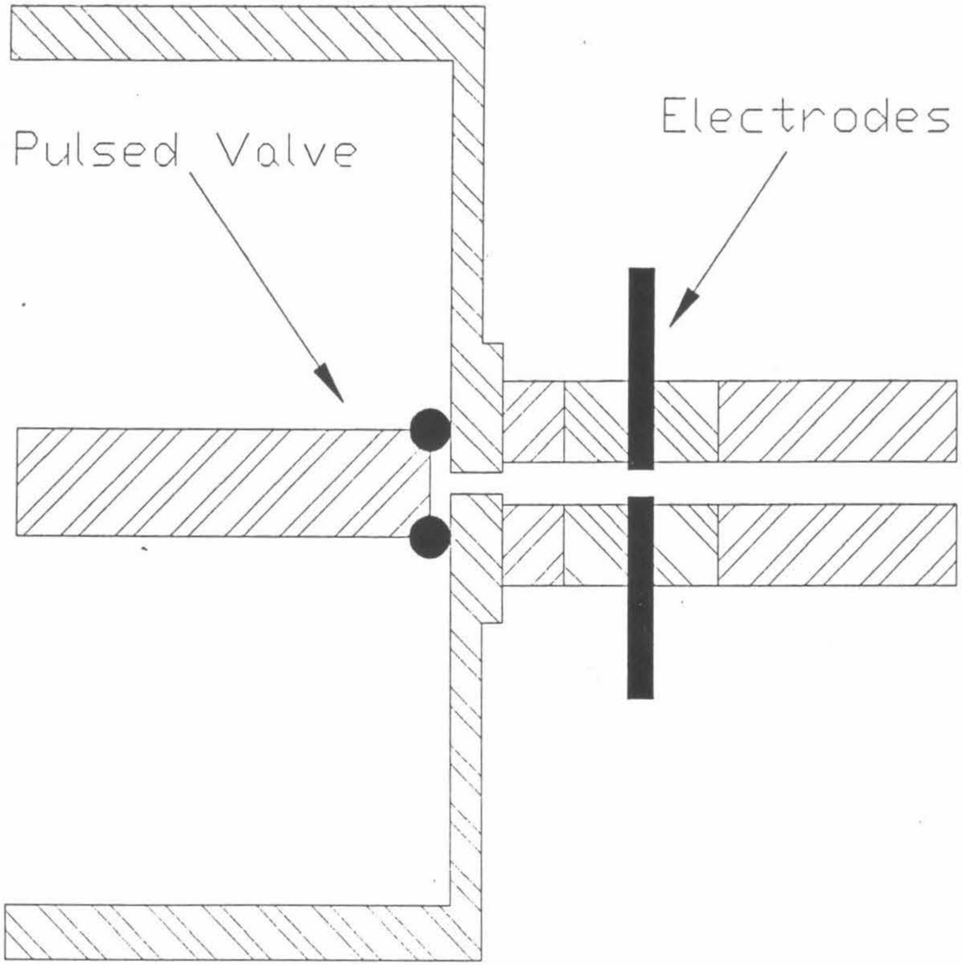


Figure 5.

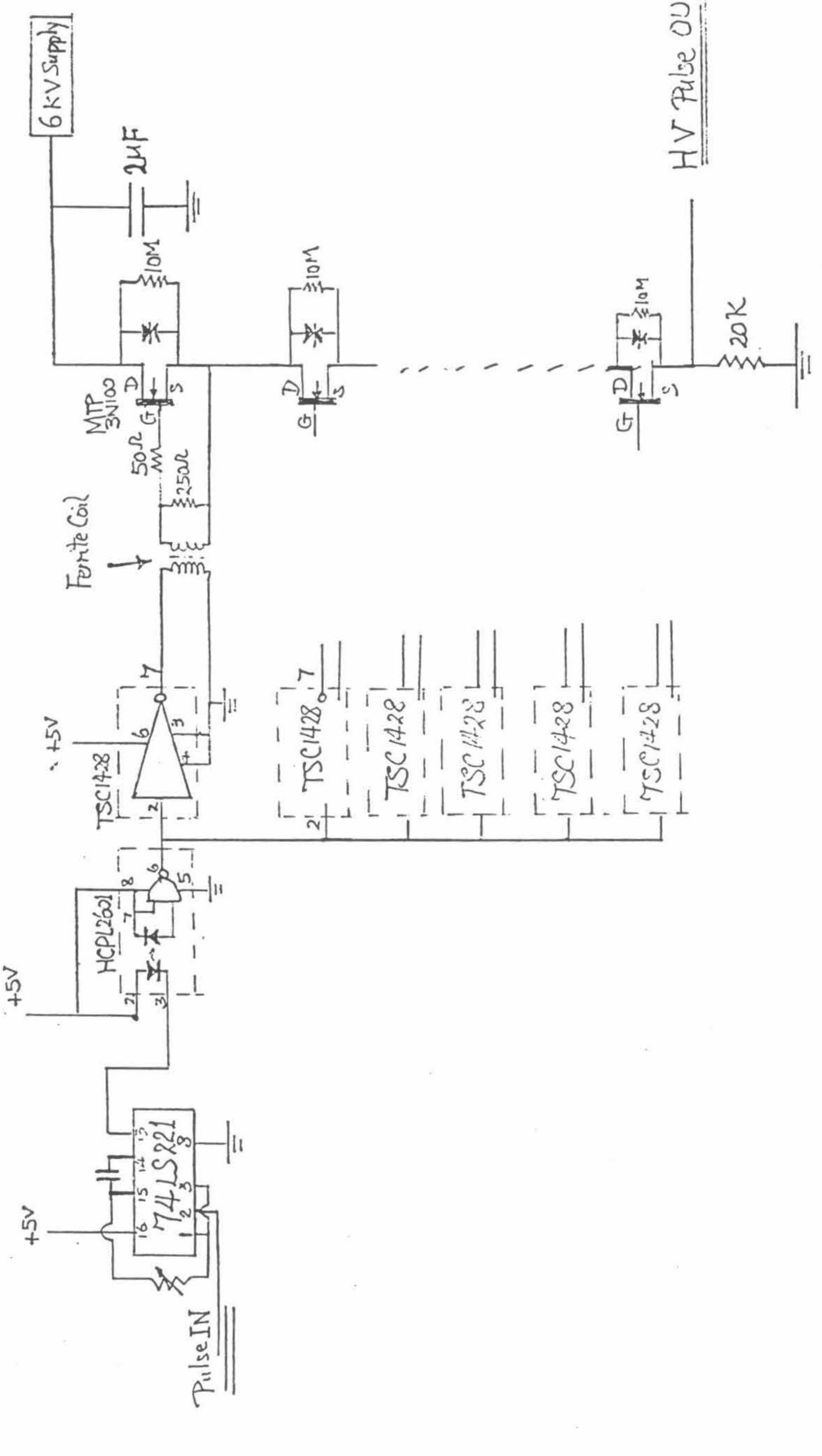

6 KV Supply
2μF
10M
10M
10M
MTP 3N100
D
G
S
D
G
S
D
G
S
50Ω
250Ω
Ferrite Coil
HV Pulse OUT
20K
+5V
TSC1428
6
3
4
2
7
TSC1428
2
7
TSC1428
TSC1428
TSC1428
TSC1428
HCPL2601
8
7
6
5
2
3
+5V
+5V
74LS221
16
15
14
13
8
1
2
3
Pulse IN

Figure 6.

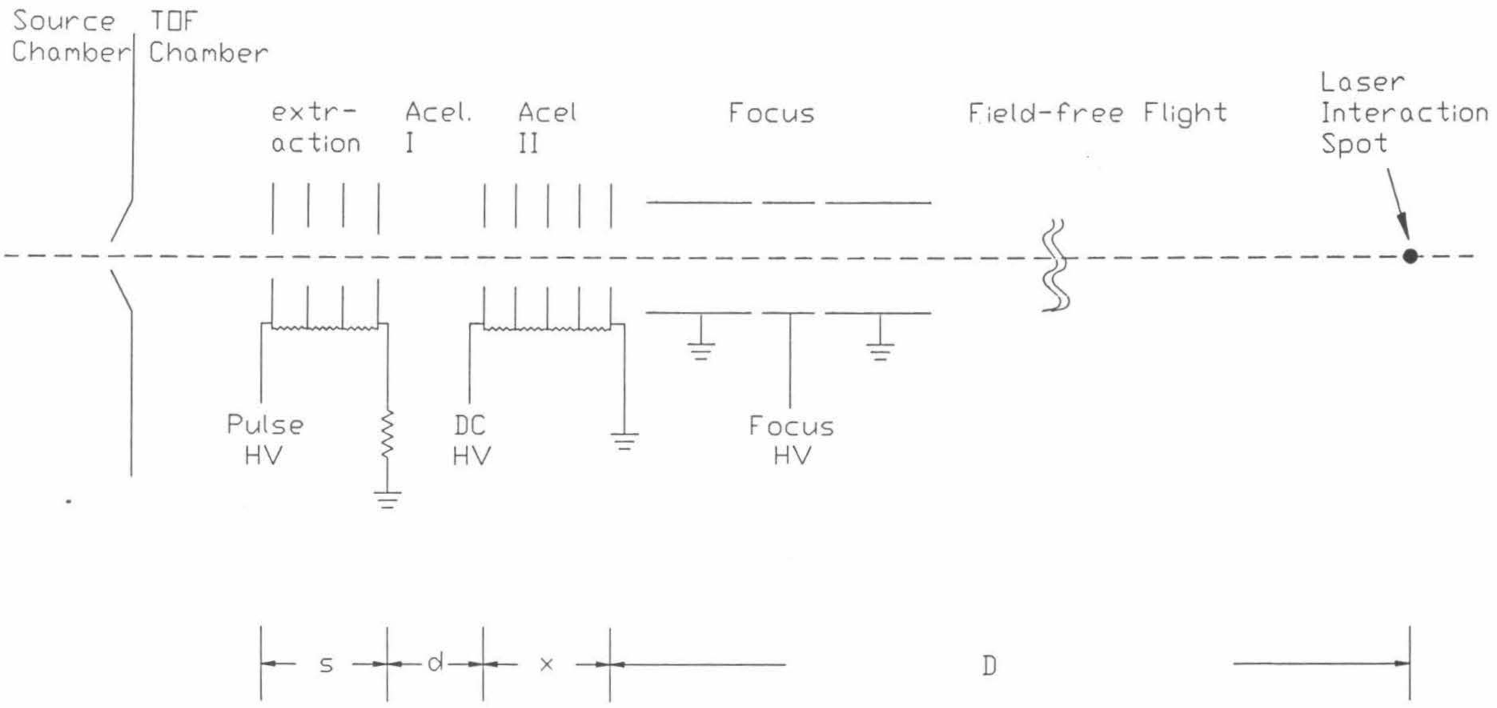

Source
Chamber
TOF
Chamber
extr-
action
Acel.
I
Acel
II
Focus
Field-free Flight
Laser
Interaction
Spot
Pulse
HV
DC
HV
Focus
HV
s
d
x
D

Figure 7.

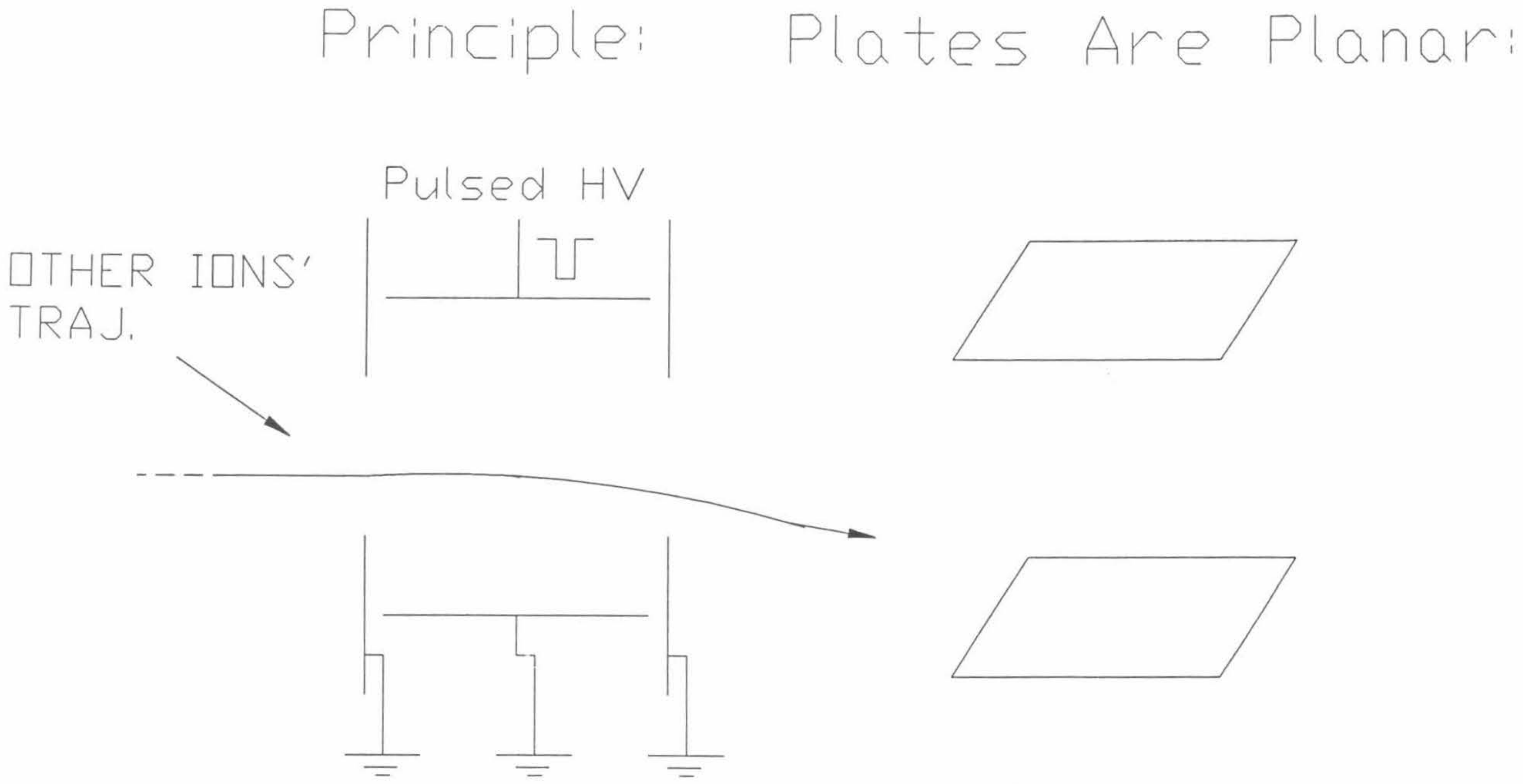


Figure 8.

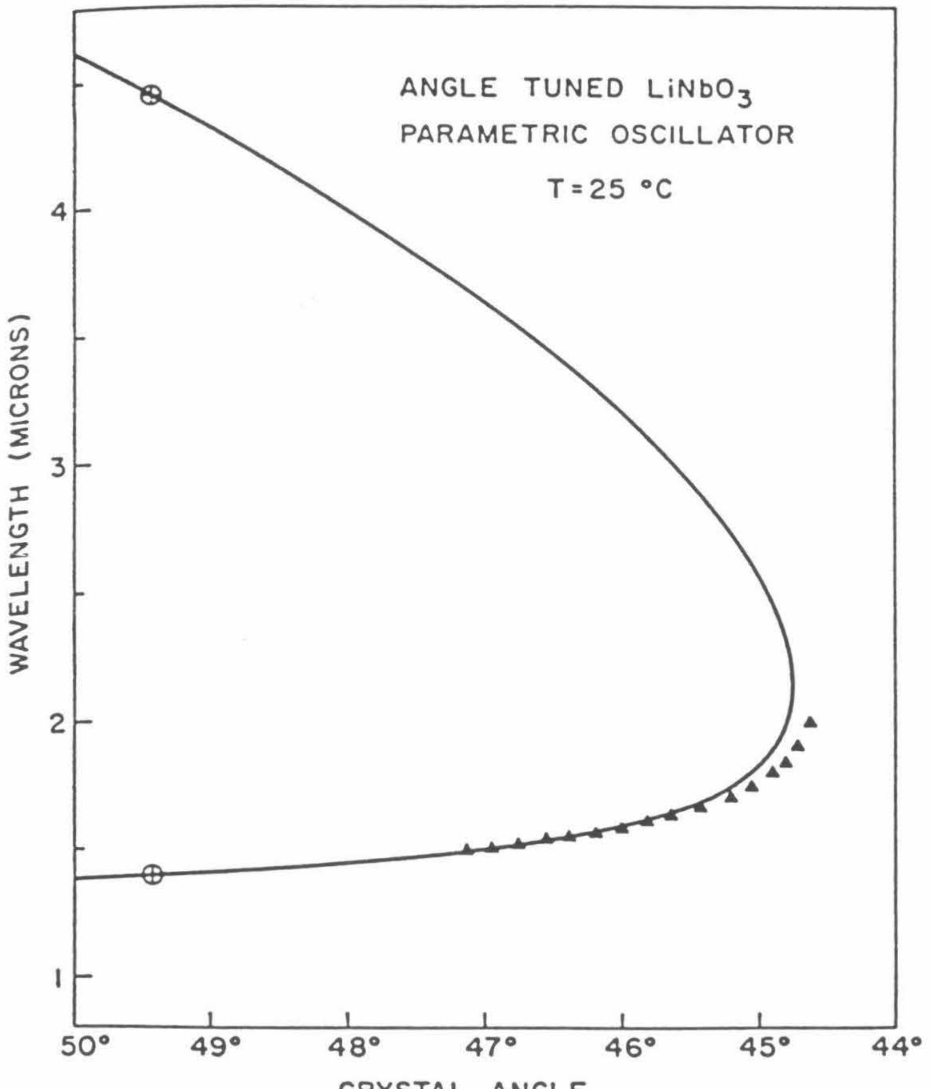

ANGLE TUNED $LiNbO_3$
PARAMETRIC OSCILLATOR
T = 25 °C
WAVELENGTH (MICRONS)
1
2
3
4
50°
49°
48°
47°
46°
45°
44°
CRYSTAL ANGLE

Figure 9.

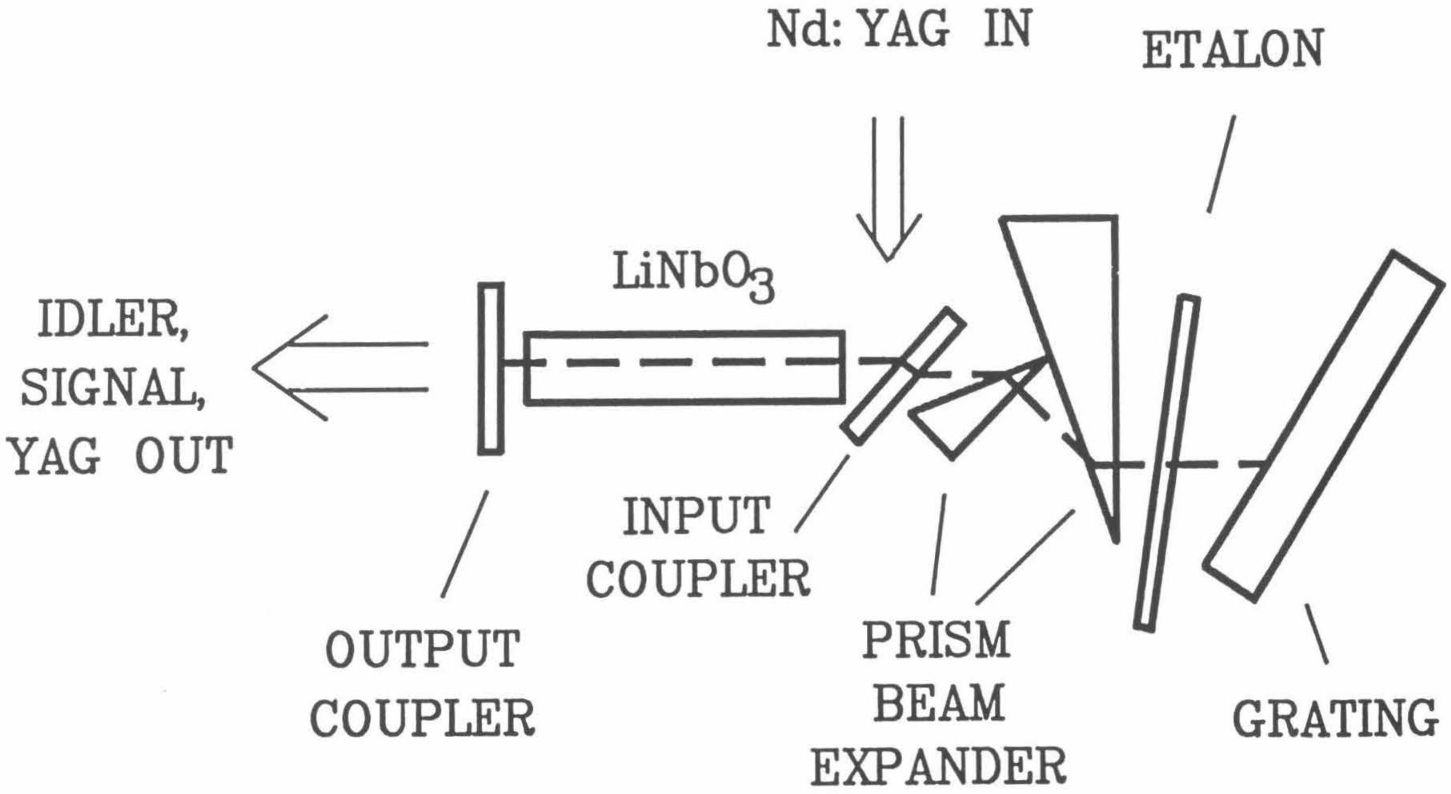

Nd: YAG IN
ETALON
LiNbO$_3$
IDLER,
SIGNAL,
YAG OUT
INPUT
COUPLER
OUTPUT
COUPLER
PRISM
BEAM
EXPANDER
GRATING

Figure 10.

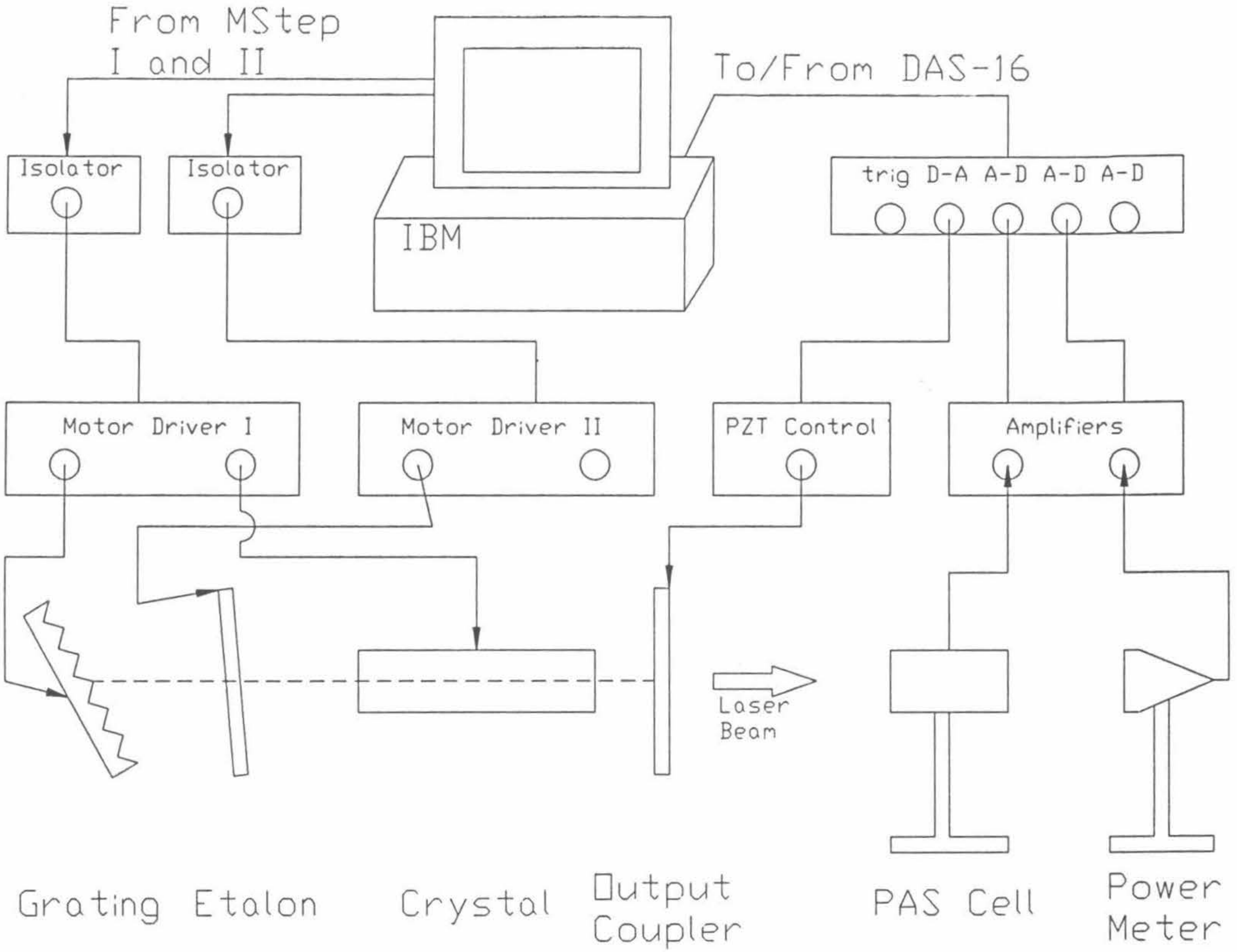

From MStep
I and II
To/From DAS-16
Isolator
Isolator
IBM
trig D-A A-D A-D A-D
Motor Driver I
Motor Driver II
PZT Control
Amplifiers
Laser
Beam
Grating
Etalon
Crystal
Output
Coupler
PAS Cell
Power
Meter

Figure 11.

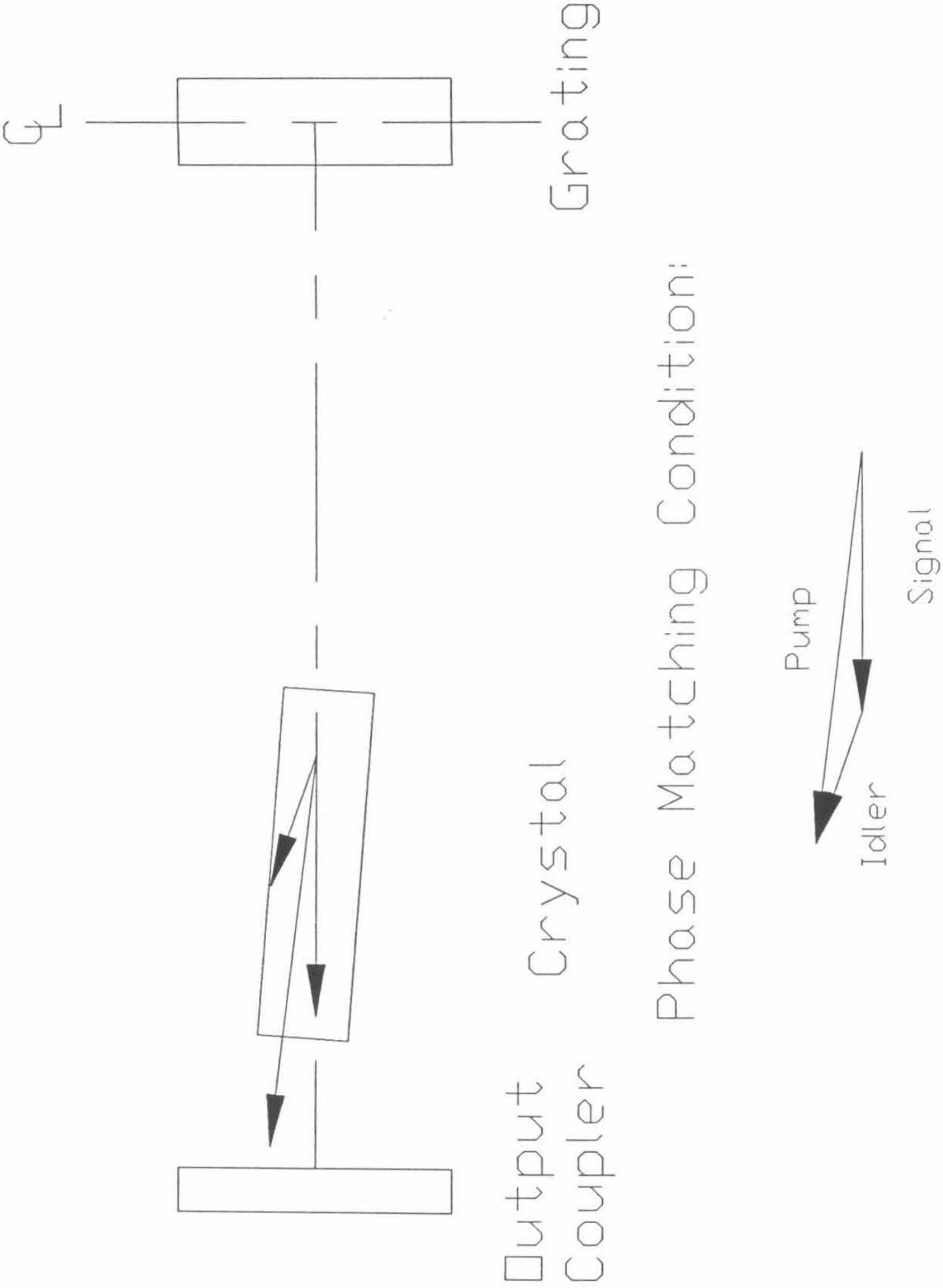

GL
Grating
Crystal
Output
Coupler
Phase Matching Condition:
Pump
Signal
Idler

Figure 12.

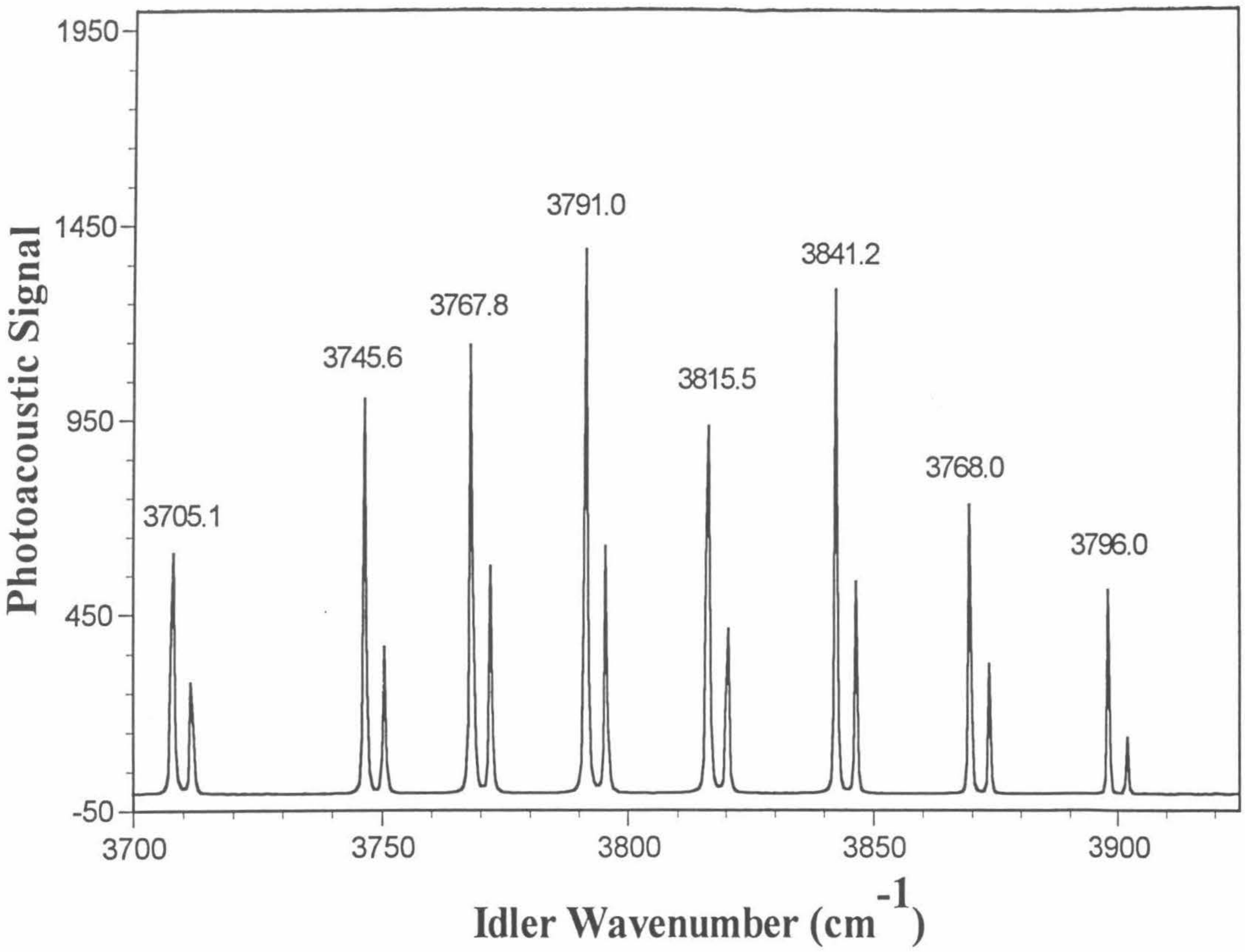

1950
1450
950
450
-50
Photoacoustic Signal
3705.1
3745.6
3767.8
3791.0
3815.5
3841.2
3768.0
3796.0
3700
3750
3800
3850
3900
Idler Wavenumber (cm^{-1})

Figure 13.

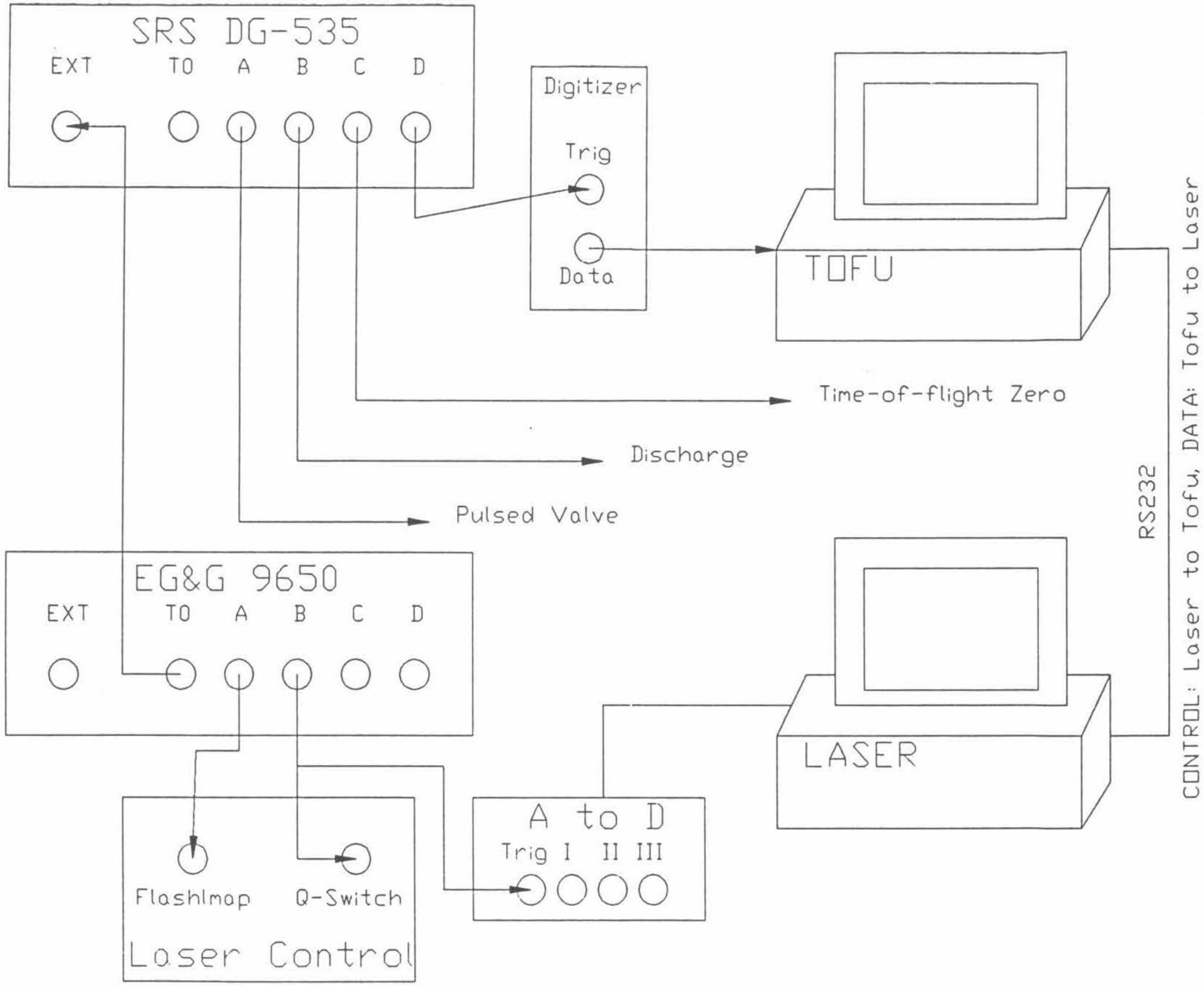

SRS DG-535
EXT
TO
A
B
C
D
Digitizer
Trig
Data
TOFU
Time-of-flight Zero
Discharge
Pulsed Valve
RS232
CONTROL: Laser to Tofu, DATA: Tofu to Laser
EG&G 9650
EXT
TO
A
B
C
D
LASER
A to D
Trig I II III
Flashlmap
Q-Switch
Laser Control

Figure 14.

Chapter 3. SiH_7^+

3.1 INTRODUCTION

Silanium ions are an important class of hypervalent molecules. Investigations of the chemistry of silicon hydride cations have been motivated by the possible role of ion-molecule reactions in the gas-phase silane chemistry occurring during chemical vapor deposition (CVD) of silicon films, especially plasma-enhanced CVD. Both experimental[2-10] and theoretical[11-14] studies have focused on reactions of Si^+ and SiH_3^+ with silane that produce hydrogenated silicon cluster ions $Si_nH_m^+$. The theoretical calculations[11-14] suggest that these cluster ions exhibit rather novel nonclassical bonding, analogous to but distinct from the nonclassical bonding observed in carbonium ions.[15] Such bonding is believed to occur even in monosilane cluster ions such as SiH_5^+.[16]

Although the gas-phase chemistry of silicon hydride ions has been studied extensively, there have been few spectroscopic studies to date. Whitham *et al.*[17-19] have obtained electronic predissociation spectra of SiH^+ and SiH_2^+ by using a fast beam apparatus. Dyke *et al.*[20] have reported the frequency of the out-of-plane bending mode of SiH_3^+ from the photoelectron spectrum of the SiH_3 radical. Smith, Martineau and Davies[21] have recently reported a high resolution infrared absorption spectrum of the ν_3 band of the SiH_3^+ cation detected in a glow discharge. SiH_3^+ is the dominant ion in low pressure silane discharges, and no other silanium species have thus far been detected by this method.

Based on the analysis of the rotationally resolved ν_3 band, the SiH_3^+ cation has a classical trigonal planar structure like CH_3^+. In the case of SiH_5^+, *ab initio* calculations by Hu *et al.*[16] suggest that an H_2 is bound to SiH_3^+ through a three-center, two-electron bond. This bonding is analogous to the case of CH_5^+, but the $SiH_3^+–H_2$ bond is much weaker. The best theoretical estimate of the dissociation energy D_0 for SiH_5^+ is 10.3 kcal/mol, which is less than the experimental value of 17.8 kcal/mol.[5] Larger silicon hydride cations are expected to be even more weakly bound. The SiH_7^+ ion has not been previously observed experimentally, and theoretical calculations on its structure have not yet been carried out.

The technique of vibrational predissociation spectroscopy has proved to be a powerful method for obtaining infrared spectra of mass-selected clusters.[22] We have applied this technique to study silicon hydride cations and our initial results on the infrared spectrum of SiH_7^+ will be presented in this chapter.

3.2 EXPERIMENTAL

The ion infrared spectrometer is described in detail in chapter 2. A brief description of the experiment is given as follows.

The SiH_7^+ ions were generated in a pulsed high pressure discharge source. A mixture of 1% silane in ultra-high purity (UHP) helium was mixed with UHP hydrogen using an MKS flow system to form a mixture of approximately 30 ppm SiH_4 in H_2 at 1000 Torr. The gas expanded through a piezoelectric pulsed valve with a pulse width of 200 µs into a 1 mm diameter, 1 cm long channel where a high voltage pulse was applied between two tungsten electrodes to initiate a glow discharge. The resulting plasma underwent supersonic expansion and was collimated by a skimmer into a second chamber, where the ions were accelerated by a time-of-flight mass spectrometer. The effectiveness of the pulsed discharge source in generating cold clusters was demonstrated by the formation of cluster ions with binding energies of 3 kcal/mol or less, such as $H_3O^+(H_2)$.

A typical TOF mass spectrum is shown in figure 1. The source, operated at a repetition rate of 30 Hz, produced intense and stable mono-silanium ion beams for hours during which no adjustments were needed.

The SiH_7^+ ions were excited by the idler beam of the $LiNbO_3$ OPO (low resolution configuration, linewidth at ~1.5 cm^{-1}). The resulting fragment ions were mass-analyzed by a reflectron spectrometer, and detected by a microchannel plate detector. The signal was amplified and then collected by a LeCroy 8818 transient digitizer and 6010 signal averager.

Spectra were recorded by stepping the OPO wavelength and averaging the photofragment signal for 400 laser shots at each wavelength. To achieve better signal-to-noise ratio, we averaged 10 such scans. Background fragmentation signal, primarily caused by collision-induced dissociation in the ion optics chamber, was subtracted. The data were then normalized with respect to the OPO laser intensity. The OPO laser wavelength was calibrated during the scan by simultaneously recording the vibrational spectrum of acetylene in a photoacoustic cell. The optical path from the OPO to the detection chamber was purged with dry air to eliminate absorption due to ambient water vapor.

3.3 RESULTS AND DISCUSSION

An absorption band of SiH_7^+ was observed in the 3800 - 3925 cm^{-1} region, as shown in figure 2. This band has clear P, Q, and R branches, with the Q branch centered at 3866 cm^{-1}. The only photofragment ion observed was SiH_5^+, indicating that the photodissociation process involved was $SiH_7^+ \rightarrow SiH_5^+ + H_2$. Due to the broad *linewidth of the* OPO laser (1.5 cm^{-1}), rotational structure of this band was not resolved. There were no other absorptions found from 3500 cm^{-1} to 4200 cm^{-1}.

On the basis of the relatively high frequency and the strength of this absorption band, we assign it to an H-H stretching motion. This stretch is strongly perturbed, as evidenced by its 295-cm^{-1} frequency shift from the free H_2 fundamental of 4161 cm^{-1}. This assignment supports the notion that SiH_7^+ is an ionic cluster with at least one H_2 molecule bound to the charge center. The H–H stretch obtains its transition dipole moment through coupling with degrees of freedom that possess oscillator strength, e.g., motion of the charge center relative to the center of mass.

The observed frequency of 3866 cm^{-1} for SiH_7^+ is close to the harmonic frequency of 3991 cm^{-1} for SiH_5^+ predicted by Hu *et al.*,[16] given that *ab initio* frequencies are generally 5-10% higher than experimental values. This agreement at first suggests that SiH_7^+ could be a complex with an H_2 molecule very weakly bound to the SiH_5^+ ion core, analogous to the structure proposed by Boo and Lee for the CH_7^+ ion (figure 3B).[23] However, the H_2 moiety would be perturbed by the SiH_5^+ core and therefore exhibit an absorption band slightly below 4161 cm^{-1}, the vibrational frequency of a free H_2. In all previous cases involving an ion-molecule cluster with H_2 ligands, such a shifted absorption band has been observed. In the case of CH_7^+ ion, the H_2 absorption band occurs at 4077 cm^{-1}.[23] We scanned the region from 3500 cm^{-1} to 4200 cm^{-1}, but SiH_7^+ showed no evidence of a second absorption band. In addition, we observed that the SiH_7^+ ions could exist even under "hot" source conditions that make mostly SiH^+ ions, which suggests that the SiH_7^+ binding energy is quite large, much larger than the CH_7^+ binding energy of 1–2 kcal/mol.[23]

We therefore propose that SiH_7^+ has the structure shown in figure 3C: a planar SiH_3^+ with two H_2 ligands bound to opposite faces of the ion, equidistant from the Si atom. The H_2 most likely bind perpendicular to the SiH_3^+ symmetry axis, with the σ electrons donating into the empty *p* orbital of the silicon cation. Sideways binding generally occurs for molecular hydrogen in ion-molecule complexes, e.g., SiH_5^+ and H_{2n+1}^+, as well as in transition-metal dihydrogen complexes.[24] This structure is a near-prolate top. The two H_2 ligands rotate around the symmetry axis with a small energy barrier. This structure should result in two H–H stretching bands: the symmetric and antisymmetric combinations of the two H_2 stretches. The antisymmetric motion, with the two H_2 ligands vibrating 180° out of phase, will cause the Si^+ charge center to oscillate along the symmetry axis as it moves toward the extended H_2. This gives rise to the observed parallel band, which has apparent P, Q, and R branches. The symmetric combination, in which the two H_2 ligands expand and contract in phase, will induce little movement of the charge center and thus should have much weaker absorption intensity. Our current interpretation of the spectrum is that the symmetric combination is in fact weaker and obscured by the stronger antisymmetric mode.

In previous spectroscopic work on ion-molecule clusters with an H_2 ligand, a semi-quantitative correlation between the frequency of the H–H stretching mode and the binding energy of that cluster was observed. The larger the frequency shift from the free H_2 fundamental, the more strongly the H_2 is bound. With a frequency shift of 295 cm^{-1}, the SiH_7^+ binding energy is estimated as 7–9 kcal/mol (Table I). This crude binding energy estimate is about half of the experimental value of the SiH_5^+ binding energy, supporting the proposed structure of SiH_7^+, in which both of the H_2 ligands compete for the charge on the silicon atom.

Our proposed structure for SiH_7^+ is quite distinct from the structure proposed by Boo *et al.* for CH_7^+.[23] The differences can be rationalized by comparing the CH_5^+ and SiH_5^+ ions. The dissociation energy for $CH_5^+ \rightarrow CH_3^+ + H_2$ is 40 kcal/mol, and the strong three-center two-electron bond in CH_5^+ results in charge delocalization among the three centers. In contrast, the dissociation energy for $SiH_5^+ \rightarrow SiH_3^+ + H_2$ is only 17.8 kcal/mol. The SiH_5^+ ion is thus closer to an $SiH_3^+{\bullet}H_2$ complex, as Hu *et al.* predict.[16] A second H_2 will bind to the Si atom, where most of the charge remains localized.

An estimate of the rotational constant *B* can be obtained from the spectrum. By using the rotational temperature of the ion beam of 80 - 100 K (measured by other experiments using the same ion source) and fitting the rotational contour of the spectrum, we estimate *B* to be ~0.85 cm^{-1}. A more definitive structure determination requires recording the rotationally resolved spectrum by running the OPO in a higher resolution configuration.

3.4 COMPARISON WITH NEW *AB INITIO* RESULTS

Since the publication of our results, researchers from two separate groups have carried out *ab initio* calculations of SiH_7^+.[25,26] Extensive basis sets and complex methods such as perturbative triple excitation coupled cluster technique [CCSD(T)] were used in the calculations. Most conclusions in our paper are strongly supported by these calculations. Both calculations found our proposed structure (figure 3C) to be the global minimum on the potential energy surface of SiH_7^+. Both calculations obtained a large transition intensity for the antisymmetric combination mode of the H_2 stretch, and a negligible one for the symmetric combination.[25,26] Their theoretical values on the frequency shift of the H_2 stretch are 297 and 259 cm^{-1} respectively, very close to the experimental value of 295 cm^{-1}. Both calculation also yielded a rotational constant B of 0.86 cm^{-1}, in excellent agreement with our value of 0.85 cm^{-1} obtained from the fitting of the spectrum.

The discrepancy between theory and experiment lies in the binding energies of the silicon hydride clusters (table II). Although accurate cluster binding energies are very difficult to obtain from *ab initio* calculations, the theoretical values for SiH_5^+ and SiH_7^+, 10.3 kcal/mol and 4.6 kcal/mol, are both significantly lower than values procured from the experiments. More accurate binding energy measurements, as well as theoretical investigations on the convergence of the binding energy values with deeper level calculations, will be helpful to understand the systematic deviation between the current theoretical and experimental values.

3.5 CONCLUSION

In conclusion, we have discovered the SiH_7^+ ion and obtained its vibrational predissociation spectrum. Our proposed structure for SiH_7^+ was supported by recent *ab initio* calculations. A comparison with the carbonium ions was presented, illustrating the difference in chemical binding between carbon and silicon.

We also attempted to obtain the IR spectrum of SiH_5^+ using the vibrational predissociation spectroscopy (*i.e.*, monitoring the predissociation process $SiH_5^+ \rightarrow SiH_3^+ + H_2$ by detecting SiH_3^+ ions). However, within the signal to noise ratio, SiH_3^+ photofragment ions are not observed. This is due to a large background of metastable fragment ions coupled with the fact that SiH_5^+ probably needs two IR photons to predissociate. The problem of metastable ions may be solved by the installation of a new device designed to deflect the metastable fragment. The principle and operation of this device will be discussed in chapter 6.

Larger silicon hydride ions are also believed to possess unusual geometries. In $Si_2H_7^+$, for example, the two Si atoms are thought to bond through an H atom, with a structure of $H_3Si{-}H^+{-}SiH_3$. The structure and novel bonding of these novel species may also be studied with multiphoton dissociation techniques.

3.6 REFERENCES

(1) This chapter is modified from: Cao, Y. B.; Choi, J.-H.; Haas, B.-M.; Johnson, M. S.; Okumura, M. *J. Phys. Chem.* **1993**, *97*, 5215.

(2) Yu, T. Y.; Cheng, T. M. H.; Kempten, V.; Lampe, F. W. *J. Phys. Chem.* **1972**, *76*, 3321.

(3) Henis, J. M. S.; Stewart, G. W.; Tripodi, M. K.; Gaspar, P. P. *J. Chem. Phys.* **1972**, *57*, 389.

(4) Cheng, T. M. H.; Yu, T. Y.; Lampe, F. W. *J. Phys. Chem.* **1974**, *78*, 1184.

(5) Boo, B. H.; Armentrout, P. B. *J. Am. Chem. Soc.* **1987**, *109*, 3549.

(6) Mandich, M. L.; Reents Jr., W. D.; Jarrold, M. F. *J. Chem. Phys.* **1988**, *88*, 1703.

(7) Reents Jr., W. D.; Mandich, M. L. *J. Phys. Chem.* **1988**, *92*, 2908.

(8) Mandich, M. L.; Reents Jr., W. D. *J. Chem. Phys.* **1989**, *90*, 3121.

(9) Mandich, M. L.; Reents Jr., W. D.; Kolenbrander, K. D. *J. Chem. Phys.* **1990**, *92*, 437.

(10) Reents Jr., W. D.; Mandich, M. L. *J. Chem. Phys.* **1990**, *93*, 3270.

(11) Raghavachari, K. *J. Chem. Phys.* **1988**, *88*, 1688.

(12) Raghavachari, K. *J. Phys. Chem.* **1988**, *92*, 6284.

(13) Raghavachari, K. *J. Chem. Phys.* **1990**, *92*, 452.

(14) Al-Laham, M. A.; Raghavachari, K. *J. Chem. Phys.* **1991**, *95*, 2560.

(15) Olah, G. A.; Prakash, G. K. S.; Williams, R. E.; Field, L. D.; Wade, K. *Hypercarbon Chemistry*; Wiley-Interscience: New York, **1987**.

(16) Hu, C. H.; Shen, M.; Schaefer III, H. F. *Chem. Phys. Lett.* **1992**, *190*, 543.

(17) Curtis, M. C.; Jackson, P. A.; Sarre, P. J.; Whitham, C. J. *J. Mol. Phys.* **1985**, *56*,

485.

(18) Sarre, P. J.; Walmsley, J. M.; Whitham, C. J. *Philos. Trans. R. Soc. London, A* **1988**, *324*, 233.

(19) Hall, D. I.; Levick, A. P.; Sarre, P. J.; Whitham, C. J.; Alijah, A.; Duxbury, G. *J. Chem. Soc., Faraday Trans.* **1993**, *89*, 177.

(20) Dyke, J. M.; Jonathan, N.; Morris, A.; Ridha, A.; Winter, M. J. *J. Chem. Phys.* **1983**, *81*, 481.

(21) Smith, D. M.; Martineau, P. M.; Davies, P. B. *J. Chem. Phys.* **1992**, *96*, 1741.

(22) Okumura, M.; Yeh, L. I.; Lee, Y. T. *J. Chem. Phys.* **1988**, *88*, 79. Yeh, L. I.; Okumura, M.; Myers, J. D.; Price, J. M.; Lee, Y. T. *J. Chem. Phys.* **1989**, *91*, 7319. Price, J. M.; Crofton, M. W.; Lee, Y. T. *J. Chem. Phys.* **1991**, *95*, 2182.

(23) Boo, D. W.; Lee, Y. T. *Chem. Phys. Lett.* **1993**, *211*, 358.

(24) Kubas, G. J. *Acc. Chem. Res.* **1988**, *21*, 120.

(25) Liu, R.; Zhou, X. J. *J. Phys. Chem.* **1993**, *97*, 9555.

(26) Hu, C. H.; Schreiner, P. R.; Schleyer, P. v. R.; Schaefer III, H. F. *J. Phys. Chem.* sumbited December **1993**.

TABLE I. The relationship between H-H stretching frequencies shifts[a] and binding energies of cluster ions with hydrogen ligands.

Ions	Frequency Shift, cm^{-1}	Binding Energy, kcal/mol
$CH_5^+(H_2)$	-83.6	1 – 2
$H_7^+(H_2)$	-141	≤ 3.0
$H_5^+(H_2)$	-181	3.1 ± 0.1
$C_2H_5^+(H_2)$	-197	4.0 ± 0.5
$H_3^+(H_2)$	-251	6.6 ± 0.3
$SiH_5^+(H_2)$	-295	(7 – 9 ?)

[a] Relative to the H-H stretching frequency of monomer H_2, 4161 cm^{-1}.

TABLE II. The binding energies of the carbonium and silanium cations in kcal/mol.

	CH_5^+	CH_7^+	SiH_5^+	SiH_7+
Theory	40-45	1.5	10.3	4.6
Experiment	41.5	1-2	17.8	7-9

3.7 FIGURE CAPTIONS

Figure 1. Time-of-flight mass spectrum of the silanium ions. The pulsed valve stagnation pressure was 1000 Torr. The gas mixing ratio SiH_4 : H_2 was 1 : 30000.

Figure 2. Vibrational predissociation spectrum of the SiH_7^+ ion in the 3800–3925 cm^{-1} region. No other predissociation bands were found from 3500 to 4200 cm^{-1}.

Figure 3. (A): SiH_5^+ structure predicted by Hu, Shen and Schaefer (B): CH_7^+ structure based on the experimental results of Boo and Lee (C): The proposed structure of SiH_7^+.

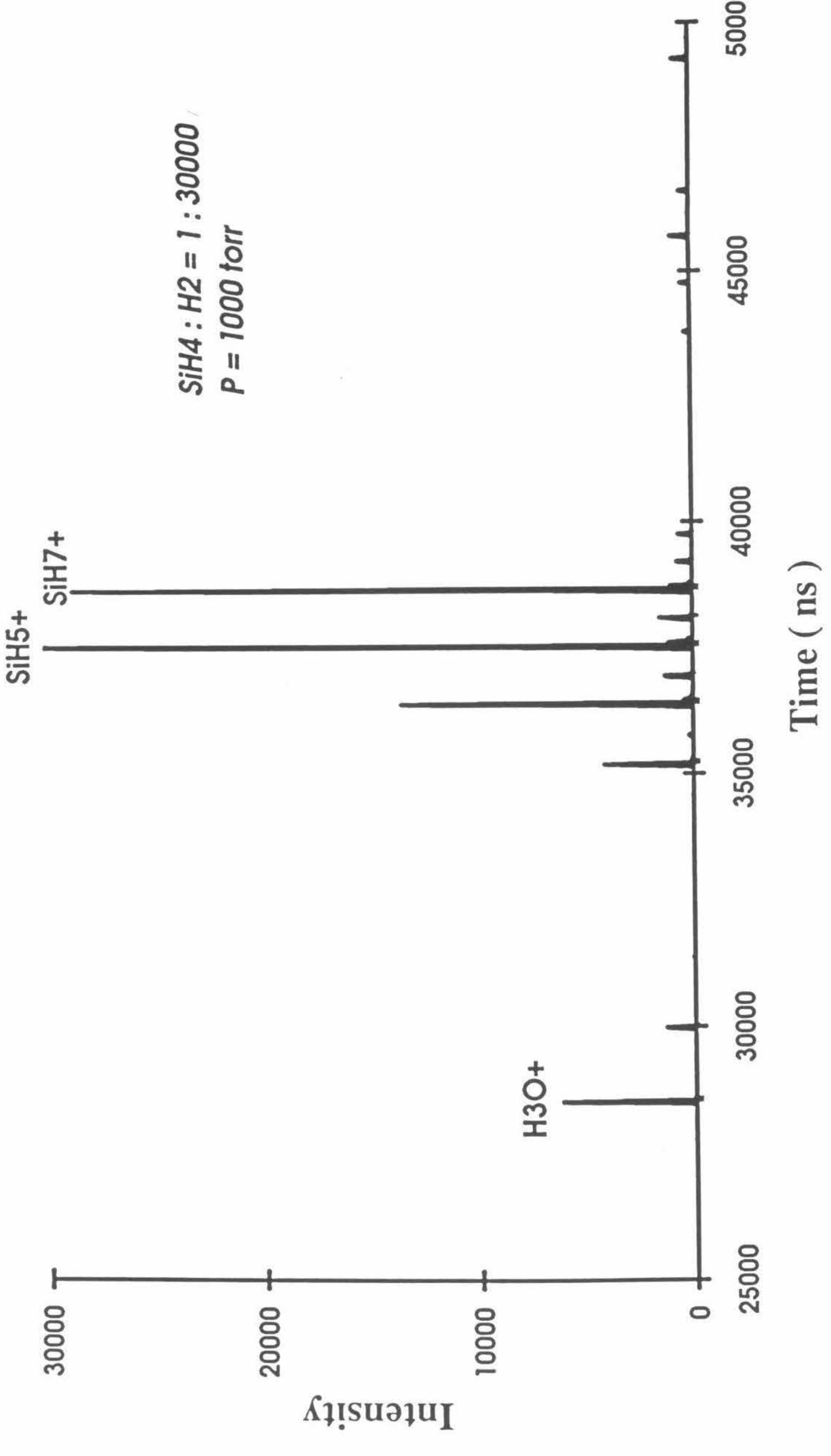

SiH4 : H2 = 1 : 30000
P = 1000 torr
SiH5+
SiH7+
H3O+
Intensity
30000
20000
10000
0
25000
30000
35000
40000
45000
50000
Time (ns)

Figure 1.

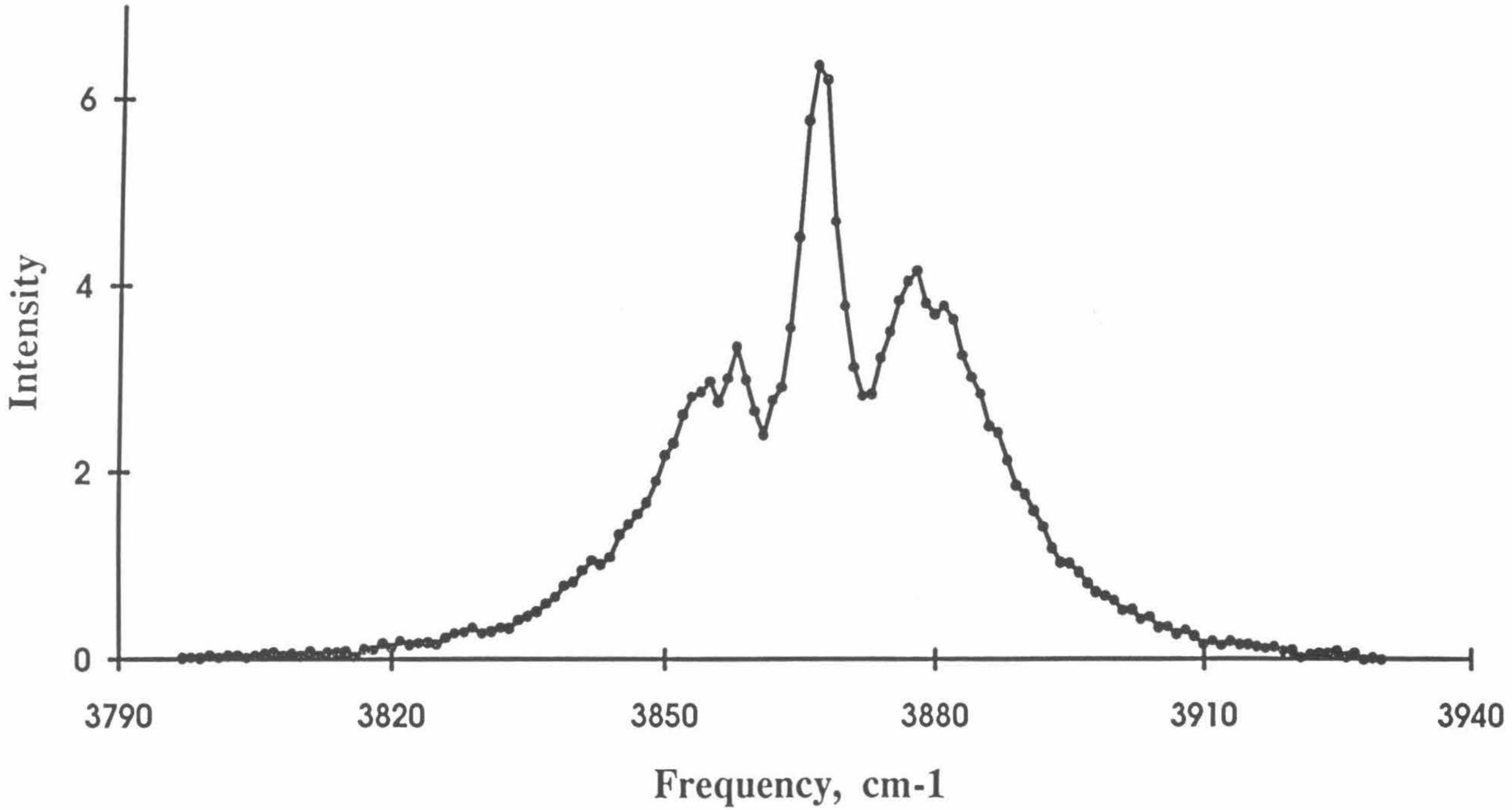


Figure 2.

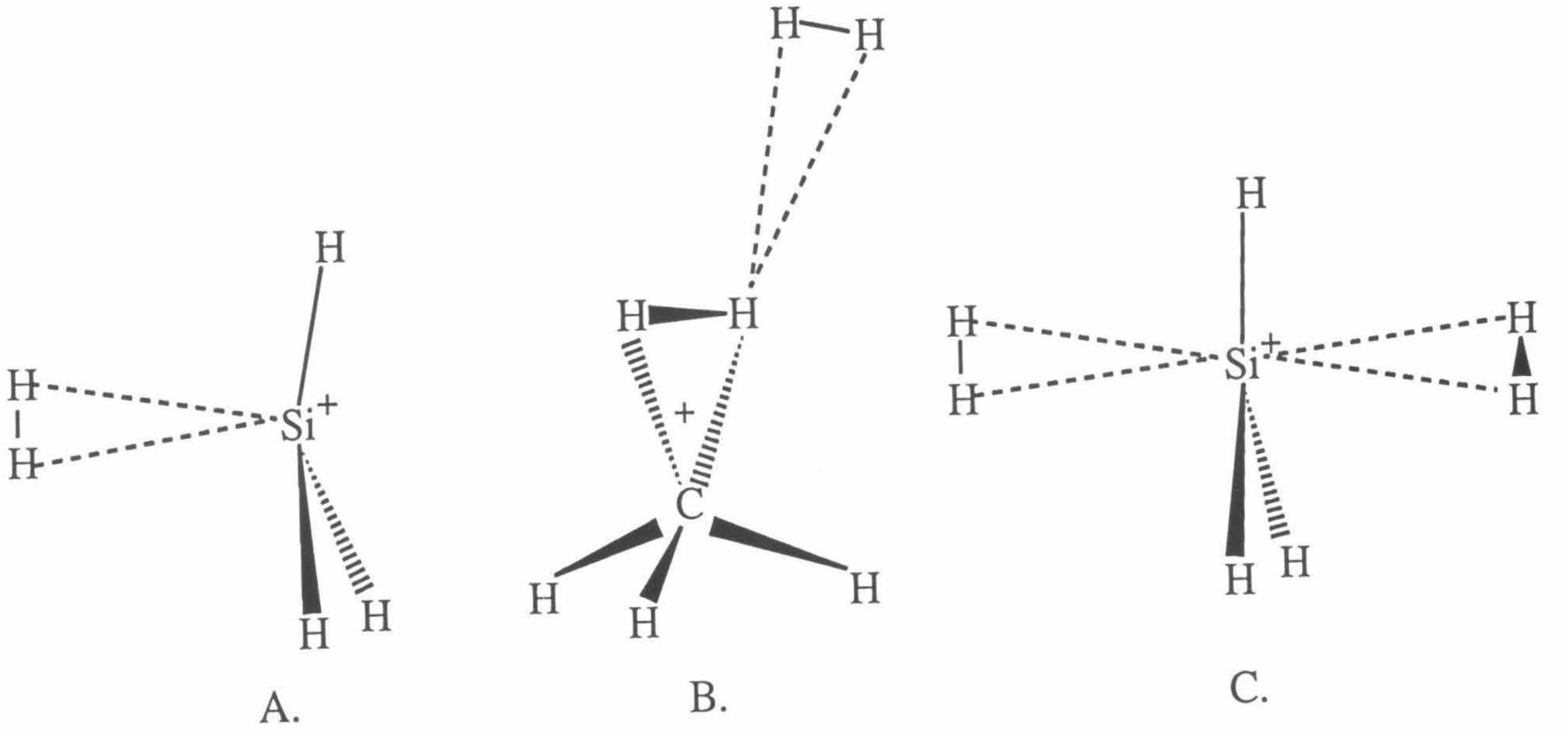


Figure 3.

Chapter 4. $NO_2^+(H_2O)_n$ Clusters

4.1 INTRODUCTION

It is well-known that a molecule behaves differently in solution than in the gas phase. HNO_3 is a strong acid in aqueous solution, *i.e.*, the equilibrium $HNO_3(aq) \rightleftarrows H^+(aq) + NO_3^-(aq)$ lies overwhelmingly to the right. But in the gas phase, the reaction $HNO_3 + H_3O^+ \rightarrow H_2NO_3^+ + H_2O$ is exothermic, indicating that HNO_3 is a better base than water. The seemingly contradictory behavior of HNO_3 highlights the dramatic effect of solvation on ion chemistry.

Condensed-phase studies of HNO_3 reveal that protonation occurs in anhydrous nitric acid by the reaction $H^+ + HNO_3 \rightarrow NO_2^+ + H_2O$.[2] Upon addition of water, the nitronium ion (NO_2^+) disappears by undergoing the reverse reaction rapidly; the only cation in aqueous nitric acid is $H^+(aq)$.

Experimental[3,4] and theoretical[5,6] studies have concluded that gas-phase protonated nitric acid $H_2NO_3^+$ exists as several isomers (Figure 1). The most stable form is a weakly-bound complex of NO_2^+ and H_2O, *i.e.*, $NO_2^+{\bullet}H_2O$. Cacace *et al.*[4] found evidence for a second isomer, $(HO)_2NO^+$, which lies 10-20 kcal/mol higher and is formed by the protonation of a terminal oxygen atom of HNO_3. *Ab initio* calculations[6] by Lee and Rice supported the structures and relative stability of the two isomers. However, their computed proton affinity for nitric acid (the energetics of the reaction $H^+ + HNO_3 \rightarrow H_2NO_3^+$, where $H_2NO_3^+$ is in its most stable isomeric form) is 182±4 kcal/mol, in disagreement with both Cacace's ICR bracketing measurement of 168±3 kcal/mol and an earlier experimental measurement[3] of 176±7 kcal/mol. These values, in turn, lead to a binding energy of the $NO_2^+{\bullet}H_2O$ complex ranging from 5 to 19 kcal/mol.

Hydrated clusters of $H_2NO_3^+$ have been the subject of a number of studies.

(Henceforth, the hydrated clusters of $H_2NO_3^+$ will be referred to as $NO_2^+(H_2O)_n$. This designation is not meant to indicate a particular structure). Fehsenfeld *et al.*[4] studied gas-phase reactions of $NO_2^+(H_2O)_n$ clusters in a flowing afterglow apparatus at room temperature. They found that $NO_2^+(H_2O)$ was generated by rapid proton-transfer from H_3O^+ to HNO_3. However, when more water vapor was added to the flow tube, larger $NO_2^+(H_2O)_n$ clusters ($n \geq 2$) were not observed in the mass spectrometer. This result is surprising, since thermodynamically $NO_2^+(H_2O)$ should undergo further hydration. The absence of these clusters therefore must be due to the existence of another reaction which consumed the larger $NO_2^+(H_2O)_n$ clusters. Following this line of thought, Fehsenfeld *et al.* postulated the reaction

$$NO_2^+(H_2O)_2 + H_2O \rightarrow H_3O^+(H_2O) + HNO_3 \tag{1}$$

and assigned a very fast rate ($> 10^{-10}$ cm^3 s^{-1}) to it in order to explain the complete absence (to within the detection sensitivity) of the larger $NO_2^+(H_2O)_n$ clusters.

$NO_2^+(H_2O)_n$ clusters are prevalent in the Earth's stratosphere.[7] $NO_2^+(H_2O)$ is the intermediate of the ion-catalyzed reaction which converts N_2O_5 to HNO_3 in the stratosphere. $NO_2^+(H_2O)$ is also generated by the reaction of HNO_3 with H_3O^+ (two species that are abundant in the stratosphere). However, the predominant ionic species in the stratosphere are $H_3O^+(H_2O)_n$ clusters, not $NO_2^+(H_2O)_n$ clusters. Ferguson *et al.*[7] thus concluded that the reaction (1) also occurs in the stratosphere.

We have investigated the clusters $NO_2^+(H_2O)_n$ for n = 1 to 5 using infrared vibrational predissociation spectroscopy. In this chapter, the structural assignments for these clusters will be discussed. Special emphasis will be given to presenting spectroscopic evidence for a solvent-induced intracluster reaction within $NO_2^+(H_2O)_n$ clusters.

4.2 EXPERIMENTAL

The apparatus has been described in detail in chapter 2; therefore, only a brief description of the experiment is given here. UHP helium was seeded with HNO_3/H_2O vapor by passage over the surface of concentrated (70%) nitric acid at 0°C. The $HNO_3/H_2O/He$ mixture at a total stagnation pressure of 1000 Torr was then pulsed into a small channel where a high voltage pulse was applied between two electrodes to initiate a glow discharge. The resulting ions underwent many thermalizing collisions as the gas flowed through the channel and expanded into the vacuum, promoting cooling and clustering. This discharge source was verified to produce cold, stable ions (Figure 2).

These parent ions were mass-analyzed by a TOF mass spectrometer. A pulsed $LiNbO_3$ optical parametric oscillator with 1.5 cm^{-1} resolution was used to excite clusters of a selected mass. Upon photon absorption, clusters could vibrationally predissociate. The resulting fragment ions were mass-analyzed by a reflectron mass spectrometer. Infrared spectra were obtained by measuring the photofragment intensity as a function of the laser wavelength.

We typically generated $NO_2^+(H_2O)_n$ clusters with n ranging from 0 to 6, as well as $H_3O^+(H_2O)_n$ and small amounts of $NO^+(H_2O)_n$. We obtained the vibrational predissociation spectra of $NO_2^+(H_2O)_n$ clusters, with $n = 1$ to 5, in the range 2650 cm^{-1} to 3850 cm^{-1}.

To investigate the second isomer of $H_2NO_3^+$, "hotter" ions were generated by electron-beam ionization of the pulsed jet expansion. The vibrational predissociation spectrum of these "hotter" $H_2NO_3^+$ ions was also obtained.

4.3 RESULTS

The experimental results on $NO_2^+(H_2O)_n$ clusters will be described in terms of the cluster size, *n*. (The locations of the band centers are summarized in table I.)

***n* = 1.** When the $H_2NO_3^+$ ion produced by the discharge source was excited by an infrared OPO photon, NO_2^+ was the only photofragment ion observed. Therefore, protonated nitric acid underwent the predissociation process

$$H_2NO_3^+ \rightarrow NO_2^+ + H_2O \ . \tag{2}$$

However, the dissociation signal from $H_2NO_3^+$ was very weak, and its dependence on laser intensity was clearly nonlinear. The power dependence could be fit to the sum of a linear term and a quadratic term (Figure 3a), which indicates that the photodissociation of $H_2NO_3^+$ was due to the combination of a sequential two-photon absorption by vibrationally cold clusters and one-photon absorption by vibrationally excited clusters. We conclude that the binding energy of $NO_2^+(H_2O)$ must be larger than $h\nu_{OH}$ (~10.5 kcal/mol), but smaller than $2h\nu_{OH}$ (~21 kcal/mol).

The vibrational predissociation spectrum of $NO_2^+(H_2O)$ consisted of two prominent bands, centered at 3626 cm^{-1} and 3704 cm^{-1}, respectively (Figure 4). These bands are only slightly red-shifted from the O-H stretching bands of a free H_2O molecule (3657 and 3756 cm^{-1}).

When $H_2NO_3^+$ was produced by electron-beam ionization, a new band at 3375 cm^{-1} was observed (Figure 5). The predissociation channel for this band is also given by reaction (2).

***n* = 2.** $NO_2^+(H_2O)_2$ clusters exhibited a similar behavior upon vibrational excitation, releasing a water molecule:

$$NO_2^+(H_2O)_n \rightarrow NO_2^+(H_2O)_{n-1} + H_2O \ . \tag{3}$$

The signal from $NO_2^+(H_2O)_2$ dissociation was much stronger than in the case of the monohydrate, and was nearly linear with respect to laser intensity (Figure 3b). Consequently, the binding energy of $NO_2^+(H_2O)_2$ is slightly larger than $h\nu_{IR}$ of ~10.5 kcal/mol. The only two bands observed are centered around 3635 cm^{-1} and 3715 cm^{-1}, respectively (Figure 6a).

***n* = 3.** For $NO_2^+(H_2O)_3$, a new photodissociation channel appeared:

$$NO_2^+(H_2O)_n \rightarrow H_3O^+(H_2O)_{n-2} + HNO_3 . \qquad (4)$$

However, the water loss channel, reaction (3), was still dominant, accounting for about 85% of the photofragments. Linear power dependence was observed for this cluster. Its infrared spectrum was almost identical to that of $NO_2^+(H_2O)_2$, with two O-H stretching bands centered at 3636 and 3716 cm^{-1} (Figure 6b). This striking spectroscopic resemblance of $NO_2^+(H_2O)_2$ to $NO_2^+(H_2O)_3$ suggests that these two clusters have similar structures. We found no band in the vicinity of 3551 cm^{-1}, the O-H stretching frequency of a free HNO_3.

***n* = 4.** For $NO_2^+(H_2O)_4$, reaction (4), the HNO_3 loss channel, became the major pathway for predissociation. Five bands were observed in the infrared predissociation spectrum (Figures 6c and 7a). In addition to bands at 3644 and 3717 cm^{-1} that were slightly to the red of the H_2O stretch frequencies, two sharp bands were found near the O-H stretch frequency of HNO_3 monomer, at 3519 and 3550 cm^{-1}, respectively. The onset of a strong broad band occurring below 2700 cm^{-1} could also be observed. The fragmentation branching ratio of the HNO_3 loss channel to the H_2O loss channel was found to be wavelength dependent, with a ratio of 7:1 upon photon excitation in the 3550 cm^{-1} region and 2:1 in the 3600-3800 cm^{-1} region.

n = 5. For $NO_2^+(H_2O)_5$ photodissociation, the HNO_3 loss channel was dominant, accounting for 90% of the photofragments. Its infrared spectrum was similar to that of $NO_2^+(H_2O)_4$ but exhibited some additional features (Figure 7b). In the H_2O stretch region, three bands were observed at 3644, 3717, and 3733 cm^{-1}. In the 3500 - 3600 cm^{-1} region, a sharp band at 3523 cm^{-1} and a broad band at 3550 cm^{-1} were observed. The most prominent feature in the spectrum is a strong, broad band peaked at 2700 cm^{-1}. This band is blue-shifted with respect to the corresponding band of $NO_2^+(H_2O)_4$.

4.4 DISCUSSION

4.4.1 Protonated Nitric Acid (*n* = 1): Two Isomeric Structures

As described in the introduction, recent experiments and *ab initio* calculations suggest that protonated nitric acid exists in two isomeric forms. The first isomer, $NO_2^+\bullet H_2O$, is a weakly-bound cluster of NO_2^+ and H_2O. The second isomer, a covalently bound $(HO)_2NO^+$, is calculated to be higher in energy by 20 kcal/mol.[6] The (scaled) *ab initio* O-H stretch frequencies, which are very different between the two isomers, are listed in table II.

For $H_2NO_3^+$ produced by the discharge source, the infrared spectrum is composed of two bands at 3626 and 3704 cm^{-1}. The frequencies of these two bands agree very well with the (scaled) *ab initio* frequencies for $NO_2^+\bullet H_2O$ (3617 and 3710 cm^{-1}).[6] In conjunction with the observation of the predissociation products NO_2^+ and H_2O, we conclude that our infrared spectrum is indeed that of the $NO_2^+\bullet H_2O$ isomer. Consequently, the observed bands are assigned as the symmetric and antisymmetric stretches of the H_2O ligand, respectively.

The remarkably good agreement between experimental and *ab initio* frequencies also suggests that the *ab initio* structure of $NO_2^+ \bullet H_2O$ (a planar ion with a nearly-linear NO_2^+ bound to a H_2O ligand 2.45 Å away[6]) may be quantitatively correct. *Ab initio* $NO_2^+ \bullet H_2O$ is a highly asymmetric rotor, with rotational constants of 0.398, 0.202, and 0.134 cm^{-1}. In the rigid rotor approximation, the symmetric stretching mode of H_2O would be an *a*–type transition and the antisymmetric mode a *b*–type transition.

The predicted selection rules are consistent with the observed spectra, which exhibit a single maximum (with two shoulders) in the 3626 cm^{-1} band and two closely spaced maxima (Q-branch features) in the band at 3704 cm^{-1}. Using the *ab initio* rotational constants and a rotational temperature of 80 K (characterized by prior experiments), we simulated the band contours of these two transitions.[9] The general features of the simulated rotational contours agree with the experimental results (Figure 4). In particular, the 8 cm^{-1} spacing between the two maxima in the 3704 cm^{-1} bands is well simulated. This good agreement between experiment and simulation offers further quantitative confirmation for the *ab initio* structure of $NO_2^+ \bullet H_2O$.

The detection of a new band at 3375 cm^{-1} in $H_2NO_3^+$ produced by electron-beam ionization points to the existence of a second isomer. The (scaled) *ab initio* O-H stretch frequencies of the $(HO)_2NO^+$ isomer are 3393 cm^{-1} (antisymmetric stretch) and 3406 cm^{-1} (symmetric stretch). (The *ab initio* calculations on vibrational frequencies of $(HO)_2NO^+$ were carried out at the MP2-DZP level[6]). The antisymmetric stretch, with an infrared intensity ~20 times higher, overwhelms the symmetric stretch band. The excellent agreement between the experimental frequency with the *ab initio* O-H antisymmetric stretch frequency confirms that the second isomer is $(HO)_2NO^+$. The absence of this isomer in the (colder) discharge source is due to its instability in terms of energy.

Because $(HO)_2NO^+$ is covalently bound, an activation barrier for predissociation obviously exists. The mechanism for $(HO)_2NO^+$ predissociation involves a concerted motion of a hydrogen-transfer from one of the OH groups to the other, with the formation of a NO double bond (Figure 8). With its light mass, the hydrogen atom may pass through the transition state by tunneling. The fact that $(HO)_2NO^+$ readily dissociates into NO_2^+ and H_2O upon the excitation of its O-H stretch implies an activation barrier of no more than 20 kcal/mol. The so-called 1,3 hydrogen-transfer involved in the predissociation of $(HO)_2NO^+$ is commonly observed in reactions of organic molecules.[10]

4.4.2 $NO_2^+(H_2O)_2$ and $NO_2^+(H_2O)_3$: NO_2^+ is the Central Ion Core

The strong resemblance of the spectra and fragmentation patterns of $NO_2^+{\bullet}H_2O$ to that of $NO_2^+(H_2O)_n$ ($n = 2, 3$) indicates underlying structural similarities. In particular, the absence of additional O-H stretching bands suggests that H_2O moieties of these clusters are all in similar environments. Therefore, the first two hydrated clusters of protonated nitric acid are identified as complexes which have a NO_2^+ central ion core and two or three H_2O ligands, i.e., $NO_2^+{\bullet}(H_2O)_n$.[11] With the additional ligands, the charge-dipole interaction between the NO_2^+ and each H_2O ligand is weakened. This results in a lower binding energy for the larger clusters, which is confirmed by the stronger photodissociation signals and their increasingly linear dependence on laser intensity.

The NO_2^+ positive ion core attracts the electron density of the oxygen atom of the H_2O ligand, weakening the O-H bond and causing a slight red-shift of the O-H stretching frequency. Consequently, the magnitude of the frequency red-shift can be used to qualitatively infer the strength of the solvating interactions: the stronger the interaction (exhibited by a larger binding energy), the larger the red-shift. The experimental result, which shows that the red-shift is the largest for n = 1 and the smallest for n = 3 among $NO_2^+\bullet(H_2O)_n$ (n = 1, 2, 3), is in accord with a decrease in binding energy with n.

4.4.3 $NO_2^+(H_2O)_4$ and $NO_2^+(H_2O)_5$: New Structures

For the $NO_2^+(H_2O)_4$ and $NO_2^+(H_2O)_5$ clusters, significant changes in their IR spectra and photodissociation patterns occur, pointing to a major transformation of the cluster structure. In contrast to the smaller clusters, the neutral products of $NO_2^+(H_2O)_4$ and $NO_2^+(H_2O)_5$ dissociation are primarily HNO_3 molecules. This result suggests the existence of intact HNO_3 in these clusters. In addition, spectroscopic studies revealed sharp bands at 3519 and 3550 cm^{-1} for $NO_2^+(H_2O)_4$ and a band at 3523 cm^{-1} for $NO_2^+(H_2O)_5$, very close to the O-H stretch frequency in a HNO_3 monomer. From the positions of these bands, we conclude that they are O-H stretches of an HNO_3 moiety in these clusters.

The presence of HNO_3 in the $NO_2^+(H_2O)_4$ and $NO_2^+(H_2O)_5$ clusters means that the NO_2^+ ion core no longer exists. Hence the positive charge should now be in the form of H_3O^+, to minimize the total energy of the clusters. This is verified by the observation of an intense broad band around 2700 cm^{-1} for both $NO_2^+(H_2O)_4$ and $NO_2^+(H_2O)_5$. This band was observed by Schwarz[12] and by Lee and co-workers[13] in their IR spectroscopic studies of $H_3O^+(H_2O)_n$ clusters, and was assigned as the O-H stretching band for the H_3O^+ ion. The enormous intensity of this band is due to its ionic origin, and its substantial red-shift and broadening are caused by the strong ionic hydrogen bonds between the hydronium ion and its ligands.

From the preceding discussion, we conclude that $NO_2^+(H_2O)_4$ and $NO_2^+(H_2O)_5$

clusters are complexes which consist of H_3O^+, HNO_3, and H_2O moieties, *i.e.*, $H_3O^+{\bullet}(H_2O)_2{\bullet}HNO_3$ and $H_3O^+{\bullet}(H_2O)_3{\bullet}HNO_3$.

In order to have a complete picture of the clusters' structures, the relative strengths of the various intermolecular interactions must be considered. The ionic hydrogen bonds that H_3O^+ forms with neutral ligands are much stronger than intermolecular forces between neutral molecules. Therefore, the former are the most important factors in establishing the cluster structures. The attractions between H_3O^+ and water ligands are particularly strong, as evidenced by the large binding energies of $H_3O^+(H_2O)$ (31.6 kcal/mol) and $H_3O^+(H_2O)_2$ (19.5 kcal/mol).[14] The binding of HNO_3 with H_3O^+ would be considerably weaker due to less favorable hydrogen-bonding. Consequently, $H_3O^+{\bullet}(H_2O)_2{\bullet}HNO_3$ consists of a central H_3O^+ ion and three directly hydrogen-bonded ligands (two H_2O and one HNO_3), while $H_3O^+{\bullet}(H_2O)_3{\bullet}HNO_3$ is comprised of a central H_3O^+ ion core which is surrounded by water ligands in the first solvation shell, with the more weakly-bound HNO_3 occupying the second solvation shell (Figure 9).

These structural assignments can be further verified by the spectroscopic data. First, the bands of $H_3O^+{\bullet}(H_2O)_2{\bullet}HNO_3$ in the 3600 - 3800 cm^{-1} region are qualitatively different from the corresponding bands of the smaller clusters $NO_2^+(H_2O)_n$ (n = 1, 2, 3); the former are sharper and exhibit a $\nu_3 : \nu_1$ intensity ratio of ~ 2 : 1. Similar characteristics were also observed by Lee and co-workers in the ligand O-H bands of $H_3O^+(H_2O)_3$.[13] The resemblance of these bands offers additional evidence for the existence of H_3O^+.

In addition, for $H_3O^+{\bullet}(H_2O)_3{\bullet}HNO_3$, a third water O-H band was found in the 3600 - 3800 cm^{-1} region. The discovery of this band, centered at 3717 cm^{-1}, reveals that the three neutral H_2O moieties do not occupy chemically equivalent positions in the $H_3O^+\bullet(H_2O)_3\bullet HNO_3$ cluster. Price *et al.*[13] have observed a similar band at 3708 cm^{-1} in $H_3O^+(H_2O)_4$ and have identified it as the free O-H band of a first shell neutral H_2O which is hydrogen-bonded to a second shell H_2O. Therefore, the existence of the 3717 cm^{-1} band, plus the observation of a distinct, reproducible shoulder at 3500 cm^{-1} (the water O-H stretch hydrogen-bonded to HNO_3), provides clear evidence of the presence of two solvation shells in the $H_3O^+\bullet(H_2O)_3\bullet HNO_3$ cluster.

Finally, in the $H_3O^+\bullet(H_2O)_2\bullet HNO_3$ cluster, two sharp bands are observed near the HNO_3 monomer frequency of 3551 cm^{-1}, one at 3550 cm^{-1} and the other at 3519 cm^{-1}. We believe that this is due to the two different ways that an HNO_3 can hydrogen-bond to H_3O^+: by using either a terminal O atom or the O atom on the hydroxy group. Those forming hydrogen bonds using the terminal O atom exhibit the 3550 cm^{-1} band, since in that configuration the O-H group is essentially unperturbed. On the other hand, hydrogen-bonding with the hydroxy group red-shifts the O-H stretch of HNO_3 by 32 cm^{-1}. For the $H_3O^+\bullet(H_2O)_3\bullet HNO_3$ cluster, the presence of only one sharp band at 3523 cm^{-1} suggests that most of the nitric acid forms hydrogen bonds using the OH group.

4.4.3 Intracluster Reaction

In the last section, we have concluded that the following intracluster reaction occurs for the larger clusters:

$$NO_2^+\bullet(H_2O)_n \rightarrow H_3O^+\bullet(H_2O)_{n-2}\bullet HNO_3 \quad . \qquad (5)$$

Studies of solvent-induced intracluster reactions can provide important information on the solute-solvent interaction. One class of these reactions is intracluster proton transfer. Castleman *et al.* studied $NH_4^+(C_5H_5N)_n$ clusters and found that depending on cluster size, the proton associates with different moieties.[15] While the higher intrinsic proton affinity of pyridine leads to a $C_5H_6N^+$ core ion in the smaller clusters ($n < 4$), NH_4^+ becomes the charge center in larger clusters ($n \geq 4$) due to its ability to provide four solvent binding sites for the C_5H_5N. A more interesting example is the metastable decomposition of protonated methanol clusters.[16] Besides evaporation of a methanol molecule, the dissociation reaction

$$CH_3OH_2^+(CH_3OH)_n \rightarrow CH_3OHCH_3^+(CH_3OH)_{n-1} + H_2O$$

was observed only in smaller clusters ($n < 3$), while

$$CH_3OH_2^+(CH_3OH)_n \rightarrow H_3O^+(CH_3OH)_{n-1} + CH_3OCH_3$$

occurred for larger clusters. This study raised an interesting question about the identity of the ion core of $CH_3OH_2^+(CH_3OH)_n$ clusters. However, all the previous research on intracluster reactions have been thermodynamic studies.[17,18] Our study on $NO_2^+(H_2O)_n$ provides the first spectroscopic evidence of such reactions.

4.4.4 Thermochemical Analysis

To illustrate the driving force for the intracluster reaction, we consider the energetics of the following reaction (Figure 10):

$$NO_2^+\bullet(H_2O)_{n-1} + H_2O \rightarrow [\ NO_2^+(H_2O)_n\ ] \rightarrow H_3O^+\bullet(H_2O)_{n-2} + HNO_3 \ . \qquad (6)$$

Using thermodynamic data from the literature,[19,20,21] we have the following values of the heat of formation for the species involved in the reaction: $\Delta H_f^0(NO_2^+) = 229.7$ kcal/mol, $\Delta H_f^0(H_2O) = -57.1$ kcal/mol, $\Delta H_f^0(HNO_3) = -30$ kcal/mol and $\Delta H_f^0(H_3O^+) = 143$ kcal/mol. To calculate the heat of formation for $NO_2^+(H_2O)_n$ and $H_3O^+(H_2O)_n$, the clustering enthalpies are also needed. For $H_3O^+(H_2O)_n$, we use results by Kebarle *et al.*:[14] $\Delta H^0_{0,1} = -31.6$ kcal/mol, $\Delta H^0_{1,2} = -19.5$ kcal/mol and $\Delta H^0_{2,3} = -17.5$ kcal/mol. For

$NO_2^+(H_2O)_n$ accurate clustering enthalpies are not available for $n > 1$. The best estimates from recent experiments are $\Delta H^0_{0,1} = -14.8$ kcal/mol, $\Delta H^0_{1,2} = -12.5$ kcal/mol and $\Delta H^0_{2,3} = -10.0$ kcal/mol, with uncertainties of ±2.5 kcal/mol.

For $n = 2$, reaction (6) is the direct proton transfer reaction from $H_2NO_3^+$ to H_2O. This process is endothermic by 12.3 kcal/mol. Our experiment indicates that the reaction intermediate has a structure of $NO_2^+(H_2O)_2$ and is bound by ~12 kcal/mol with respect to the entrance channel, $NO_2^+(H_2O) + H_2O$. Because the reverse reaction was observed to be very fast,[3] this proton transfer process is believed to proceed with no activation barrier.

For $n = 3$, reaction (5) is marginally exothermic, by about 6 kcal/mol. This reaction is claimed by Fehsenfeld *et al.*[4] to occur at a very fast rate, which implies that no activation barrier is present along the reaction pathway. However, we have observed a complex with an NO_2^+ ion core and three H_2O ligands as a stable intermediate bound by about 10 kcal/mol with respect to the entry channel. According to Fehsenfeld *et al.*, the interaction between the products $H_3O^+(H_2O)$ and HNO_3 should lead to a stable complex of $H_3O^+ \bullet (H_2O) \bullet HNO_3$. This was not seen in our experiment. The absence of $H_3O^+ \bullet (H_2O) \bullet HNO_3$ indicates the existence of a barrier to cluster isomerization. Therefore reaction (1), in spite of Fehsenfeld *et al.*'s claim, is probably slow.

We did, however, observe that for ~10% of the ions, absorption of one infrared photon was enough to promote the thermalized $NO_2^+ \bullet (H_2O)_3$ cluster over the isomerization barrier, and dissociate it into $HNO_3 + H_3O^+(H_2O)$. This is an interesting example of a reaction which is initiated by the absorption of an infrared photon.

For $n = 4$, reaction (5) is significantly exothermic, by 16.2 kcal/mol. Since the binding energy of the $NO_2^+(H_2O)_4$ cluster is ~8 kcal/mol, the $NO_2^+(H_2O)_4$ reaction intermediate lies higher in energy than the product channel of $H_3O^+(H_2O)_2$ + HNO_3. Therefore, we are confident that the cluster $H_3O^+\bullet(H_2O)_2\bullet HNO_3$ is the global minimum on the reaction potential surface. This conclusion is in good agreement with our experimental results, which showed HNO_3 as the primary fragment formed upon photon absorption. However, the existence of a minor H_2O dissociation channel suggests the possible existence of a second isomer $NO_2^+\bullet(H_2O)_4$, since one IR photon cannot provide enough energy for $H_3O^+\bullet(H_2O)_2\bullet HNO_3$ to reach the reactant channel of $NO_2^+\bullet(H_2O)_3$ + H_2O.

The possible co-existence of two isomeric structures of the $NO_2^+(H_2O)_4$ cluster is also implied by the wavelength dependence of the predissociation branching ratio of that cluster. The $NO_2^+\bullet(H_2O)_4$ isomer absorbs only at 3600-3800 cm^{-1} and probably produces only H_2O fragments. The $H_3O^+\bullet(H_2O)_2\bullet HNO_3$ isomer, on the other hand, should absorb at both HNO_3 and H_2O frequencies and produce both fragments. Therefore a mixture of the two isomers would have wavelength-dependent branching ratios.

4.5 CONCLUSION

In conclusion, we have established molecular structures for the $NO_2^+(H_2O)_n$ structures that are consistent with our experimental findings. With the structural determination, we have uncovered the intracluster reaction $NO_2^+ + 2H_2O \rightarrow HNO_3 + H_3O^+$, which occurs at a critical solvent number of $n = 4$. The driving force for the reaction is essentially the solvent-induced stabilization of H_3O^+, which is due to the ability of H_3O^+ to form three very strong hydrogen bonds. Our infrared spectra provide evidence for the formation of a second solvation shell in the $NO_2^+(H_2O)_5$ cluster.

We have also presented a detailed analysis of the cluster reactions in terms of energetics. We find that the clusters in the ion beam were the isomers with the most thermodynamically stable structures. We have also demonstrated at the molecular level how solvent molecules modify the thermicity and activation barriers of chemical reactions.

Our technique provides a direct spectroscopic probe of solvent-induced phenomena in clusters. In our recent investigation of another system, the $NO^+(H_2O)_n$ clusters, a similar intracluster reaction was observed.[22]

NOTE: After the submission of our paper, the binding energy of $NO_2^+(H_2O)$ was measured directly and found to be 14.8 ± 2.0 kcal/mol,[23] within the bracket of 11 - 20 kcal/mol that we determined experimentally.

4.6 REFERENCES

(1) This chapter is modified from: Cao, Y.; Choi, J.-H.; Haas, B.-M.; Johnson, M. S.; Okumura, M. *J. Chem. Phys.* **1993**, *99*, 9307.

(2) Addison, C. C. *Chem. Rev.* **1980**, *80*, 21.

(3) Fehsenfeld, F. C.; Howard, C. J.; Schmeltekopf, A. L. *J. Chem. Phys.* **1975,** *63*, 2835.

(4) Cacace, F.; Attina, M.; de Petris, G.; Speranza, M. *J. Am. Chem. Soc.* **1989**, *111*, 5481*; J. Am. Chem. Soc.* **1990**, *112*, 1014.

(5) Nguyen, M.-T.; Hegarty, A. F. *J. Chem. Soc. Perkin Trans.* **1984**, *II*, 2043.

(6) Lee, T. J.; Rice, J. E. *J. Phys. Chem.* **1992**, *96*, 650; *J. Am. Chem. Soc.* **1992**, *114*, 8247.

(7) Ferguson, E. E.; Fehsenfeld, F. C.; Albritton, D. L. In *Gas Phase Ion Chemistry, Vol I*; Bowers, M. T. Eds.; Academic Press: New York, **1979**; Chapter 2.

(8) Cao, Y.; Choi, J.-H.; Haas, B.-M.; Johnson, M. S.; Okumura, M. *J. Phys. Chem.* **1993**, *97*, 5215.

(9) Program ASMWIRP developed at the University of Chicago, Jagod, M.-F., private communication.

(10) Lowry, T. H.; Richardson, K. S. *Mechanism and Theory in Organic Chemistry*; Harper and Row: New York, **1976**.

(11) Grandinetti, F.; Bencivenni, L.; Ramondo, F. *J. Phys. Chem.* **1992**, *96*, 4354.

(12) Schwarz, H. A. *J. Chem. Phys.* **1977**, *67*, 5525.

(13) Yeh, L. I.; Okumura, M.; Myers, J. D.; Price, J. M.; Lee, Y. T. *J. Chem. Phys.* **1989**, *91*, 7319; Price J. M. *Ph. D. Thesis, University of California at Berkeley* **1990**.

(14) Cunningham, A. J.; Payzant, J. D.; Kebarle, P. *J. Am. Chem. Soc.* **1972**, *94*, 7627.

Lau, Y. K.; Ikuta, S.; Kebarle, P. *J. Am. Chem. Soc.* **1982**, *104*, 1462.

(15) Castleman, Jr., A. W. *Int. J. Mass Spectrum. Ion Processes* **1992**, *118*, 167.

(16) Zhang, X.; Yang, X.; Castleman, Jr, A. W. *Chem. Phys. Lett.* **1991**, *185*, 298.

(17) Deakyne, C. A.; Mautner, M.; Campbell, C. L.; Hughes, M. G.; Murphy, S. P. *J. Chem. Phys.* **1986**, *84*, 4958.

(18) Stace, A. J.; Winkel, J. F.; Lopez Martens, R. B.; Upham, J. E. *J. Phys. Chem.* **1994**, *98*, 2012.

(19) Baulch, D. L.; Cox, R. A.; Crutzen, P. J.; Hampson Jr., R. F.; Kerr, J. A.; Troe, J.; Watson, R. T. *J. Phys. Chem. Ref. Data* **1982**, *11*, 327.

(20) H. M. Rosenstock, K. Draxl, B. W. Steiner and J. T. Herron, *J. Phys. Chem. Ref. Data* **1977**, *6,* Supp. 1.

(21) Bryant, G.; Jiang, Y.; Grant, E. *Chem. Phys. Lett.* **1992**, *200*, 495.

(22) Choi, J.-H.; Kuwata, K. T.; Haas, B.-M.; Cao, Y.; Johnson, M. S.; Okumura, M. *J. Chem. Phys.* to be appeared in the issue of May 15, **1994**.

(23) Sunderlin, L. S.; Squires, R. R. *Chem. Phys. Lett.* **1993**, *222*, 333.

TABLE I. Vibrational predissociation band centers for $NO_2^+(H_2O)_n$ in cm^{-1}.

	H_3O^+	HNO_3 NO–H	H_2O H-bond	H_2O sym	H_2O free	H_2O antisym
$NO_2^+ \bullet (H_2O)$				3626		3704
$NO_2^+(H_2O)_2$				3635		3715
$NO_2^+(H_2O)_3$				3636		3716
$NO_2^+(H_2O)_4$	~2600	3519 3550		3644		3717
$NO_2^+(H_2O)_5$	~2700	3523	3550	3644	3717	3733
H_2O				3657		3756
HNO_3		3551				

TABLE II. The experimental and scaled *ab initio* O-H stretching frequencies of $H_2NO_3^+$.

	$NO_2^+ \cdot (H_2O)$ sym.	$NO_2^+ \cdot (H_2O)$ antisym.	$(HO)_2NO^+$ sym.	$(HO)_2NO^+$ antisym.
ab initio	3617	3710	3406	3393
experimental	3626	3704	not observed	3375

4.7 FIGURE CAPTIONS

Figure 1. Graphical representation of the two isomers of $H_2NO_3^+$ (*ab initio* structures by Lee and Rice).

Figure 2. TOF mass spectrum of the $NO_2^+(H_2O)_n$ clusters (n = 0-5). The ions were formed in a pulsed discharge of nitric acid and water vapor seeded in 1000 Torr of helium. The other major species in the mass spectrum are $H_3O^+(H_2O)_n$, the hydrated hydronium clusters.

Figure 3. The dependence on laser pulse energy of the fragment signal from photodissociation of the parent ions $NO_2^+(H_2O)_n$ (n = 1-2). The lines are fits to a sum of linear and quadratic terms. The figure reveals (*a*) a near-quadratic dependence for n = 1 and (*b*) a near-linear dependence for n = 2.

Figure 4. The O-H stretching bands of the $NO_2^+{\bullet}(H_2O)$ isomer. a) The predissociation spectrum obtained experimentally. b) Simulation of the rotational band contour using rotational constants calculated from the *ab initio* structure. We used a rotational temperature of 80 K and convoluted the predicted spectrum with a Gaussian to simulate a resolution (linewidth, FWHM) of 1.5 cm^{-1}.

Figure 5. The O-H stretching band of the $(HO)_2NO^+$ isomer.

Figure 6. Vibrational predissociation spectra of $NO_2^+(H_2O)_n$ clusters (n = 2-4) in the 3500

to 3850 cm^{-1} region. The ordinates for the first three figures are the photofragment yield from the H_2O loss channel. For the last figure, the ordinate is the photofragment yield from the HNO_3 loss channel of the $n = 4$ cluster.

Figure 7. Vibrational predissociation spectra of $NO_2^+(H_2O)_4$ and $NO_2^+(H_2O)_5$ in the 2650 to 3500 cm^{-1} region. The ordinate is the photofragment yield from the HNO_3 loss channel.

Figure 8. Proposed mechanism for the vibrational predissociation of $(HO)_2NO^+$.

Figure 9. Proposed structures of the $NO_2^+(H_2O)_n$ clusters, for $n = 1$ to 5.

Figure 10. Reaction paths for $NO_2^+(H_2O)_{n-1} + H_2O \rightarrow (NO_2^+(H_2O)_n) \rightarrow H_3O^+(H_2O)_{n-2} + HNO_3$. Each reaction pathway for a cluster n is lower relative to the n-1 pathway by the hydration enthalpies. Numbers are in kcal/mol; those in parentheses are estimates only.

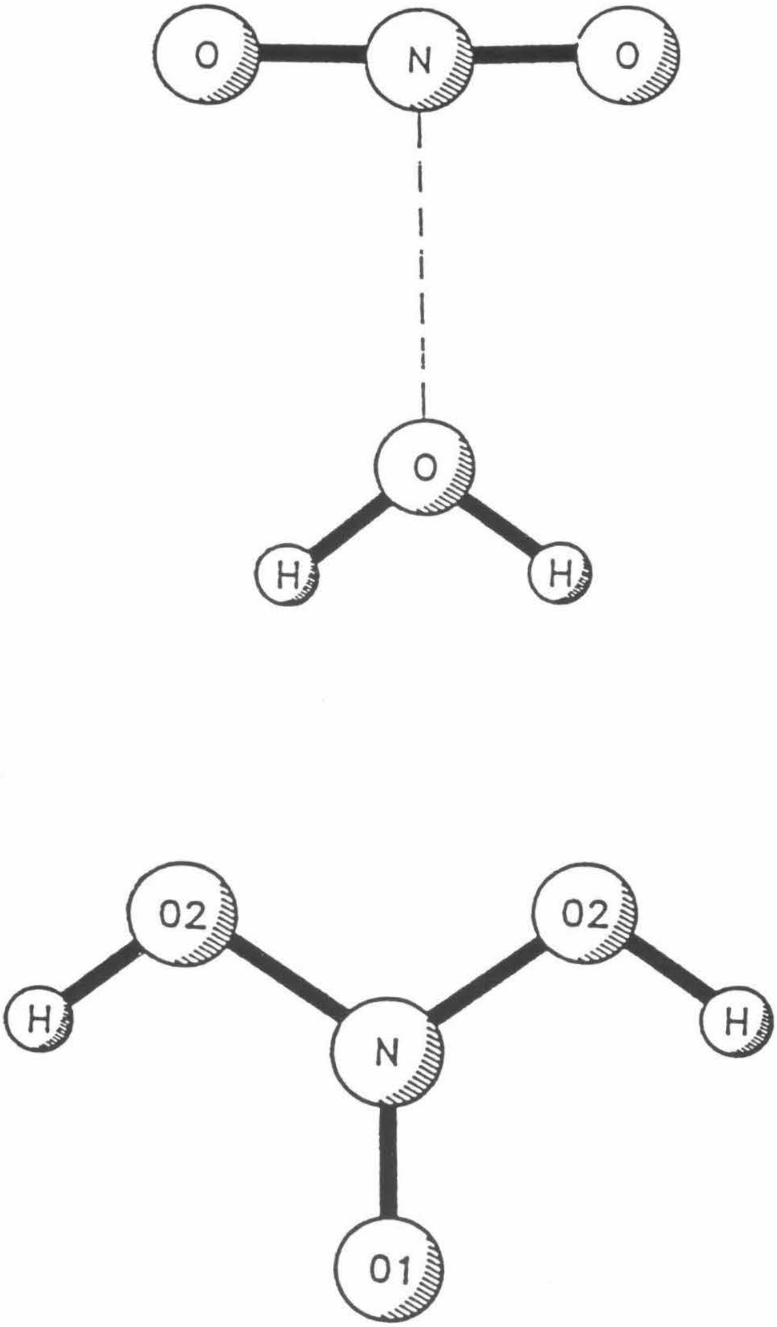


Figure 1.

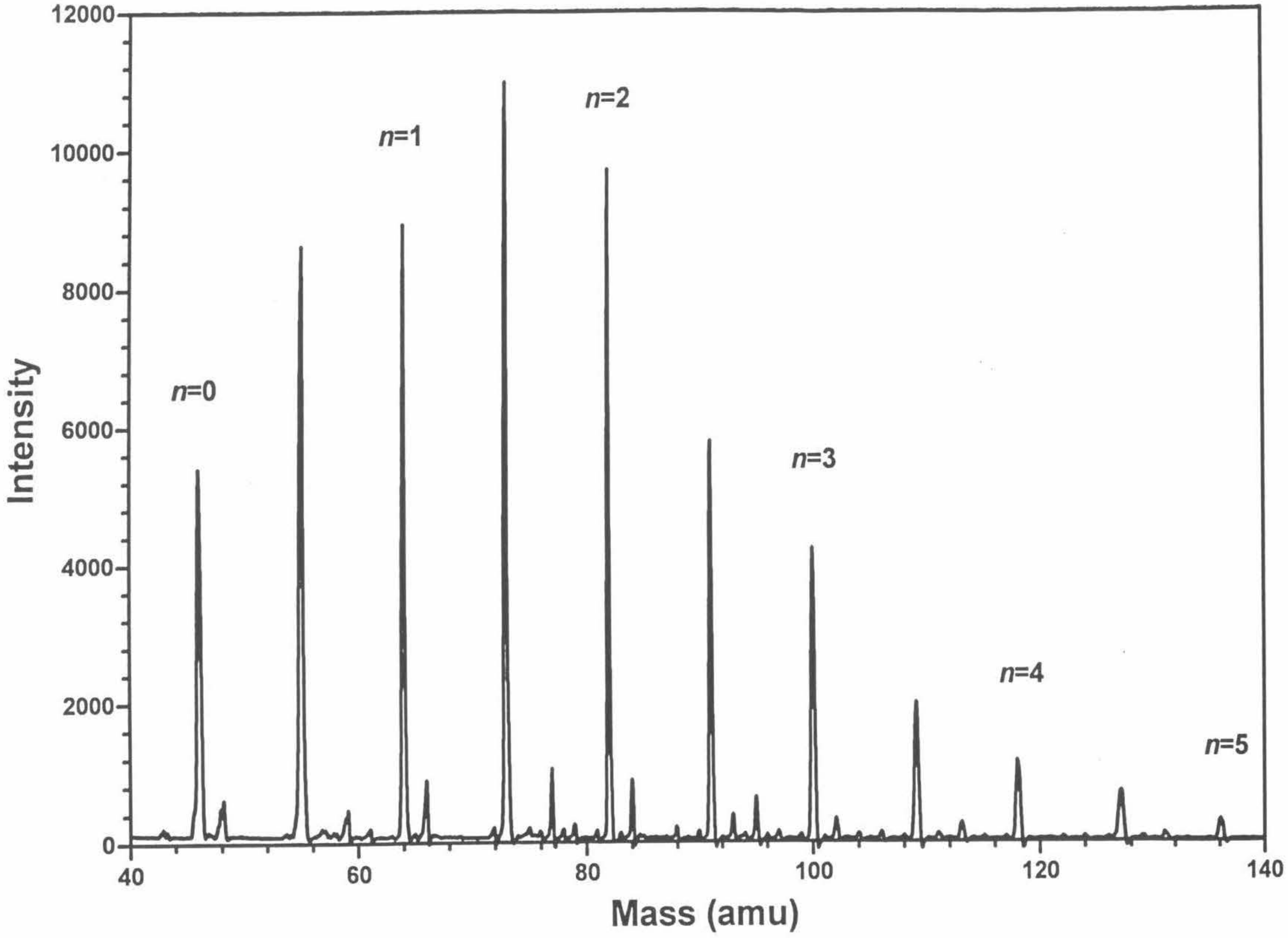

12000
10000
8000
6000
4000
2000
0
Intensity
40
60
80
100
120
140
Mass (amu)
n=0
n=1
n=2
n=3
n=4
n=5

Figure 2.

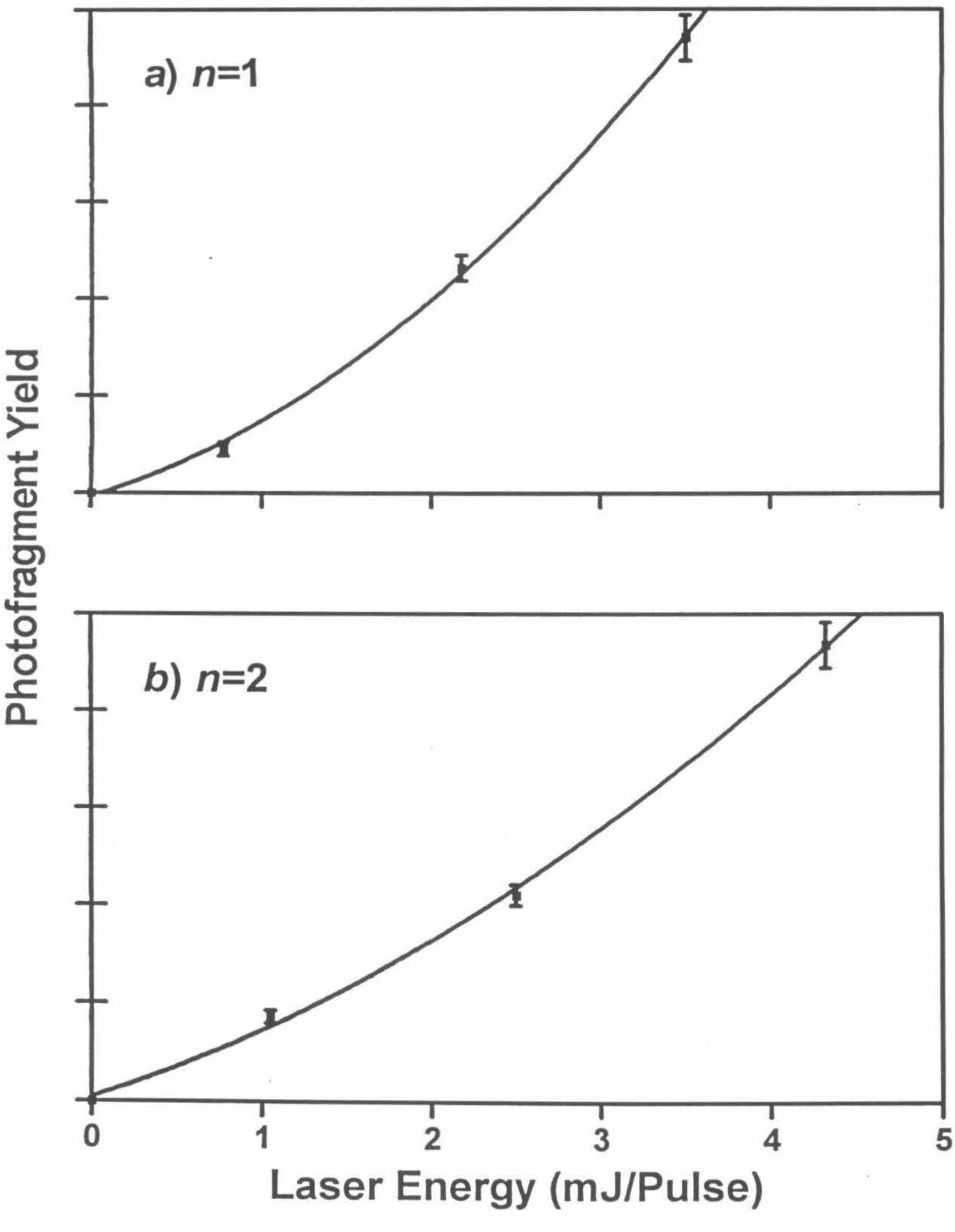


Figure 3.

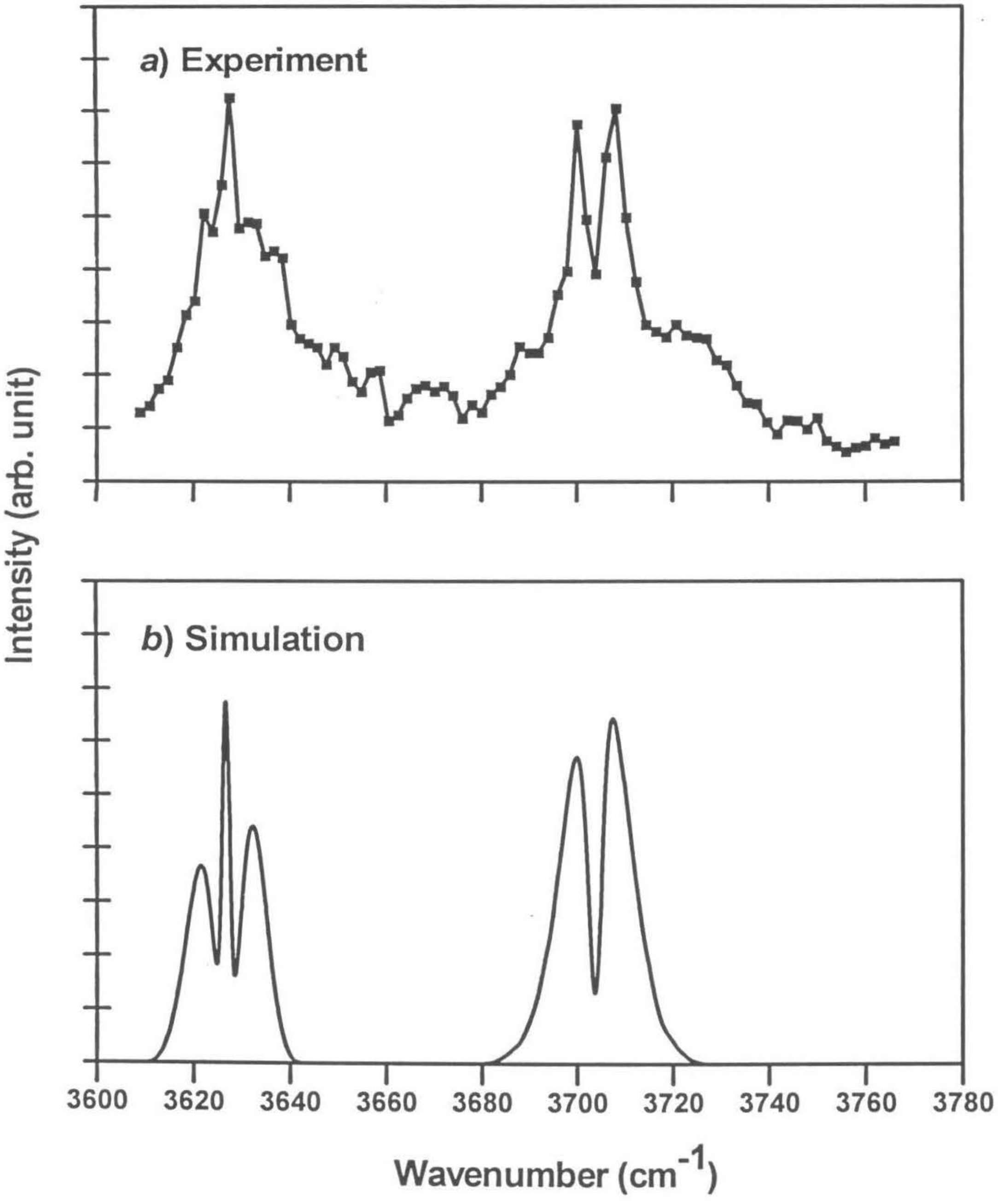

a) Experiment
b) Simulation
Intensity (arb. unit)
3600
3620
3640
3660
3680
3700
3720
3740
3760
3780
Wavenumber (cm^{-1})

Figure 4.

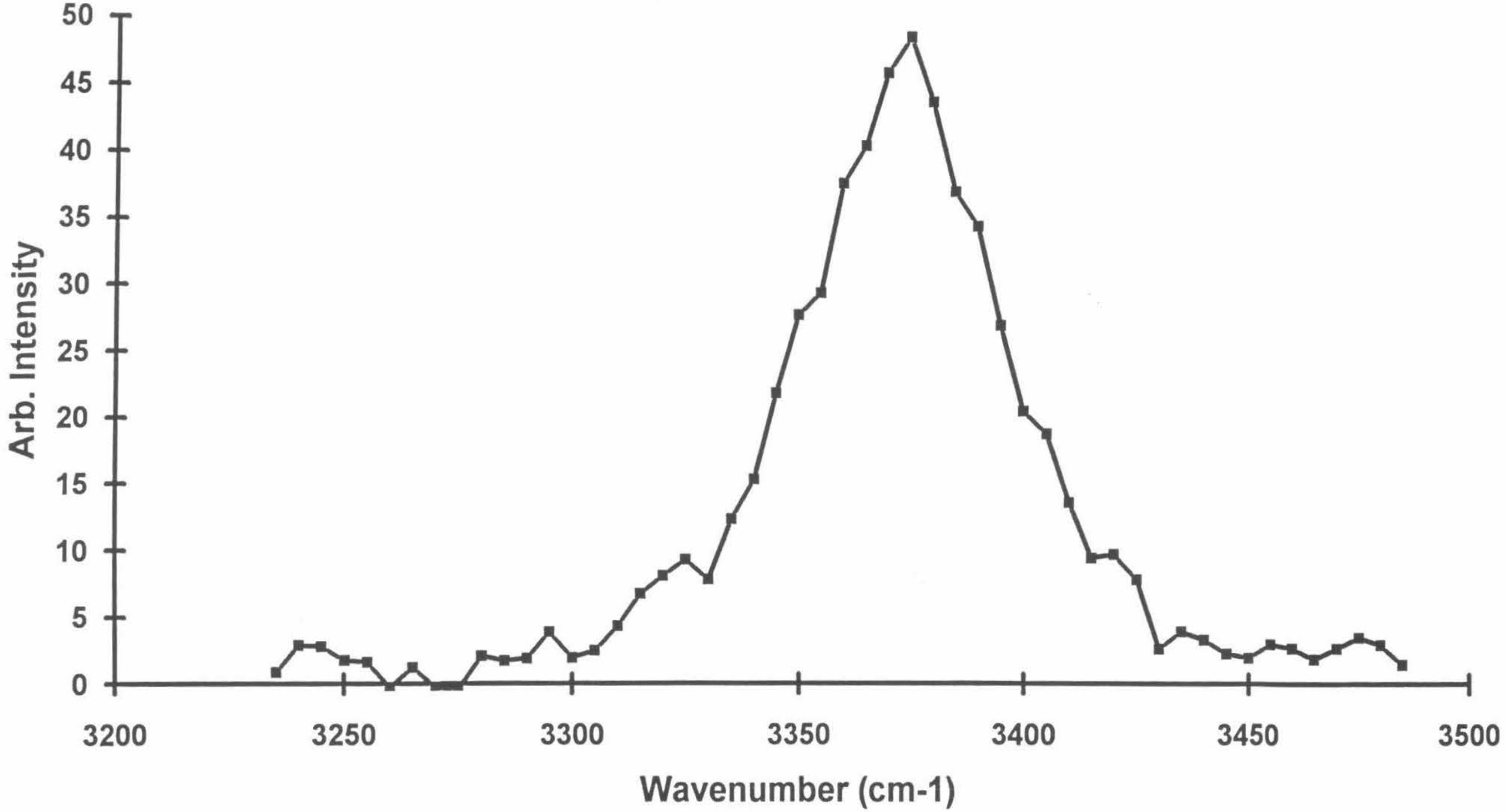
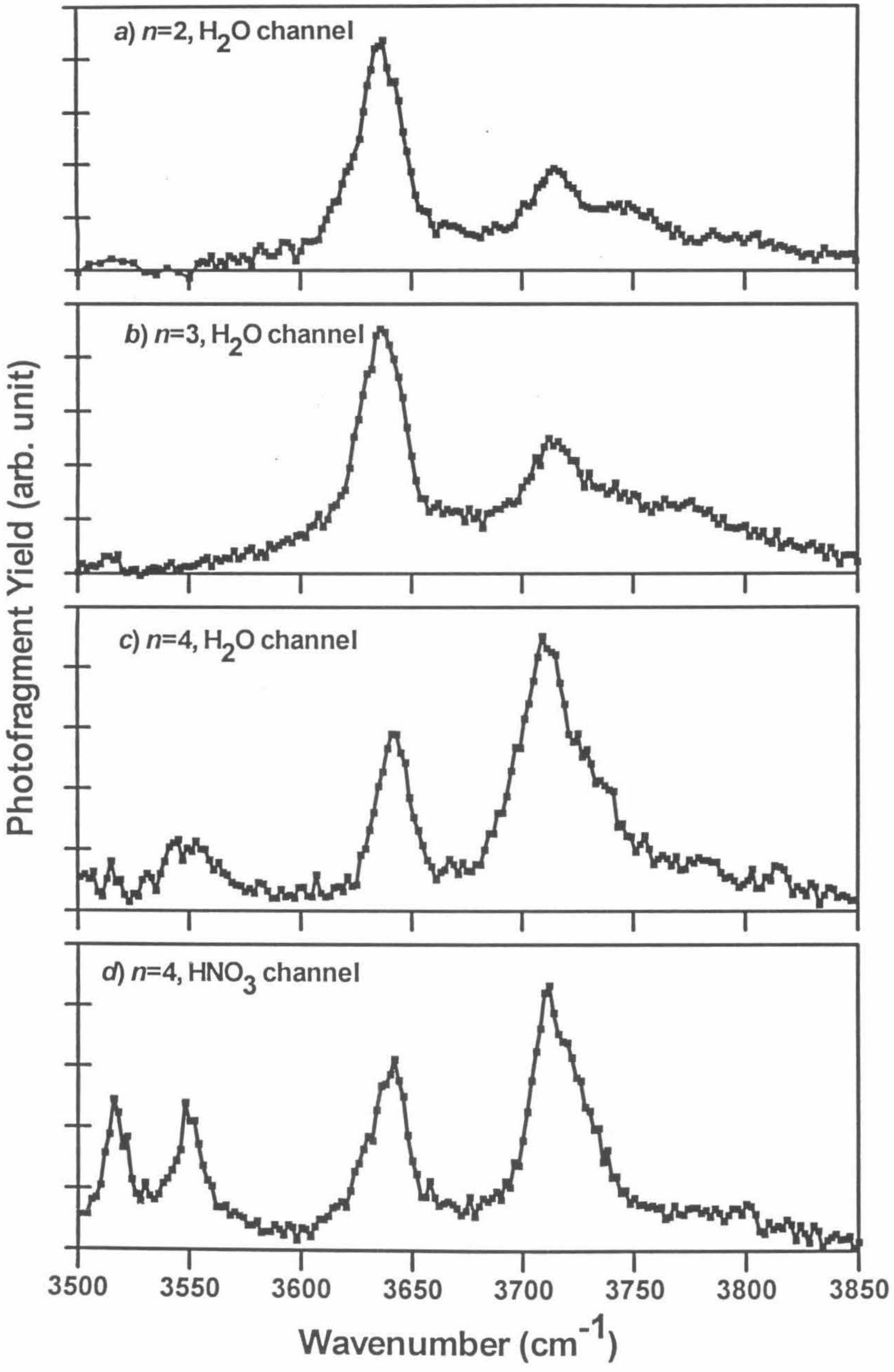

Arb. Intensity
50
45
40
35
30
25
20
15
10
5
0
3200
3250
3300
3350
3400
3450
3500
Wavenumber (cm-1)

Figure 5.



a) n=2, H_2O channel
b) n=3, H_2O channel
c) n=4, H_2O channel
d) n=4, HNO_3 channel
Photofragment Yield (arb. unit)
3500
3550
3600
3650
3700
3750
3800
3850
Wavenumber (cm^{-1})

Figure 6.

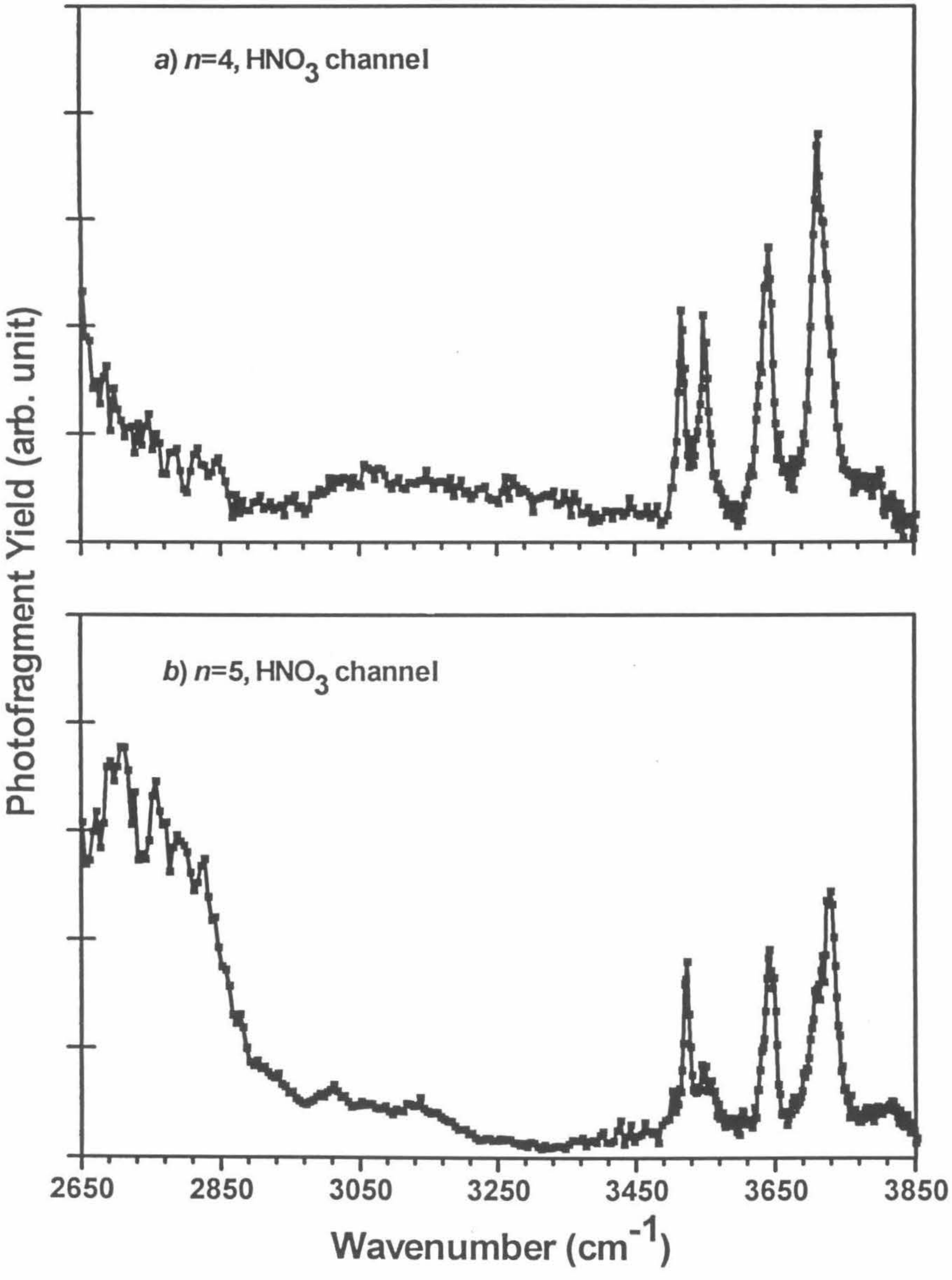
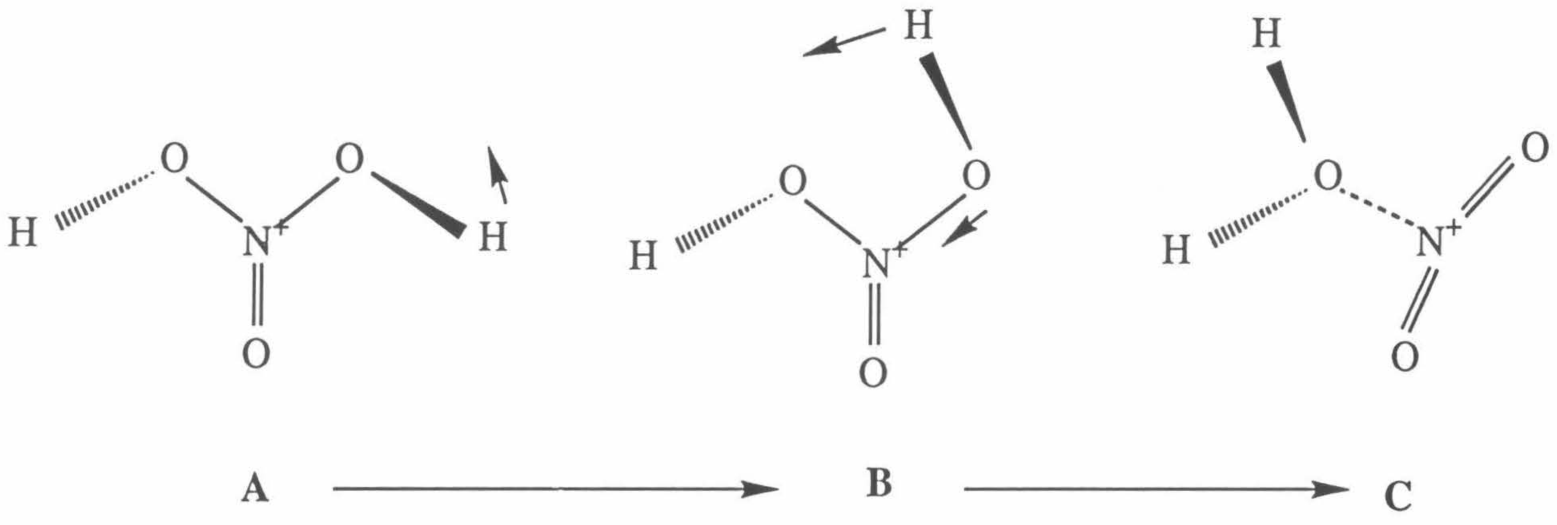
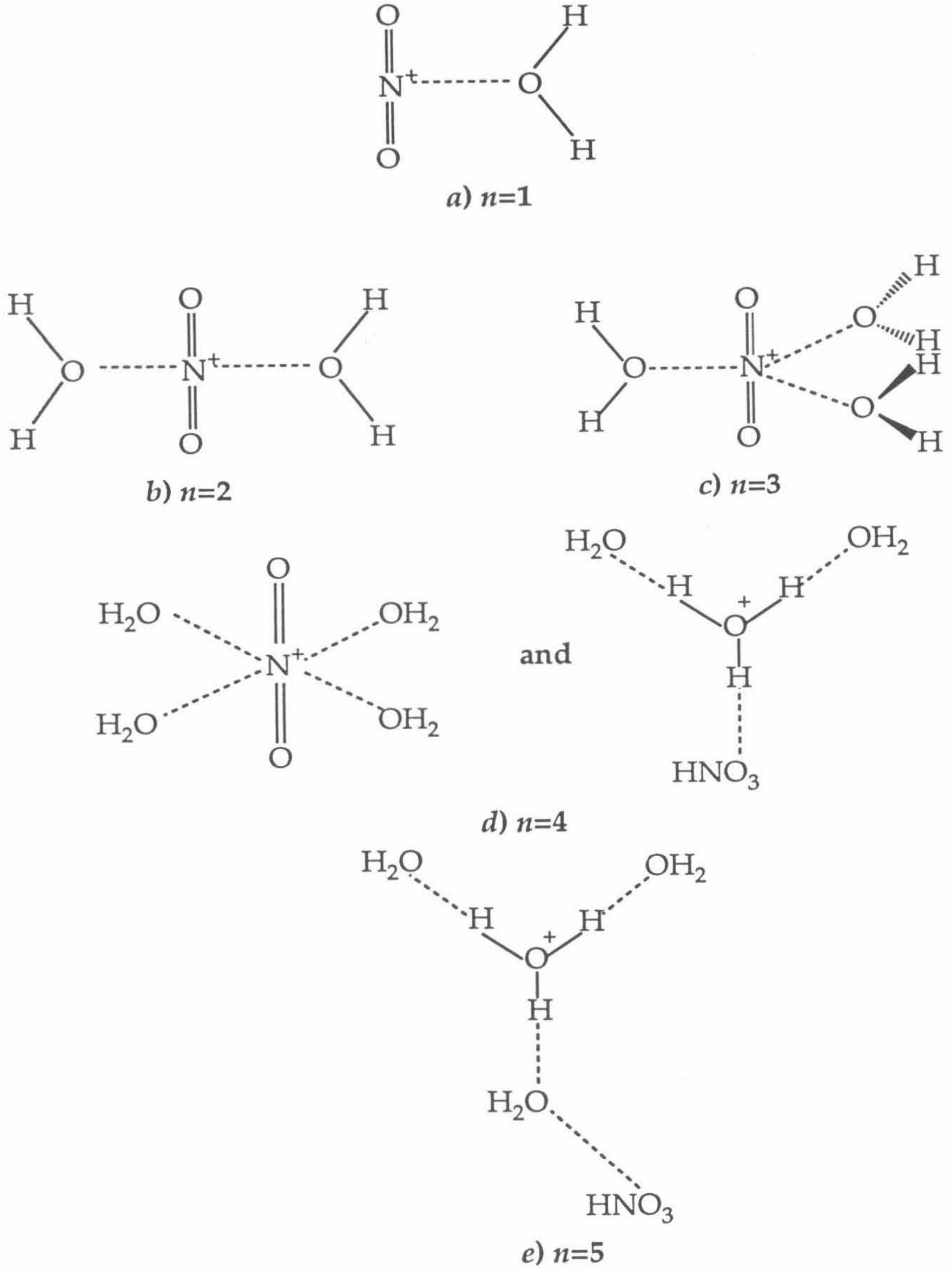

a) n=4, HNO3 channel
b) n=5, HNO3 channel
Photofragment Yield (arb. unit)
2650
2850
3050
3250
3450
3650
3850
Wavenumber (cm-1)

Figure 7.



Hydrogen Transfer

Figure 8.



a) n=1

b) n=2

c) n=3

d) n=4

e) n=5

Figure 9.

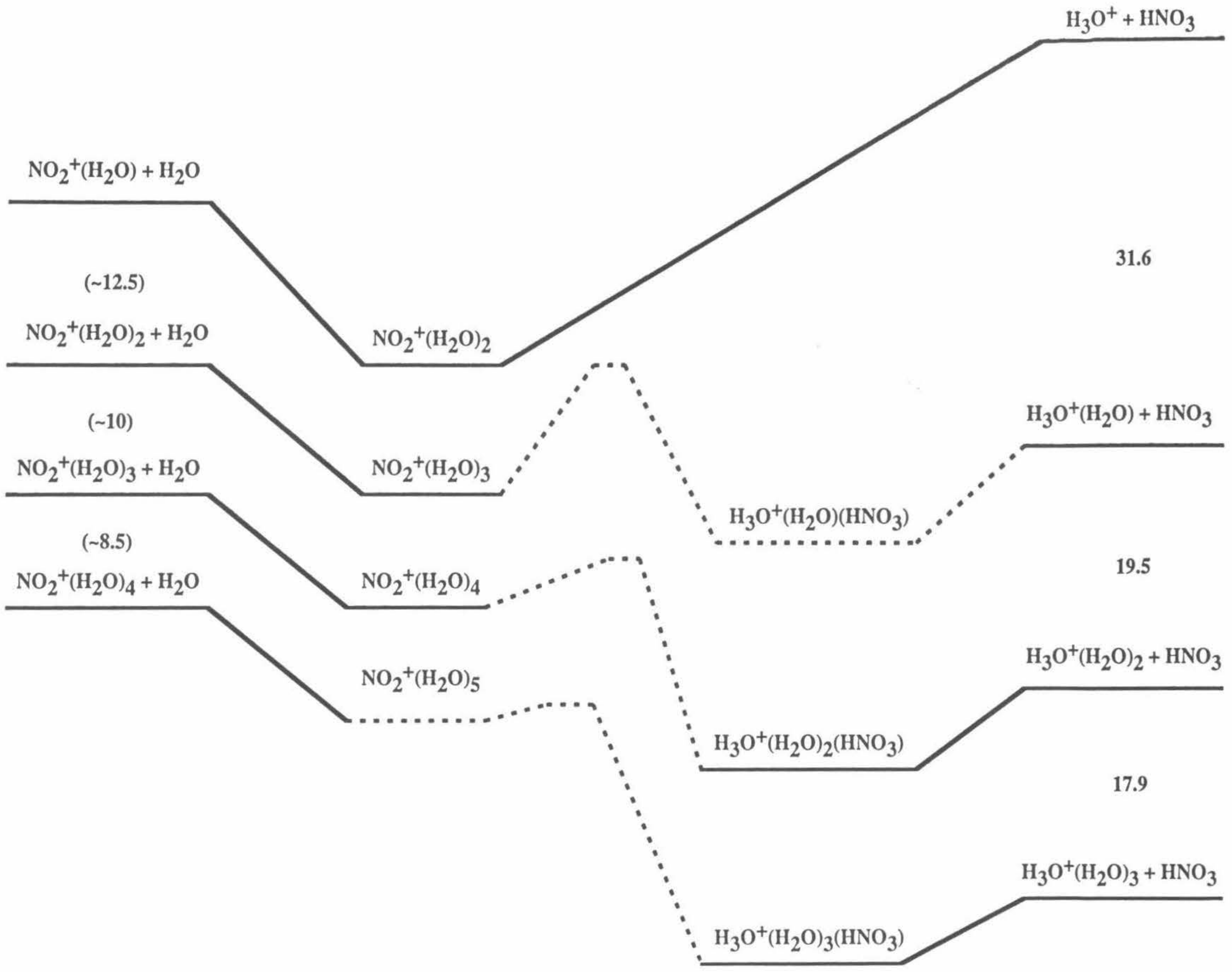

$H_3O^+ + HNO_3$
$NO_2^+(H_2O) + H_2O$
31.6
(~12.5)
$NO_2^+(H_2O)_2 + H_2O$
$NO_2^+(H_2O)_2$
$H_3O^+(H_2O) + HNO_3$
(~10)
$NO_2^+(H_2O)_3 + H_2O$
$NO_2^+(H_2O)_3$
$H_3O^+(H_2O)(HNO_3)$
(~8.5)
$NO_2^+(H_2O)_4 + H_2O$
$NO_2^+(H_2O)_4$
19.5
$H_3O^+(H_2O)_2 + HNO_3$
$NO_2^+(H_2O)_5$
$H_3O^+(H_2O)_2(HNO_3)$
17.9
$H_3O^+(H_2O)_3 + HNO_3$
$H_3O^+(H_2O)_3(HNO_3)$

Figure 10.

Chapter 5. $HClONO_2^+$

5.1 INTRODUCTION

Probably the most startling recent discovery about the Earth's atmosphere was that of the "Antarctic ozone hole."[2] For the past decade, massive depletion of ozone has occurred over Antarctica during the austral spring. For example, in 1989, about 90% of the ozone that was present between 14 and 20 km of altitude in August 23 had been destroyed by October 20.[3] Such a catastrophic loss of stratospheric ozone in such a short time could not have been predicted with our understanding of atmospheric chemistry before this discovery.

The catalytic cycles that are responsible for the ozone depletion have now been well established.[4] The most important cycle among them is the ClO dimer mechanism:[5]

$$\mathrm{ClO + ClO + M \rightarrow (ClO)_2 + M}$$

$$\mathrm{(ClO)_2 + h\nu \rightarrow Cl + ClOO}$$

$$\mathrm{ClOO + M \rightarrow Cl + O_2 + M}$$

$$\mathrm{2 \times (Cl + O_3 \rightarrow ClO + O_2)}$$

$$\text{Net:} \qquad \mathrm{2\,O_3 \rightarrow 3\,O_2}$$

Photolysis of Cl_2 and HOCl in the Antarctic spring produces Cl, which reacts rapidly with ozone to form ClO. This radical then destroys far more ozone molecules through the preceding catalytic cycle. However, chlorine atoms in the stratosphere exist primarily in the form of $ClONO_2$ or HCl, and the gas-phase reactions which convert these stable chlorine reservoirs into the photo-sensitive species Cl_2 and HOCl are too slow to be observed at room temperature.[6,7,8] Hence, gas-phase chemistry alone cannot explain the highly elevated concentration of ClO in the Antarctic winter and spring.[4]

There is now strong evidence that heterogeneous reactions occurring on the surfaces of polar stratospheric clouds (PSCs) are responsible for the conversions of chlorine reservoir species into Cl_2 and HOCl.[9] (PSCs are classified as type I, which is believed to be nitric acid trihydrate (NAT), and type II, which is water ice). Field measurements have shown that air parcels containing high concentrations of ClO had previously experienced the low temperatures necessary for PSC formation, suggesting that such air parcels had been "processed" by PSCs.[10] Laboratory studies in flow tubes and Knudsen cells have clearly demonstrated that such heterogeneous reactions can occur on model film substrates.[11-15] These heterogeneous reactions are

$$\mathrm{ClONO_2} + \mathrm{H_2O}(s) \rightarrow \mathrm{HOCl}(g) + \mathrm{HNO_3}(ad)\ , \tag{1}$$

$$\mathrm{HOCl} + \mathrm{HCl}(ad) \rightarrow \mathrm{Cl_2}(g) + \mathrm{H_2O}(s)\ , \tag{2}$$

$$\mathrm{ClONO_2} + \mathrm{HCl}(ad) \rightarrow \mathrm{Cl_2}(g) + \mathrm{HNO_3}(ad)\ . \tag{3}$$

It should be noted that reaction (3) may be due to the sequential occurrence of (1) and (2). Reaction (1), the hydrolysis of chlorine nitrate, is found to occur on both water ice and water-rich nitric acid trihydrate films. The efficiency of these reactions on ice and water-rich surfaces at lower temperatures has led to the speculation that the mechanisms of the heterogeneous processes might be ionic. For example, Molina *et al.* have postulated that reaction (1) might occur by an acid-catalyzed mechanism:[16]

$$\mathrm{ClONO_2} + \mathrm{H^+} \rightarrow \mathrm{HNO_3{-}Cl^+} \tag{4}$$

with Cl^+ being the leaving group of the intermediate complex.

In order to explore the mechanism of heterogeneous chlorine nitrate reactions, Nelson and Okumura studied the reactions of hydrated hydronium ions with chlorine nitrate in a collision experiment.[17] In that experiment, a beam of hydrated hydronium ions $H_3O^+(H_2O)_n$ formed in the expansion of a pulsed discharge plasma passed through a gas cell containing $ClONO_2$. The ionic products of the ion-molecule reactions were subsequently mass analyzed. The only reaction products detected were $NO_2^+(H_2O)_n$ clusters, with $n = 0, 1, 2$. Since $NO_2^+(H_2O)$ is equivalent to $H_2NO_3^+$ (protonated nitric acid), their results indicated that reaction (1) is acid-catalyzed. However, instead of the mechanism (4) proposed by Molina *et al.*, an alternative mechanism is more plausible:

$$H^+ + ClONO_2 \rightarrow NO_2^+ + HOCl\,. \qquad (5a)$$

In this new mechanism proposed by Nelson and Okumura.,[17] the proton directly attacks the chlorine nitrate at the center oxygen to form HOCl, which is followed by a charge transfer to NO_2^+. NO_2^+ subsequently is solvated by water to form HNO_3.

Lee and Rice have performed *ab initio* calculations on both protonated nitric acid and protonated chlorine nitrate using extended basis sets and a coupled cluster method for configuration interaction, CCSD(T).[18,19] They found that the lowest energy isomers of $HClONO_2^+$ and $H_2NO_3^+$ are weakly bound complexes of $NO_2^+{\bullet}(HOCl)$ and $NO_2^+{\bullet}(H_2O)$, respectively. This theoretical finding strongly supports the mechanism of Nelson and Okumura, since the protonated chlorine nitrate, with its weakly-bound structure, can easily decompose into NO_2^+ and HOCl. Lee and Rice also predicted that a second isomer of chlorine nitrate, $(ClO)(HO)NO^+$, could be formed by protonating one of the terminal O atoms of the nitrate group.[19] This isomer was calculated to lie 20 kcal/mol higher in energy than the $NO_2^+{\bullet}(HOCl)$ complex. The predicted structures for both isomers are shown in figure 1.

The results of Nelson and Okumura have been recently confirmed and extended in selected-ion flow tube (SIFT) experiments by van Doren *et al.*[20] These authors found that, under thermal conditions (223 K), proton transfer from H_3O^+ to $ClONO_2$ leads to two major product channels

$$H_3O^+ + ClONO_2 \rightarrow H_2NO_3^+ + HOCl \ (63\%) , \quad (5b)$$

$$\rightarrow HClONO_2^+ + H_2O \ (31\%) . \quad (5c)$$

This experimental result implies that the $HClONO_2^+$ formed in the drift tube was the weakly bound complex NO_2^+•(HOCl), the lowest energy isomer predicted by Lee and Rice.

We have studied the infrared spectrum of $H_2NO_3^+$ using the technique of vibrational predissociation spectroscopy.[21] Its spectra, given in chapter 4, revealed that the lowest energy isomer is NO_2^+•(H_2O). A second isomer that was predicted by *ab initio* calculations was also found when ions were generated by a "hotter" source, which involved electron ionization of a pulsed jet at the throat of its supersonic expansion.

We have recently applied this experimental technique to obtain the infrared spectrum of $HClONO_2^+$. Our results, presented in this chapter, present direct experimental evidence for the structure of protonated chlorine nitrate.

5.2 EXPERIMENTAL

Details of the experimental apparatus have been described in chapter 2. In this experiment, electron beam ionization was used to generate the $HClONO_2^+$ ion beam.

The chlorine nitrate was synthesized from the reaction of Cl_2O and N_2O_5.[22] Cl_2O was generated by condensing 10 l-atm Cl_2 gas into 44 g of HgO. The reagents were allowed to react in a flask for 3 to 4 days at -78 °C (an acetone - dry ice slush). The reaction flask was then warmed up, transferring Cl_2O into a storage flask cooled at liquid nitrogen temperature (-196 °C). N_2O_5 was made by the dehydration of 68 ml of fuming HNO_3 with 175 g of P_2O_5 in the presence of an ozone flow for about 8 hours. The temperature of the reaction flask was slowly raised from -40 °C to 50 °C during the process. The N_2O_5 product was collected by a storage flask at -78 °C (acetone - dry ice). Finally, small portions of Cl_2O were condensed into a flask containing N_2O_5 at -196 °C; the flask was then warmed to -25 °C to allow reaction. The reaction product, $ClONO_2$, was transferred to a flask at -196 °C by warming up the reaction flask to -78 °C. HNO_3 impurities in the product mixture, which could be readily seen as a white solid in the yellow supernatant at about –70 °C, were removed by vacuum distillation. The purified chlorine nitrate is a pale yellow solid at -196 °C.

Chlorine nitrate was introduced into the gas line by passing 1000 Torr of helium carrier gas over a sample of $ClONO_2$ held in a pyrex flask at -20 °C. Since chlorine nitrate decomposes very easily on surfaces at room temperature, special attention was paid to the gas inlet system. The gas line was made entirely of stainless steel. The piezoelectric element driving the pulsed valve was coated with halocarbon grease to reduce $ClONO_2$ decomposition on the surface. Even so, we needed to passivate the gas inlet system with several Torr of $ClONO_2$ for 2 to 3 days before a stable $HClONO_2^+$ ion beam could be generated.

5.3 RESULTS

The time-of-flight mass spectrum of the ion source is shown in figure 2. The most intense mass peaks and their probable assignments are mass 46 (NO_2^+), 64 ($H_2NO_3^+$), 98 ($H^{35}ClONO_2^+$), 100 ($H^{37}ClONO_2^+$), 109 [$H^+(H_2O)_6$], 143 [$NO_2^+(^{35}ClONO_2)$], and 145

[NO_2^+(^{37}ClONO$_2$)]. The observation of a 3 : 1 intensity ratio between mass 98 and mass 100, which agrees with the natural isotope ratio for species containing a chlorine atom, supports our peak assignments.

Vibrational predissociation spectra were obtained for ions of mass 98 and 100. Their spectra are identical, indicating that these ions are indeed the same chemical species differing only in the chlorine isotope. (Isotope effects are too small to be observed in low resolution spectra.) With this additional spectroscopic evidence, we are confident that mass 98 is $H^{35}ClONO_2^+$ and mass 100 is $H^{37}ClONO_2^+$.

Vibrational excitation of protonated chlorine nitrate ($H^{35}ClONO_2^+$) led solely to formation of NO_2^+ photoproducts (mass 46), indicating that this ion underwent vibrational predissociation by

$$HClONO_2^+ \rightarrow NO_2^+ + HOCl\ . \qquad (6)$$

The infrared spectrum of $HClONO_2^+$, shown in Figure 3, consisted of three bands in the 3000 - 4000 cm^{-1} range. The strongest band is centered at 3582 cm^{-1}, while a band of about half the intensity is observed at 3386 cm^{-1}. The weakest band occurs at 3712 cm^{-1}. The intensity of each of these bands depended linearly on the laser fluence.

5.4 DISCUSSION

The strongest band of protonated chlorine nitrate, at 3578 cm^{-1}, is only ≈20 cm^{-1} to the red of the O-H stretching frequency of HOCl. This suggests the existence of an intact HOCl moiety within $HClONO_2^+$. The possible existence of such a moiety, combined with the observation that the photofragment ion of $HClONO_2^+$ is NO_2^+ (*m*/*e*=46), leads us to propose that the structure of protonated chlorine nitrate is the weakly bound complex predicted by Lee and Rice, NO_2^+•(HOCl).[19] The fact that the experimental band position (3582 cm^{-1}) is in excellent agreement with the scaled *ab initio* frequency (3580 cm^{-1}) further confirms our proposed structure.

For $HClONO_2^+$, only the O-H stretching frequency lies higher than 3000 cm^{-1}. Therefore, the fact that two additional bands were observed was at first quite surprising. The band at 3712 cm^{-1} possess about one fifth the intensity of the 3582 cm^{-1} band, suggesting that it may arise from a strong combinational mode. The most likely candidate is the $\nu_1+\nu_3$ combination band of the NO_2^+ ion core (ν_1 is the N-O symmetric stretch and ν_3 is the antisymmetric stretch), because its intensity would be greatly enhanced by the ionic character of this mode. Jacox has recently observed both the ν_3 (2348 cm^{-1}) and $\nu_1+\nu_3$ (3711 cm^{-1}) bands of NO_2^+ in matrix isolation experiments.[23] *Ab initio* calculations for the two stretching modes of the NO_2^+ moiety in the NO_2^+•(HOCl) isomer suggest that these band frequencies are almost unperturbed by complex formation;[19] therefore, we would expect the combination band to lie near the matrix value.

This assignment of the band is then consistent with the conclusion that the structure of protonated chlorine nitrate is NO_2^+•(HOCl). To verify the origin of the 3712 cm^{-1} band, we intend to study the infrared spectroscopy of NO_2^+•Ar. A similar band around 3712 cm^{-1} in NO_2^+•Ar would confirm our current assignment. This experiment will be carried out soon.

The observation of a third band at 3386 cm^{-1} cannot be explained with the NO_2^+•(HOCl) structure. It may, in fact, arise from another isomer of protonated chlorine nitrate. The location of the band is in excellent agreement with the scaled O-H stretching frequency of 3379 cm^{-1} from the *ab initio* calculations (at the MP2-TZ2P level) for the isomer $(ClO)(HO)NO^+$ (Figure 1). We therefore assign the 3386 cm^{-1} band to this isomer. The $(ClO)(HO)NO^+$ isomer is covalently bound; observation of vibrational predissociation would indicate that the barrier for H transfer to the O atom of the ClO group must be on the order of one photon, *i.e.*, ~10 kcal/mol.

Our band assignments are supported by our earlier studies of the vibrational spectrum of protonated nitric acid, $H_2NO_3^+$. A simple comparison reveals that HNO_3 and $ClONO_2$ have essentially the same structure, with a terminal hydrogen atom replaced by a terminal chlorine atom. For $H_2NO_3^+$, we have found that its lowest energy isomer is $NO_2^+{\bullet}(H_2O)$, and the second isomer has a covalently bonded structure of $(HO)_2NO^+$. These two isomers are formed by protonation of HNO_3 on either the O atom of the OH group or a terminal O atom. Both theory and experiment conclude that $(HO)_2NO^+$ lies higher in energy than the weakly bound complex $NO_2^+{\bullet}(H_2O)$, and is therefore a metastable conformer. The experimental evidence was provided by the change in intensity of the 3375 cm^{-1} band with source conditions. Specifically, this band was observed only in spectra of ions formed by the electron gun source, the same source used in the current chlorine nitrate experiments (Figure 4). When the ions were formed in the pulsed discharge source, this band was absent or very weak. In the discharge source, the ions are expected to undergo many more collisions prior to expansion; the absence of the 3375 cm^{-1} band is then attributed to the quenching of the metastable $(HO)_2NO^+$ by sufficient collisions.

The two structures we propose for $HClONO_2^+$ are analogous to those for $H_2NO_3^+$, the underlying reason being the similarity in structure and reactivity of $ClONO_2$ and HNO_3. Assignment of the $HClONO_2^+$ 3386 cm^{-1} band is thus supported by the analogous assignment for protonated nitric acid. We can confirm our hypothesis about the 3386 cm^{-1} band (and therefore the existence of the (ClO)(HO)NO^+ isomer) by recording the spectra of $HClONO_2^+$ formed in the pulsed discharge source and looking for the intensity decrease of this band. However, to date, we have been unable to produce protonated chlorine nitrate in our discharge source. It appears that any $HClONO_2^+$ produced undergoes sufficient collisions in the discharge to form protonated nitric acid. The most likely mechanism is ligand switching, since theory suggests that the binding energy of HOCl to NO_2^+ is $\approx$4 kcal/mol less than that of H_2O. Further experiments in which we change the condition of the discharge source, or change the buffer gas used in the expansion, may produce $HClONO_2^+$ ions with a different isomer ratio. Investigation of these spectra should provide the confirmation for the existence of the second isomer.

Our spectroscopic results provide experimental confirmation that the most stable form of protonated chlorine nitrate is the weakly bound complex $NO_2^+\bullet$(HOCl), as predicted by Lee and Rice. The observed gas-phase behavior supports the suggestion by Nelson and Okumura that proton catalysis is the mechanism for the reaction of H_2O with $ClONO_2$ on the surfaces of PSCs, especially water ice (see Figure 5). The rapidity of this reaction on ice surfaces could be explained by the readiness of the $NO_2^+\bullet$(HOCl) reaction intermediate to give up HOCl.

In the condensed phase, the reaction would initially involve initial proton migration within the ice by the Grottius mechanism,[24] which should be rapid. The PSCs are expected to be acidic, given the partial pressures of HCl in the stratosphere. Since $ClONO_2$ possesses a proton affinity higher than that of H_2O, it will act as a trap site for a mobile proton. The $ClONO_2$ will be bound to the surface by hydrogen bonds, with *surrounding* H_2O. The O atoms all possess large negative partial charges (on the order of –0.45), and will therefore act as proton acceptors. In the proposed mechanism, proton attack at the ClO group will lead to formation of an intermediate complex $NO_2^+{\bullet}(HOCl)$. The NO_2^+ will then undergo an hydration reaction with the surrounding water molecules:[25]

$$NO_2^+(H_2O)_n \rightarrow H_3O^+(H_2O)_{n-2} + HNO_3 \qquad (7)$$

analogous to the reactions observed in the $NO_2^+(H_2O)_n$ clusters reported in Chapter 4. The net reaction is then

$$ClONO_2 + H_2O \rightarrow HOCl + HNO_3\ . \qquad (8)$$

The molecular nitric acid may undergo dissociative ionization $HNO_3 \rightarrow H^+(aq) + NO_3^-(aq)$; if so, the total reaction will become exothermic. The pivotal step in the above reaction is protonation of the ClO group and formation of the complex, which should lead directly to formation of HOCl.

The proposed mechanism is in essence acid-catalyzed hydrolysis of chlorine nitrate. A key prediction of the proposed reaction is the existence of a nitronium ion intermediate on or near the surface. The observation of NO_2^+ on a model surface dosed with $ClONO_2$ would provide strong evidence in favor of this reaction pathway.

The nitronium ion has recently been observed on the surface of ice. Horn *et al.* have studied the reflectance absorption infrared spectroscopy (RAIRS) of water ice surfaces dosed with N_2O_5.[26] At temperatures of 160 K and higher in which desorption of N_2O_5 occurred, they observed a strong band at 2390 cm^{-1}, which they assigned to the antisymmetric stretch of NO_2^+. In addition, they found that the 2390 cm^{-1} band and the bands attributed to NO_3^- grew together in intensity. These authors therefore proposed that chemical desorption of N_2O_5 on PSCs follows a mechanism analogous to that proposed by Nelson and Okumura for chlorine nitrate:

$$H_3O^+ + N_2O_5 \rightarrow H_2O + NO_2^+\bullet(HNO_3) , \qquad (9)$$

i.e., protonation of N_2O_5 forms the weakly bound complex $NO_2^+\bullet(HNO_3)$. HNO_3 is subsequently released and dissociatively ionized, *i. e.*, $HNO_3 \rightarrow H^+(aq) + NO_3^-(aq)$.

5.5 CONCLUSIONS

We have reported the vibrational predissociation spectrum of protonated chlorine nitrate, which dissociates into HOCl + NO_2^+ products. With the help of *ab initio* predictions by Lee and Rice, we assign the observed bands to two isomers of $HClONO_2^+$. One is a weakly bound complex of the nitronium ion with HOCl, $NO_2^+\bullet(HOCl)$, formed by protonation of the O atom of the ClO group and exhibiting an HOCl-like OH stretch and a combination band of NO_2^+ stretches. The other has a covalently bonded structure of $(ClO)(HO)NO^+$, formed by protonation of a terminal O atom, with a vibrational band at 3386 cm^{-1}. These observed vibrational frequencies are in excellent agreement with the *ab initio* frequencies of Lee and Rice. The observation of two isomers formed in our electron bombardment source is consistent with similar behavior exhibited by protonated nitric acid.

Our experimental observation of the complex $NO_2^+\bullet(HOCl)$ upon protonation of chlorine nitrate provides strong support to the ionic mechanism of $ClONO_2$ reaction on PSC surfaces proposed by Nelson and Okumura. The ionic mechanism could explain the rapid hydrolysis of $ClONO_2$ on Polar Stratospheric Cloud particles, a critical step leading to the dramatic springtime depletion of ozone over Antarctica. This mechanism is supported by recent experiments from other groups and in its general form, may apply to other important heterogeneous reactions in the stratosphere (such as N_2O_5 chemical desorption on PSC) as well.

5.6 REFERENCES

(1) This chapter is prepared from a manuscript to be submitted for publication.

(2) Farman, J. C.; Gardiner, B. G.; Shankin, J. D. *Nature* **1985**, *315*, 207.

(3) Deshler, T.; Hoffmann, D. J.; Hereford, J. V.; Sutter, C. B. *Geophys. Res. Lett.* **1990**, *17*, 151.

(4) Anderson, J. G.; Toohey, D. W.; Brune, W. H. *Science* **1991**, *251*, 39.

(5) Molina, L. T.; Molina, M. J. *J. Phys. Chem.* **1987**, *91*, 433.

(6) Atkinson, R.; Tuazon, E. C.; MacLeod, H.; Aschmann, S. M.; Winer, A. M. *Geophys. Res. Lett.* **1986**, *13*, 117.

(7) Atkinson, R.; Aschmann, S. M.; Tuazon, E. C.; Goodman, M. A.; Winer, A. M. *J. Atmos. Chem.* **1987**, *5*, 83.

(8) Molina, L. T.; Molina, M. J.; Stachnik, R. A.; Tom, R. D. *J. Phys. Chem.* **1985**, *89*, 3779.

(9) Solomon, S.; Garcia, R. R.; Rowland, F. S.; Wuebbles, D. J. *Nature* **1986**, *321*, 755.

(10) Toohey, D. W.; Avallone, L. M.; Lait, L. R.; Newman, P. A.; Schoeberl, M. R. *Science*, **1993**, *261*, 1134.

(11) Leu, M. T. *Geophys. Res. Lett.* **1988**, *15*, 17.

(12) Tolbert, M. A.; Rossi, M. J.; Golden, D. M. *Geophys. Res. Lett.* **1988**, *15*, 847.

(13) Tolbert, M. A.; Rossi, M. J.; Malhotra, P.; Golden, D. M. *Science* **1987**, *238*, 1258.

(14) Rossi, M. J.; Malhotra, R.; Golden, D. M. *Geophys. Res. Lett.* **1987**, *14*, 127.

(15) Leu, M. T.; Moore, S. B.; Keyser, L. F. *J. Phys. Chem.* **1991**, *95*, 7763.

(16) Wofsy, S. C.; Molina, M. J.; Salawitch, R. J.; Fox, L. E.; McElroy, M. B. *J. Geophys. Res.* **1988**, *93*, 2442.

(17) Nelson, C. M.; Okumura, M. *J. Phys. Chem.* **1992**, *96*, 6112.

(18) Lee, T. J.; Rice, J. E. *J. Phys. Chem.* **1992**, *96*, 650.

(19) Lee, T. J.; Rice, J. E. *J. Phys. Chem.* **1993**, *97*, 6637.

(20) van Doren, J. M.; Viggiano, A. A.; Morris, R. A. *J. Amer. Chem. Soc.*, submitted Feb. **1994**.

(21) Cao, Y.; Choi, J.-H.; Haas, B. M.; Okumura, M. J. Chem. Phys., to be submitted.

(22) Schmeisser, M. *Inorg. Synth.* **1967**, *9*, 127.

(23) Forney, D.; Thompson, W. E.; Jacox, M. E. *J. Chem. Phys.* **1993**, *99*, 7393.

(24) Fletcher, N. H. The *Chemical Physics of Ice*, Cambridge University Press, Cambridge, **1970**.

(25) Cao, Y.; Choi, J.-H.; Haas, B.-M.; Johnson, M. S.; Okumura, M. *J. Chem. Phys.* **1993**, *99*, 9307.

(26) Horn, A. B.; Koch, T.; Chesters, M. A.; McCoustra, M. R. S.; Sodeau, J. R. *J. Phys. Chem.* **1994**, *98*, 946.

5.7 FIGURE CAPTIONS

Figure 1. The *ab initio* structures of protonated nitric acid and protonated chlorine nitrate (from Lee and Rice, ref. 18 and 19).

Figure 2. The time-of-flight mass spectrum of the protonated chlorine nitrate source. $HClONO_2^+$, at mass 98 and 100, was formed by electron beam ionization of a pulsed gas expansion. The gas was a mixture of chlorine nitrate (5 Torr) and helium (1000 Torr).

Figure 3. The infrared vibrational predissociation spectra of protonated chlorine nitrate, $HClONO_2^+$, formed in electron beam source.

Figure 4. The infrared vibrational predissociation spectra of protonated nitric acid. (a) by $H_2NO_3^+$ formed in the pulsed discharge source of $HNO_3/H_2O/He$ (b) the additional peak observed in the electron beam ionization source.

Figure 5. A proposed mechanism for the acid-catalyzed hydrolysis of chlorine nitrate on ice surfaces. This ionic mechanism could explain the rapid destruction of chlorine nitrate on Polar Stratospheric Cloud particles. The upper panel shows the reactants, mobile protons in ice in the form of H_3O^+ and $ClONO_2$ hydrogen-bonded near the surface. The O atom on the ClO group can act as a trap for the proton, since chlorine nitrate has a higher proton affinity. In the middle panel, proton transfer to chlorine nitrate occurs, leading to formation of the weakly-bound complex of HOCl and NO_2^+. In the lower panel, ligand switching has occurred and the HOCl is displaced by H_2O, which binds

more strongly to the nitronium ion.

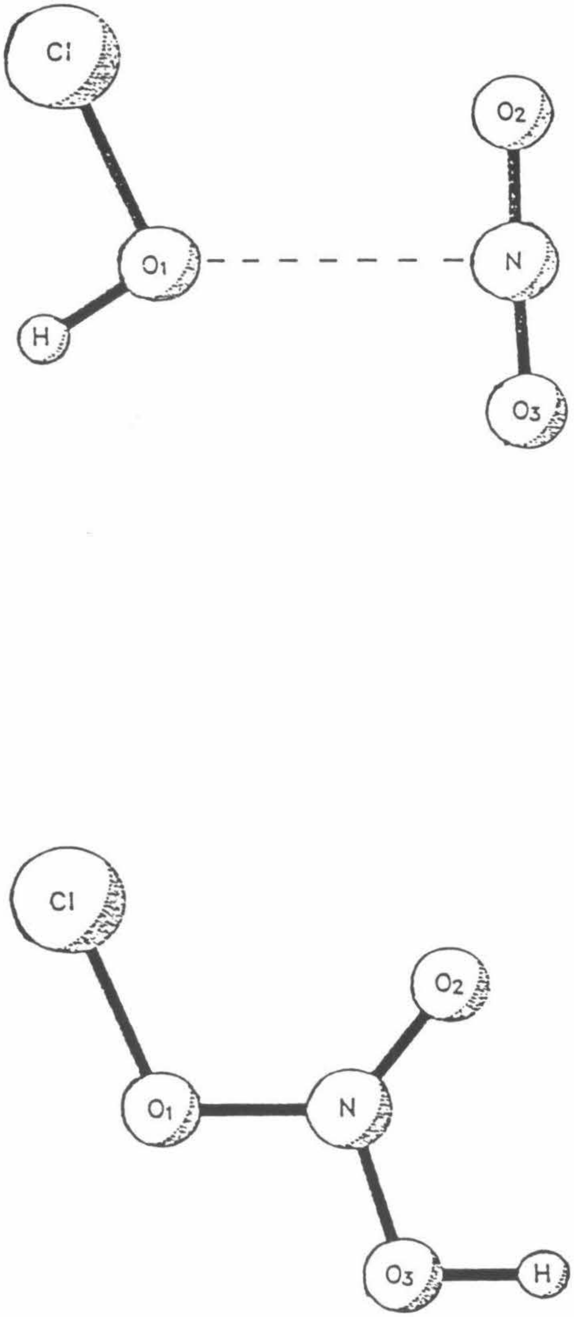
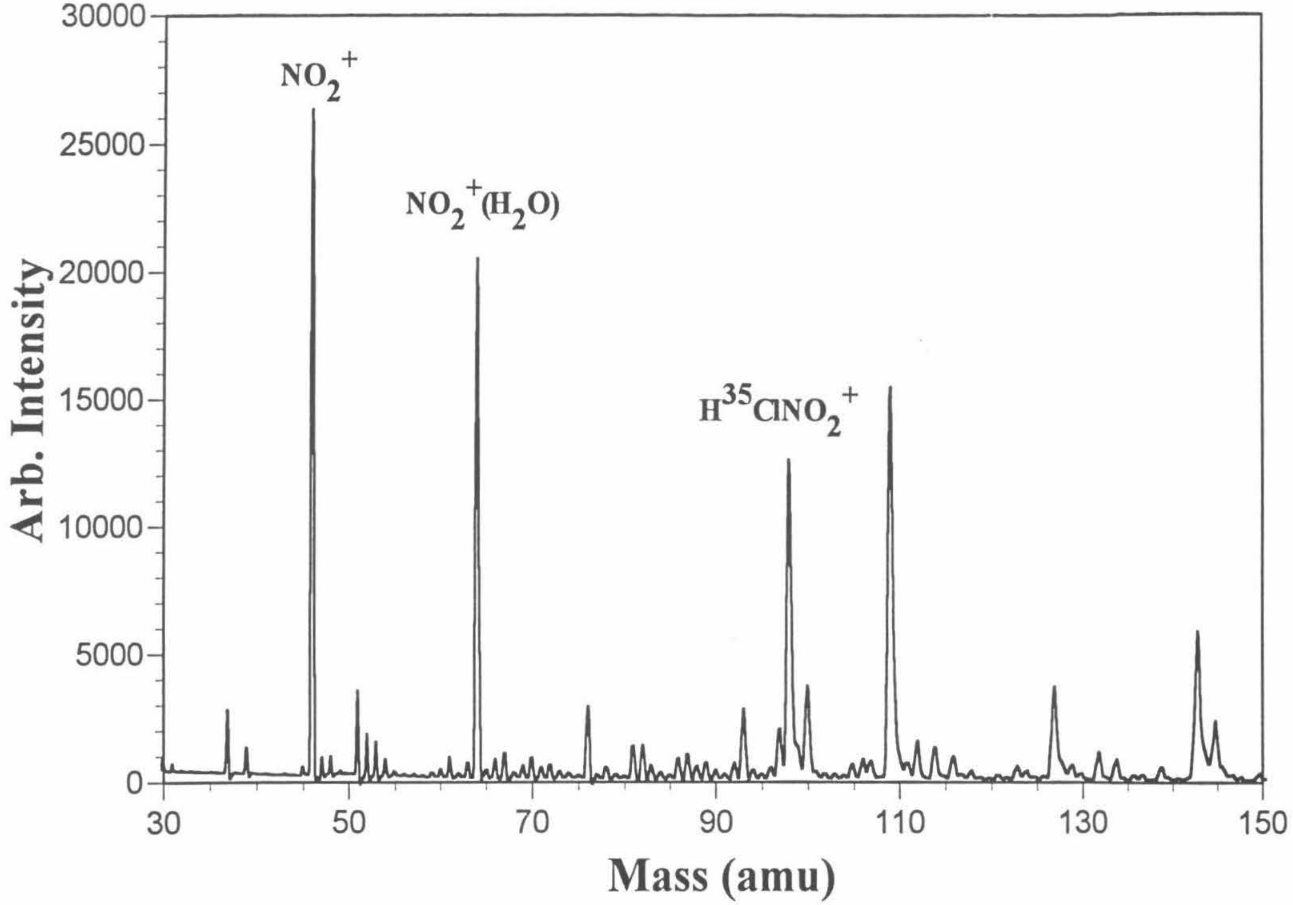


Figure 1.



NO_2^+
$NO_2^+(H_2O)$
$H^{35}ClNO_2^+$
Arb. Intensity
30000
25000
20000
15000
10000
5000
0
30
50
70
90
110
130
150
Mass (amu)

Figure 2.

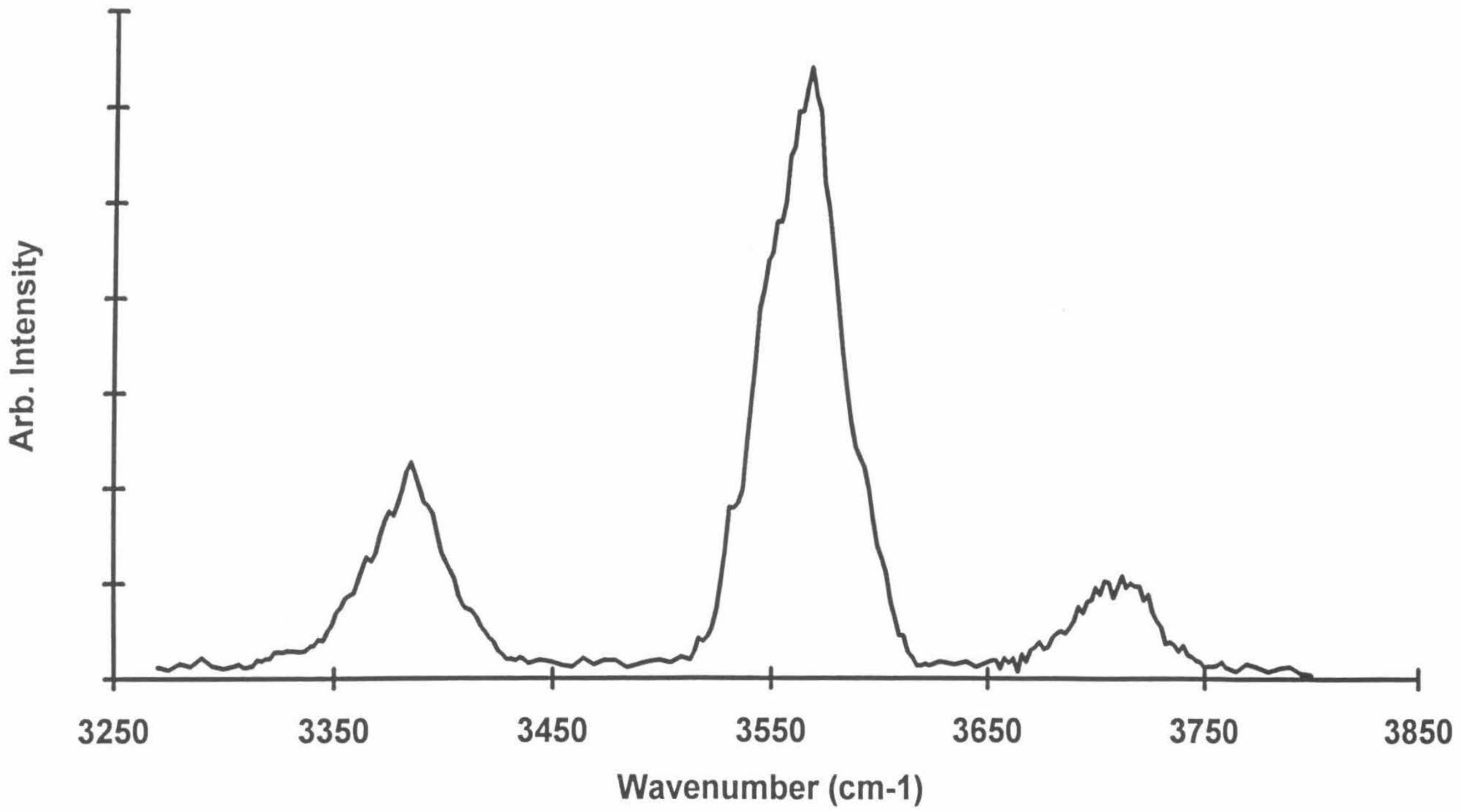

Arb. Intensity
3250
3350
3450
3550
3650
3750
3850
Wavenumber (cm-1)

Figure 3.

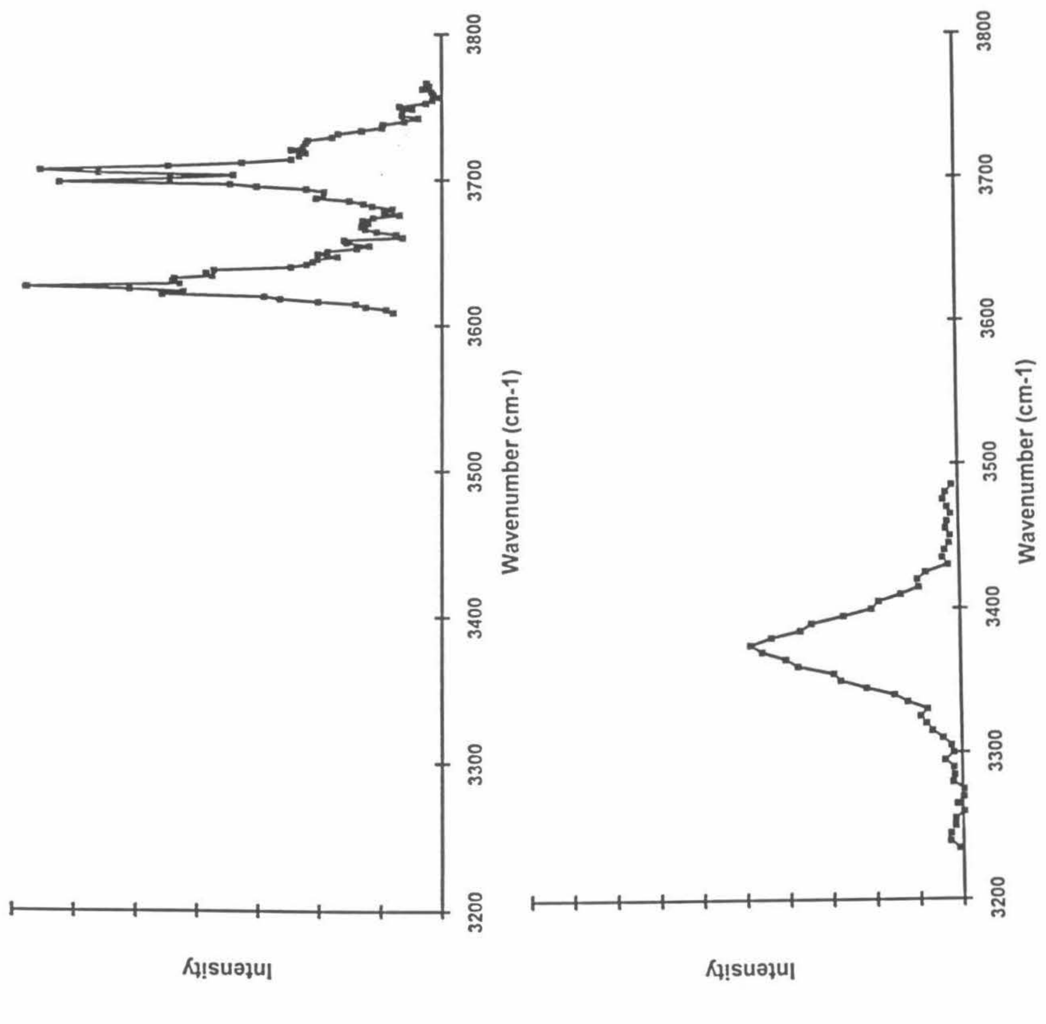
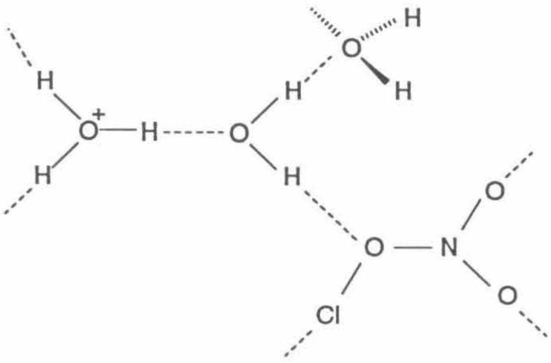

Intensity
Wavenumber (cm-1)
3200
3300
3400
3500
3600
3700
3800
Intensity
Wavenumber (cm-1)
3200
3300
3400
3500
3600
3700
3800

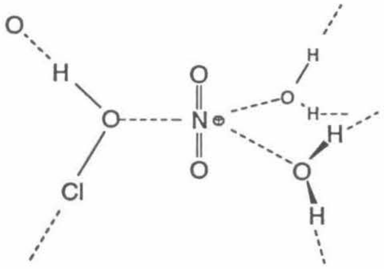
Figure 4.

PROPOSED MECHANISM FOR $ClONO_2$ DECOMPOSITION ON ICE

1. Mobile proton in ice trapped by $ClONO_2$



2. Adduct Formation



3. Hydration of NO_2^+

$$NO_2^+ + 2H_2O \rightarrow HNO_3 + H_3O^+$$

$$HNO_3 \rightarrow H^+ + NO_3^-$$

Figure 5.

Chapter 6. Ongoing Research

6.1 SOLVATED HALIDE CLUSTERS

Solution-phase solvation of electrolytes is one of the most fundamental chemical processes. In fact, chloride ion solvation by bulk water is often discussed in chapter one of general chemistry textbooks. However, many details concerning the solvation of halide ions are still not well understood.

In an aqueous solution, the water molecules that hydrogen bond to halide ions undergo rapid exchange ($\sim 10^{11}$ s^{-1}).[1] Therefore, solution-phase methods that can be used to study halide solvation are limited. For example, the nuclear magnetic resonance technique is ineffective, as it responds on a much longer time scale.

However, techniques that are applicable to aqueous halide solutions have achieved some success. In a recent neutron diffraction study of aqueous $NiCl_2$ solution, Powell *et al.* was able to obtain the radial pair distribution functions, $g_{ClH}(r)$ and $g_{ClO}(r)$,[2] and from them determine the distances between Cl^- and the first solvation shell water molecules ($r_{Cl\text{-}H(1)} \approx 2.28$ Å, $r_{Cl\text{-}O} \approx 3.1$ Å, $r_{Cl\text{-}H(2)} \approx 3.7$ Å). Since the vibrational frequencies of solvent molecules depend strongly on their neighboring interactions, infrared spectroscopy can also be used to probe solvation. However, infrared spectra obtained in aqueous solutions are usually very broad, containing only some information about the averaged anion - solvent interaction.[3]

Clusters are an ideal environment for the study of anion solvation. By adding solvent molecules to the halide anion sequentially and studying the change in cluster properties, a detailed understanding of ion-solvent interaction can be obtained. To date, experiments on solvated halide clusters have been mostly thermodynamic studies of cluster association reactions such as

$$X^-(ROH)_{n-1} + ROH \rightarrow X^-(ROH)_n$$

Through these studies, Kebarle, Castleman, and Hiraoka have obtained binding energies of F^-, Cl^-, Br^- and I^- with various polar solvents.[4] Recently, photoelectron spectroscopy was used to obtain the size-dependence of solvent stabilization energy E_{stab} of an anion. Cheshnovsky *et al.* observed a leveling-off in the increase of E_{stab} in $I^-(H_2O)_n$ clusters containing more than six water molecules, an indication that six water molecules form the first solvation shell around the iodide ion.[5] These authors also suggested the same behavior in hydrated chloride and bromide clusters.

Theoretical work on halide-solvent clusters have provided much insight into the nature of solvation. *Ab initio* and molecular simulation calculations have yield optimized geometries for systems such as $F^-(H_2O)_n$, $Br^-(H_2O)_n$,$Cl^-(H_2O)_n$ and $Cl^-(CH_3OH)_n$. For example, the angular deviation from the collinear $X^-\cdots H{-}O$ configuration in the monohydrate clusters is calculated to increase from Br^- to Cl^- to F^- (Figure 1). However, for the larger clusters, there exists a controversy among different calculations as to whether the chloride anion is solvated by surrounding ligands, or resides on the cluster surface instead.[6]

We are attempting to study halide-solvent clusters using vibrational predissociation spectroscopy. The system under current study is $Cl^-(H_2O)_n$. Experimentally, CCl_4 and H_2O vapors are mixed with the helium carrier gas. The resulting mixture is expanded and the supersonic gaseous jet is ionized by an electron gun. With this technique, we have generated $Cl^-(H_2O)_n$ clusters (Figure 2). The underlying process can be described as a dissociative electron attachment to CCl_4, followed by association of Cl^- with water molecules. Initial studies of $Cl^-(H_2O)$ reveal that the O-H stretch of the water ligand has a very large bandwidth. This may be due to the extremely low potential barrier (≤ 1 kcal/mol) for the pseudo in-plane rotation of H_2O in this cluster.

More extensive work on $Cl^-(H_2O)_n$ clusters is being carried out in our laboratory. Studies on $Cl^-(H_2O)_2$ and larger clusters will provide information that might resolve the current dispute on the position of halide ions within the cluster. Characteristics of Cl^- solvation can also be obtained by changing the solvent molecule. For example, studies on $Cl^-(CH_3OH)_n$ clusters will help us understand the roles that functional groups OH and CH_3 play in the Cl^- solvation.

6.2 METAL–LIGAND COMPLEXES

In recent years, remarkable progress has been made on the gas phase studies of transition metal ion chemistry. Examples of research in this area include ion-beam studies of organometallic chemistry, photodissociation studies, and metal cluster reactivities.[7] However, spectroscopic studies have been few and have all probed the electronic transitions of the metal. Infrared spectroscopy, on the other hand, can directly probe the vibrations of a ligand. Therefore, low resolution IR spectroscopy can yield vibrational frequencies of the ligand, providing a test for the high level *ab initio* calculations that have been performed on metal-ligand complexes. Spectra at higher resolution may even provide rotational constants of some simple complexes.

One system that has been extensively studied is Co^+-L_n. For example, Bowers *et al.* have obtained binding energies of Co^+-$(H_2)_n$ clusters.[8,9] They have found that the first and second H_2's have roughly equal binding energies (18.2 and 17.0 kcal/mol), the third and fourth binding energies are also equal (9.6 kcal/mol), the fifth and sixth cluster *binding* energies are 4.3 and 4.0 kcal/mol. The seventh H_2 has a binding energy of 0.8 kcal/mol, indicating that it is not directly bound to the cobalt ion. The proposed structures for the first six clusters are shown in figure 3. *Ab initio* vibrational frequencies of the H_2 ligands are also available.[8]

Co^+-$(H_2)_n$ is an excellent system with which to start our spectroscopic studies. Similar to the SiH_7^+ ion discussed in chapter 3, H_2 vibrations in these clusters are also highly perturbed, and thus are dipole-allowed transitions. Due to its large binding energy, Co^+-(H_2) and Co^+-$(H_2)_2$ may not predissociate upon excitation, but larger clusters should. An interesting question arises when one examines the structure of Co^+-$(H_2)_3$, in which the three H_2 ligands bind differently with Co^+. Will the vibrational excitation of a strongly bound H_2 ligand causes a release of the most weakly bound H_2 ligand from the cluster?

We attempted to study Co^+-$(H_2)_n$ clusters before, generating them by sputtering of the discharge electrodes with little success. I have since designed a laser vaporization source, analogous to one developed by Smalley *et al.*[10] A schematic of the source is presented in figure 4. A few milijoules/pulse of a frequency doubled Nd:YAG beam is focused onto the metal rod, generating a plasma plume above the metal surface. This plume is then swept out by a supersonic expansion of a hydrogen carrier gas and cooled by the thermalizing collisions. Clustering association with the carrier gas molecules could thus occur. To avoid ablating at the same position on the metal rod, the metal rod undergoes a continuous screwing motion, driven by a stepper motor mechanism. With the laser vaporization source now completed, we are resuming our research on metal-ligand complexes.

6.3 METASTABLE ELIMINATION

The principal factor limiting the detection sensitivity of our current apparatus is the presence of background fragment ions. After a parent ion is accelerated to its final kinetic energy, it may still dissociate if (*a*) it is in an excited state where the internal energy is higher than the binding energy, or (*b*) it collides with a residual molecule along its flight path. The resulting fragment ion would attain approximately the same velocity as its parent ion, thus arriving at the detector at about the same time as a genuine photofragment. The statistical noise of the background fragments prevents us from detecting single photofragment ions.

In order to enhance the sensitivity of the experiment, interfering background due to these so-called "metastable" ions must be drastically reduced. The reduction of metastable interference is especially helpful in high-resolution spectroscopy, where photofragment signals are small because only ions that belong to a particular quantum state can be excited at each moment.

A variety of attempts have been made to reduce the metastable background. The most straightforward method is to reduce the fragmentation rate of the parent ion. When a properly designed, supersonically cooled ion source is employed, ions generated have little internal energy for predissociation, thus making dissociation through collision with background gas the primary mechanism of fragmentation. Specifically, in our apparatus, where the pressure in the time-of-flight chamber is about 10^{-6} Torr (corresponding to a mean-free-path length of 50 meters), collisional dissociation is the predominant channel of background fragmentation. Consequently, reducing the background pressure in the time-of-flight chamber, by employing either larger vacuum pumps or superior differential pumping schemes, is a viable route to reducing metastable background.

Another way to reduce the metastable background is to separate metastable ions from the photofragments so that they do not interfere with the photodissociation signal. One idea is dissociating the parent ions inside the reflectron (a configuration similar to the one employed by Duncan *et al.*[11]). With this configuration, photofragment ions will have a different kinetic energy as well as a separate flight path from that of the metastable ions, which are mostly generated in the time-of-flight section. The net result is that the photofragment and the metastable arrive at the detector at different times. However, there are several disadvantages to this technique. First of all, ions diverge in the reflectron field, therefore, a complete overlap between the laser beam and the parent ions inside the reflectron cannot be easily achieved. Second, since reflectron fields are tilted with respect to the beam axis, dissociation inside the reflectron causes a transverse deviation of the photofragment beam. Finally, this technique is not very effective: in most situations, temporal separation between the photofragment and metastable ions is too small to allow complete separation of these two peaks.

I have conceived a device, which I will refer to as the "meta-deflector," that could effectively eliminate the metastable background. In the following paragraphs, I will illustrate its operation principle and discuss the possible designs of this device.

The meta-deflector is shown in figure 5a. It consists of three regions of equal length l, a central field-free region with two deflection regions on each side. This device is placed along the path of the ion beam, with the laser interaction spot lying in the middle of its central region.

The mass of the parent and fragment ion are designated m_p and m_f, respectively. We choose the electric fields of the meta-deflector to be E (pointing down) in the first deflection region and $-(m_f / m_p)\ E$ (pointing up) in the second region. For an ion which does not change its mass inside the meta-deflector, these electric fields introduced a net downward deflection: the ion emerges from the meta-deflector tilting slightly downward. However, if an ion enters the meta-deflector as a parent ion (mass m_p), undergoes predissociation after absorbing a photon at the laser interaction spot (changing its mass from m_p to m_f), a different situation occurs. Specifically, the downward acceleration that the ion has experienced in the first region

$$a_1 = q\ E\ /\ m_p$$

and the upward acceleration that the ion has experienced in the second region

$$a_2 = q\ [\ -(m_f / m_p)\ E\ \ ]\ /\ m_f\ =\ -q\ E\ /\ m_p$$

are the same. Since the ion spends equal amount of time in both regions, the net deflection it has experienced is therefore zero, *i. e.*, the ion will leave the meta-deflector travelling horizontally.

In other words, after passing through the meta-deflector, the photofragment and the metastable proceeds in a slightly different direction: the metastable ion travels slightly downward while the photofragment ion moves horizontally. The subsequent journey of these ions causes a transverse separation between them.

The transverse separation between the photofragment and the metastable can be evaluated using simple trajectory calculations. Specifically, if the kinetic energy of the parent ion is E_k, the equivalent field-free length between the meta-deflector and the detector is D, the transverse separation Δz is

$$\Delta z = (\sqrt{m_p/m_f} - \sqrt{m_f/m_p})\frac{El}{E_k}D \ .$$

Using the parameter values of our apparatus, *i. e.*, E_k = 2.3 keV and D = 70 cm, and appropriate values of 1 cm and 400 v/cm for l and E, we then have

$$\Delta z = 13.2(\sqrt{m_p/m_f} - \sqrt{m_f/m_p}) \quad cm \ .$$

For fragment-parent mass ratios of 0.8 and 0.9, the transverse separation distances are 2.7 cm and 1.3 cm respectively. With the active area of the MCP detector at about 2 cm in diameter, separations caused by the meta-deflector are large enough to allow the detection of photofragments while the metastable ions completely miss the detector.

The calculations above have assumed that the meta-deflector consists of regions of constant electric field. An exact field distribution like this cannot be realistically achieved. Two different designs of the meta-deflector, which generate electric fields close to the ideal distribution, are discussed in the following paragraphs.

The first design, shown in figure 5b, consists of planar potential plates: deflection fields are generated by horizontal plates while the penetration of the deflection fields is reduced by the vertical plates. Using the ion simulation program SIMION,[12] I have obtained the electric field distribution as well as ion trajectories through this device. The results showed that a small beam divergence is introduced by the edge effect.

A more promising design takes advantage of the "octopole deflector".[13] Shown in figure 5c, it is a hollow cylinder consisting of eight segments, with different voltages applied to them. If a is chosen to be 0.35 to 0.45, the deflector field is very close to uniform over more than half the deflector area.

There is a slight inconvenience in using the meta-deflector. Its deflection fields cause a modest downward offset of the ion beam axis. Due to this slight offset, the photofragment ions arrive at the detector ~0.4 cm lower. This can be easily corrected with a slight increase of the electric field in the second region which introduces a small net upwards acceleration for the photofragments.

In conclusion, the meta-deflector is a simple device, easy to operate, and requires only DC electric fields. The theoretical calculations and simulations indicate that for cases where fragment-parent mass ratio is smaller than 0.9, metastable ions can be mostly eliminated (*i.e.*, not reaching the detector) with this device. Therefore, this new technique should have an immediate impact on predissociation experiments of $Cl^-(H_2O)_n$, $F^-(H_2O)_n$, and $K^+(H_2O)_n$.

6.4 REFERENCES

(1) Bopp, P. In *The Physics and Chemistry of Aqueous Ionic Solutions*; Bellissent-Funel, M.-C., Neilson, G. W., Eds.; Reidel: Dordrecht, **1987**.

(2) Powell, D. H.; Neilson, G. W.; Enderby, J. E. *J. Phys., Condens. Matter* **1993**, *5*, 5723.

(3) Bergstrom, P. A.; Lindgren, J.; Kristiansson, O. *J. Phys. Chem.* **1991**, *95*, 8575.

(4) Arshadi, M.; Yamdagni, R.; Kebarle, P. *J. Phys. Chem.* **1970**, *74*, 1475. Yamdagni, R.; Payzant, J. D.; Kebarle, P. *Can. J. Chem.* **1973**, *51*, 2507. Keesee, R. G.; Castleman Jr.; A. W. *Chem Phys. Lett.* **1980**, *74*, 139. Hiraoka, K.; Mizuse, S. *Chem. Phys.* **1987**, *118*, 457.

(5) Markovich, G.; Ginger, R.; Levin, M.; Cheshnovsky, O. *J. Chem. Phys.* **1991**, *95*, 9416.

(6) Perera, L.; Berkowitz, M. L. *J. Chem. Phys.* **1993**, *99*, 4222.

(7) Hettich, R. L.; Jackson, T. C.; Stanko, E. M.; Freiser, B. S. *J. Am. Chem. Soc.* **1986**, *108*, 5086. Hettich, R. L.; Fresier, B. S. *J. Am. Chem. Soc.* **1986**, *18*, 2537. Benedek, G. In *Elemental and Molecular Clusters*; Springer Verlag: Berlin **1988**.

(8) Kemper, P. R.; Bushnell, J.; von Helden, G.; Bowers, M. T. *J. Phys. Chem.* **1993**, *97*, 52.

(9) Kemper, P. R.; Bushnell, J.; von Koppen, P.; Bowers, M. T. *J. Phys. Chem.* **1993**, *97*, 1810.

(10) Brucat, P. J.; Zheng, L.-S.; Pettiette, C. L.; Yang, S.; Smalley, R. E. *J. Chem. Phys.* **1986**, *84*, 3078.

(11) Cornett, D. S.; Peschke, M.; LaiHing, K.; Cheng, P. Y.; Willey, K. F.; Duncan, M.

A. *Rev. Sci. Instrum.* **1992**, *63*, 2177.

(12) SIMION is available through Idaho National Engineering laboratory.

(13) Kelly, J. In *Advances in Electronics and Electron Optics, Vol 43*; Marton, L. Ed.; **1977**.

6.5 FIGURE CAPTIONS

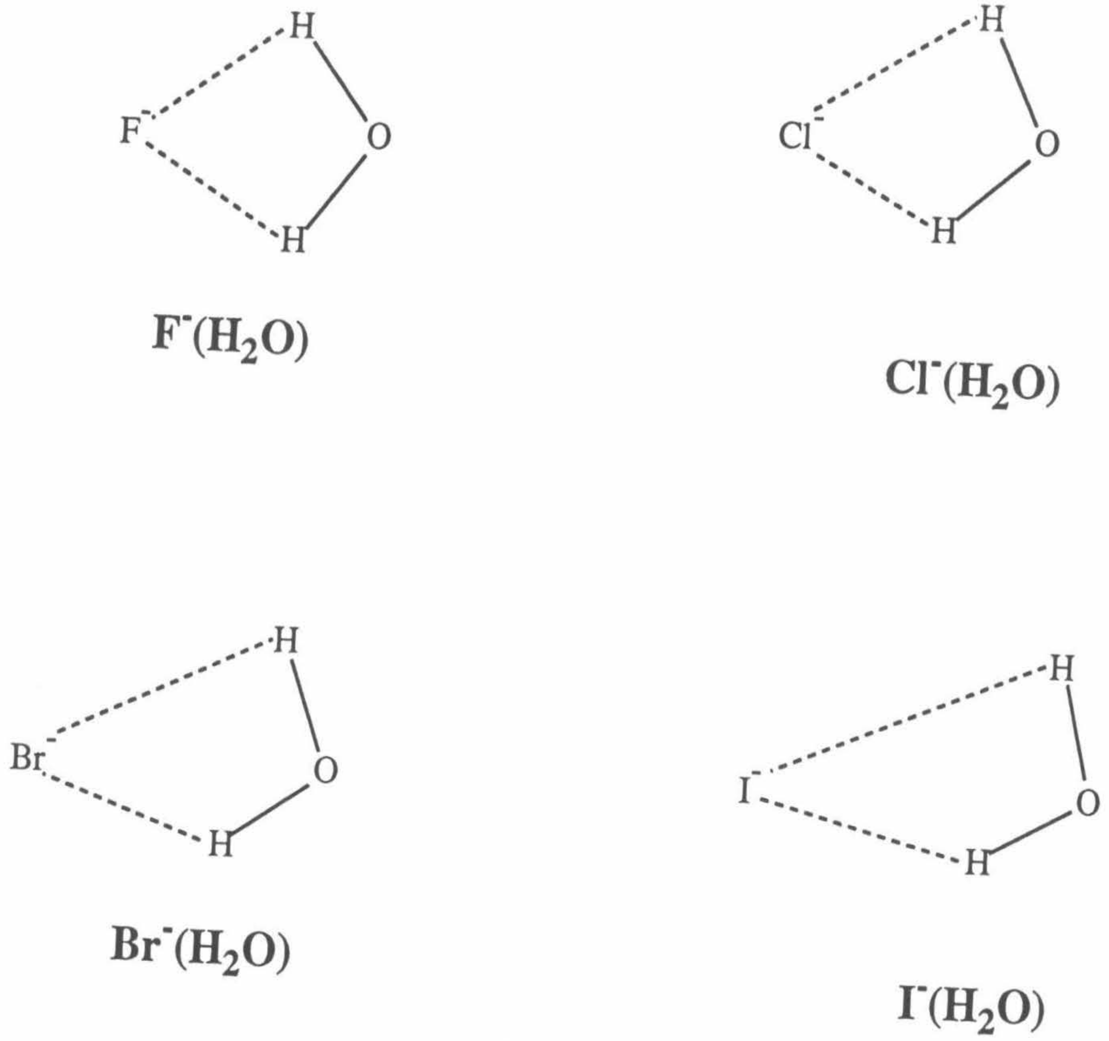
(1) Theoretical structures of $Cl^-(H_2O)$, $Br^-(H_2O)$ and $I^-(H_2O)$ clusters.

(2) Time-of-flight mass spectrum of $Cl^-(H_2O)_n$ clusters.

(3) Proposed structures by Bowers et al. of $Co^+(H_2)_n$ clusters.

(4) A schematic diagram of the laser vaporization source in our laboratory.

(5) The meta-deflector. (a) Ideal situation (b) The planar design (c) The octopole design.



$F^-(H_2O)$

$Cl^-(H_2O)$

$Br^-(H_2O)$

$I^-(H_2O)$

Figure 1.

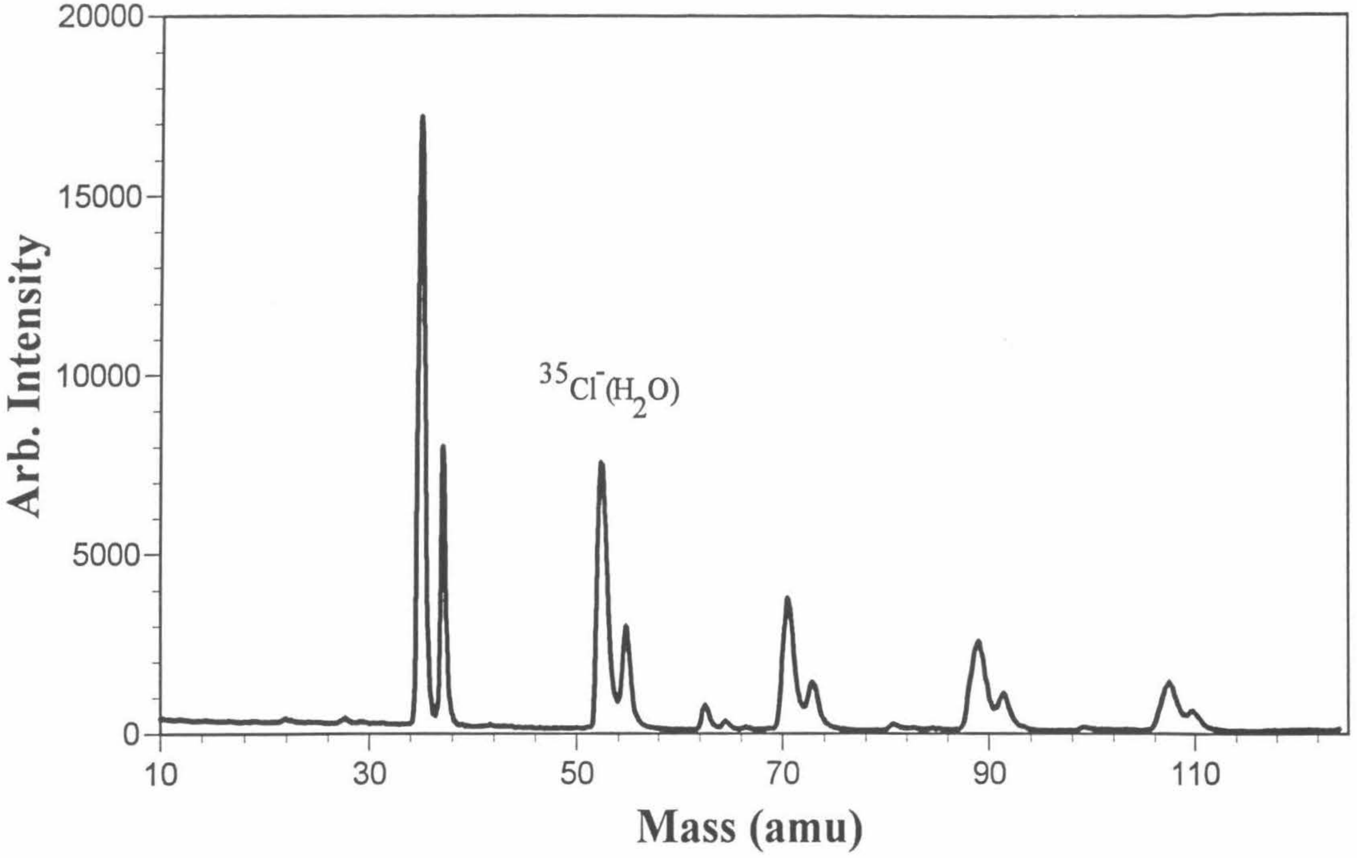

20000
15000
10000
5000
0
Arb. Intensity
$^{35}Cl^{-}(H_2O)$
10
30
50
70
90
110
Mass (amu)

Figure 2.

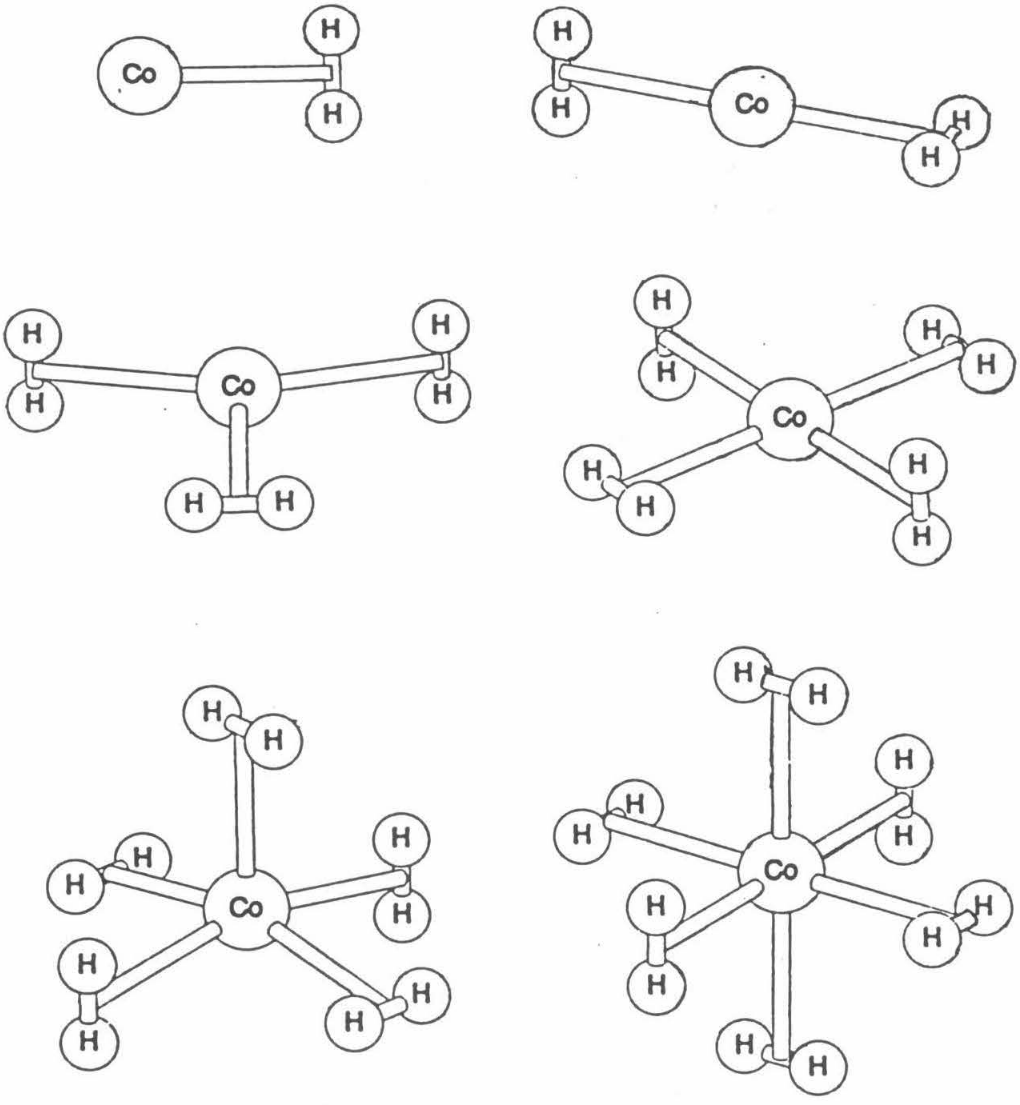

Co
H
H
H
H
Co
H
H
H
H
Co
H
H
H
H
H
H
H
H
Co
H
H
H
H
H
H
H
H
H
H
Co
H
H
H
H
H
H
H
H
H
H
H
H
Co
H
H
H
H
H
H
H
H
H
H

Figure 3.

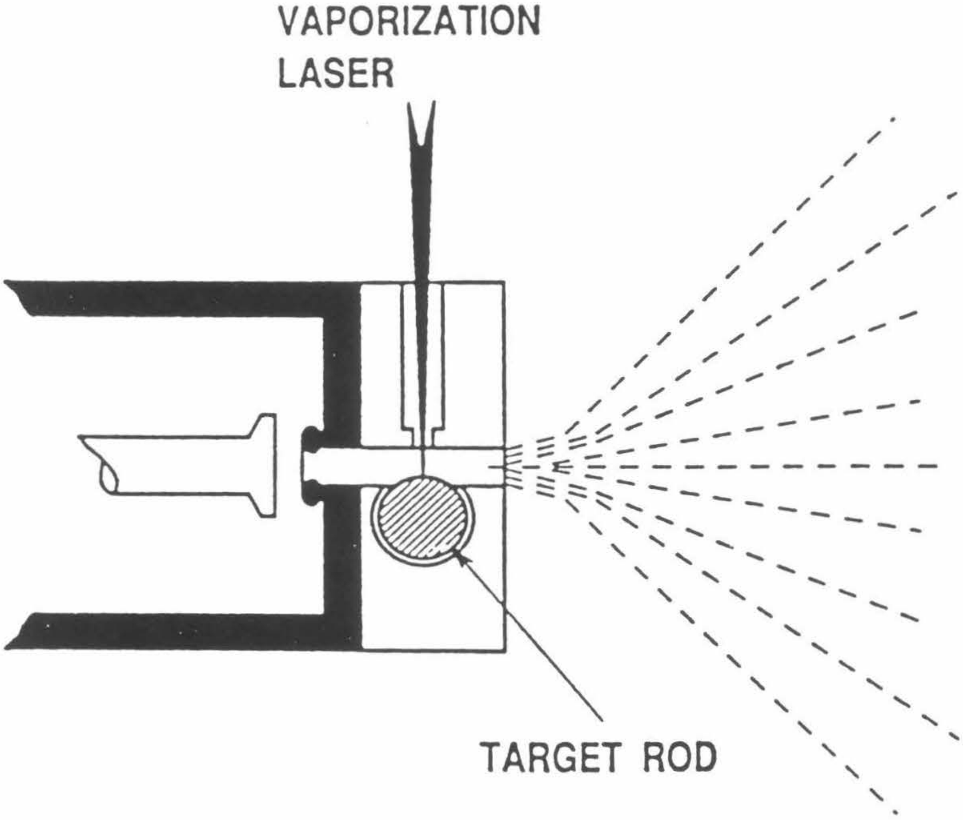
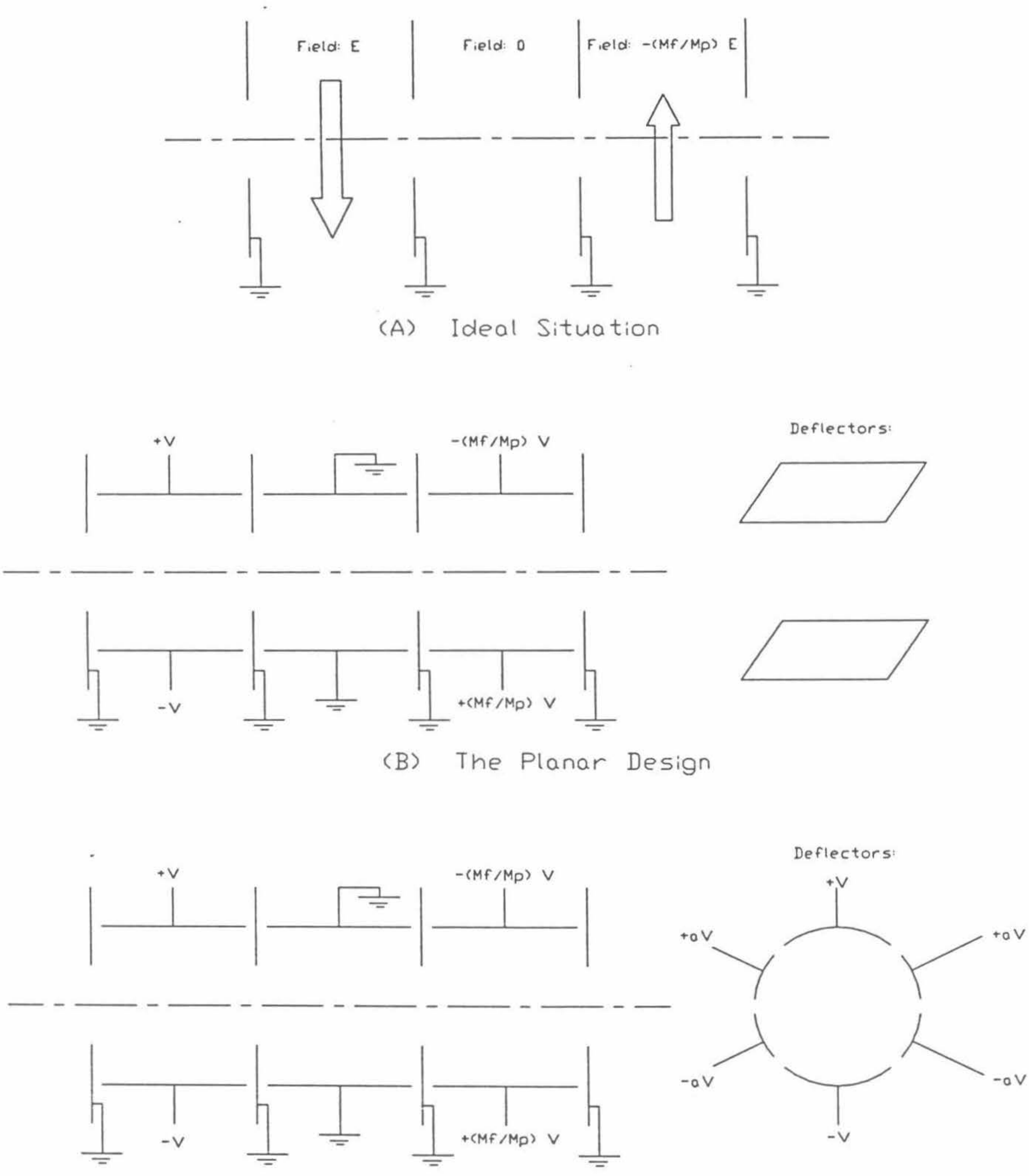


Figure 4.



(A) Ideal Situation

(B) The Planar Design

(C) The Octopole Design

Figure 5.

Appendix. Software for the $LiNbO_3$ OPO

Source Files of OPODATA are Listed as Follows:

```
Main Program:
OPODATA   PAS
Global Variables:
FILEVAR   PAS
Data Collection:
LCOLLECT  PAS
MCOLLECT  PAS
HCOLLECT  PAS
 CGLOBAL  PAS
 UTIL     PAS
 TP4D16   PAS   {from Keithley-Metrabyte, for data acquisition board}
 GDATARUT PAS
 SUBRUT   PAS
 MSTEP1   ASM   {from Keithley-Metrabyte, for stepper motor board}
 MSTEP2   ASM   {same as above}
Data Graphing, Printing:
GRAPHRUT  PAS   {unessential, omitted here}
 PSRUT    PAS   {unessential, omitted here}
```

Scanning information is updated into a file named OPOPOSI.DAT, which is listed here:

```
OPO scan info file
Ion Spec
  100
1. offset info:
 0.000000E+0000  0.000000E+0000  0.000000E+0000
2. position info:
    18267     21102      5626   2.7979
   50   50   80
3. LowResScan info:
  3610.0000  3670.0000   10.0000       300
   9.1761     0.5378 1000.0000
-1.00024182E+0004  3.03460146E+0000 -1.06521254E-0004  1.34966466E-0009
4. MedResScan info:
2943.0000 2948.0000    0.1000     30
   0.9561     0.0643   31.9480
0.000000  0.00000000E+0000  3.26800000E-0006  0.00000000E+0000
0.00000000E+0000
5. HighResScan info:
2907.5000 2908.0000    0.0100        10
  14.9872     0.1331    0.6720
0.001000
```

OPODATA.PAS:

```
{***************************************************}
{    Ion spectroscopic Data Acquisition Program    }
{    using Low, Medium and High resolution OPO      }
{***************************************************}
program OPODATA(input, output);
{$N+,O+,F+}
{$M 65220,0,20000}
uses
   Dos,Crt,Objects,Drivers,Views,Menus,App,
   FileVar,LCollect,MCollect,HCollect,GraphRut;
const
   cmLoad     = 101;
   cmSave     = 102;
   cmSaveAs   = 103;
   cmCD       = 104;
   cmCollectL = 105;
   cmCollectM = 106;
   cmCollectH = 107;
   cmAssign   = 108;
   cmAssign2  = 109;
   cmMeasure  = 110;
   cmMeasure2 = 111;
   cmPower    = 112;
   cmTOF1     = 113;
   cmTOF12    = 114;   {event.commands}
type
   PYbcApp = ^TYbcApp;
   TYbcApp =  object(TApplication)
       procedure HandleEvent(var Event :TEvent); virtual;
       procedure InitMenubar; virtual;
       procedure InitStatusLine ; virtual;
   end;
var
   SName  : String[14];
   CDName : PathStr;
   TApp   : TYbcApp;
   PApp   : PYbcApp;
{File Input/Output Procedures:}
Procedure ReadFile;
     var
         F       : Text;
         S       : String[128];
         S1,S2   : String[14];
         i,JunkI : integer;
     Begin
      Assign(F,ActiveFileName);
      {$I-}
      Reset(F);
      if IOResult <> 0 then
          begin
               Sound(800); Delay(2500); NoSound;Exit;
          end;
      {$I+}
      Readln(F,S1);
      if S1 <>’Opo Data’ then
        begin
           Close(F);
           ActiveFileName:=’None’;
           Sound(800);Delay(1500);NoSound;
        end
      else
        begin
```

```
    Readln(F,SName);
    ReadLn(F,Resolution);
    Readln(F,S2,ScanWNFirst,ScanWNLast,ScanWNStep);
    Readln(F,S);
    Readln(F,ScanRep,NDataPoints,
           FileNAssign,FileNMeasure);
    Readln(F,S);
    if NDataPoints >= 1 then
      For i:=1 to NDataPoints do
        Readln(F,JunkI,FileData[3,i],FileData[0,i],FileData[1,i],
              FileData[2,i],FileData[4,i],FileData[5,i]);
    if FileNAssign >= 1 then
       begin
         Readln(F,S);
         For i:=1 to FileNAssign do
         Readln(F,FileAssignData[i].position,
                  FileAssignData[i].wavenumber);
       end;
    if FileNMeasure >= 1 then
        begin
          Readln(F,S);
          For i:=1 to FileNMeasure do
            Readln(F,FileMeasureData[i].position,
                     FileMeasureData[i].wavenumber);
        end;
    Close(F);
  end;
End;                { proc ReadFile }
Procedure SaveFile;
 var
     F : Text;
     S : String[128];
     i : integer;
 Begin
  Assign(F,ActiveFileName);
  {$I-}
  Rewrite(F);
  if IOResult <> 0 then
      begin
           Sound(800); Delay(1500); NoSound; Exit;
      end;
  {$I+}
  Writeln(F,'Opo Data');
  Writeln(F,SName);
  Writeln(F,Resolution);
  Writeln(F,'ScanRangeStep: ',ScanWNFirst:9:4,Char(9),ScanWNLast:9:4,
             Char(9),ScanWNStep:9:4);
  Writeln(F,'ScanRep,NDataPoints,NAssign,NMeasure:');
  Writeln(F,ScanRep:7,Char(9),NDataPoints:7,Char(9),
            FileNAssign:7,Char(9),FileNMeasure:7);
  Writeln(F,'#'+Char(9)+'Wavenumber'+Char(9)+'Data'+Char(9)+'Power'
            +Char(9)+'Reference'+Char(9)+'TOF1'+Char(9)+'TOF2');
  if NDataPoints >= 1 then
    For i:=1 to NDataPoints do
   Writeln(F,i:3,Char(9),FileData[3,i]:9:4,Char(9),FileData[0,i]:9:4,
                 Char(9),FileData[1,i]:9:4,Char(9),FileData[2,i]:9:4,
                Char(9),FileData[4,i]:9:4,Char(9),FileData[5,i]:9:4);
  if FileNAssign >= 1 then
     begin
       Writeln(F,'Assigned Lines:');
       For i:=1 to FileNAssign do
          Writeln(F,FileAssignData[i].position:7,
                    Char(9),FileAssignData[i].wavenumber:9:4);
        end;
     if FileNMeasure >= 1 then
         begin
            Writeln(F,'Measured Lines:');
            For i:=1 to FileNMeasure do
              Writeln(F,FileMeasureData[i].position:7,
                        Char(9),FileMeasureData[i].wavenumber:9:4);
         end;
     Close(F);
   End;            { proc SaveFile }
Procedure ShowFiles ( SFB : integer );
   var
       F            : Text;
       i,j,IoNum    : integer;
       haha         : Char;
       TempFileName,TempCD : string;
       TempIn,CheckIn: PathStr;
       DirInfo      : SearchRec;
   begin
       ClrScr;
       Writeln('The Current Directory is: ' +
                  CDName + '.  Files *.OPO are: ');
       i := 0 ;
       FindFirst('*.OPO',Archive,DirInfo);
       While DosError = 0 do
          begin
              TempFileName  := DirInfo.Name;
              j := Length(TempFileName);
              Delete(TempFileName, j-3 ,4 );
              GoToXY(1 + 10 * ( i mod 7 ) , 3 + (i div 7 ) );
              Write(TempFileName);
              i := i + 1;
              if ( i mod 140 = 0 ) and  ( i > 0 ) then
                    repeat until KeyPressed;
              FindNext(DirInfo);
          end;
       if SFB = 1 then
          begin
             Repeat
                  GoToXY(1,23);
                  Write('                                        ');
                  GoToXY(1,23);
                  Write('Read file. Enter filename["Q" to quit]: ');
                  Readln(TempIn);
                  if  (TempIn ='Q') or ( TempIn ='q' ) then exit;
                  TempIn  := TempIn +'.OPO';
                  CheckIn := FSearch(TempIn,CDName);
                  if CheckIn ='' then
                     begin
                        GoToXY(1,24);
                        Write('                                       ');
                        GoToXY(1,24);
                        Write('File does not exist.Try again?<Y/N>:');
                        Repeat haha:=Readkey until haha in
                                   ['Y','y','N','n'];
                        Write(haha);
                     end;
             until ( haha in ['N','n'] ) or ( CheckIn <>'');
             if CheckIn <>'' then
                    begin
                          ActiveFileName[0] := TempIn[0];
                      For i := 1 to Length(TempIn) do
                          ActiveFileName[i] := UpCase(TempIn[i]);
                    ReadFile;
              end;
    end
 else if SFB = 2 then
       Repeat
            GoToXY(1,23);
            Write(’                                        ’);
            GoToXy(1,23);
            Write(’Save file. Enter filename["Q" to quit]: ’);
            Readln(TempIn);
            if  (TempIn =’Q’) or ( TempIn =’q’ ) then exit;
            TempIn  := TempIn +’.OPO’;
            CheckIn := FSearch(TempIn,CDName);
            if CheckIn  <>’’ then
               begin
                  GoToXY(1,24);
                  Write(’                                            ’);
                  GoToXY(1,24);
                  Write(’File already exists.Overwrite?<Y/N>: ’);
                  Repeat haha := Readkey until haha in
                    [’Y’,’y’,’N’,’n’];
                  Write(haha);
              end;
            if (CheckIn = ’’) or ( haha in [’Y’,’y’] ) then
               begin
                   Assign(F,TempIn);
                  {$I-}
                   Rewrite(F); IoNum := IoResult;
                  {$I+}
                   if IoNum = 0 then
                      begin
                        ActiveFileName := TempIn;
                        SaveFile;
                        GotoXY(1,25);
                        Write(’File Saved!’);
                        Delay(1000);
                      end;
               end;
       until IoNum = 0
else
       Repeat
           GoToXY(1,23);
           Write(’                                                ’);
           GoToXy(1,23);
           Write(’Enter directory name["Q" to quit]: ’);
           Readln(TempCD);
           if  (TempCD =’Q’) or ( TempCD =’q’ ) then exit;
           {$I-} ChDir(TempCD); IoNum := IoResult; {$I+}
           if IoNum <> 0 then
              begin
                  GoToXY(1,24);
                  Write(’                                            ’);
                  GoToXY(1,24);
                  Write(’No such directory.Try again? <Y/N>: ’);
                  Repeat haha := Readkey until  haha in
                    [’Y’,’y’,’N’,’n’];
                  Write(haha);
             end;
           GetDir( 0, TempCD);
           if TempCD <> CDName then
                  begin
                       CDName := TempCD;
                       ActiveFileName := ’None’;
```

```
                        end;
             Until ( haha in ['N','n'] ) or ( IoNum = 0 );
   end;         { ShowFiles }
{*********** TYbcApp: ******************}
Procedure TYbcApp.HandleEvent( var Event : TEvent);
  begin
    TApplication.HandleEvent(Event);
    if Event.What = evCommand  then
       begin
               Case  Event.Command of
                      cmLoad   :  begin
                                      TYbcApp.Done;
                                      ShowFiles(1);
                                      TYbcApp.Init;
                                  end;
                      cmSave   :  begin
                                   IF ActiveFileName <> 'None' then
                                     begin
                                        SaveFile;
                                        TYbcApp.Done;
                                        clrscr;GotoXY(35,15);
                                        Writeln('File Saved!');
                                        Delay(2000);
                                        TYbcApp.Init;
                                     end
                                   ELSE
                                     begin
                                        Sound(800);Delay(2500);NoSound;
                                     end;
                                  end;
                      cmSaveAs :  begin
                                   IF ActiveFileName <> 'None' then
                                     begin
                                        TYbcApp.Done;
                                        ShowFiles(2);
                                        TYbcApp.Init;
                                     end
                                   ELSE
                                     begin
                                        Sound(800);Delay(2500);NoSound;
                                     end;
                                  end;
                      cmCD       : begin
                                       TYbcApp.Done;
                                       ShowFiles(3);
                                       TYbcApp.Init;
                                  end;
                      cmCollectL: begin
                                          TYbcApp.Done;
                                          ActiveFileName :='None';
                                          LRuns;
                                          TYbcApp.Init;
                                  end  ;
                      cmCollectM: begin
                                          TYbcApp.Done;
                                          ActiveFileName :='None';
                                          MRuns;
                                          TYbcApp.Init;
                                  end  ;
                      cmCollectH: begin
                                          TYbcApp.Done;
                                          ActiveFileName :='None';
                                          HRuns;
```

```
                    TYbcApp.Init;
            end;
cmAssign :  begin
             IF ActiveFilename <> 'None' then
                begin
                    TYbcApp.Done;
                    DDraw(1);
                    TYbcApp.Init;
                end
             ELSE
                begin
                   Sound(800);Delay(2500);NoSound;
                end;
            end;
cmAssign2:  begin
             IF ActiveFilename <> 'None' then
                begin
                    TYbcApp.Done;
                    DDraw(2);
                    TYbcApp.Init;
                end
             ELSE
                begin
                   Sound(800);Delay(2500);NoSound;
                end;
            end;
cmMeasure:  begin
             IF ActiveFilename <> 'None' then
                begin
                    TYbcApp.Done;
                    DDraw(3);
                    TYbcApp.Init;
                end
             ELSE
                begin
                   Sound(800);Delay(2500);NoSound;
                end;
            end;
cmMeasure2: begin
             IF ActiveFilename <> 'None' then
                begin
                    TYbcApp.Done;
                    DDraw(4);
                    TYbcApp.Init;
                end
             ELSE
                begin
                   Sound(800);Delay(2500);NoSound;
                end;
            end;
cmPower   : begin
             IF ActiveFilename <> 'None' then
                begin
                    TYbcApp.Done;
                    DDraw(5);
                    TYbcApp.Init;
                end
             ELSE
                begin
                   Sound(800);Delay(2500);NoSound;
                end;
            end;
cmTOF1   : begin
```

```
                                   IF ActiveFilename <> 'None' then
                                      begin
                                          TYbcApp.Done;
                                          DDraw(6);
                                          TYbcApp.Init;
                                      end
                                   ELSE
                                      begin
                                         Sound(800);Delay(2500);NoSound;
                                      end;
                                  end;
                      cmTOF12  : begin
                                   IF ActiveFilename <> 'None' then
                                      begin
                                          TYbcApp.Done;
                                          DDraw(7);
                                          TYbcApp.Init;
                                      end
                                   ELSE
                                      begin
                                         Sound(800);Delay(2500);NoSound;
                                      end;
                                  end;
               end;
         ClearEvent(Event);
       end;
  end;                    {TYbcApp.HandleEvent}

Procedure  TYbcApp.InitMenuBar;
   var
     R: TRect;
   begin
      GetExtent(R);
      R.B.Y :=R.A.Y + 1;
      MenuBar := New(PMenuBar, Init(R, NewMenu(
                  NewSubMenu('~F~ile ', hcNoContext, NewMenu(
                   NewItem('~L~oad File ' ,'F1', KBF1, cmLoad,
                        hcNoContext,
                   NewItem('~S~ave       ' ,'F2', KBF2, cmSave,
                        hcNoContext,
                   NewItem('Save ~A~s    ' ,'F3', KBF3, cmSaveAs,
                        hcNoContext,
                   NewItem('~C~hange Dir' ,'F4', KBF4, cmCD,
                        hcNoContext,
                   nil))))),
                  NewSubMenu('~C~ollectData ', hcNoContext, NewMenu(
                   NewItem('~L~owResScan ',
                              'Alt-L',KBAltL,cmCollectL,hcNoContext,
                   NewItem('~M~edResScan ',
                              'Alt-M',KBAltM,cmCollectM,hcNoContext,
                   NewItem('~H~ighResScan',
                              'Alt-H',KBAltH,cmCollectH,hcNoContext,
                   nil)))),
                  NewSubMenu('~G~raphMode ',hcNocontext, NewMenu(
                   NewItem('Reference','F5',KbF5,cmAssign,hcNoContext,
                   NewItem('Normalized
                              Ref','F6',KbF6,cmAssign2,hcNoContext,
                   NewItem('Data','F7',KbF7,cmMeasure,hcNoContext,
                   NewItem('Normalized
                              Data','F8',KbF8,cmMeasure2,hcNoContext,
                   NewItem('Power','F9',kbF9,cmPower,hcNoContext,
                   NewItem('TOF1','F10',kbF10,cmTOF1,hcNoContext,
                   NewItem('Normalized
                              TOF1','Alt-T',kbAltT,cmTOF12,hcNoContext,
                   nil)))))))),
                  nil)))))));
      end;                          {TYbcApp.InitMenuBar}
Procedure   TYbcApp.InitStatusLine;
   var
     R: Trect;
   Begin
     GetExtent(R);
     R.A.Y:= R.B.Y -1;
     StatusLine := New(PStatusLine, Init(R,
                     NewstatusDef(0, $FFFF,
                       NewStatusKey('', KbF10, cmMenu,
                       NewStatusKey('~Alt-X~ Exit ', KbAltX, cmQuit,
                       NewStatusKey('~F4~ ChDir. CuDir is '+ CDName +
                               ' CuFile is '+ ActiveFileName, kbF4,cmCD,
                       nil))),
                     nil)));
     end;                          {TYbcApp.InitStatusLine}
{******************** Main Program **************************}
begin
    ActiveFileName :='None';
    GetDir(0,CDName);
    TApp.Init;
    TApp.Run;
    TApp.Done;
end.                       { Program  OPODATA }
```

FILEVAR.PAS:

```
{******************}
{ Golbal variables }
{******************}
Unit FileVar;
{F+}
INTERFACE
 Uses
   Dos;
 Const
 F1=#59;F2=#60;F3=#61;F4=#62;F5=#63;F6=#64;F7=#65;F8=#66;F9=#67;F10=#68;
 left=#75;right=#77;up=#72;down=#80;Alt_F=#33;Alt_B=#48;
 home=#71;pg_up=#73;ending=#79;pg_dn=#81;Ins=#82;Del=#83;
 Alt_M=#50;Alt_J=#36;Alt_C=#46;Alt_P=#25;Alt_Q=#16;Alt_W=#17;Alt_S=#31;
                                                       {IBM key codes}
   NMax      = 2;
   AMMax     = 100;
   NPointMax = 500;
                  { maximum channels/assignable peaks/data points }
   ScreenX = 639;  ScreenY = 479;   {VGA}
 Type
   OPODataType    = Array[0..5,1..NPointMax] of Real;
     {0:Data;1:Power;2:Ref;3:Wavenumber/steps;4,5: TOFVIEW para }
   AMDataType1    = Record
                      Position   : integer;
                      Wavenumber : Real;
                    end;
   AMDataType     = Array[1..AMMax] of AMDataType1;
 var
   ActiveFileName : String[12];
   FileData       : OPODataType;
   FileNAssign    : integer;
   FileNMeasure   : integer;
   FileAssignData : AMDataType;
   FileMeasureData: AMDataType;
   Resolution      : String[7];
   ScanWNFirst,ScanWNLast,ScanWNStep : Real;
   ScanRep    : integer;
   NDataPoints: integer;
IMPLEMENTATION
end.
```

LCOLLECT.PAS:

```
{********************************}
{ The Low Resolution OPO Data    }
{  Collecting Workhorse          }
{********************************}
{*******************SCANNING METHOD DESCRIPTION********************}
{1.     Wavenumber is determined by grating position NGrating        }
{          through a spline function WNNG (reverse function is NGWN) }
{2.     For every NGrating, there is a crystal position that would   }
{          cause lasing, this position is determined by a global     }
{          cubic function NCNG, plus an offset (to account for       }
{          the day-to-day difference in the pumping laser angle)     }
{3.     When scanning, OPO goes to the particular NGrating which     }
{          corresponds to scanning wavenumber, then moves crystal    }
{          position to NCrystal :=NCNG+offset.  OPO should lase now. }
{          To acheive best result, NCrystal tuning is allowed during }
{          OPO scan                                                  }
{*****************************************************************}
Unit LCollect;
{$B-,N+,V-,R-,D+,F+}
INTERFACE
 Uses
  Dos,Crt,Graph,FileVar,CGlobal,Util,SubRut,GDataRut;
 Procedure LRuns;
IMPLEMENTATION
  Procedure RefreshScreen;
    const
     head1A='< >:    ^v:     F1:      F2:      HOME/END:       PGUP/DN:
AltC:    AltM:     ';
     head1B='    jogC   jogG   moveC   moveG          jog20C
jog20G     CalWN      MvWN';
     head2A='F3      F4    AltS    AltQ        AltW      F5     F6          F7
 F8    F9    F10  ';
     head2B='  AdjPa  Mode    Scan    QuitScn    WData  Z/UZ  AutoZoom
TOF  Init  TChk   Ex';
    Begin
     SetViewPort(0,0,GetMaxX,GetMaxY,ClipOn);
     SetColor(red);
     Rectangle(0,0,GetMaxX,150);
     Line(0,15,GetMaxX,15);
     Line(0,40,GetMaxX,40);
     Line(0,55,GetMaxX,55);
     Line(0,90,GetMaxX,90);
     Line(0,100,GetMaxX,100);
     Line(0,150,GetMaxX,150);                    {draw the box edges}

     SetColor(Cyan);
     SetTextJustify( CenterText,CenterText);
     OutTextXY(GetMaxX div 2,8,' COMMAND MENU ');
     OutTextXY(GetMaxX div 2,48,' CURRENT STATUS ');
     OutTextXY(GetMaxX div 2,96, ' DIALOG ');
     SetColor(LightGreen);
     if AcouOnly then
         OutTextXY(100,48,'Acoustic')
     else
         OutTextXY(100,48,'Ion Spec');
  if DataSaved then
    OutTextXY(480,48,'Data Saved')
  else
    OutTextXY(480,48,'Data Unsaved');
                                         {write the box names}
  SetColor(White);
  OutTextXY(GetMaxX div 2,22,head1A);
  OutTextXY(GetMaxX div 2,34,head2A);
  SetColor(LightGreen);
  OutTextXY(GetMaxX div 2 -250,62,'NCrystal=');
  OutTextXY(GetMaxX div 2 +200,62,'NGrating=');
  OutTextXY(GetMaxX div 2 -100,62,'CSpeed=');
  OutTextXY(GetMaxX div 2 + 50,62,'GSpeed=');
  OutTextXY(GetMaxX div 2,22,head1B);
  OutTextXY(GetMaxX div 2,34,head2B);

  OutTextXY(320,72,'CurrentWaveNumber=                        cm-1');
  OutTextXY(320,82,'From             To             cm-1;Step=
               cm-1;Shots/Point=           ');
  OutTextXY(50,72,'NotScanning');
                     {draw the text contents}
  ClearDrawings;
 end;                        { proc RefreshScreen }
Procedure TrackPara;
 Var
  TMString1, TMString2 : String[3];  { motor speeds }
  TSString1,TSString2  : String[9];  { ScanFirst , ScanLast }
  TSString3     : String[7];         { Scan StepSize }
  TSString4     : String[6];         { ShotsPerPoint }
  Begin
   Str(NcSpeed,TMString1); Str(NgSpeed,TMString2);
   SetViewPort(GetMaxX div 2 -70,57,GetMaxX div 2-25,67,ClipOn);
   ClearViewPort;
   SetViewPort(GetMaxX div 2 +80,57,GetMaxX div 2+125,67,ClipOn);
   ClearViewPort;
   SetViewPort(10,57,GetMaxX,67, ClipOn);
   OutTextXY(getMaxX div 2 -50,5,TMString1);
   OutTextXY(GetMaxX div 2 +100,5,TMString2);
                                   { Write out motor speeds }
   Str(ScanWNFirst:9:3,TSString1);
   Str(ScanWNLast:9:3,TSString2);
   Str(ScanWNStep:7:3,TSString3);
   Str(ScanRep:6,TSString4);
   SetViewPort(40,77,130,87,ClipOn); ClearViewPort;
   SetViewPort(160,77,250,87,ClipOn);ClearViewPort;
   SetViewPort(340,77,410,87,ClipOn);ClearViewPort;
   SetViewPort(570,77,620,87,ClipOn);ClearViewPort;
   SetViewPort(0,77,639,87,ClipOn);
   OutTextXY( 85,5,TSString1);
   OutTextXY(205,5,TSString2);
   OutTextXY(375,5,TSString3);
   OutTextXY(595,5,TSString4);
                              { write out other paras }
  end;                                    { proc TrackPara }
Procedure AdjustPara ;
  const
   AD1 ='Adjust Parameters: motor speeds; scanning parameters.';
   ADM1 ='Crystal Motor: (slowest: 255; fastest: 20) new speed:
              ';
   ADM2 ='Grating Motor: (slowest: 255; fastest: 20) new speed:
              ';
                      { Motor's parameters }
   ADS1    ='Enter SCANWNFIRST:                                     ';
 ADS2    ='Enter SCANWNLAST:                                    ';
 ADS3    ='Enter SCANWNSTEP:                                    ';
 ADS4    ='Enter SHOTS/PIONT: (  50000 )                        ';
                    { Laser scan parameters }
 ADS5    =' Enter new value: ';
 ADS6    ='DOffset is: ';
 ADS7    ='POffset is: ';
 ADS8    ='ROffset is: ';
 ADS9    ='TrigDelay in us: ';
var
 Vertposi                : integer;
 Tempspeed,Temp4,JunkI   : longint;
 JunkR,Temp1,Temp2,Temp3 : Real;
 AdStr                   :  String[11];
Begin
 SetViewPort(10,103,629,147,ClipOn);
 VertPosi := VertStart;
 OutTextXY(GetMaxX div 2,VertPosi,AD1);
 VertPosi := VertPosi + VertStep;
 OutTextXY(GetMaxX div 2,VertPosi,ADM1);
 MoveTo(550, VertPosi);
 GetNum(Tempspeed,JunkR,True);
 If (TempSpeed>=MaxMotorSpeed) and (TempSpeed<=MinMotorSpeed)  then
    NcSpeed :=Tempspeed ;
 VertPosi := VertPosi + VertStep;
 OutTextXY(GetMaxX div 2,VertPosi,ADM2);
 MoveTo(550, VertPosi);
 GetNum (TempSpeed,JunkR,True);
 If (TempSpeed>=MaxMotorSpeed) and (TempSpeed<=MinMotorSpeed)  then
    NgSpeed := TempSpeed;
 Delay(500);
 ClearViewPort;                 { We have got the motor speeds ! }
 TrackPara;

 SetViewPort(10,103,629,147,ClipOn);
 VertPosi:= VertStart - 4 ;
 OutTextXY(GetMaxX div 2,VertPosi,ADS1);
 MoveTo(520, VertPosi);
 GetNum(JunkI,Temp1,False);
 If ( Temp1 >=2000.0 ) and ( Temp1 <= 4700.0)  then
    ScanWNFirst:=Temp1;
 VertPosi:= VertPosi + 9;
 OutTextXY(GetMaxX div 2,VertPosi,ADS2);
 MoveTo(520, VertPosi);
 GetNum(JunkI,Temp2,False);
 If ( Temp2 >=2000.0 ) and ( Temp2 <= 4700.0)  then
   ScanWNLast:=Temp2;

 VertPosi:= VertPosi + 9;
 OutTextXY(GetMaxX div 2,VertPosi,ADS3);
 MoveTo(520, VertPosi);
 GetNum(JunkI,Temp3,False);
 If (Abs(Temp3) >= 0.001 ) and ( Abs(Temp3) <= 1000.0 )
   and ( (Temp3 + 1.01) > 0.000001 ) then
    ScanWNStep:=Temp3;

 NDataPoints:=Trunc(Abs((ScanWNLast- ScanWNFirst)/ ScanWNStep
                          +1.001));
 if NDataPoints > NPointMax  then
    begin
      Sound(1000);Delay(3000);NoSound;
      NDataPoints := NPointMax;
      ScanWNStep  := Abs(ScanWNLast-ScanWNFirst) / (NPointMax - 1);
```

```
         end;

      VertPosi:= VertPosi + 9;
      OutTextXY(GetMaxX div 2,VertPosi,ADS4);
      MoveTo(520, VertPosi);
      GetNum(Temp4,JunkR,True);
      If ( Temp4 > 0 ) and ( Temp4 <= 50000 )  then
         ScanRep :=Temp4;
      Delay(800);
      ClearViewPort;                             { We have got scan paras }
      TrackPara;

      SetViewPort(10,103,629,147,ClipOn);
      VertPosi := VertStart;
      Str(OOffset[0]:11:5,ADStr);
      OutTextXY(320,Vertposi,ADS6+ADStr+ADS5);
      MoveTo(550,Vertposi);
      GetNum(JunkI,Temp1,False);
      if Abs( Temp1 + 1.01 ) >  0.000001  then
         OOffset[0] := Temp1;

      VertPosi :=VertPosi + VertStep;
      Str(OOffset[1]:11:5,ADStr);
      OutTextXY(320,Vertposi,ADS7+ADStr+ADS5);
      MoveTo(550,Vertposi);
      GetNum(JunkI,Temp1,False);
      if Abs( Temp1 + 1.01 ) >  0.000001  then
         OOffset[1] := Temp1;

      VertPosi :=VertPosi +VertStep;
      Str(OOffset[2]:11:5,ADStr);
      OutTextXY(320,Vertposi,ADS8+ADStr+ADS5);
      MoveTo(550,Vertposi);
      GetNum(JunkI,Temp1,False);
      if Abs( Temp1 + 1.01 ) >  0.000001  then
         OOffset[2] := Temp1;

      ClearViewPort;
      Str(TDelay:11:2,ADstr);
      OutTextXY(320,Vertstart,ADS9+ADStr+AdS5);
      MoveTo(550,VertStart);
      GetNum(JunkI,temp1,False);
      if ( Temp1 >= 20 )  and ( Temp1 <= 33300 ) then
         TDelay := Temp1;
      ClearViewPort;
    end;                                       { proc adjustpara }
(********************************************************)
Procedure LRuns;
 var
  Key                        : Char;
  CollectFlag                : Boolean;
  GraphDetect,GraphMode,
  i,error                    : Integer;
  Computerfile               : Text;
  JunkStr,str1,str2,str3,str4,str5 ,str6   : String[90];
  JNEtalon : Longint;
  JNeSpeed : integer;
  JPZTVolts : Real;
Begin  { Procedure LRuns }
   Resolution := 'Low';
   Clrscr;
   GraphDetect:=Detect;
   Graph.DetectGraph(GraphDetect, GraphMode);
```

```
InitGraph(GraphDetect,GraphMode,'C:\ybc\aqdata');
              { NOTE : I have a copy of EGAVGA.BGI in C:\ybc\aqdata }
SetBkColor(DarkGray); SetColor(LightGreen);

Assign(Computerfile, 'C:\YBC\AQDATA\OPOPOSI.DAT');
Reset(computerfile);
Readln(computerfile, Junkstr);
Readln(computerfile, Junkstr);
if junkstr ='Acoustic' then
       AcouOnly := True
else
       AcouOnly := False;
DataSaved := False;
Readln(computerfile, TDelay);
Readln(computerfile, JunkStr);
Readln(computerfile, OOffset[0],OOffset[1],OOffset[2]);
Readln(Computerfile, JunkStr);
Readln(computerfile, NCrystal,NGrating,JNEtalon,JPZTVolts);
Readln(computerfile, NcSpeed, NgSpeed,JNeSpeed);
Readln(computerfile, JunkStr);
Readln(computerfile, ScanWNFirst,ScanWNLast,ScanWNStep,ScanRep);
Readln(Computerfile, WindowRatio[1],WindowRatio[2],WindowRatio[3]);
Readln(Computerfile,FitSYNData[1,1],FitSYNData[1,2],
                    FitSYNData[1,3],FitSYNData[1,4]);
Readln(computerfile, JunkStr);
Readln(Computerfile, str1);
Readln(Computerfile, Str2);
Readln(Computerfile, Str3);
Readln(computerfile,JunkStr);
Readln(computerfile,str4);
Readln(computerfile,str5);
Readln(computerfile,str6);
Close(computerfile);
              { last time positions loaded in from file OPOPOSI.DAT }
NDataPoints    := Round((ScanWNLast- ScanWNFirst)/ ScanWNStep + 1.0) ;
NPresentPoint:= 0;
Scanning      := False;
For i := 0 to Nmax do
   WindowOffset[i+1] := 0.0;
                  { initial setting of parameters }
Spline;
RefreshScreen;
KeepTrackL(CollectFlag);
TrackPara;

InitStuff(CollectFlag);

Repeat
  Repeat until Readkey = #0;
  key := Readkey;
                    { until first key is "Escape" }
  if key in [left,right,up,down, home,pg_up,ending,pg_dn,F1,F2,
                              {Motor moving}
           F3,                {Para setting}
           F4,                {Acoustic Only}
           F5,                {Zoom/unzoom}
           F6,                {AutoZoom}

           F7,                {TOF Control}
           F8,                {Initialize the motors/Reset pos}
           F9,                {Timing Check on Acoustic Signal}
           Alt_C,             {Cal NC,NG }
           Alt_M,             {Move to a new Wavenumber position}
```

```
      Alt_S,                  {StartScan}
      Alt_Q,                  {QuitScan}
      Alt_W,                  {Write data to file}
      F10 ]                   {Exit}
then
 begin
  case key of
   left,right,home,ending:
         if ((not Scanning) or (ScanRep-NPresentRep > 10 )) and
            ( NCrystal > CGMin ) and ( NCrystal < CGMax ) then
             begin
                if key = left then
                    JogL(0,-1,CollectFlag)
                else if key = right  then
                    JogL(0,1,CollectFlag)
                else if key = home then
                    Jog20L(0,20,CollectFlag)
                else
                    Jog20L(0,-20,CollectFlag);
                KeepTrackL(CollectFlag);
                Delay(400);
             end;
   up,down,pg_up,pg_dn:
         if ( not Scanning ) and
            ( NGrating > CGMin ) and ( NCrystal < CGMax ) then
             begin
                if key = up then
                    JogL(1,1,CollectFlag)
                else if key = down then
                    JogL(1,-1,CollectFlag)
                else if key = pg_up then
                    Jog20L(1,20,CollectFlag)
                else
                    Jog20L(1,-20,collectFlag);
                KeepTrackL(CollectFlag);
                Delay(400);
             end;
   F1,F2:    begin
                if not scanning then
                  begin
                     if key = F1 then
                        MoveLMotor(0,CollectFlag)
                     else
                        MoveLMotor(1,CollectFlag);
                     KeepTrackL(CollectFlag);
                  end;
             end;
   F3 :      begin
                  if not scanning  then
                     AdjustPara;
             end;
   F4:       Begin
                  if not Scanning then
                      AcouYesNo;
             End;
   F5:       if (not Scanning) or (ScanRep-NPresentRep>30) then
             Zoom123;
   F6:       if (not Scanning) or (ScanRep-NPresentRep>30) then
             AutoZoom;
   F7:       Begin
                if not Scanning then
                   ToTOF;
             End;
```

```
        F8:         begin
                       if not scanning then
                          begin
                             InitStuff(CollectFlag);
                             GetStartPosition;
                             KeepTrackL(CollectFlag);
                          end;
                    end;
        F9:         Begin
                       if not Scanning then
                         begin
                            TChkDial(i,CollectFlag);
                            if CollectFlag then
                           GetTCheck(0,i,0,ScanRep,NDataPoints,error);
                         end;
                    End;
        Alt_C:      begin
                       if not Scanning then
                                   CalCG;
                    end;
        Alt_M:      begin
                       if not scanning then
                           begin
                              MoveToWN(CollectFlag);
                              KeepTrackL(CollectFlag);
                           end;
                    end;
        Alt_S:      begin
                      if not scanning then
                          begin
                             ScanDial(CollectFlag);
                             if CollectFlag then
                            GetData(0,0,0,2,ScanRep,NDataPoints,error);
                          end;
                    end;
        Alt_Q:      begin
                       if scanning then
                            begin
                                StopScan;
                                KeepTrackL(CollectFlag);
                            end;
                    end;
        Alt_W:      begin
                       if not scanning then
                           DataSave;
                    end;
        F10:        begin
                       if not scanning then
                           begin
                              QuitCollect(key);
                              if key in ['N','n'] then key:='N'
                              else key :=F10;
                           end;
                    end;
        end;  { Case }
   end;  {if}
until key= F10;  {exit}

Rewrite(computerfile);
Writeln(computerfile,'OPO scan info file');
if AcouOnly then
   Writeln(computerfile,'Acoustic')
else
```

```
      Writeln(computerfile,'Ion Spec');
   Writeln(computerfile,TDelay:5:0);
   Writeln(computerfile,'1. offset info:');
   Writeln(computerfile,OOffset[0]:15,' ',
                    OOffset[1]:15,' ',OOffset[2]:15);
   Writeln(computerfile,'2. position info:');
   Writeln(computerfile,NCrystal:9,NGrating:9,JNEtalon:9,JPZTVolts:9:4);
   Writeln(computerfile,NcSpeed:5,NgSpeed:5,JNeSpeed:5);
   Writeln(computerfile,'3. LowResScan info:');
   Writeln(computerfile,ScanWNFirst:11:4,ScanWNLast:11:4,
                        ScanWNStep:10:4,ScanRep:9);
   Writeln(computerfile,WindowRatio[1]:9:4,' ',WindowRatio[2]:9:4,' ',
                        WindowRatio[3]:9:4);
   Writeln(Computerfile,FitSYNData[1,1]:17,' ',FitSYNData[1,2]:17,' ',
                        FitSYNData[1,3]:17,' ',FitSYNData[1,4]:17);
   Writeln(Computerfile,'4. MedResScan info:');
   Writeln(Computerfile,str1);
   Writeln(Computerfile,str2);
   Writeln(Computerfile,str3);
   Writeln(computerfile,'5. HighResScan info:');
   Writeln(Computerfile,str4);
   Writeln(Computerfile,str5);
   Writeln(computerfile,Str6);
   Close(computerfile);
            { Save back to C:\YBC\AQDATA\OPOPOSI.DAT }
   CloseGraph;
 end;          {Procedure  LRuns}
end.                        { unit lcollect }
```

MCOLLECT.PAS:

```
{********************************}
{ The Medium Resolution OPO Data }
{  Collecting Workhorse          }
{********************************}
{*******************SCANNING METHOD DESCRIPTION********************}
{                                                                  }
{1.     For rough determination of the lasing wavenumber:          }
{          it is determined by grating position NGrating through   }
{          through a spline function WNNG (reverse function is NGWN),}
{          and the correct crystal position is determined by NCNG  }
{2.     The exact lasing wavenumber is determined by etalon position }
{          with the following equation:                            }
{            wavenumber = PumpWN - N / ( 2.0 * n * d * Cos(theta) )  }
{                in our case,  d = 0.2 cm                          }
{                              n = 1.442                           }
{  synthetic fused silica: n = 1.4458@1.4um, 1.4395@1.9um          }
{                and  theta  = ArcSin(1/n * Sin(thetax))           }
{                    thetax = C1+C2*NEtalon+c3*NEtalon^2           }
{                            +C4*NEtalon^4                         }
{                         approx. = C1 + C2 * NEtalon              }
{         ***NOTE***   lever-length = 20.5 +/- 0.5 cm              }
{                      micrometer cm/step = 6.70E-005  (Accurate)  }
{                      therefore     C2   = 3.268E-006        ***  }
{                      C1 = 0 : at step 0 95% etalon causes lasing }
{3.     Things to consider carefully:                              }
{          a. Reasonably accurate RoughWN, which means Grating     }
{             position shoud be calibrated.                        }
{          b. The linear assumption on theta vs NEtalon. The accuracy}
{             of Coeffs                                            }
{4.     When scanning, OPO first moves NGrating & NCrystal to the  }
{          rough wavenumber.  Then start with order Nr, move the   }
{          etalon to the exact wavenumber.  Now, do the scan by move }
{          theta smaller while keep N const, NGrating and NCrystal }
{          will move small amount to keep RoughWavenumber closest to }
{          CurrentWavenumber.  When Nr=0, order will jump to Nr,     }
{          and continue the above process again.  (this way, the     }
{          scan range is improved).  During the scan, NCrystal is    }
{          allowed to move for the best result.                      }
{********************************************************************}
Unit  MCollect;
{$B-,N+,V-,R-,D+,F+}
INTERFACE
 Uses
  Dos,Crt,Graph,Filevar,CGlobal,Util,SubRut,GDataRut;
 Procedure MRuns;
Implementation
  Procedure RefreshScreen;
   Const
     head1A='< >:  ^ v:  Hm/End:    PgUp/Dn:    AltF/B:  Ins/Dele:
AltM:     AltJ:      ';
     head1B='    JC    JG       J20C        J20G        JE         J20E
 MoveWN    JmpOrd';
     head2A='F1    F2   F3    F4    AltS   AltQ    AltW    F5    F6
F7   F8    F9    F10  ';
     head2B='  CalW  MvE  AjPa  Mode    Scn    QScn    WDat  Z/UZ  AtZm
TOF  Init  TChk   Ex';
    Begin
     SetViewPort(0,0,GetMaxX,GetMaxY,ClipOn);
     SetColor(Red);
     Rectangle(0,0,GetMaxX,150);
     Line(0,15,GetMaxX,15);
     Line(0,40,GetMaxX,40);
     Line(0,55,GetMaxX,55);
     Line(0,90,GetMaxX,90);
     Line(0,100,GetMaxX,100);
     Line(0,150,GetMaxX,150);               {draw the box edges}

     SetColor(Cyan);
     SetTextJustify(CenterText,CenterText);
     OutTextXY(GetMaxX div 2,8,' COMMAND MENU ');
     OutTextXY(GetMaxX div 2,48,' CURRENT STATUS ');
     OutTextXY(GetMaxX div 2,96, ' DIALOG ');
     SetColor(LightGreen);
     if AcouOnly then
          OutTextXY(100,48,'Acoustic')
     else
          OutTextXY(100,48,'Ion Spec');
     if DataSaved then
       OutTextXY(480,48,'Data Saved')
     else
       OutTextXY(480,48,'Data Unsaved');
                                            {write the box names}
     SetColor(White);
     OutTextXY(GetMaxX div 2,22,head1A);
     OutTextXY(GetMaxX div 2,34,head2A);
     SetColor(LightGreen);
     OutTextXY(GetMaxX div 2 -250,62,'NCrystal=');
     OutTextXY(GetMaxX div 2 +200,62,'NGrating=');
     OutTextXY(GetMaxX div 2,62,'RoughWN =                cm-1');
     OutTextXY(GetMaxX div 2,22,head1B);
     OutTextXY(GetMaxX div 2,34,head2B);
     OutTextXY( 50,72,'NotScanning');
     OutTextXY(150,72,'NEatlon=');
     OutTextXY(285,72,'OrderN=');
     OutTextXY(515,72,'CurrentWN=             cm-1');
     OutTextXY(320,82,'From           To            cm-1;Step=
cm-1;Shots/Point=           ');
                                   {draw the text contents}
    ClearDrawings;
    end;            { proc RefreshScreen }
  Procedure TrackPara;
    Var
     TSString1,TSString2  : String[9];
     TSString3     : String[8];
     TSString4     : String[6];
    Begin
     Str(ScanWNFirst:9:4,TSString1);
     Str(ScanWNLast:9:4,TSString2);
     Str(ScanWNStep:8:4,TSString3);
     Str(ScanRep:6,TSString4);
     SetViewPort(40,77,130,87,ClipOn);
     ClearViewPort;
     SetViewPort(160,77,250,87,ClipOn);
     ClearViewPort;
     SetViewPort(340,77,410,87,ClipOn);
     ClearViewPort;
     SetViewPort(570,77,620,87,ClipOn);
     ClearViewPort;
     SetViewPort(0,77,639,87,ClipOn);
     OutTextXY(85,5,TSString1);
     OutTextXY(205,5,TSString2);
     OutTextXY(375,5,TSString3);
     OutTextXY(595,5,TSString4);                {write out scan paras}
   end;                         { proc TrackPara }
  Procedure AdjustPara ;
    const
     ADC1    ='CFactor is: ';
     ADS1    ='Enter ScanWNFirst: ';
     ADS2    ='Enter ScanWNLast: ';
     ADS3    ='Enter ScanWNStep: ';
     ADS4    ='Enter SHOTS/PIONT: ( < 50000 ) ';
     ADS5    =' Enter new value: ';
     ADS6    ='DOffset is: ';
     ADS7    ='POffset is: ';
     ADS8    ='ROffset is: ';
     ADS9    ='TrigDelay in us: ';
  var
     Vertposi   : integer;
     Tempspeed,Temp4,JunkI             : longint;
     JunkR,Temp1,Temp2,Temp3           : Real;
     ADStr  : String[11];
Begin
     SetViewPort(10,103,629,147,ClipOn);
     Str(CFactor:8:6,ADStr);
     OutTextXY(320,Vertstart,ADC1+ADStr+ADS5);
     MoveTo(550,Vertstart);
     GetNum(JunkI,Temp1,False);
     if Abs(Temp1) <  0.001  then
        CFactor := Temp1;          { CFactor should be quite small! }
     ClearViewPort;

     SetViewPort(10,103,629,147,ClipOn);
     VertPosi:= VertStart - 4 ;
     OutTextXY(GetMaxX div 2,VertPosi,ADS1);
     MoveTo(520, VertPosi);
     GetNum(JunkI,Temp1,False);
     If ( Temp1 >= 2000 ) and ( Temp1 <= 4700)  then
        ScanWNFirst:=Temp1;
     VertPosi:= VertPosi + 9;
     OutTextXY(GetMaxX div 2,VertPosi,ADS2);
     MoveTo(520, VertPosi);
     GetNum(JunkI,Temp2,False);
     If ( Temp2 >= ScanWNFirst ) and ( Temp2 <= 4700)  then
        ScanWNLast:=Temp2;

     VertPosi:= VertPosi + 9;
     OutTextXY(GetMaxX div 2,VertPosi,ADS3);
     MoveTo(520, VertPosi);
     GetNum(JunkI,Temp3,False);
     If (Abs(Temp3 + 1.01)>0.000001) and ( Abs(Temp3) <= PZTMax )  then
        ScanWNStep:=Temp3;
     NDataPoints := Round(Abs(ScanWNLast - ScanWNFirst)
                 / ScanWNStep + 1.0000);

     VertPosi:= VertPosi + 9;
     OutTextXY(GetMaxX div 2,VertPosi,ADS4);
     MoveTo(520, VertPosi);
     GetNum(Temp4,JunkR,True);
     If ( Temp4 > 0 ) and ( Temp4 <= 50000 )  then
        ScanRep :=Temp4;
     Delay(800);
     ClearViewPort;                           { We have got scan paras }
     TrackPara;
     SetViewPort(10,103,629,147,ClipOn);

     VertPosi := VertStart;
     Str(OOffset[0]:11:5,ADStr);
     OutTextXY(320,Vertposi,ADS6+ADStr+ADS5);
     MoveTo(550,Vertposi);
     GetNum(JunkI,Temp1,False);
     if Abs( Temp1 + 1.01 ) >  0.000001  then
        OOffset[0] := Temp1;

     VertPosi :=VertPosi + VertStep;
     Str(OOffset[1]:11:5,ADStr);
     OutTextXY(320,Vertposi,ADS7+ADStr+ADS5);
     MoveTo(550,Vertposi);
     GetNum(JunkI,Temp1,False);
     if Abs( Temp1 + 1.01 ) >  0.000001  then
        OOffset[1] := Temp1;
     VertPosi :=VertPosi +VertStep;
     Str(OOffset[2]:11:5,ADStr);
     OutTextXY(320,Vertposi,ADS8+ADStr+ADS5);
     MoveTo(550,Vertposi);
     GetNum(JunkI,Temp1,False);
     if Abs( Temp1 + 1.01 ) >  0.000001  then
        OOffset[2] := Temp1;

     ClearViewPort;
     Str(TDelay:11:2,ADstr);
     OutTextXY(320,Vertstart,ADS9+ADStr+AdS5);
     MoveTo(550,VertStart);
     GetNum(JunkI,Temp1,False);
     if ( Temp1 >= 20 )  and ( Temp1 <= 30000 ) then
        TDelay := Temp1;
     ClearViewPort;
    end;                    { proc adjpara }
(************************************************)
Procedure MRuns;
 var
  Key                          : Char;
  CollectFlag                 : Boolean;
  GraphDetect,GraphMode,
  i, error                   : Integer;
  Computerfile                : Text;
  JPZTVolts                   : Real;
  JunkStr,str1,str2,str3,
  str4,str5                   : String[90];
Begin { Procedure  MRuns }
  Resolution :='Med';
  Clrscr;
  GraphDetect:=Detect;
  Graph.DetectGraph(GraphDetect, GraphMode);
  InitGraph(GraphDetect,GraphMode,'C:\ybc\aqdata');
                  { NOTE : I have a copy of EGAVGA.BGI in C:\ybc\aqdata }
  SetBkColor(DarkGray); SetColor(LightGreen);
  Assign(Computerfile, 'C:\YBC\AQDATA\OPOPOSI.DAT');
  Reset(computerfile);
  Readln(computerfile, Junkstr);
  Readln(computerfile, Junkstr);
  if junkstr ='Acoustic' then
        AcouOnly := True
  else
        AcouOnly := False;
  DataSaved := False;
  Readln(computerfile, TDelay);
  Readln(computerfile, JunkStr);
  Readln(computerfile, OOffset[0],OOffset[1],OOffset[2]);
  Readln(computerfile, JunkStr);
  Readln(computerfile, NCrystal,NGrating,NEtalon,JPZTVolts);
  Readln(computerfile, NcSpeed,NgSpeed,NeSpeed);
  Readln(computerfile, JunkStr);
  Readln(computerfile, Str1);
  Readln(Computerfile, Str2);
  Readln(Computerfile, FitSYNData[1,1],FitSYNData[1,2],
                        FitSYNData[1,3],FitSYNData[1,4]);
  Readln(computerfile, JunkStr);
  Readln(computerfile, ScanWNFirst,ScanWNLast,ScanWNStep,ScanRep);
  Readln(computerfile, WindowRatio[1],WindowRatio[2],WindowRatio[3]);
  Readln(Computerfile, CFactor,FitSYNData[2,1],FitSYNData[2,2],
                        FitSYNData[2,3],FitSYNData[2,4]);
  Readln(computerfile, JunkStr);
  Readln(computerfile, Str3);
  Readln(computerfile, Str4);
  Readln(computerfile, Str5);
  Close(computerfile);
            { last time positions loaded in from file OPOPOSI.DAT }
  NDataPoints := Round(Abs(ScanWNLast-ScanWNFirst) / ScanWNStep + 1.0 );
  NPresentPoint := 0;
  Scanning := False;
  For i := 0 to NMax do
        WindowOffset[i+1] := 0.0;
  Spline;
  RoughWN := WNNG(NGrating);
  OnWN(RoughWN,OrderN);
  RefreshScreen;
  KeepTrackM(collectFlag);
  TrackPara;
  InitStuff(CollectFlag);

  Repeat
     Repeat until Readkey = #0;
     key := Readkey;
     if key in [left,right,up,down, home,pg_up,ending,pg_dn,
                                {LMotor Jogging and jogging20 }
              Alt_F,Alt_B,Ins, Del,
                                {HMotor Jogging ang jogging20 }
              Alt_M,            {Move to a new wavenumber position}
              Alt_J,            {Jump Order}
              F1,               {Calibrate / reset Wavenumber}
              F2,               {Etalon Moving}
              F3,               {Para setting}
              F4,               {Acoustic Only}
              F5,               {Zoom/unzoom}
              F6,               {AutoZoom}
              F7,               {TOF Control}
              F8,               {Initialize the boards}
              F9,               {Timing Check}
              Alt_S,            {StartScan}
              Alt_Q,            {QuitScan}
              Alt_W,            {Write data to file}
              F10 ]             {Exit}
     then
      begin
       case key of
left,right,home,ending:
          if (( not Scanning ) or ( ScanRep - NPresentRep > 10 )) and
             ( NCrystal > CGMin ) and ( NCrystal < CGMax ) then
             begin
                if key = left then
                          JogL(0,-1,CollectFlag)
                else if key = right then
                          JogL(0,1,CollectFlag)
                else if key = home then
                          Jog20L(0,20,CollectFlag)
                else
                          Jog20L(0,-20,CollectFlag);
                KeepTrackM(CollectFlag);
                Delay(400);
             end;
up,down,pg_up,pg_dn :
        if ( not Scanning )  and
           ( NGrating > CGMin ) and ( NGrating < CGMax ) then
            begin
               if key = up then
                  JogL(1,1,CollectFlag)
               else if key = down then
                  JogL(1,-1,CollectFlag)
               else if key = pg_up then
                  Jog20L(1,20,CollectFlag)
               else
                  Jog20L(1,-20,CollectFlag);
               KeepTrackM(CollectFlag);
               Delay(400);
            end;
Alt_F,Alt_B:
        if ( not Scanning ) and
           ( NEtalon > EMin ) and ( NEtalon < EMax ) then
           begin
              if key = Alt_F then
                 JogH(0,1,CollectFlag)
              else
                 JogH(0,-1,CollectFlag);
              KeepTrackM(CollectFlag);
              Delay(400);
           end;
Ins,Del:
        if ( not Scanning )  and
           ( NEtalon > EMin ) and ( NEtalon < EMax ) then
            begin
               if key = Ins then
                  Jog20H(0,20,CollectFlag)
               else
                 Jog20H(0,-20,CollectFlag);
               KeepTrackM(collectFlag);
               Delay(400);
            end;
F2    : begin
        if not scanning then
            begin
                MoveHMotor(0,CollectFlag);
                KeepTrackM(CollectFlag);
            end;
       end;
F1:    begin
          if not scanning then
            begin
                OnWN(RoughWN,OrderN);
                KeepTrackM(CollectFlag);
            end;
       end;
Alt_M: begin
          if not scanning then
             begin
                MoveToWN(CollectFlag);
                KeepTrackM(collectFlag);
                OnWN(RoughWN,OrderN);
                KeepTrackM(CollectFlag);
             end;
       end;
Alt_J: begin
          if not scanning then
              begin
                 JumpOrN;
                 KeepTrackM(CollectFlag);
              end;
       end;
F3 :   begin
         if not scanning  then
              AdjustPara;
              OnWN(RoughWN,OrderN);
              KeepTrackM(CollectFlag);
       end;
F4:    Begin
            if not Scanning then
                AcouYesNo;
       End;
F5:    if ( not Scanning ) or ( ScanRep - NPresentRep > 30 ) then
               Zoom123;
F6:    if ( not Scanning ) or ( ScanRep - NPresentRep > 30 ) then
               AutoZoom;
F7:    Begin
            if not Scanning then
                ToTOF;
       End;
F8:    begin
           if not scanning then
               begin
                   Initstuff(CollectFlag);
                   GetStartPosition;
                   KeepTrackM(collectFlag);
                   OnWN(RoughWN,OrderN);
                   KeepTrackM(collectFlag);
               end;
       end;
F9:      Begin
            if not Scanning then
               begin
                 TChkDial(i,CollectFlag);
                 if CollectFlag then
                    GetTCheck(0,i,0,ScanRep,NDataPoints,error);
               end;
         End;
Alt_S:   begin
            if not scanning then
            begin
               ScanDial(CollectFlag);
               if CollectFlag then
                  GetData(0,0,0,2,ScanRep,NDataPoints,error);
            end;
         end;
Alt_Q:   begin
            if scanning then
            begin
                   StopScan;
                   KeepTrackM(CollectFlag);
            end;
         end;
Alt_W:   begin
            if not scanning then
                   DataSave;
         end;
F10:     begin
            if not scanning then
                begin
                   QuitCollect(key);
                   if key in [’N’,’n’] then key:=’N’ else key:=F10;
                end;
         end;
  end; { Case }
end; {if}
until key= F10;                         {exit}
   Rewrite(computerfile);
   Writeln(computerfile,’OPO scan info file’);
   if AcouOnly then
       Writeln(computerfile,’Acoustic’)
   else
       Writeln(computerfile,’Ion Spec’);
   Writeln(computerfile,TDelay:5:0);
   Writeln(computerfile,’1. offset info:’);
   Writeln(computerfile,OOffset[0]:15,’ ’,
                OOffset[1]:15,’ ’,OOffset[2]:15);
   Writeln(computerfile,’2. position info:’);
   Writeln(computerfile,NCrystal:9,NGrating:9,NEtalon:9,JPZTVolts:9:4);
   Writeln(computerFile,NcSpeed:5,NgSpeed:5,NeSpeed:5);
   Writeln(computerfile,’3. LowResScan info:’);
   Writeln(computerfile,Str1);
   Writeln(computerfile,Str2);
   Writeln(Computerfile,FitSYNData[1,1]:17,’ ’,FitSYNData[1,2]:17,’ ’,
   FitSYNData[1,3]:17,’ ’,FitSYNData[1,4]:17);
   Writeln(Computerfile,’MedResScan info:’);
   Writeln(computerfile,ScanWNFirst:9:4,’ ’,ScanWNLast:9:4,’ ’,
                        ScanWNStep:9:4,’ ’,ScanRep:6);
   Writeln(Computerfile,WindowRatio[1]:9:4,' ',WindowRatio[2]:9:4,' ',
   WindowRatio[3]:9:4);
   Writeln(Computerfile,CFactor:8:6,' ',FitSYNData[2,1]:17,' ',
          FitSYNData[2,2]:17,' ',FitSYNData[2,3]:17,
            ' ',FitSYNData[2,4]:17);
   Writeln(computerfile,'HighResScan info:');
   Writeln(Computerfile,Str3);
   Writeln(Computerfile,Str4);
   Writeln(computerfile,Str5);
   Close(computerfile);             { update : save back to OPOPOSI.DAT }
   CloseGraph;
   end;                              { Procedure  MRuns }
end.                                  { unit Mcollect }
```

HCOLLECT.PAS:

```
{********************************}
{ The High Resolution OPO Data   }
{  Collecting Workhorse          }
{********************************}
{************************************************************}
{                   SINGLE MODE SCAN                         }
{    The scan range is about 1 cm-1.                         }
{       The scan is like this:                               }
{        1. Crystal and grating angles remain fixed.         }
{        2. Find a position (NEtalon,PZTVolt) that lases.    }
{        3. Scan NEtalon just like the MCollect case         }
{        4. Matching The NEtalon with the proper PZTVolt:    }
{               WN := NN/(2 * L)  therefore                  }
{               d(WN)/dL := - NN/(2 * L * L) := -WN/L        }
{               therefore d(WN)/dV  := (dL/dV) * (-WN/L)     }
{               L-V is a non-linear "hysteresis" curve, if   }
{               use linear approximation, then:              }
{               dL/dV := 3.5E-0005 * (cm/volt)               }
{        5. CHFactor is added to take care of the slight     }
{               mismatch                                     }
{*notes on Burleigh piezo motion system**********************}
{  (element PZS-50, which can travel 35um@100v)              }
{  (driver PZ-300, 3-channel low-current low-frequency model)}
{   1. We only use Channel 1 now, Channel 2,3 are not used   }
{   2. VZTVolts := PZTGain * (DAInput + FrontKnob)           }
{             GZTGain := 20.0; DAInput: 0-5v                 }
{             Please set FrontKnob :=0!!!!!                  }
{   3. To get optimum repeatability, repole the element if   }
{      not used for 24 hours.                                }
{      prodecure: Apply maximum working voltage for 30 sec.  }
{                 then return to 0v for 60 sec.              }
{   4. PZTs have inherent characteristics:                   }
{          hysteresis, linearity, creep, step response       }
{       a) hysteresis: best to approach the voltage from     }
{          same direction.                                   }
{          Spec. hysteresis:  20%                            }
{       b) non-linearity: it is best to operate the lower    }
{          hysteresis curve, i.e., increase the voltage only }
{          Spec. non-linearity: 3.0%                         }
{       c) Creep: after the voltage change is applied, PZT   }
{          extends a few percent more before settling down.  }
{          creep time is defined when the creep drops to     }
{          less than 1 percent.                              }
{          Spec. Creep:                                      }
{            INCREASING V                DECREASING V        }
{            3-4%  time: 20-40sec        4-5% time: 50-70sec }
{       d) Step response:                                    }
{            This problem has been taken care of by PZ-300   }
{           driver.  Just limit the input D-A voltage to    }
{           zero to five volts.  Nothing to worry about     }
{           on this one!                                    }
{***********************************************************}
Unit  HCollect;
{$B-,N+,V-,R-,D+,F+}
INTERFACE
 Uses
  Dos,Crt,Graph,Filevar,CGlobal,Util,SubRut,GDataRut;
 Procedure HRuns;
Implementation
  Procedure RefreshScreen;
    const
     head1A='< >:    ^ v:    Home/End:      PgUp/Dn:      AltF/B:
Ins/Del:      AltP:     ';
     head1B='    JogC    JogG          Jog20C          Jog20G        JogE
   Jog20E     PZT ';
     head2A='F1    F2   F3    F4    AltS    AltQ    AltW    F5    F6
F7   F8   F9    F10  ';
     head2B='  CalW  MvE  AjPa  Mode    Scn    QScn    WDat  Z/UZ  AtZm
TOF  Int  TChk   Ex';
    Begin
     SetViewPort(0,0,GetMaxX,GetMaxY,ClipOn);
     SetColor(Red);
     Rectangle(0,0,GetMaxX,150);
     Line(0,15,GetMaxX,15);
     Line(0,40,GetMaxX,40);
     Line(0,55,GetMaxX,55);
     Line(0,90,GetMaxX,90);
     Line(0,100,GetMaxX,100);
     Line(0,150,GetMaxX,150);                {draw the box edges}

     SetColor(Cyan);
     SetTextJustify( CenterText,CenterText);
     OutTextXY(GetMaxX div 2,8,' COMMAND MENU ');
     OutTextXY(GetMaxX div 2,48,' CURRENT STATUS ');
     OutTextXY(GetMaxX div 2,96, ' DIALOG ');
     SetColor(LightGreen);
     if AcouOnly then
          OutTextXY(100,48,'Acoustic')
     else
          OutTextXY(100,48,'Ion Spec');
     if DataSaved then
       OutTextXY(480,48,'Data Saved')
     else
       OutTextXY(480,48,'Data Unsaved');
                                             {write the box names}
     SetColor(White);
     OutTextXY(GetMaxX div 2,22,head1A);
     OutTextXY(GetMaxX div 2,34,head2A);
     SetColor(LightGreen);

     OutTextXY(GetMaxX div 2 -250,62,'NCrystal=');
     OutTextXY(GetMaxX div 2 +200,62,'NGrating=');
     OutTextXY(GetMaxX div 2,62,'RoughWN=                        cm-1');
     OutTextXY(GetMaxX div 2,22,head1B);
     OutTextXY(GetMaxX div 2,34,head2B);

     OutTextXY( 50,72,'NotScanning');
     OutTextXY(150,72,'NEatlon=');
     OutTextXY(370,72,'CurrentWN=               cm-1');
     OutTextXY(530,72,'DAVolt=');
     OutTextXY(320,82,'From            To               cm-1;Step=
  cm-1;Shots/Point=           ');
                                    {draw the text contents}
     ClearDrawings;
    end;             { proc RefreshScreen }
   Procedure TrackPara;
    Var
     TSString1,TSString2  : String[9]; { SFirst , SLast }
     TSString3    : String[8];         { SStepSize }
     TSString4    : String[6];          { ScanRep }
     Begin
      Str(ScanWNFirst:9:4,TSString1);
      Str(ScanWNLast:9:4,TSString2);
      Str(ScanWNStep:8:4,TSString3);
      Str(ScanRep:6,TSString4);
      SetViewPort(40,77,130,87,ClipOn);ClearViewPort;
      SetViewPort(160,77,250,87,ClipOn);ClearViewPort;
      SetViewPort(340,77,410,87,ClipOn);ClearViewPort;
      SetViewPort(570,77,620,87,ClipOn);ClearViewPort;
      SetViewPort(0,77,639,87,ClipOn);
      OutTextXY(85,5,TSString1);
      OutTextXY(205,5,TSString2);
      OutTextXY(375,5,TSString3);
      OutTextXY(595,5,TSString4); {write out scan paras }
    end;                                { proc TrackPara }
   Procedure AdjustPara ;
     const
      AD1 ='Adjust Parameters.                                            ';
      AD2 ='CHFactor is: ';
      ADS1     ='Enter ScanWNFirst: ';
      ADS2     ='Enter ScanWNLast: ';
      ADS3     ='Enter ScanWNStep: ';
      ADS4     ='Enter SHOTS/PIONT: ( < 50000 ) ';
      ADS5     =' Enter new value: ';
      ADS6     ='DOffset is: ';
      ADS7     ='POffset is: ';
      ADS8     ='ROffset is: ';
      ADS9     ='TrigDelay in us is: ';
     var
      Vertposi : integer;
      Tempspeed,Temp4,JunkI            : longint;
      JunkR,Temp1                      : Real;
      ADStr : String[11];
Begin
  SetViewPort(10,103,629,147,ClipOn);
  VertPosi := VertStart;
  OutTextXY(GetMaxX div 2,VertPosi,AD1);

  VertPosi := VertPosi + VertStep;
  Str(CHFactor:9:7,ADStr);
  OutTextXY(320,Vertposi,AD2+ADStr+ADS5);
  MoveTo(550, VertPosi);
  GetNum(JunkI,Temp1,False);
  If Abs(Temp1)  <= 0.1  then
     CHFactor :=Temp1 ;
  Delay(800);
  ClearViewPort;
  TrackPara;
             { We have got CHFactor ! }
  SetViewPort(10,103,629,147,ClipOn);
  VertPosi:= VertStart - 4 ;
  OutTextXY(GetMaxX div 2,VertPosi,ADS1);
  MoveTo(520, VertPosi);
  GetNum(JunkI,Temp1,False);
  If ( Temp1 >= 2700 ) and ( Temp1 <= 4700)  then
      ScanWNFirst:=Temp1;

  VertPosi:= VertPosi + 9;
  OutTextXY(GetMaxX div 2,VertPosi,ADS2);
  MoveTo(520, VertPosi);
  GetNum(JunkI,Temp1,False);
  If ( Temp1 >= 2700 ) and ( Temp1 <= 4700)  then
      ScanWNLast:=Temp1;

  VertPosi:= VertPosi + 9;
  OutTextXY(GetMaxX div 2,VertPosi,ADS3);
  MoveTo(520, VertPosi);
  GetNum(JunkI,Temp1,False);
  If Abs(Temp1) <= 0.1   then
      ScanWNStep:=Temp1;

  NDataPoints := Round((ScanWNLast - ScanWNFirst)/ScanWNStep + 1.0000 );

  VertPosi:= VertPosi + 9;
  OutTextXY(GetMaxX div 2,VertPosi,ADS4);
  MoveTo(520, VertPosi);
  GetNum(Temp4,JunkR,True);
  If ( Temp4 > 0 ) and ( Temp4 <= 50000 )  then
      ScanRep :=Temp4;
  Delay(800);
  ClearViewPort;                              { We have got scan paras }
  TrackPara;
  SetViewPort(10,103,629,147,ClipOn);

  VertPosi := VertStart;

  Str(OOffset[0]:11:5,ADStr);
  OutTextXY(320,Vertposi,ADS6+ADStr+ADS5);
  MoveTo(550,Vertposi);
  GetNum(JunkI,Temp1,False);
  if Abs( Temp1 + 1.01 ) >  0.000001  then
      OOffset[0] := Temp1;
  VertPosi :=VertPosi + VertStep;
  Str(OOffset[1]:11:5,ADStr);
  OutTextXY(320,Vertposi,ADS7+ADStr+ADS5);
  MoveTo(550,Vertposi);
  GetNum(JunkI,Temp1,False);
  if Abs( Temp1 + 1.01 ) >  0.000001  then
      OOffset[1] := Temp1;

  VertPosi :=VertPosi +VertStep;
  Str(OOffset[2]:11:5,ADStr);
  OutTextXY(320,Vertposi,ADS8+ADStr+ADS5);
  MoveTo(550,Vertposi);
  GetNum(JunkI,Temp1,False);
  if Abs( Temp1 + 1.01 ) >  0.000001  then
      OOffset[2] := Temp1;

  ClearViewPort;
  Str(TDelay:11:2,Adstr);
  OutTextXY(320,Vertstart,ADS9+Adstr+AdS5);
  MoveTo(550,VertStart);
  GetNum(JunkI,Temp1,False);
  if ( Temp1 >= 20 )  and ( Temp1 = 33000 ) then
      TDelay := Temp1;
  ClearViewPort;
end;                 { proc adjpara }
```

```
(**********************************************)
Procedure HRuns;
 var
  Key                         : Char;
  CollectFlag                 : Boolean;
  GraphDetect,GraphMode,
  i, error                    : Integer;
  Computerfile                : Text;
  JunkStr,str1,str2,str3,str4   : String[90];

Begin { Procedure  HRuns }
  Resolution :='High';
  Clrscr;
  GraphDetect:=Detect;
  Graph.DetectGraph(GraphDetect, GraphMode);
  InitGraph(GraphDetect,GraphMode,'C:\ybc\aqdata');
                 { NOTE : I have a copy of EGAVGA.BGI in C:\ybc\aqdata }
  SetBkColor(DarkGray); SetColor(LightGreen);
  Assign(Computerfile, 'C:\YBC\AQDATA\OPOPOSI.DAT');
  Reset(computerfile);
  Readln(computerfile, Junkstr);
  Readln(computerfile, Junkstr);
  if junkstr ='Acoustic' then
        AcouOnly := True
  else
        AcouOnly := False;
  DataSaved := False;
  Readln(computerfile, TDelay);
  Readln(computerfile, JunkStr);
  Readln(computerfile, OOffset[0],OOffset[1],OOffset[2]);
  Readln(computerfile, JunkStr);
  Readln(computerfile, NCrystal,NGrating,NEtalon,DAVolts);
  Readln(computerfile, NcSpeed,NgSpeed,NeSpeed);
  Readln(computerfile, JunkStr);
  Readln(computerfile, Str1);
  Readln(Computerfile, Str2);
  Readln(Computerfile, FitSYNData[1,1],FitSYNData[1,2],
                        FitSYNData[1,3],FitSYNData[1,4]);
  Readln(computerfile,JunkStr);
  Readln(computerfile,Str3);
  Readln(computerfile,Str4);
  Readln(Computerfile, CFactor,FitSYNData[2,1],FitSYNData[2,2],
                        FitSYNData[2,3],FitSYNData[2,4]);
  Readln(computerfile,Junkstr);
  Readln(computerfile,ScanWNFirst,ScanWNLast,ScanWNStep,ScanRep);
  Readln(computerfile,WindowRatio[1],WindowRatio[2],WindowRatio[3]);
  Readln(computerfile,CHFactor);
  Close(computerfile);
                 { last time positions loaded in from file OPOPOSI.DAT }
  NDataPoints := Round((ScanWNLast - ScanWNFirst) / ScanWNStep + 1.0 );
  NPresentPoint := 0;
  Scanning := False;
  For i := 0 to NMax do
        WindowOffset[i+1] := 0.0;
  Spline;
  RoughWN := WNNG(NGrating);
  OnWN(RoughWN,OrderN);
  RefreshScreen;
  KeepTrackH(collectFlag);
  TrackPara;
  InitStuff(CollectFlag);

  Repeat
```

```
     Repeat until Readkey = #0;
     key := Readkey;
     if key in [left,right,up,down, home,pg_up,ending,pg_dn,
                               {LMotor Jogging and jogging20 }
              Alt_F,Alt_B,Ins,Del,
                               {HMotor Jogging ang jogging20 }
                          Alt_P,
                          F1,  {Calibrate WN}
                          F2,  {Etalon Moving}
                          F3,  {Para setting}
                          F4,  {Acoustic Only}
                          F5,  {zoom/unzoom}
                          F6,  {AutoZoom}
                          F7,  {TOF Control}
                          F8,  {Initialize the boards}
                          F9,  {Timing Check}
                       Alt_S,  {StartScan}
                       Alt_Q,  {QuitScan}
                       Alt_W,  {Write data to file}
                        F10 ]  {Exit}
                         then
                        begin
                   case key of
left,right,home,ending:
        if ( not Scanning ) and
           ( NCrystal > CGMin ) and ( NCrystal < CGMax ) then
            begin
               if key = left then
                  JogL(0,-1,CollectFlag)
               else if key = right then
                  JogL(0,1,CollectFlag)
               else if key = home then
                  Jog20L(0,20,CollectFlag)
               else
                  Jog20L(0,-20,CollectFlag);
               KeepTrackH(CollectFlag);
               Delay(400);
            end;
up,down,pg_up,pg_dn :
        if ( not Scanning ) and
        ( NGrating > CGMin ) and ( NGrating < CGMax ) then
            begin
               if key = up then
                  JogL(1,1,CollectFlag)
               else if key = down then
                  JogL(1,-1,CollectFlag)
               else if key = pg_up then
                  Jog20L(1,20,CollectFlag)
               else
                  Jog20L(1,-20,CollectFlag);
               KeepTrackH(CollectFlag);
               Delay(400);
            end;
Alt_F,Alt_B:
        if (( not Scanning ) or ( ScanRep - NPresentRep > 10 )) and
        ( NEtalon > EMin ) and ( NEtalon < EMax ) then
            begin
               if key = Alt_F then
                 JogH(0,1,CollectFlag)
               else
                 JogH(0,-1,CollectFlag);
               KeepTrackH(CollectFlag);
               Delay(400);
            end;
Ins,Del:
        if (( not Scanning ) or ( ScanRep - NPresentRep > 10 )) and
        ( NEtalon > EMin ) and ( NEtalon < EMax ) then
            begin
               if key = Ins then
                 Jog20H(0,20,CollectFlag)
               else
                 Jog20H(0,-20,CollectFlag);
               KeepTrackH(collectFlag);
               Delay(400);
            end;
Alt_P:      begin
               if not scanning then
                 begin
                   MovePZT(CollectFlag);
                   KeepTrackH(collectFlag);
                 end;
            end;
F1:    begin
          if not scanning then
            begin
                OnWN(RoughWN,OrderN);
                KeepTrackH(CollectFlag);
            end;
       end;
F2:    begin
          if not scanning then
          begin
                MoveHMotor(0,CollectFlag);
                KeepTrackH(CollectFlag);
             end;
             end;
F3 :   begin
         if not scanning  then
             AdjustPara;
       end;
F4:      Begin
            if not Scanning then
                AcouYesNo;
         End;
F5:      if ( not Scanning ) or ( ScanRep - NPresentRep > 30 ) then
               Zoom123;
F6:      if ( not Scanning ) or ( ScanRep - NPresentRep > 30 ) then
               AutoZoom;
F7:      Begin
            if not Scanning then
                ToTOF;
         End;
F8:      begin
           if not scanning then
               begin
                   Initstuff(CollectFlag);
                   GetStartPosition;
                   KeepTrackH(collectFlag);
               end;
         end;
F9:      Begin
            if not Scanning then
              begin
                TChkDial(i,CollectFlag);
                if CollectFlag then
                   GetTCheck(0,i,0,ScanRep,NDataPoints,error);
              end;
         End;
Alt_S:   begin
             if not scanning then
               begin
                  ScanDial(CollectFlag);
                  if CollectFlag then
                     GetData(0,0,0,2,ScanRep,NDataPoints,error);
               end;
         end;
Alt_Q:   begin
            if scanning then
               begin
                   StopScan;
                   KeepTrackH(CollectFlag);
               end;
               end;
Alt_W:   begin
            if not scanning then
                DataSave;
         end;
F10:     begin
            if not scanning then
            begin
            QuitCollect(Key);
            if key in ['N','n'] then key :='N' else key:=F10;

                         end;
                   end;
          end; { Case }
       end; {if}
    until key= F10;                             {exit}
    Rewrite(computerfile);
    Writeln(computerfile,'OPO scan info file:');
    if AcouOnly then
       Writeln(computerfile,'Acoustic')
    else
       Writeln(computerfile,'Ion Spec');
    Writeln(computerfile,TDelay:5:0);
    Writeln(computerfile,'1. offset info:');
    Writeln(computerfile,OOffset[0]:15,' ',OOffset[1]:15,' ',
          OOffset[2]:15);
    Writeln(computerfile,'2. position info:');
    Writeln(computerfile,NCrystal:9,NGrating:9,NEtalon:9,DAVolts:9:4);
    Writeln(computerFile,NcSpeed:5,NgSpeed:5,NeSpeed:5);
    Writeln(computerfile,'3. LowResScan info:');
    Writeln(computerfile,Str1);
    Writeln(computerfile,Str2);
    Writeln(Computerfile,FitSYNData[1,1]:17,' ',FitSYNData[1,2]:17,' ',
        FitSYNData[1,3]:17,' ',FitSYNData[1,4]:17);
    Writeln(computerfile,'4. MedResScan info:');
    Writeln(computerfile,Str3);
    Writeln(computerfile,Str4);
    Writeln(Computerfile,CFactor:9:7,' ',FitSYNData[2,1]:17,' ',
          FitSYNData[2,2]:17,' ',FitSYNData[2,3]:17,' ',
          FitSYNData[2,4]:17);
    Writeln(computerfile,'5. HighResScan info:');
    Writeln(Computerfile,ScanWNFirst:9:4,' ',ScanWNLast:9:4,' ',
                        ScanWNStep:9:4,' ',ScanRep:9);
    Writeln(Computerfile,WindowRatio[1]:9:4,' ',WindowRatio[2]:9:4,' ',
                        WindowRatio[3]:9:4);
    Writeln(computerfile,CHFactor:9:6);
    Close(computerfile);
```

```
                        { update : save back to OPOPOSI.DAT }
    CloseGraph;
    end;             { Procedure  HRuns }
 end.                        { unit Hcollect }
```

CGLOBAL.PAS:

```
{*****************************}
{ Collect "Global" Variables  }
{*****************************}
Unit CGlobal;
{$F+}
INTERFACE
 Const
   PumpWN     =9398.0;          {+/- 4 cm-1, YAG 1.064 micron pump}
   CGMax      = 35000; CGMin = 0;
   EMax       = 16000; EMin = -500;
   MaxMotorSpeed = 20; MinMotorSpeed = 255;
                                { stepping motors' parameters }
   IndexE         = 1.442;      { Infrasil, see MCollect for details }
   ThickE         = 0.20;       { cm }
   LCavity        = 15;         { cm, not that accurate yet }
   dLdV           = 3.5E-0005;  { cm/volt }
   PZTGain        = 20.0;       { PZT output / DA input }
   PZTMax         = 100.0;      { Largest safe Pizeo votage output }
   DASAddress     = $310;
   LMotorAddress  = $330;
   HMotorAddress  = $300;
   DASIntLev      = 7;          { Bus addresses and interrupt level }
   VertStart = 10;
   VertStep  = 10;
   DTop = 155; DBot = 285; DZero =DBot-15 ;
   PTop = 290; PBot = 340; PZero =PBot-10 ;
   RTop = 345; RBot = 475; RZero =RBot-15 ;
                                { VGA screen paras }
 type
   OpoDMaxMinType = Array[1..2,0..2] of real;
   WindowType     = Array[1..3] of Real;
   SS             = String[128];
 var
   OOffset                    : Array[0..2] of Real;
                                        { Signal Offset Settings }
   TDelay                     : Real;
                                        { Programable timer delay }
   OpoDMaxmin                 : OpoDMaxMinType;
   WindowRatio,WindowOffset   : WindowType;
                                        { Data display during the scan }
   CFactor,CHFactor           : Real;
   FitSYNData                 : Array[1..2,1..4] of Real;
                                        { Scan Schronization parameters }
   NcSpeed,NgSpeed,NeSpeed    : Integer;
   NCrystal,NGrating,NEtalon: longint;
   DAVolts                    : Real;
   OrderN                     : integer;
   RoughWN,CurrentWN          : Real;
                                         { Current position paramenters }
   SMode                      : String[8];
   EtaFirst,EtaLast,EtaStep   : Longint;
   NPresentPoint              : Integer;
   NPresentRep                : Longint;
   Scanning,AcouOnly          : Boolean;
   DataSaved                  : Boolean; {Current Scaning parameters}
IMPLEMENTATION
end.                                 { unit CGlobal }
```

UTIL.PAS:

```
{************************************************}
{  MSTEP5   MOTOR  & PZT DRIVERS FOR L,M,HCollect}
{************************************************}
{****************NOTE!!!!!!!******************}
{         HMOTOR motion is INVERTED            }
{         with respect to LMOTOR.              }
{*********************************************}
Unit Util;
  {$N+}                            {N+ to activate 8087 mode}
  {$L MStep.obj}  {$L MStep2.obj}
  {$F+}                            {force external proc to be far}
interface
  uses
   Dos,Crt,FileVar,CGlobal,Tp4d16;

  Procedure InitLmotor( IMWhich : integer;var InitFlag :integer);
  Procedure JogL(JWhich,JDir: integer;var JFlag:Boolean);
  Procedure Jog20L(J20Which,J20Dir: integer;var J20Flag :Boolean);
  Procedure TurnLMotor (TMWhich: byte; TurnStep: longint;var TMFlag:
integer);
  Procedure InitHMotor( IMWhich : integer;var InitFlag :integer);
  Procedure JogH(JWhich,JDir: integer;var JFlag:Boolean);
  Procedure Jog20H(J20Which,J20Dir: integer; var J20Flag :Boolean);
  Procedure TurnHMotor (TMWhich: byte; TurnStep: longint;var TMFlag:
integer);
  Procedure GoPZT (var GPZTFlag: Boolean);
implementation
 type
   ParaArray  =array [0..9] of integer;
 var
   d: ParaArray;
   md, flag: integer;
   stp     : longint;

  Procedure Mstep (var md: integer; var d : ParaArray;
                   var stp: longint; var flag: integer);  external;
                   {md=mode,d=channel and other status data, stp=step #,
                       flag=error#. External tells Pascal to look for
                       external proc that was linked by $L Mstep.obj}
  Procedure Mstep2 (var md: integer; var d : ParaArray;
                   var stp: longint; var flag: integer);  external;
                   {md=mode,d=channel and other status data, stp=step #,
                       flag=error#. External tells Pascal to look for
                       external proc that was linked by $L Mstep2.obj}

  Procedure InitLMotor;
   Begin
     d[1] :=  255;     {start rate divider      255--->20}
     d[2] :=  100;     {high speed run rate divider 255--->20}
     d[3] :=  200;     {acceleration/deceleration steps  1---->65535}
     d[4] :=  2;       {motor phase:1=three phase 2=four ph. 3=five ph.}
     d[5] :=  0;       {excitation  : 0=full step   1=half step}
     d[6] :=  0;       {logic parity data: 0= inverted  1=positive true}
     d[7] :=  0;       {clock source : 0=internal 1=100kHz+ divider
2=external}
     d[8] :=  0;       {switching at standstill 0=off   1=on}
     d[9] :=  LMotorAddress;     { low motor board base address }
     md   :=  12;            { initialization ! }

     d[0] :=IMWhich;
     MStep(md,d,stp,InitFlag);
   end;                           {proc initmotor}
```

```
Procedure JogL;
 var
   fun : longint;
  begin
    md   :=2;
    d[0] :=JWhich;
    if JDir >= 0 then
        fun  :=1
    else
        fun  := -1;
    Mstep(md,d,fun,flag);
    if JWhich = 0 then
       NCrystal := NCrystal + fun
    else
       NGrating := NGrating + fun;
  end;
                            {proc  jogL}
 Procedure Jog20L;
   var
     fun        :  longint;
   begin
     md :=4;
     d[0] := J20Which;
    if J20Dir >= 0 then
        fun  :=20
    else
        fun  := -20;
     Mstep(md,d,fun,flag);
     if J20Which = 0 then
        NCrystal  := NCrystal + fun
     else
        NGrating := NGrating + fun;
   end;
                      {proc  jog20L}
 Procedure TurnLMotor;
   var
     DelayPeriod : longint;
     DelayT,i,n : word;
   Begin
      md :=4;
      d[0] :=TMWhich;
      if TMWhich =0 then
        begin
          d[1] :=NcSpeed;
          NCrystal := NCrystal + TurnStep;
        end
      else
        begin
          d[1] :=NgSpeed;
          NGrating  := NGrating + TurnStep;
        end;
      MStep(md,d,TurnStep,TMFlag);

      Delay(500);
      DelayPeriod := Round ( Abs(TurnStep) * d[1] / 12.5 );
                           { MOTOR RATE (per milisec) = 12.5 / speed ! }
      if DelayPeriod < 5000 then
          begin
            DelayT := DelayPeriod;
            Delay(DelayT);
          end
      else
          begin
```

```
              n := Trunc(DelayPeriod / 5000) + 1;
              for i := 1 to n do
                 Delay(5000);
           end;
                           { Wait until the motor stops }
    End;
          { Proc TurnLMotor }
  Procedure InitHMotor;
   Begin
     d[1] :=  255;     {start rate divider      255--->20}
     d[2] :=  100;     {high speed run rate divider 255--->20}
     d[3] :=  200;     {acceleration/deceleration steps  1---->65535}
     d[4] :=  2;       { motor phase :1=three phase 2=four phase 3=five
phase}
     d[5] :=  0;       {excitation  : 0=full step   1=half step}
     d[6] :=  0;       {logic parity data: 0= inverted  1=positive true}
     d[7] :=  0;       {clock source : 0=internal 1=100kHz+ divider
2=external}
     d[8] :=  0;       {switching at standstill 0=off   1=on}
     d[9] :=  HMotorAddress;     { high motor board base address }
     md   :=  12;            { initialization ! }

     d[0] := IMWhich;
     MStep2(md,d,stp,InitFlag);
   end;
         { proc initHmotor }
 Procedure JogH;
  var
    fun ,realfun : longint;
  begin
     md:=2;
     d[0] :=JWhich;
     if JDir >= 0 then
         fun  :=1
     else
         fun  := -1;
     realfun  := -1 * fun;
     Mstep2(md,d,realfun,flag);
     if JWhich = 0 then
        NEtalon := NEtalon + fun
  end;
                          { proc  jogH }
  Procedure Jog20H;
    var
     fun,realfun:  longint;
    begin
      md :=4;
      d[0] :=J20Which;
     if J20Dir >= 0 then
         fun  :=20
     else
         fun  := -20;
      realfun := -1 * fun;
      Mstep2(md,d,realfun,flag);
      if J20Which = 0 then
        NEtalon := NEtalon + fun
    end;
           { proc  jog20H }
  Procedure TurnHMotor;
    var
      DelayPeriod,realstep : longint;
      DelayT,i,n : word;
    Begin
```

```
       md :=4;
       d[0] :=TMWhich;
       if TMWhich =0 then
         begin
           d[1] :=NeSpeed;
           NEtalon := NEtalon + TurnStep;
         end
       else
         begin
           d[1] :=NeSpeed;
         end;
       RealStep := -1 * TurnStep;
       MStep2(md,d,RealStep,TMFlag);

       Delay(500);
       DelayPeriod := Round ( Abs(TurnStep) * d[1] / 12.5 );
                         { MOTOR RATE (per milisec) = 12.5 / speed ! }
       if DelayPeriod < 5000 then
           begin
             DelayT := DelayPeriod;
             Delay(DelayT);
           end
       else
           begin
              n := Trunc(DelayPeriod / 5000) + 1;
              for i := 1 to n do
                 Delay(5000);
           end;         { Wait until the motor stops }
    End;
          { Proc TurnHMotor }
  Procedure GoPZT;
      Const
          Max_Buffer = 1000;
      var
          Board_Num, Err_Code, dataval, chanlo : integer;
      begin
          board_num :=0; chanlo :=0; err_code := 0;
          dataval := Round( DAVolts /5.0  * 4095.0 );
          if dataval < 0 then
               dataval := 0
          else if dataval > 4095 then
               dataval :=4095;
          d16_aous(board_num,chanlo,dataval,err_code);
          if err_code = 0 then
             GPZTFlag := True
          else
             GPZTFlag := False;
          Delay(500);
      end;
              { proc GoPZT }
end.                          { unit Util }
```

GDATARUT.PAS:

```
{***********************************}
{ The procedures that sets up the   }
{ interrupts and the interrupt      }
{ procedures themselves             }
{***********************************}
 Unit GDataRut;
 {$N+,D+,F+}
 Interface
   uses
     Dos,Crt,Graph,FileVar,CGlobal,Util,SubRut,Tp4d16;
```

```
  procedure GetTCheck(num,channel,trig,GTCShots,GTCNData:integer;
                    var Err:integer);
  procedure GetData(num,channello,channelhi,trig,GDShots,
                  GDNData:integer;var  Err: integer);
  procedure StopScan;
Implementation
  var
    fcount,frep,number,trigway,trep : integer;
    xxx : longint;
    CHF : integer;
procedure intHandler0; interrupt;
  var
    k,flag  : integer;
  begin
    xxx := xxx+ 1;
  with d16_gv[number] do
    begin
      if  xxx > 1 then
            begin
                 FileData[0,1]:=((swap(port[ain_hb]) or Port[ain_lb])
                                  shr 4) -vofs  + FileData[0,1];

                 Port[d16_stat]:=0;
                 Port[cntr_enable] :=0;
                 For  k:=2 to frep do
                    begin
                      port[mux_ctrl]:=( CHF shl 4) or CHF;
                      port[ain_lb]:=$FF;
                      repeat until ((port[d16_stat] and $80) <> $80);
                      FileData[0,k]:=((swap(port[ain_hb]) or
                          Port[ain_lb])  shr 4) -vofs + FileData[0,k];
                    end;
                                        { take the acouistic data }
                    Port[mux_ctrl] := ( CHF shl 4 ) or CHF;
                    Port[cntr_enable]:=2;
                    NPresentRep :=NPresentRep+1;

                    if  NPresentRep=fcount  then
                       begin
                          { Following: Plot the Data point!}
                          SetViewPort(10,DTop,GetMaxX-10,DBot,ClipOn);
                          OPODmaxMin[1,0] :=FileData[0,1]/ fcount
                                           -OOffset[CHF];
                          OPODmaxMin[2,0] :=FileData[0,1]/ fcount
                                           -OOffset[CHF];
                          For K:=1 to frep do
                            begin
                               FileData[0,k]:=FileData[0,k]/fcount
                                             - OOffset[CHF];
                               if FileData[0,k] > OPODmaxMin[2,0] then
                                 OPODMaxMin[2,0] := FileData[0,k]
                               else if FileData[0,k] < OPODMaxMin[1,0]
                                   then
                                 OPODMaxMin[1,0] := FileData[0,k];

                               PutPixel(Round((300.0 * (2*k-1))/ frep),
                                  Round(DZero-DTop-WindowRatio[1] *
                              (FileData[0,k]-WindowOffset[1])),Yellow);
                            end;

                            Port[D16_ctrl] :=0;
                            Port[d16_stat] :=0;
```

```
                             SetIntVec( 8+ intlev, oldvec);
                             port[$21] :=imr;
                             port[cntr_enable] :=0;
                             intflg :=0;

                             SetViewPort(5,67,95,77,ClipOn);
                             ClearViewPort;
                             SetColor(LightGreen);
                             OutTextXY(45,5,'NotScanning');
                             SetViewPort(10,103,629,147,ClipOn);

                             Scanning:=False;
                             NPresentPoint := NDataPoints;
                        end;   { NPresentRep = fcount }
             end;                 { if xxx  1 }
       Port[d16_stat] :=0;
       Port[$20] :=$20;
     end;       { with d16_gv }
  end;
                         { proc inthandler0 }

procedure intHandler1; interrupt;
   type
     TValueType   = Array[0..5] of integer;
     DrawDataType = Array[0..NMax] of integer;
     TOFType = Array[1..3] of Real;
   var
     tvalue         : TValueType;
     TOF1,TOF2      : TOFType;
     DrawData       : DrawDataType;
     i,j,Flag,flag1,flag2,Junk: integer;
     NM,NC,NG       : Longint;
     aa,bb,cc       : SS;
     ww,bing      : Boolean;
   begin
     xxx := xxx + 1;
     with d16_gv[number] do
        begin
          if xxx > 1 then
             begin
                tvalue[0]:=((swap(port[ain_hb]) or Port[ain_lb])
                       shr 4) -vofs;
                Port[d16_stat]:=0;
                Port[cntr_enable] :=0;
                port[mux_ctrl]:=( 6 shl 4) or 6;
                port[ain_lb]:=$FF;
                repeat until ((port[d16_stat] and $80) <> $80);
                Junk :=  ((swap(port[ain_hb]) or
                            Port[ain_lb])  shr 4) -vofs;
                     { Software read on channel 6:"buffer" reading }
                For   i:= 1 to NMax  do
                  begin
                    port[mux_ctrl]:=( i shl 4) or i;
                    port[ain_lb]:=$FF;
                    repeat until ((port[d16_stat] and $80) <> $80);
                    tvalue[i]:=((swap(port[ain_hb]) or Port[ain_lb])
                              shr 4) -vofs;
                  end;                            { take the data ! }

                Port[mux_ctrl] := 0;
                Port[cntr_enable]:=2;
                NPresentRep := NPresentRep + 1;
                if NPresentRep = fcount + 1 then
   begin
     trep :=trep+1;
     NPresentRep :=NPresentRep -fcount;
   end;

For i:=0 to NMax do
   FileData[i,trep] :=FileData[i,trep]+tvalue[i];

if  NPresentRep=fcount  then
   begin
      port[d16_stat] :=0;
      FileData[(NMax + 1),trep] :=CurrentWN;
      {Get the TOF integral from the TOF computer: }
      if not AcouOnly then
      begin
        GetFromTOF(bb,50.0);
        aa :=’’; cc :=’’;
        i :=2 ; j := 1;
        if ( bb[1] = ’$’) and ( bb[2] =’Y’ ) then
        repeat
           i := i + 1;
           While  bb[i] <>’X’ do
              begin
                 aa := aa + bb[i];
                 i  := i + 1;
              end;
           Val(aa,TOF1[j], Flag1);
           aa :=’’;
           i := i + 1;

           While ( bb[i] <>’Y’ ) and (bb[i] <>’@’) do
              begin
                  cc := cc + bb[i];
                  i  := i + 1;
              end;
              Val(cc,TOF2[j],flag2);
              cc :=’ ’;
              if (flag1 <> 0 ) or (flag2 <> 0 ) then
                 TOF1[j] := 0 - 200000
              else if Abs( TOF1[j] - TOF2[j] ) > 0.01
              then
                 TOF1[j] := 0 - 100000
              else
                 TOF1[j] := TOF1[j] / fcount ;
              j := j + 1;
           until j > 3 ;      { if bb[1]=’$’ }

           bb :=’’;
           for j:= 1 to 3 do
              begin
                 Str(TOF1[j]:14:5,aa);
                 bb :=bb +’ ’+aa;
              end;
           WriteResp(bb);
      end;    { not AcouOnly }
      {The data:}

      For i:=1 to NMax do
   FileData[i,trep]:=FileData[i,trep]/fcount-OOffSet[i];
      if not AcouOnly then
         begin
             FileData[2,trep] := FileData[0,trep]/fcount
                                - OOffSet[0];
```

```
       FileData[0,trep] := TOF1[1] ;
       FileData[4,trep] := TOF1[2] ;
       FileData[5,trep] := TOF1[3] ;
   end
else
    begin
        FileData[0,trep] :=FileData[0,trep]/fcount
         - ooffSet[0];
        FileData[4,trep] :=-1.0;
        FileData[5,trep] :=-1.0;
    end;

    {Move to new position:}
if Resolution =’Low’ then
   begin
        CurrentWN := ScanWNFirst+ trep*ScanWNStep;
        NM := NGWN(CurrentWN);
        NC := NCNG(NGrating);
        TurnLMotor(1,NM-NGrating,flag);
        NM := NCNG(NGrating);
        TurnLMotor(0,NM-NC,flag);
        KeepTrackL(ww);
   end
else if Resolution =’Med’ then
   begin
    if Smode =’Full’ then
      begin
        CurrentWN := ScanWNFirst+ trep*ScanWNStep;
        NG := NGWN(RoughWN);
        NC := NCNG(NGrating);
        RoughWN := CurrentWN;
        NM := NGWN(RoughWN);
        TurnLMotor(1,NM-NG,flag);
        NM := NCNG(NGrating);
        TurnLMotor(0,NM-NC,flag);
        NM :=NEWN(CurrentWN);
        { stay within 0.1 - 0.9 * EMax }
        if (NM >= 0.1 * EMax) and
              (NM <= 0.9 * EMax) then
          begin
            TurnHMotor(0,NM-NEtalon,flag);
            KeepTrackM(ww);
          end
        else if NM < 0.1 * EMax then
          Repeat
             OrderN := OrderN - 1;
             if NEWN(CurrentWN) <= 0.9 * EMax then
                begin
                   NM :=NEWN(CurrentWN) - NEtalon;
                   TurnHMotor(0,NM,Flag);
                   KeepTrackM(ww);
                   bing := False;
                end
             else
                begin
                  OrderN := OrderN + 1;
                  bing := True;
                end;
          until  bing
        else if NM > 0.9 * EMax then
          Repeat
             OrderN := OrderN + 1;
             if NEWN(CurrentWN) >= 0.1 * EMax then
```

```
                begin
                   NM :=NEWN(CurrentWN) - NEtalon;
                   TurnHMotor(0,NM,Flag);
                   KeepTrackM(ww);
                   bing := False;
                end
             else
                begin
                  OrderN := OrderN - 1;
                  bing := True;
                end;
          until  bing;
     end
   else  if SMode ='Etalon' then
     begin
         TurnHMotor(0,EtaStep,flag);
         KeepTrackM(ww);
     end;
end
else {'High'}
   begin
      CurrentWN:= ScanWNFirst + trep * ScanWNStep;
      NM :=NEWN(CurrentWN);
      { stay within 0.1 - 0.9 * EMax }
      if (NM >=0.1*EMax) and (NM <=0.9*EMax) then
          begin
            TurnHMotor(0,NM-NEtalon,flag);
            KeepTrackH(ww);
          end
      else if NM < 0.1 * EMax then
          Repeat
             OrderN := OrderN - 1;
             if NEWN(CurrentWN) <= 0.9 * EMax then
                begin
                   NM :=NEWN(CurrentWN) - NEtalon;
                   TurnHMotor(0,NM,Flag);
                   KeepTrackH(ww);
                   bing := False;
                end
             else
                begin
                  OrderN := OrderN + 1;
                  bing := True;
                end;
          until  bing
      else if NM > 0.9 * EMax then
          Repeat
             OrderN := OrderN + 1;
             if NEWN(CurrentWN) >= 0.1 * EMax then
                begin
                   NM :=NEWN(CurrentWN) - NEtalon;
                   TurnHMotor(0,NM,Flag);
                   KeepTrackH(ww);
                   bing := False;
                end
             else
                begin
                  OrderN := OrderN - 1;
                  bing := True;
                end;
          until  bing;
      DAVolts :=DAVolts+(dPZTdWN(CurrentWN)/
        PZTGain  * ScanWNStep;
```

```
               GoPZT(ww);
               KeepTrackH(ww);
            end;

         { Following: Plot the Data point!}
         if trep = 1 then
             for i := 0 to NMax do
                 begin
                    OPODMaxMin[1,i] := FileData[i,trep];
                    OPODMaxMin[2,i] := FileData[i,trep];
                 end;

         For i := 0 to NMax do
             if FileData[i,trep] > OPODMaxMin[2,i]  then
                 OPODMaxMin[2,i] := FileData[i,trep]
             else if FileData[i,trep]<OPODMaxMin[1,i] then
                 OPODMaxmin[1,i] := FileData[i,trep];

         DrawData[0] := Round(DZero-DTop-WindowRatio[1] *
                    (FileData[0,trep] - WindowOffset[1]) );
         DrawData[1] := Round(PZero-PTop-WindowRatio[2] *
                    (FileData[1,trep] - WindowOffset[2]) );
         DrawData[2] := Round(RZero-RTop-WindowRatio[3] *
                     (FileData[2,trep] - WindowOffset[3]));
         SetViewPort(10,DTop,629,DBot,ClipOn);
         PutPixel( Round( (300.0 * (2 * trep -1) ) /frep ),
                  Drawdata[0],Yellow);
         SetViewPort(10,PTop,629,PBot,ClipOn);
         PutPixel( Round( (300.0 * (2 * trep -1) )/frep ) ,
                   Drawdata[1],Yellow);
         SetViewPort(10,RTop,629,RBot,ClipOn);
         PutPixel( Round( (300.0 * (2 * trep -1) ) /frep) ,
                   Drawdata[2],Yellow);
         SetViewPort(10,103,629,147,ClipOn);
         NPresentPoint :=trep;
      end;      { if NPresentRep = fcount ! }

      if trep =frep+1  then
         begin
           Port[D16_ctrl] :=0;
           Port[d16_stat] :=0;
           SetIntVec( 8+ intlev, oldvec);
           Port[$21] :=imr;
           Port[cntr_enable] :=0;
           intflg :=0;

           SetViewPort(5,67,95,77,ClipOn);
           ClearViewPort;
           SetColor(LightGreen);
           OutTextXY(45,5,'NotScanning');
           SetViewPort(10,103,629,147,ClipOn);
           Scanning:=False;
         end; { if trep = frep + 1 ! }
end;                        { if xxx > 1 }

if ( ( NPresentRep =fcount ) and ( trep < frep ) )
      or ( xxx = 1 ) then
   begin
      if not AcouOnly then
         begin
            aa := '$Start!@';
            ww :=True;
            SendToTOF(aa,bb,ww,5.0);
```

```
                      end;
                end;
                Port[d16_stat] :=0;
                Port[$20] :=$20;
        end;          { with }
 end;
               { proc inthandler1 }

procedure GetTCheck;
  var
    i,j,temp  : integer;
    WaitTime  : Word;
  begin
    for j:= 1 to GTCNData  do
      FileData[0,j] :=0;

      fcount:=GTCShots;
      frep  :=GTCNData;
      number:= num;
      trigway:=trig;
      xxx :=0;
      NPresentRep :=0;
      trep:=1;
      err :=0;
      CHF :=channel;
      D16_set_mux(num,channel,channel,err);

      with d16_gv[num] do
         begin
           if  ftt<>1 then
             err:=1
           else
             if (dmaflg or intflg) =1 then
                err :=20
             else
                begin
                   Scanning :=True;
                   d16_set_pit_ratio(num,$ffff,$ffff,err);
                   port[d16_ctrl] :=0;
                   Port[d16_stat] :=0;

                   Port[Cntr_enable] :=2;
                   d16_set_cntr0_config(num,1,err);
                   WaitTime := Round(TDelay / 10.0 );
                   d16_cset0(num,WaitTime,err);
                       {set up the lOOKHz clock delay / external pulse }
                   imr :=port[$21];
                   GetIntVec(8+INTLEV, oldvec);
                   setIntvec(8+intlev,@inthandler0);
                   port[$21] :=((1 shl intlev) Xor $ff) and imr;
                        {set up interrupt vector}
                   port[d16_stat] :=0;
                   temp:=port[mux_ctrl];
                   Port[mux_ctrl]:=temp;
                   intflg :=1;
                   Port[d16_ctrl] :=((intlev shl 4) or $82) or trig;
                end;
         end;   { with }
    end;
                             { proc  GetTCheck }
Procedure GetData;
   var
     i,j,temp,flag  : integer;
  WaitTime  : Word;
  NM,NC,NG  : Longint;
  cc, dd    : SS;
  KTOk      : Boolean;
  deltaP,WNx : real;
begin
  {Move to the scan start configuration:}
  if Resolution ='Low' then
     begin
        CurrentWN := ScanWNFirst;
        NM :=NGWN(CurrentWN);
        NC :=NCNG(NGrating);
        TurnLmotor(1,NM - NGrating,flag);
        NM :=NCNG(NGrating);
        TurnLmotor(0,NM-NC,flag);
        KeepTrackL(KTOk);
     end
  else if Resolution = 'Med' then
     begin
          if Smode ='Full' then
             begin
                NG  := NGWN(RoughWN);
                RoughWN :=ScanWNFirst;
                NM := NGWN(RoughWN);
                NC := NCNG(NGrating);
                TurnLMotor(1,NM-NG,flag);
                NM := NCNG(NGrating);
                TurnLMotor(0,NM-NC,flag);
                CurrentWN :=ScanWNFirst;
                ONWN(CurrentWN,OrderN);
                NM := NEWN(CurrentWN);
                TurnHMotor(0,NM-NEtalon,flag);
                KeepTrackM(KTOk);
                {Start from 0.1 * EMax to 0.9 * EMax }
                if NEtalon > 0.9 * EMax then
                  repeat
                    OrderN := OrderN + 1;
                    NM :=NEWN(CurrentWN) - NEtalon;
                    TurnHMotor(0,NM,Flag);
                    KeepTrackM(KTOk);
                  until NEtalon <= 0.9 * EMax
                else if NEtalon < 0.1 * EMax then
                  repeat
                    OrderN := OrderN - 1;
                    NM :=NEWN(CurrentWN) - NEtalon;
                    TurnHMotor(0,NM,Flag);
                    KeepTrackM(KTOk);
                  until NEtalon >= 0.1 * EMax;
             end
          else if SMode ='Etalon' then
             begin
               TurnHMotor(0,EtaFirst-NEtalon,Flag);
               KeepTrackM(KTOk);
             end;
     end
  else {'High'}
     begin
            WNx := CurrentWN;
            NG   := NGWN(RoughWN);
            RoughWN :=ScanWNFirst;
            NM := NGWN(RoughWN);
            NC := NCNG(NGrating);
            TurnLMotor(1,NM-NG,flag);
          NM := NCNG(NGrating);
          TurnLMotor(0,NM-NC,flag);
          CurrentWN :=ScanWNFirst;
          ONWN(CurrentWN,OrderN);
          NM := NEWN(CurrentWN);
          TurnHMotor(0,NM-NEtalon,flag);
          KeepTrackH(KTOk);
          {Start from 0.1 * EMax to 0.9 * EMax }
          if NEtalon > 0.9 * EMax then
              repeat
                  OrderN := OrderN + 1;
                  NM :=NEWN(CurrentWN) - NEtalon;
                  TurnHMotor(0,NM,Flag);
                  KeepTrackH(KTOk);
              until NEtalon <= 0.9 * EMax
          else if NEtalon < 0.1 * EMax then
              repeat
                  OrderN := OrderN - 1;
                  NM :=NEWN(CurrentWN) - NEtalon;
                  TurnHMotor(0,NM,Flag);
                  KeepTrackH(KTOk);
              until NEtalon >= 0.1 * EMax;
          deltaP := (dPZTdWN(CurrentWN)/PZTGain) * (CurrentWN-WNx);
          DAVolts := DAVolts + deltaP;
          GoPZT(KTOK);
          KeepTrackH(KTOK);
   end;

if not Acouonly then
   Begin
      Str(NDataPoints:5,cc);  cc :=’$’+cc+’@’;
      SendToTOF(cc,dd,False,5.0);
      delay(500);
      Str(ScanRep:8,cc); cc :=’$’+cc+’@’;
      SendToTOF(cc,dd,False,5.0);
      delay(500);
      cc :=’$3@’;
      SendToTOF(cc,dd,False,5.0);
      Delay(3500);
   end;           { Send Para to TOF computer }
for j:=1 to GDNData  do
   for  i:=0 to NMax do FileData[i,j] :=0;

fcount:=GDShots;
frep  :=GDNData;
number:= num;
trigway:=trig;
xxx := 0 ;
NPresentRep :=0;
trep:=1;
err :=0;

D16_set_mux(num,channello,channello,err);
with d16_gv[num] do
   begin
      if  ftt<>1 then
          err:=1
      else
          if (dmaflg or intflg) =1 then
             err :=20
          else
            begin
               Scanning :=True;
```

```
                    d16_set_pit_ratio(num,$ffff,$ffff,err);
                    port[d16_ctrl] :=0;
                    Port[d16_stat] :=0;

                    Port[Cntr_enable] :=2;
                    d16_set_cntr0_config(num,1,err);
                    WaitTime := Round(TDelay / 10.0);
                    d16_cset0(num,WaitTime,err);
                     {set up the 100KHz clock delay and external pulse }
                    imr :=port[$21];
                    GetIntVec(8+INTLEV, oldvec);
                    SetIntvec(8+intlev,@inthandler1);
                    port[$21] :=((1 shl intlev) Xor $ff) and imr;
                                     {set up interrupt vector}
                    port[d16_stat] :=0;
                    temp:=port[mux_ctrl];
                    Port[mux_ctrl]:=temp;
                    intflg :=1;
                    Port[d16_ctrl] :=((intlev shl 4) or $82) or trig;
                 end;
        end;   { with }
     end;                          { proc  GetData }

  Procedure StopScan;
    Begin
       with d16_gv[0] do
         begin
            Port[D16_Ctrl] :=0;
            Port[D16_Stat] :=0;
            SetIntVec(8+Intlev, oldvec);
            Port[$21] :=imr;
            Port[Cntr_enable]:=0;
            intflg:=0;
            Port[D16_Stat]:=0;
            Port[$20]:=$20;

            SetViewPort(5,67,95,77,ClipOn);
            ClearViewPort;
            SetColor(lightGreen);
            OutTextXY(45,5,'NotScanning');
         end;           { with d16 gv[board num] }

       SetViewPort(10,103,629,147,ClipOn);
       ClearViewPort;
       Scanning := False;
    End;            { Proc StopScan }
end.                          {Unit GDataRut}
```

SUBRUT.PAS:

```
{********************************************}
{Commomly Called  SubRoutines for L,M,HCollect}
{********************************************}
Unit SubRut;
{$N+,D+,F+}
INTERFACE
  uses
    Dos,Crt,Graph,tp4d16,FileVar,CGlobal,Util;
{Clear Drawing Screen: }
  Procedure  ClearDrawings;
{Board Initialization: }
  Procedure  DAS( Var ErrFlag : integer);
  Procedure  InitStuff ( var InitDone : Boolean );
{Calculations needed for scanning schronization and calibration: }
```

```
  Function   NCNG ( NG : LongInt) : LongInt;
  Procedure  Spline;
  Function   NGWN ( WN : Real )   : LongInt;
  Function   WNNG ( NG : LongInt) : Real;
  Procedure  ONWN ( WN : Real; var ON : integer );
  Function   WNNE ( NE : Longint ) : Real;
  Function   NEWN ( WN : Real )   : LongInt;
  Function   dPZTdWN( WN : Real )   : Real;
{Status Update: }
  Procedure  KeepTrackL (var KTFlag: Boolean);
  Procedure  KeepTrackM (var KTMFlag: Boolean);
  Procedure  KeepTrackH (var KTHFlag: Boolean);
{I/O Handlers: }
  Procedure  GetNum (var GNNum: longint; var GNVal:Real;GNFlag:Boolean);
  Procedure  Getstr (var  GTstr  :  SS );
  Procedure  GetStartPosition;
{Movements: }
  Procedure  MoveLMotor ( MWhich : integer; var MMFlag   : Boolean);
  Procedure  MoveHMotor ( MWhich : integer; var MMFlag   : Boolean);
  Procedure  MovePZT( var MPZTFlag : Boolean);
  Procedure  MoveToWN( var MTWFlag: Boolean);
  Procedure  JumpOrN;
{Data Display: }
  Procedure  Zoom123;
  Procedure  AutoZoom;
{RS232: }
  Procedure  WriteResp( WBuf  : SS);
  Procedure  GetFromTOF(var GFTBuf  : SS; WTime : Real ) ;
  Procedure SendToTOF(varOutBuf,InBuf:SS;Oneway:Boolean;WaitTime:Real);
  Procedure  ToTOF;
{Misc. Procedures: }
  Procedure  AcouYesNo;
  Procedure  CalCG;
  Procedure  TChkDial(var Chan: integer; var TDFlag: Boolean);
  Procedure  ScanDial(var SDFlag: Boolean);
  Procedure  DataSave;
  Procedure  QuitCollect(var QCKey: Char);
IMPLEMENTATION
  Const
     CalN  = 27;
  Type
     CubType = Array[1..CalN] of real;
     CalType = Array[1..CalN,1..2] of real;
  Const
     CalRes: CalType =
          ((2488.6,5624),(2537.7,6224),(2595.6,6844),(2672.7,7644),
           (2765.7,8644),(2852.6,9644),(2934.8,10644),(3014.9,11644),
           (3064.5,12144),(3143.3,13144),(3212.8,14144),(3288.9,15144),
           (3361.9,16144),(3395.2,16644),(3434.2,17144),(3469.5,17644),
           (3549.0,19144),(3650.6,20644),(3741.8,22144),(3831.9,23644),
           (3919.8,25144),(4002.3,26644),(4080.7,28144),(4166.8,29644),
           (4234.9,31144),(4302.5,32644),(4391.2,34644));
           { Calibration with Monochrometer,
              need to be redone periodically }
  Var
     CubRes : CubType;
(****************************PROCEDURES********************************)
   Procedure  ErrorBox (ErrNum: integer; EBResult : Char);
      Const
         EB1 ='Board Error! Error number is: ';
         EB2 ='Please fix the error.CONTINUE?<Y/N> ';
      var
         VertPosi  : integer;
```

```
        jstr2      : string[2];
    begin
        str(ErrNum:2,Jstr2);
        SetViewPort(10,103,629,147,ClipOn);
        Vertposi := VertStart;
        OutTextXY(GetMaxX div 2,VertPosi,EB1+Jstr2+'.');
        VertPosi := VertPosi + VertStep;
        OutTextXY(GetMaxX div 2,VertPosi,EB2);
        Repeat  EBResult := Readkey;
        Until EBResult in ['Y','y','N','n'];
        ClearViewPort;
    end;                    { Proc ErrorBox }

Procedure ClearDrawings;
  Var
    w1,w2,w3,w4,w5,w6 : string[9];
    x1,x2,x3  : real;
  Begin
      x1 := ( DZero - DTop ) / WindowRatio[1];
      x2 := ( PZero - PTop ) / WindowRatio[2];
      x3 := ( RZero - RTop ) / WindowRatio[3];
      Str(x1:9:4,w1);
      Str(x2:9:4,w2);
      Str(x3:9:4,w3);
      Str(WindowOffset[1]:9:3,w4);
      Str(WindowOffset[2]:9:3,w5);
      Str(WindowOffset[3]:9:3,w6);
      SetViewPort(0,153,GetMaxX,GetMaxY,ClipOn);
      ClearViewPort;
      SetViewPort(0,0,GetMaxX,GetMaxY,ClipOn);
      Rectangle(0,DTop-1,GetMaxX,DBot+1);
      Rectangle(0,PTop-1,GetMaxX,PBot+1);
      Rectangle(0,RTop-1,GetMaxX,RBot+1);
      SetViewPort(0,0,GetMaxX,GetMaxY,ClipOn);
      OutTextXY(175,DTop+6,'Data:  YScale='+w1+' Offset='+w4);
      OutTextXY(175,PTop+6,'Power: YScale='+w2+' Offset='+w5);
      OutTextXY(175,RTop+6,'Refer: YScale='+w3+' Offset='+w6);
      Line(10,DZero,GetMaxX-10,DZero);
      Line(10,PZero,GetMaxX-10,PZero);
      Line(10,RZero,GetMaxX-10,RZero);
                  {draw the Graphics Windows}
  end;                            {Proc  ClearDrawings}

Procedure DAS( var ErrFlag : integer);
  var
      board_num,base_adr,int_level,dma_level  : integer;
                                              {D16 Board Para}
  Begin
    board_num :=0; int_level:=DASIntLev; dma_level:=1;
    base_adr := DASAddress;
    d16_init(board_num,base_adr,int_level,dma_level,ErrFlag);
                 { initialize the DAS-16 board }
  End;                            { procedure DAS }

 Procedure InitStuff;
  Const
    IMB1 ='Initialize Motor and DAS board.Make sure switch is on ';
    IMB2 ='Otherwise scan synchronization will be messed up !    ';
    IMB3 ='PRESS <Y/N> FOR INITIALIZE.                           ';
  Var
    Vertposi      : integer;
    ch,resp       : Char;
    IMFlag        : integer;
  InitOK       : Boolean;
begin
      Sound(2000);Delay(1000);NoSound;
      SetViewPort(10,103,629,147,ClipOn);
      VertPosi := VertStart;
      SetColor(Red);
      OutTextXY(GetMaxX div 2,VertPosi,IMB1);
      VertPosi :=VertPosi + VertStep;
      OutTextXY(GetMaxX div 2,VertPosi,IMB2);
      VertPosi :=VertPosi + VertStep;
      OutTextXY(GetMaxX div 2,VertPosi,IMB3);
      VertPosi :=VertPosi + VertStep;
      SetColor(LightGreen);                 {write dialog texts}
      Repeat ch :=Readkey until ch in [’Y’,’y’,’N’,’n’];
      ClearViewPort;
      if ch in [’Y’,’y’] then
         begin
            InitOK := True;
            if initOK  then
               begin
                 InitLMotor(0,IMFlag);
                 if IMFlag <> 0 then InitOK := False;
               end;
            if initOk then
               begin
                 InitLMotor(1,IMFlag);
                 if IMFlag <> 0 then InitOK := False;
               end;
            if initOk then
               begin
                 InitHMotor(0,IMFlag);
                 if IMFlag <> 0 then InitOK := False;
               end;
            if initOk then
               begin
                 InitHMotor(1,IMFlag);
                 if IMFlag <> 0 then InitOK := False;
               end;
            if initOk then
               begin
                  ClearViewPort;
                  OutTextXY( 320, 10,’ Motors are initialized.’);
                  Delay(1500);
                  ClearViewPort;
                  DAS(IMFlag);
                  if IMFlag <> 0 then InitOk := False;
               end;
            if initOk then
               begin
                   ClearViewPort;
                   OutTextXY( 320, 10,’DAS is initialized.’);
                   Delay(1500);
                   ClearViewPort;
               end
            else
               begin
                   ErrorBox(IMFlag,Resp);
               end;
         end   { if ch in [ ’Y’,’y’] }
    else
         InitDone :=False;

    if not InitDone then
```

```
              begin
                 ClearViewPort;
                 OutTextXY( 320, 10,'Initialization not done.');
                 Delay(1500);
                 ClearViewPort;
              end;
       end;                         { proc InitStuff }

     Function NCNG ;
       Var
         X :real;
       begin
         X :=    FitSYNData[1,1] + FitSYNData[1,2] * NG +
                 FitSYNData[1,3] * NG * NG+
                 FitSYNData[1,4] * NG * NG * NG ;
         NCNG := Round(x);
       end;

   PROCEDURE spline;             { do a natural spline }
      Var
         i,k: integer;
         p,qn,sig,un: real;
         x,y : CubType;
         u: ^CubType;
      BEGIN
         new(u);
         for i := 1 to CalN do
             begin
                 x[i] := CalRes[i,1];
                 y[i] := CalRes[i,2];
             end;
         CubRes[1] := 0.0;
         u^[1] := 0.0;
               { natural }
         FOR i := 2 TO CalN-1 DO
            BEGIN
               sig := (x[i]-x[i-1])/(x[i+1]-x[i-1]);
               p := sig*CubRes[i-1]+2.0;
               CubRes[i] := (sig-1.0)/p;
               u^[i] := (y[i+1]-y[i])/(x[i+1]-x[i])
                          -(y[i]-y[i-1])/(x[i]-x[i-1]);
               u^[i] := (6.0*u^[i]/(x[i+1]-x[i-1])-sig*u^[i-1])/p
            END;
         qn := 0.0;
         un := 0.0;
                 { natural }
         CubRes[CalN] := (un-qn*u^[CalN-1])/(qn*CubRes[CalN-1]+1.0);
         FOR k := CalN-1 DOWNTO 1 DO
             CubRes[k] := CubRes[k]*CubRes[k+1]+u^[k];
         dispose(u)
      END;

   Function splint( x: real) : real;
      VAR
         klo,khi,k: integer;
         h,b,a: real;
      BEGIN
         klo := 1;
         khi := CalN;
         WHILE khi-klo > 1 DO
            BEGIN
               k := (khi+klo) DIV 2;
               IF CalRes[k,1] > x THEN khi := k ELSE klo := k
       END;
    h := CalRes[khi,1]-CalRes[klo,1];
    a := (CalRes[khi,1]-x)/h;
    b := (x-CalRes[klo,1])/h;
    Splint := a*CalRes[klo,2]+b*CalRes[khi,2]+
       ((a*a*a-a)*CubRes[klo]+(b*b*b-b)*CubRes[khi])*(h*h)/6.0
 END;

Function NGWN;
  begin
    NGWN := Round(Splint(WN));
  end;

Function WNNG;
  Const
    BigWN   = 4700;
    SmallWN = 2400;
  var
    i           : integer;
    TempNG      : LongInt;
    x1,x2,TempWN: real;
    Bingo       : Boolean;
  begin
    x1 := SmallWN;
    x2 := BigWN;
    Bingo := True;
    if ( NG < Splint (x1) )  then
        WNNG := SmallWN
    else if ( NG > Splint (x2) ) then
        WNNG := BigWN
    else
        Bingo := False;
    if not Bingo then
       begin
          Repeat
             TempWN := ( x1 + x2 ) / 2.0;
             if ( NG < Splint( TempWN ) ) then
                 x2 :=  TempWN
             else
                 x1 :=  TempWN;
          until ( ( x2 - x1 ) < 0.001 );
          Bingo := True;
          WNNG   := TempWN;
       end;
  end;                    { function WNNG }

Function ThetaE ( NE : longint )  : Real;
    var
       thetax : real;
    begin
       thetax :=  FitSYNData[2,1] + FitSYNData[2,2] * NE +
            FitSYNData[2,3] * NE * NE +
            FitSYNData[2,4] * NE * NE * NE ;
       ThetaE := 1.0 / IndexE  * thetaX;
     { approx. good! Sin(thetaE) = Sin(thetax)/indexE }
    end;                                        { function ThetaE }

Procedure ONWN;
    var
       i,j : integer;
       x,y,z,w   : real;
    begin
       x := PumpWN - WN;
       w := 2.0 * IndexE*ThickE *(1.0+CFactor)* x
              * Cos( ThetaE(NEtalon) );
       i := Trunc(w);
       y :=  i/( 2.0 *IndexE*ThickE *(1.0+CFactor)
              *Cos( ThetaE(NEtalon) ) );
       j := i + 1;
       z :=  j/( 2.0 *IndexE*ThickE *(1.0+CFactor)
                *Cos( ThetaE(NEtalon) ) );
       if Abs(y-x) < Abs(z-x) then
          ON := i
       else
          ON := j;
    end;                     { Procedure ONWN  }

Function WNNE;
    var
         theta , x : Real;
    begin
         theta := ThetaE (NE);
         x:= OrderN /( 2.0*IndexE*ThickE *(1.0+CFactor)* Cos(theta) );
         WNNE  := PumpWN - x;
    end;                         { function WNNE }

Function NEWN;
    var
       x1,x2,x3,y3  : real;
    begin
       x1 := 0;
       x2 := EMax;
       Repeat
          x3 := ( x1 + x2 )/ 2.0;
          y3 := WNNE(Round(x3));
          if  y3 < WN  then
              x2 := x3
          else
              x1 := x3;
       Until ( x2 - x1 < 0.1 ) ;
       NEWN := Round(x3);
    end;                     { function NEWN }

 Function dPZTdWN;
   var
      x  : Real;
   begin
      x := PumpWN - WN;
      dPZTdWN :=  LCavity * (1.0 + CHFactor)  / ( x  * dLdV );
   end;                       { Func dPZTdWN }

 Procedure KeepTrackL;
   var
       StringM  : String[7];
       StrCWN   : String[10];
   Begin
    SetViewPort(GetMaxX div 2 -200,57,GetMaxX div 2-140,67,ClipOn);
    ClearViewPort;
    SetViewPort(GetMaxX div 2 +250,57,GetMaxX div 2+310,67,ClipOn);
    ClearViewPort;
    SetViewPort(10,57,GetMaxX,67,ClipOn);
    Str(NCrystal,StringM);
    OutTextXY(getMaxX div 2 -180,5,StringM);
    Str(NGrating,StringM);
    OutTextXY(GetMaxX div 2 +280,5,StringM);
                             { Write out current Motor Position}
   SetViewPort(350,67,430,77,ClipOn); ClearViewPort;
   CurrentWN := WNNG(NGrating); Str(CurrentWN:10:4,strCWN);
   OutTextXY(40,5,strCWN);
                               { Write out current wavenumber }
  end;
               { Proc  KeepTrackL }

Procedure KeepTrackM;
  var
      RN     : integer;
      StrRON : String[2];
      StrON  : String[4];
      StringM: String[7];
      StrWN  : String[10];
  Begin
   SetViewPort(GetMaxX div 2 -200,57,GetMaxX div 2-140,67,ClipOn);
   ClearViewPort;
   SetViewPort(GetMaxX div 2 +250,57,GetMaxX div 2+310,67,ClipOn);
   ClearViewPort;
   SetViewPort(10,57,GetMaxX,67,ClipOn);
   Str(NCrystal,StringM);
   OutTextXY(GetMaxX div 2 -180,5,StringM);
   Str(NGrating,StringM);
   OutTextXY(GetMaxX div 2 +280,5,StringM);
                               { Write out current LMotor Position}

   SetViewPort(310,57,390,67,ClipOn); ClearViewPort;
   RoughWN := WNNG(NGrating); Str(RoughWN:9:4,strWN);
   OutTextXY(40,5,strWN);
                               { Write out rough wavenumber }
   Str(NEtalon,StringM);
   SetViewPort(190,67,250,77,ClipOn);
   ClearViewPort;
   OutTextXY(30,5,StringM);
                               { Write out current HMotor Position }
   CurrentWN :=WNNE(NEtalon);
   Str(CurrentWN:10:4,StrWN);
   SetViewPort(500,67,600,77,ClipOn);
   ClearViewPort;
   OutTextXY(50,5,StrWN);
                               { Write out current wavenumber }
   Str(OrderN,StrON);
   RN := OrderN-Trunc( 2.0*IndexE*ThickE*
             (1.0+CFactor)*(PumpWN -CurrentWN));
   Str(RN,strRON);
   SetViewPort(320,67,390,77,ClipOn);
   ClearViewPort;
   OutTextXY(35,5,StrON+’(’+StrRON+’)’);
                         { Write out etalon interference orders }
   SetViewPort(10,103,GetMaxX-10,147, ClipOn);
  end;                                  { Proc  KeepTrackM }

Procedure KeepTrackH;
  var
      StringM: String[7];
      StrWN : String[10];
      StrPZT : String[8];
  Begin
   SetViewPort(GetMaxX div 2 -200,57,GetMaxX div 2-140,67,ClipOn);
   ClearViewPort;
   SetViewPort(GetMaxX div 2 +250,57,GetMaxX div 2+310,67,ClipOn);
   ClearViewPort;
   SetViewPort(10,57,GetMaxX,67,ClipOn);
  Str(NCrystal,StringM);
  OutTextXY(getMaxX div 2 -180,5,StringM);
  Str(NGrating,StringM);
  OutTextXY(GetMaxX div 2 +280,5,StringM);
                               { Write out current LMotor Position}

  SetViewPort(310,57,390,67,ClipOn); ClearViewPort;
  RoughWN := WNNG(NGrating); Str(RoughWN:10:4,strWN);
  OutTextXY(40,5,strWN);
                               { Write out rough wavenumber }
  Str(NEtalon,StringM);
  SetViewPort(190,67,250,77,ClipOn);
  ClearViewPort;
  OutTextXY(30,5,StringM);
                               { Write out current HMotor Position}
  CurrentWN :=WNNE(NEtalon);
  Str(CurrentWN:10:4,StrWN);
  SetViewPort(345,67,435,77,ClipOn);
  ClearViewPort;
  OutTextXY(45,5,StrWN);
                               { Write out current wavenumber }
  Str(DAVolts:8:4,StrPZT);
  SetViewPort(560,67,630,77,ClipOn);
  ClearViewPort;
  OutTextXY(35,5,StrPZT);
                               { Write out current PZT Voltage }
  SetViewPort(10,103,GetMaxX-10,147, ClipOn);
 end;
                               { Proc  KeepTrackH }

Procedure  GetNum (var GNNum : longint; var GNVal : Real;
                       GNFlag: Boolean);
        { The Procedure that reads the input numbers }
var
 GNOk,GNReturn,GNBS    : Boolean;
 GNkey                 : Char;
 GNTemp                : String[8];
 GNstring              : String[10];
 GNCode,j,k            : Integer;
Begin
  j:= 1;
  GNstring :='          ';
  GNTEmp   :='        ';

  Repeat
     Repeat
        GNOk := False;
        GNkey :=ReadKey;
        GNOk := ( GNKey in  ['-','.','0'..'9'] );
        GNReturn := ( Ord(GNKey) = 13 );      { Carriage return }
        GNBS     := ( Ord(GNKey) =  8 );      { BackSpace }
     Until GNOk or GNReturn or GNBS;

     if GNOk then
       Begin
          GNTemp[j] := GNKey;
          OutText(GNTemp[j]);
          MoveRel(8,0);
          j := j + 1;
       End
     else if GNBS then
       Begin
          if j > 1 then
```

```
               begin
                  j:= j - 1;
                  Moverel(-8,0);
                  SetColor(DarkGray);
                  OutText(GNTemp[j]);
                  SetColor(LightGreen);
                  GNTemp[j] :=' ';
               end;
         End
       else     { GNReturn }
          begin
             K:=0;
             Repeat
                K:=K+ 1;
                if K <= (11-j) then
                    GNstring[k] :=' '
                else
                    GNstring[k] :=GNTemp[k-(11-j)];
             Until K=10;
             if  j > 1  then
                 begin
                   if GNFlag then
                         Val(GNstring, GNnum, GNCode)
                   else
                         Val(GNString, GNVal, GNCode);
                 end;
             if ( GNstring ='          ' ) or ( GNCode <> 0 )then
                 begin
                      GNVal := -1.01;
                      GNNum := -1;
                 end;
                        { ERROR ! return -1.01 and -1 ! }
          end;
    Until GNReturn;
 end;                            { proc GetNum }

Procedure Getstr ( var  GTstr  :  SS );
   var
      i,j  : integer;
      temp : SS;
      ch   : char;
   begin
       Temp :='';
       GTStr:='';
       i :=1;
       Repeat
           ch :=Readkey;
            if not ( ch in [ #13, #8 ] ) then
                begin
                  Temp[i] := ch;
                  i := i + 1;
                  OutText(Temp[i-1]);
                  MoveRel(8,0);
                end;
            if ord(ch) = 8 then
                 begin
                    i :=i -1;
                    SetColor(DarkGray);
                    MoveRel(-8,0);
                    OutText(Temp[i]);
                    SetColor(lightGreen);
                end;
          Until  ch in [ #13 ];
          if i > 1 then
                  for  j := 1 to i-1 do
                     GTStr:= GTStr + Temp[j]
          else
                  GTStr :='';
   end;                       { Proc GetStr }

 Procedure GetStartPosition;
  Const
   GSP1 = 'Input new motor positions:(Will mess up synchronization!)';
   GSP2 = 'NCrystal ? (-50000 to 50000)                             ';
   GSP3 = 'NGrating ? (-50000 to 50000)                             ';
   GSP4 = 'NEtalon  ? (-50000 to 50000)                             ';
 Var
   VertPosi      : integer;
   GSPFlag       : Boolean;
   JunkR         : Real;
   JunkI         : Longint;
  begin
     SetViewPort(10,103,629,147,ClipOn);
     VertPosi := VertStart;
     OutTextXY(GetMaxX div 2,VertPosi,GSP1);
     VertPosi := VertPosi + VertStep;

     OutTextXY(GetMaxX div 2,VertPosi,GSP2);
     MoveTo(500,Vertposi);
     GetNum(JunkI,JunkR,True);
     if junkI <> -1 then
         NCrystal := JunkI;
     VertPosi := VertPosi + VertStep;
     OutTextXY(GetMaxX div 2,VertPosi,GSP3);
     MoveTo (500,VertPosi);
     GetNum(JunkI,JunkR,True);
     if junkI <> -1 then
         NGrating := JunkI;

     ClearViewPort;
     VertPosi := VertStart;
     OutTextXY(GetMaxX div 2,VertPosi,GSP4);
     MoveTo(500,Vertposi);
     GetNum(JunkI,JunkR,True);
     if JunkI <> -1 then
            NEtalon := JunkI;
     ClearViewPort;
   end;                        {proc getstartposition}

Procedure MoveLMotor;
  var
    Movestep    : longint;
    Ch          : char;
    CheckFlag   : integer;
    Flag        : Boolean;
    i           : string[3];
  Procedure  GetMMStep (GMNum:integer;var
          GMMstep:longint;GMMnum:longint);
   const
     GMMS1A =' Move the crystal motor at constant speed             ';
     GMMS1B =' Move the grating motor at constant speed             ';
     GMMS2 = ' Please wait until motor stops to execute this command';
     GMMS3 = ' Enter the number of steps :                          ';
   Var
     VertPosi : integer;
     JunkR    : Real;
   begin
      SetViewPort(10,103,629,147,ClipOn);
      Vertposi :=VertStart;
      if GMNum = 1 then
         OutTextXY(GetMaxX div 2,VertPosi,GMMS1A)
      else
         OutTextXY(GetMaxX div 2,VertPosi,GMMS1B);
      VertPosi := VertPosi + VertStep;
      OutTextXY(GetMaxX div 2,VertPosi,GMMS2);
      VertPosi := VertPosi + VertStep;
      OutTextXY(GetMaxX div 2,VertPosi,GMMS3);
      MoveTo( 500,VertPosi);
      GetNum ( GMMstep,JunkR,True);
      if ( GMMstep + GMMnum  > CGMax )   or
         ( GMMstep + GMMnum  <  CGMin )  then
          GMMstep := 0;
       Delay (800);
       ClearViewPort;
    end;
           { SubRoutine GMMstep }
 Begin
   if MWhich = 0 then
       begin
         GetMMStep(1,MoveStep,NCrystal);
         if ( MoveStep <> 0 )  and ( MoveStep <> -1 )  then
            begin
               TurnLMotor(0,MoveStep,CheckFlag);
               if CheckFlag <> 0 then
                   ErrorBox(checkflag,ch)
               else
                   MMFlag :=True;
           end;
       end
   else
       begin
           GetMMStep(2,MoveStep,NGrating);
           if ( MoveStep <> 0 ) and ( MoveStep <> -1 ) then
              begin
                 TurnLMotor(1,MoveStep,CheckFlag);
                 if CheckFlag <> 0 then
                    ErrorBox(checkflag,ch)
                 else
                    MMFlag :=True;
             end;
        end;
end;                         {  proc MoveLMotor  }

Procedure MoveHMotor;
  var
    Movestep    : longint;
    Ch          : char;
    CheckFlag   : integer;
    Flag        : Boolean;

  Procedure  GetMMStep (var  GMMstep : longint; GMMnum : longint);
   const
     GMMS1 = ' Move the etalon motor at constant speed             ';
     GMMS2 = ' Please wait until motor stops to execute this command';
     GMMS3 = ' Enter the number of steps :                         ';
   Var
     VertPosi : integer;
     JunkR  :Real;
   begin
```

```
      SetViewport(10,103,629,147,ClipOn);
      Vertposi :=VertStart;
      OutTextXY(GetMaxX div 2,VertPosi,GMMS1);
      VertPosi := VertPosi + VertStep;
      OutTextXY(GetMaxX div 2,VertPosi,GMMS2);
      VertPosi := VertPosi + VertStep;
      OutTextXY(GetMaxX div 2,VertPosi,GMMS3);
      MoveTo( 500,VertPosi);
      GetNum ( GMMstep,JunkR,True);
      if ( GMMstep + GMMnum  >  EMax)   or
         (GMMstep  + GMMnum <   EMin)  then
          GMMstep := 0;
       Delay (800);
       ClearViewPort;
    end;
                    { SubRoutine GMMstep }
 Begin
    if MWhich = 0 then
       begin
         GetMMStep(MoveStep,NEtalon);
         CheckFlag :=0;
         if ( MoveStep <> 0 ) and ( MoveStep <> -1 ) then
            begin
              TurnHMotor(0,MoveStep,CheckFlag);
              if CheckFlag <> 0 then
                 ErrorBox( checkflag,ch)
              else
                 MMFlag :=True;
            end;
       end
end;                          {  proc MoveHMotor  }

 Procedure MovePZT;
     Const
        MPZT1='Change DAinput to PZT controller.                  ';
        MPZT2='Enter The new DAvoltage ( 0 to 5 v ):              ';
     Var
        X : Real; Y : Longint;
     Begin
        SetViewPort(10,103,629,147,Clipon);
        OutTextXY(320,vertStart,MPZT1);
        OutTextXY(320,vertStart+vertStep,MPZT2);
        MoveTo(530,VertStart+VertStep);
        GetNum(Y,X,False);
        if ( X >= 0.0 ) and ( X <= PZTMax/ PZTGain) then
            begin
               DAVolts := X;
               GoPZT(MPZTFlag);
            end;
        Delay(500);
        ClearViewPort;
     End;
                            { proc MovePZT }

 Procedure  MoveToWN;
    Const
       MTW1 ='Move Crystal/Grating to lase at a new wavenumber:     ';
       MTW2 ='Enter the wavenumber to goto:                         ';
    Var
       vertposi, flag : integer;
       JunkI, NcOld, NgOld, deltaC, deltaG   : longint;
       JunkR, NewWN : real;
       MoveYes       : Boolean;
```

```
   begin
      SetViewPort(10,103,629,147,clipOn);
      VertPosi :=vertStart;
      OutTextXY(GetMaxX div 2, VertPosi,MTW1);
      VertPosi := VertPosi + VertStep;
      OutTextXY(GetMaxX div 2, VertPosi,MTW2);
      MoveTo(500,VertPosi);
      GetNum(JunkI,JunkR, False);

      MoveYes := False;
      if (JunkR > 2600) and (JunkR < 4700) then
         begin
            NewWN := JunkR;
            MoveYes := True;
         end;
      Delay(800);
      ClearViewPort;

      if MoveYes then
         begin
            NGOld     :=NGrating;
            NcOld     :=NCNG(NGrating);
            deltaG    :=NGWN(NewWN)-NgOld;
            deltaC    :=NCNG( NGWN(NewWN) ) - NcOld;
            TurnLMotor(1,deltaG,flag);
            TurnLMotor(0,deltaC,flag);
         end;
   end;                        { porcedure MoveToWN }

Procedure  JumpOrN;
   Const
       JPN1  =’Change the order of interference by 1. ’;
       JPN2  =’Plus 1, Minus 1, or Cancel.<P/M/C>?    ’;
   Var
       ch   : char;
       flag : integer;
       nj   : Longint;
   Begin
       SetViewPort(10,103,629,147,Clipon);
       OutTextXY(320,VertStart,JPN1);
       OuttextXY(320,vertStart+vertStep,JPN2);
       Repeat ch := readkey until ch in [’P’,’p’,’M’,’m’,’C’,’c’];
       if ch in [’P’,’p’] then
          begin
             OrderN :=  OrderN + 1;
             if ( NEWN(CurrentWN) <= 0.9 * EMax )
                and ( NEWN(CurrentWN) >= 0.1 * EMax ) then
                begin
                   nj := NEWN(CurrentWN) - NEtalon;
                   TurnHMotor(0,nj,Flag);
                end
             else
                begin
                  OrderN := OrderN - 1;
                  Sound(1000);Delay(1000); Nosound;
                end;
          end
       else if  Ch in [’M’,’m’]  then
          begin
             OrderN  := OrderN - 1;
             if ( NEWN(CurrentWN) <= 0.9 * EMax )
                and ( NEWN(CurrentWN) >= 0.1 * EMax ) then
                begin
```

```
                   nj :=NEWN(CurrentWN) - NEtalon;
                   TurnHMotor(0,nj,Flag);
                end
             else
                begin
                  OrderN := OrderN + 1;
                  Sound(1000); Delay(1000); NoSound;
                end;
          end;
       ClearViewPort;
   End;
              { procedure JumpOrN }

Procedure  DrawTrace (Number : integer);
   Var
      i         : integer;
      wr,wo     : String[9];
      xr        : Real;
   Begin
       Case Number of
          1:   { window 1}
              Begin
                  xr := ( DZero  - DTop ) / WindowRatio[1] ;
                  Str(xr:9:4,wr);
                  Str(WindowOffset[1]:9:3,wo);
                  SetViewPort(2,DTop,637,DBot,ClipOn); ClearViewPort;
                  SetViewPort(0,0,639,479,Clipon);
                  OutTextXY(175,DTop+6,'Data:   YScale='+wr+
                     ' Offset='+wo);
                  Line(10,DZero,629,DZero);
                  SetViewPort(10,DTop,629,DBot,ClipOn);
                  if NPresentPoint > 0 then
                      For i :=1 to NPresentPoint do
                         PutPixel(Round((300.0 * (2 * i -
                                 1)/NDataPoints)),
                           Round(DZero-DTop-WindowRatio[1] *
                            (FileData[0,i] - WindowOffset[1]) ),
                           Yellow);
              End;
          2:   { window 2}
               Begin
                  xr := ( PZero  - PTop ) / WindowRatio[2] ;
                  Str(xr:9:4,wr);
                  Str(WindowOffset[2]:9:3,wo);
                  SetViewPort(2,PTop,637,PBot,ClipOn); ClearViewPort;
                  SetViewPort(0,0,639,479,Clipon);
                  OutTextXY(175,PTop+6,'Power:  YScale='+wr+
                     ' Offset='+wo);
                  Line(10,PZero,629,PZero);
                  SetViewPort(10,PTop,629,PBot,ClipOn);
                  if NPresentPoint > 0 then
                      For i :=1 to NPresentPoint do
                         PutPixel(Round((300.0 * (2 *
                                 i - 1)/NDataPoints)),
                           Round(PZero-PTop-WindowRatio[2] *
                              (FileData[1,i] - WindowOffset[2]) ),
                                 Yellow);
              End;
          3:   { window 3}
              Begin
                  xr := ( RZero  - RTop ) / WindowRatio[3] ;
                  Str(xr:9:4,wr);
                  Str(WindowOffset[3]:9:3,wo);
```

```
                     SetViewPort(2,RTop,637,RBot,ClipOn); ClearViewPort;
                     SetViewPort(0,0,639,479,Clipon);
                     OutTextXY(175,RTop+6,'Refer:  YScale='+wr+
                        ' Offset='+wo);
                     Line(10,RZero,629,RZero);
                     SetViewPort(10,RTop,629,RBot,ClipOn);
                     if NPresentPoint > 0 then
                         For i :=1 to NPresentPoint do
                            PutPixel(Round((300.0 *
                              (2 * i - 1)/NDataPoints)),
                              Round(RZero-RTop-WindowRatio[3] *
                               (FileData[2,i] - WindowOffset[3]) ),
                                   Yellow);
                 End;
          end;        {case}
      End;
                { Proc: DrawTrace }


  Procedure Zoom123;
   Const
     Z123A ='Window Zoom/UnZoom by 20% or zero the offset on window
1,2,3.    ';
     Z123B ='1,2,3:Zm 1,2,3; 4,5,6:Unzm 1,2,3;7,8,9:Reset offset
1,2,3.Enter: ';
   var
     Wkey     : Char;
     num,err  : Integer;

   Begin
       SetViewPort(10,123,629,147,ClipOn);
       OutTextXY(300,8,Z123A);
       OutTextXY(300,18,Z123B);
       Repeat Wkey:=ReadKey
       Until ( Wkey in ['1','2','3','4','5','6','7','8','9'] ) and
             ( ( not Scanning ) or ( ScanRep - NPresentRep > 30 ) );
       Val(Wkey,num,err);
       ClearViewPort;
       if num =1 then
            WindowRatio[1] := WindowRatio[1] * 1.2
       else if num =2 then
             WindowRatio[2] := WindowRatio[2] * 1.2
       else if num =3 then
             WindowRatio[3] := WindowRatio[3] * 1.2
       else if num =4 then
             WindowRatio[1] := WindowRatio[1] / 1.2
       else if num =5 then
             WindowRatio[2] := WindowRatio[2] /1.2
       else if num =6 then
             WindowRatio[3] := WindowRatio[3] /1.2
       else if num =7 then
             WindowOffset[1] :=0.0
       else if num =8 then
             WindowOffset[2] :=0.0
       else
             WindowOffset[3] :=0.0;

       if num > 3 then   num := num - 3 ;
       if num > 3 then   num := num - 3 ;
       DrawTrace(num);
     end;             { proc  Zoom123 }

  Procedure AutoZoom;
```

```
   Const
     AZ1  ='Autozooming. Enter window number <1/2/3>:   ';
   var
     AZkey        : Char;
     num,err      : integer;
   begin
     SetViewPort(10,130,629,147,ClipOn);
     OutTextXY(310,10,AZ1);
     Repeat AZkey := Readkey
     until ( AZkey in ['1','2','3'] )  and
           ( ( not Scanning ) or ( ScanRep - NPresentRep > 30 ) );
     Clearviewport;
     Val(AZkey,num,err);
     if NPresentPoint > 1 then
         begin
             if num = 1 then
                   begin
                      if Abs(OPODmaxMin[2,0] - OPODMaxMin[1,0])
                             < 0.0001 then
                          WindowRatio[1]  := 1000
                      else
                          WindowRatio[1]   := 0.85 * (DZero-DTop)
                              / ( OPODmaxMin[2,0] - OPODMaxMin[1,0] );
                      WindowOffset[1]  := OPODMaxMin[1,0];
                   end
              else if num = 2 then
                   begin
                      if Abs(OPODmaxMin[2,1] - OPODMaxMin[1,1])
                       < 0.0001 then
                          WindowRatio[2]  := 1000
                      else
                          WindowRatio[2]   := 0.85 * (PZero-PTop)
                              / ( OPODmaxMin[2,1] - OPODMaxMin[1,1] );
                      WindowOffset[2]  := OPODMaxMin[1,1];
                   end
              else
                   begin
                      if Abs(OPODmaxMin[2,2] - OPODMaxMin[1,2])
                            < 0.0001 then
                          WindowRatio[3]  := 1000
                      else
                          WindowRatio[3]   := 0.85 * (RZero-RTop)
                              / ( OPODmaxMin[2,2] - OPODMaxMin[1,2] );
                      WindowOffset[3]  := OPODMaxMin[1,2];
                   end;
        DrawTrace(num);
     end;
   end;                          { Proc AutoZoom }

Procedure WriteResp( WBuf : SS);
    begin
       SetViewPort(10,103,619,115,ClipOn);
       ClearViewPort;
       OutTextXY(310,6,WBuf);
    end;                                 { proce WriteResp }

Procedure GetFromTOF;
    var
       Reg2                     : registers;
       SignalY,CountY,Done      : Boolean;
       BeginT,TTime,FirstT,LastT,SigTime      :  Real;
       Th,Tm,Ts,Ts100   : Word;
    begin
```

```
       Reg2.Dx := 0;         { Com1: port }
       Reg2.Ah  := 0;
       Reg2.Al  := $E7;      { Com1: 9600, n,8,2,- }
       Intr(20,Reg2);        { Initialize Port com1 }

       GFTBuf :='';
       CountY := False;
       Gettime ( Th,Tm,Ts,Ts100);
       BeginT :=60 * Tm + Ts + Ts100 / 100.0 ;

       Repeat
         GetTime ( Th, Tm , Ts, Ts100);
         TTime :=60 * Tm + Ts + Ts100 / 100.0 ;
         Reg2.Ah :=3;
         Intr(20, Reg2);
         SignalY := (( Reg2.Ah and $01) = $01 );
                                         { signal coming in }
         if SignalY then
            begin
                SigTime := TTime;
                Reg2.ah :=2;
                Intr(20,Reg2);
                GFTBuf := GFTBuf + Chr( Reg2.Al );
                                        { Get the signal }
                if not CountY then
                   begin
                      FirstT := SigTime ;
                      CountY  := True;
                   end;
            end;
         Done :=  ( TTime - BeginT  > WTime )  or
         (( TTime < BeginT ) and ( 3600.0 + TTime - BeginT > WTime ))
          or  (  CountY and  ( (TTime - SigTime > 0.2) or
         ( (TTime < SigTime) and (3600.0+ TTime- SigTime > 0.2) ) ) );
       Until    Done;
                                   { only wait for 10 Seconds }
       LastT := SigTime;

       if not CountY  then
          begin
            GFTBuf :='';
            Sound(1000);Delay(2000);NoSound;
          end;
    end;                     { Procedure GetFromTOF  }

Procedure SendToTOF;
  Var
      Reg1       : Registers;
  Procedure Send;
    var
       LBuf,i : Word;
     begin
       Reg1.Dx  := 0;         { Subject to Change! Activate Com1 Port }
       Reg1.Ah  := 0;
       Reg1.Al  := $E7;       { Com1: 9600, n,8,2,- }
       Intr(20,Reg1);         { Initialize Port com1 }

       LBuf :=length(OutBuf);
       for  i := 1 to LBuf do
           Begin
               Reg1.Ah :=1;
               Reg1.Al :=Ord(OutBuf[i]);
               Intr(20, Reg1);
```

```
             End;              { Send Message }
       end;                                   { SubRoutine Send }

  Begin
       Send;
       if not Oneway then
           GetFromTOF(InBuf, WaitTime);
  End;
          { procedure SendToTOF }

  Procedure ToTOF;
      { YBC :      Send Character strings '$'+xxx+'@' to
                   TOF computer, wait for response.
                   A great try-out tool for testing port communication }
    Const
       ToTOF1 = 'Type in message ( end with <Return> ): ';
       ToTOF2 = 'Message: ';
       ToTOF3 = 'The message has been sent out, wait for response. ';
       ToTOF4 = 'The TOF response is:  ';
    var
        Vertposi   : Integer;
        ee,ff      : SS;
        yy         : Boolean;
                             { YBC : temperary ?  }
    Begin
       SetViewPort(10,103,629,147,ClipOn);
       VertPosi := VertStart;
       OutTextXY(GetMaxX div 2,VertPosi,ToTOF1);
       VertPosi := Vertposi + VertStep;
       OutTextXY(150, VertPosi, ToTOF2);
       MoveTo(300,VertPosi);
       GetStr(ee);

       { RS232c communication: }
       if not ( ee='' ) then
          begin
              VertPosi := Vertposi + VertStep;
              OutTextXY(GetMaxX div 2, VertPosi, ToTOF3);
              ee := '$'+ee+'@';
              yy :=False;
              SendToTOF(ee,ff,yy,5.0);
              VertPosi := Vertposi + VertStep;
              OutTextXY(GetMaxX div 2,vertposi,ToTOF4+ff);
          end;
        Delay(1000);
        ClearViewPort;
   end;
                    { prcoedure ToTOF }

  Procedure AcouYesNo;
     Const
         AYN = 'Toggle the acoustic/ion beam experimental mode switch?
<Y/N>: ';
     var
        ch       : char;
     begin
       SetViewPort(10,103,629,147,ClipOn);
       OutTextXY(320,VertStart,AYN);
       Repeat ch :=Readkey until ch in ['Y','y','N','n'];
       Delay(500);
       ClearViewPort;
       if ch in ['Y','y'] then
          begin
          SetColor(DarkGray);
          SetViewPort(0,0,GetMaxX,GetMaxY,ClipOn);
          if  AcouOnly then
              OutTextXY(100,48,'Acoustic')
          else
              OutTextXY(100,48,'Ion Spec');
                                                        { Erase ! }
          SetColor(LightGreen);
          if AcouOnly then
               OutTextXY(100,48,'Ion Spec')
          else
               OutTextXY(100,48,'Acoustic');
                                                        { Write ! }
          AcouOnly  :=  Not AcouOnly ;
        end;
   end;
                               { proc acouYesNo }

Procedure CalCG;
   Const
     CCG1 ='Calculate NC, NG for a certain wavenumber.';
     CCG2 ='Enter wavenumber: ';
   var
     JunkI,TempC,TempG  : longint;
     TempWN,WNTry       : Real;
     StrC, StrG         : String[7];
     ch                 : Char;
   Begin
     SetViewPort(10,103,629,147,ClipOn);
     OutTextXY(310, Vertstart, CCG1);
     OutTextXY(200,Vertstart+VertStep,CCG2);
     MoveTo(460, VertStep+VertStart);
     GetNum(JunkI,WNTry,False);
     if ( WNTry > 2200 ) and  ( WNTry < 4700 ) then
        TempWN :=WNTry
     else
        TempWN := CurrentWN;
     TempG :=NGWN(TempWN);
     TempC :=NCNG(TempG);
     Str(TempG:7,StrG);
     Str(TempC:7,StrC);

     OutTextXY(310,VertStart+2 * VertStep,'NC = '+StrC+
              ' NG = '+StrG );
     ch :=Readkey;
     ClearViewPort;
  end;                                      { proc CalCG }

Procedure TChkDial;
  const
    TC1 = 'Do A-D Time Check (check the trigger)?<Y/N>: ';
    TC2 = 'Select Channel Number (0..2): ';
  var
    Tckey: Char;
    i, chft : integer;
  begin
    setviewport(10,103,629,147,ClipOn);
    OuttextXY(310,vertStart,TC1);
    Repeat Tckey:=Readkey Until TCKey in ['Y','y','n','N'];
    if tckey in ['Y','y'] then
       begin
          TDFlag := true;
          OuttextXY(310,vertStart+vertStep, tc2);
          Repeat tckey :=readkey until tckey in [’0’,’1’,’2’];
          MoveTo(550,vertStart+vertStep);
          if tckey =’0’ then
             begin
                chan := 0; Outtext(’0’);
             end
          else if tckey =’1’ then
             begin
                chan := 1; Outtext(’1’);
             end
          else
             begin
                chan := 2; Outtext(’2’);
             end;
          delay(500);

          Clearviewport;
          SetViewPort(5,67,95,77,ClipOn);
          Clearviewport;
          Setcolor(red);
          OutTextXY(45,5,’Scanning..’);

          DataSaved := False;
          SetColor(darkGray);
          SetViewPort(0,0,GetMaxX,GetMaxY,ClipOn);
          OutTextXY(480,48,’Data Saved’);
          SetColor(Lightgreen);
          OutTextXY(480,48,’Data UnSaved’);
          SetViewPort(10,103,629,147,clipOn);

          Scanning := True;
          NPresentPoint  := 0;
          ClearDrawings;
       end
    else
       begin
          TDFlag := False;
          ClearViewPort;
       end;
  end;
                                        { Proc Tchkdial }

Procedure ScanDial;
  const
     ss1 =’Start Scan(Make sure trigger pulse is connected)? <Y/N>: ’;
     ss2 =’Scanning full mode rather than etalon only mode? <Y/N>: ’;
  var
     sskey    : Char;
     i        : integer;
     zz       : Boolean;
     cc,dd    : SS;
     NGOld,NCOld,DeltaC,deltaG   : Longint;
     FFlag : integer;
 begin
    setviewport(10,103,629,147,ClipOn);
    OuttextXY(310,vertStart,ss1);
    Repeat sskey:=Readkey Until ssKey in [’Y’,’y’,’n’,’N’];
    if sskey in [’y’,’Y’] then
        begin
          if resolution =’Med’ then
            begin
              OuttextXY(310,vertStart+vertstep,ss2);
              Repeat sskey:=Readkey Until ssKey in [’Y’,’y’,’n’,’N’];
              if ssKey in ['Y','y'] then
                Smode :='Full'
              else
                begin
                  Smode :='Etalon';
                  EtaFirst:=Round(0.1*EMax);EtaLast:=Round(0.9*EMax);
                  EtaStep:=Round(0.01*EMax);
                end;
            end;
          SDFlag := True;
          Clearviewport;
          SetViewPort(5,67,95,77,ClipOn);
          Clearviewport;
          Setcolor(red);
          OutTextXY(45,5,'Scanning..');

          DataSaved := False;
          SetColor(darkGray);
          SetViewPort(0,0,GetMaxX,GetMaxY,ClipOn);
          OutTextXY(480,48,'Data Saved');
          SetColor(Lightgreen);
          OutTextXY(480,48,'Data UnSaved');
          SetViewPort(10,103,629,147,clipOn);

          Scanning :=True;
          NPresentpoint :=0;
          ClearDrawings;
        end
    else
        begin
          SDFlag := False;
          ClearViewPort;
        end;
 end;
                          { Procedure ScanDial }

Procedure DataSave;
  const
    DS1  ='Save the collected data to file. Enter <Y/N>: ';
    DS2A ='Current data directory is:  ';
    DS3  ='Enter your personal code name (1 character):';
    DS4A ='Saving data into file : ';
    DS5  ='No data to save into the file ! ';
  var
    DSkey,ch               : Char;
    DSV                    : String[60];
    Cudir                  : SS;
    VertPosi,i,j,num       : integer;
    year,month,day,dw      : word;
    stry,strm              : String[1];
    strd                   : string[2];
    strnum                 : string[3];
    name                   : String[12];
    S                      : PathStr;
    YBCfile                : Text;
    ScanLast1              : Real;

  begin
     SetViewport(10,103,629,147,ClipOn);
     VertPosi :=VertStart;
     OutTextXY(GetMaxX div 2 , VertPosi,DS1);

     Repeat   DSkey :=ReadKey until DSkey in ['Y','y','N','n'];
```

```
if DSkey in ['N','n']  then
    begin
       VertPosi := VertPosi + VertStep;
       OutTextXY(300,VertPosi,'Abort DATASAVING as requested');
    end
else
    begin
      GetDir(0, CuDir);
      DSV := DS2A  + CuDir +'.';
      VertPosi := VertPosi + 8;
      OutTextXY(300,VertPosi,DSV);
                         { Get the Datafile directory }
      Vertposi := VertPosi + 8;
      OutTextXY(300,VertPosi,DS3);
      Repeat  ch := ReadKey
      Until  ( Ord(ch) > 96 ) and ( Ord(ch) < 123 );
                                        { 'a' to 'z' ! }
      OutTextXY(550,Vertposi,ch);

      GetDate(year,month,day,dw);
      year := year Mod 10 ;
      Str(year,stry);
      if month = 10  then
           strm :='A'
      else if month = 11 then
           strm :='B'
      else if month = 12 then
           strm :='C'
      else
           Str(month,strm);
      Str(day,strd);
      if day   < 10 then    strd :='0'+strd;

      num := 0;
      Repeat
         num := num + 1;
         Str(num,strnum);
         if num < 10   then   strnum := '00' + strnum
         else if num < 100  then  strnum := '0' + strnum
         else   strnum := strnum;

         name := stry+strm+strd+ch+strnum+'.OPO'   ;
                         { the format for datafile name }
         S :=FSearch(name,CuDir);
      Until S ='' ;

      DSV :=DS4A + name+ ' .';
      VertPosi :=VertPosi + 8;
      OutTextXY(300,Vertposi,DSV);

     if NPresentPoint >0 then
       begin
         Assign(YBCfile,name);
         Rewrite(YBCfile);
         Writeln(YBCFile,'Opo Data');
         Writeln(YBCFile,name);
         Writeln(YBCFile,Resolution+'Res');
         ScanLast1 := ScanWNFirst+
               (NPresentPoint-1)* ScanWNStep;
         Writeln(YBCFile,'ScanRangeStep:
               ',ScanWNFirst:11:4,Char(9),
                ScanLast1:11:4,Char(9),ScanWNStep:11:4);
         WriteLn(YBCFile,'ScanRep,NDataPoints,NAssign,NMeasure');
```

```
                  writeln(YBCfile, ScanRep:10,Char(9),NPresentPoint:10,
                              Char(9),'0',Char(9),'0');
                  Writeln(YBCFile,'#'+Char(9)+'WaveNumber'+Char(9)+
                  'Data'+Char(9)+'Power'+Char(9)+'Reference'+Char(9)
                            +'TOF1'+Char(9)+'TOF2');
                  for j:=1 to NPresentPoint do
                    begin
                     Write(YBCFile,j:3,Char(9));
                     Writeln(YBCFile,FileData[3,j]:9:4,Char(9),
                        FileData[0,j]:9:4,Char(9),FileData[1,j]:9:4,
                        Char(9),FileData[2,j]:9:4,Char(9),
                        FileData[4,j]:9:4,Char(9),FileData[5,j]:9:4);
                    end;
                  Close(YBCfile);
                  DataSaved := True;
                  SetColor(darkGray);
                  SetViewPort(0,0,GetMaxX,GetMaxY,ClipOn);
                  OutTextXY(480,48,'Data UnSaved');
                  SetColor(Lightgreen);
                  OutTextXY(480,48,'Data Saved');
                  SetViewPort(10,103,629,147,clipOn);
                end
              else
                begin
                  Sound(800);Delay(2000);NoSound;
                  ClearViewPort;
                  VertPosi := VertStart;
                  OutTextXY(300,VertPosi,DS5);
                end;
            end;                 { 'Y','N' }
            Delay(1000);
            ClearviewPort;
     end;
                 { proc Datasave }
  Procedure QuitCollect;
    Const
     QC1  ='Quit the DATACOLLECT, return to Main Desktop.';
     QC2  ='Unsaved data WILL BE LOST!!!                 ';
     QC3  ='Enter <Y/N>:                                 ';
    Begin
     SetViewPort(10,103,629,147,ClipOn);
     OutTextXY(GetMaxX div 2,VertStart,QC1);
     OuttextXY(GetMaxX div 2,VertStart+ VertStep,QC2);
     OuttextXY(GetMaxX div 2,VertStart+ 2 * VertStep,QC3);
     Repeat     QCkey   :=Readkey;
     Until QCkey in [ 'Y','y','N','n'];
     Delay(500);
     ClearViewPort;
    End;  { proc  QuitCollect }
end.                   { unit  SubRut }
```