LBL-34924
UC-411

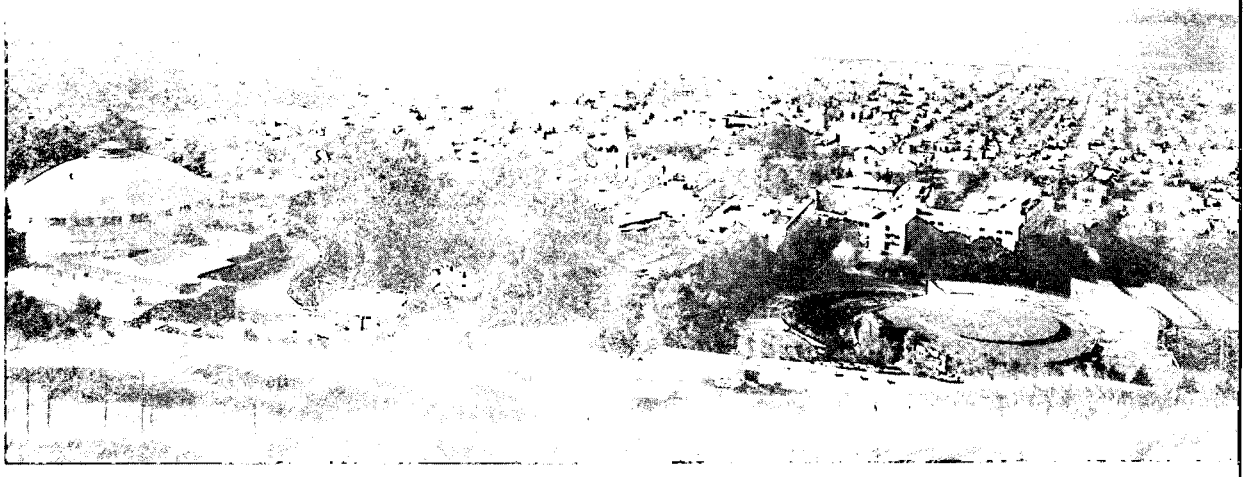
Lawrence Berkeley Laboratory
UNIVERSITY OF CALIFORNIA

CHEMICAL SCIENCES DIVISION

Fast Beam Studies of Free Radical Photodissociation

D.R. Cyr
(Ph.D. Thesis)

November 1993



Prepared for the U.S. Department of Energy under Contract Number DE-AC03-76SF00098

DISCLAIMER

LBL–34924

Fast Beam Studies of Free Radical Photodissociation

Douglas Robert Cyr
Ph. D. Thesis

Department of Chemistry
University of California

and

Chemical Sciences Division
Lawrence Berkeley Laboratory
University of California
Berkeley, CA 94720

November 1993

This work was supported in part by the Director, Office of Energy Research, Office of Basic Energy Sciences, Chemical Science Division of the U.S. Department of Energy under contract No. DE-AC03-76SF00098, and in part by the National Science Foundation under Grant No. CHE 91-08145.

Fast Beam Studies of Free Radical Photodissociation

by

Douglas Robert Cyr

Abstract

Fast Beam Studies of Free Radical Photodissociation

by

Douglas Robert Cyr

Doctor of Philosophy in Chemistry

University of California at Berkeley

Professor Daniel M. Neumark, Chair

The photodissociation of free radicals is studied in order to characterize the spectroscopy and dissociation dynamics of the dissociative electronic states in these species. To accomplish this, a novel method of radical production, based on the photodetachment of the corresponding negative ion, has been combined with a highly complementary form of photofragment translational spectroscopy. The optical spectroscopy of transitions to dissociative states is determined by monitoring the total photofragment yield as a function of dissociation photon energy. Branching ratios to various product channels, internal energy distributions of the fragments, bond dissociation energies, and the translational energy-dependent photofragment recoil angular distributions are then determined at selected excitation energies. A detailed picture of the dissociation dynamics can then be formulated, allowing insight concerning the interactions of potential energy surfaces involved in the dissociation.

After an introduction to the concepts and techniques mentioned above, the experimental apparatus used in these experiments is described in detail.

The basis and methods used in the treatment of data, especially in the dissociation dynamics experiments, are then put forward.

A brief illustration of the capabilities of the experiment is then given, using as examples results from the the initial photodissociation studies completed on the prototypical test systems of O_2 and the N_3 radical.

An investigation of the spectroscopy and dissociation dynamics of the NCO radical following excitation of the $\tilde{B}\ ^2\Pi \leftarrow \tilde{X}\ ^2\Pi$ electronic transition is then detailed. Measurements of the photodissociation cross section as a function of dissociation wavelength show that even the lowest vibrational levels of the $\tilde{B}\ ^2\Pi$ state predissociate. Analysis of fragment kinetic energy release reveals that the spin-forbidden N (^4S) + CO ($^1\Sigma^+$) products are exclusively produced until 20.3 kcal/mol above the origin, at which point the spin-allowed N (^2D) + CO product channel becomes energetically accessible. The spin-allowed channel dominates above this threshold. By determining the location of this threshold we obtain a new $\Delta_f H^o_{(298K)}$ for NCO of 30.5 ±1 kcal/mol, several kcal/mol lower than the previously accepted value.

A photodissociation study of the nitromethyl radical, CH_2NO_2, is then reported. Two major dissociation product channels are observed at each of three dissociation wavelengths investigated in the range from 240 to 270 nm, and are identified as (I) $CH_2NO_2 \rightarrow CH_2NO + O$ and (II) $CH_2NO_2 \rightarrow H_2CO + NO$. No evidence is found for simple C-N bond fission to give (III) $CH_2NO_2 \rightarrow CH_2 + NO_2$. Translational energy and angular distributions are presented for the two observed channels. The translational energy distribution of channel (I) peaks at only 5-8 kcal/mol, while the distribution for channel (II) peaks at ~60 kcal/mol. The angular distributions for both channels are largely isotropic. The nature of the electronic excitation and dissociation dynamics are considered at length. The upper state in the electronic transition is assigned to the $1\ ^2B_1$ state, and the data is found to be consistent with both dissociation processes occurring primarily on excited state surfaces.

This thesis is dedicated

to my wife

Lejla

and my daughters

Ana Katarina and Jelena Kristina

Table of Contents

Acknowledgments

Much credit for the success of this experiment has to go to those I have been fortunate to work with in the last five years. Above all, Professor Daniel Neumark deserves recognition as the originator of this ambitious and rewarding project, and my thanks for being a superb research advisor. Dan has managed to foster an atmosphere within the group that is both ambitious and confident, while at the same time very supportive and unified. In addition, Dan actively encourages his students to pursue ideas of their own; as just one example, this encouragement lead to the development of our pulsed discharge negative ion source.

Looking back, I feel very fortunate to have been at the right place at the right time, having joined the Neumark Group soon after the free radical photodissociation proposal received funding. As I surveyed the completely empty laboratory that was D-21 Latimer, I felt very excited and eager. But how was I, a naive, inexperienced, first-year graduate student, going to build, trouble-shoot and operate such a complex experiment? The answer, of course, was that I would not do it alone. Through every step of the building process Dan was always accessable, providing clever ideas to solve difficult problems as well as a reasonable amount of patience and understanding when my inexperience occasionally showed through. Not long into the project, Ricky Metz and then Bob Continetti signed on, bringing not only enthusiasm but also a wealth of knowledge, which they always took time to confer to me.

Ricardo Metz had a major role in building the machine; he was spirited away from the operational fixed-frequency photoelectron spectrometer in the early stages of construction. Ricky was very multi-talented and managed to keep a hand in explaining transition state data from his old machine even as he worked on the radical project. He taught me the basics of photoelectron spectroscopy, taking me under his wing when we slipped across the hall and spent a few weeks on his old stomping grounds, using the

fixed-frequency photoelectron spectrometer to scout for negative ions suitable for use as radical precursors.

Dr. Robert Continetti joined the experiment at a critical point, when the major building was nearly complete and the first experiments were only a few months away. Initially, he provided some critical technology (piezo valve and fast ion gauge) and 'hard experiment' experience that surely made a world of difference. Later, Bob played a leading role in developing and implementing other essential components, including the time-and-position sensitive detector. As the first postdoc on the experiment, Bob lead by example, working long hours and giving his all. I have no doubt he will be successful in his position on the faculty at UC San Diego.

As a new member of the FRBM team, David Osborn had an ideal philosophy, striving to learn as much as possible while enthusiastically doing whatever was asked, all the while growing more independent. Dave patiently waited for his opportunity to lead a project, knowing that he would inherit an experiment with all major wrinkles ironed out, in part due to his own hard work. I am fully confident that he is ready to help lead the machine into new and exciting areas in the next few years. He is also thanked for his (often thankless) work as system manager for the various Sun workstations since the departure of Steve Bradforth.

Dr. David Leahy (DaveBob) was a welcome addition, arriving to help fill the void left by the departure of Bob Continetti to San Diego. Although excellent in all regards, Dave has made major contributions to our time- and position-sensitive data analysis effort, which was functional but a somewhat underdeveloped area on his arrival. He has also been active in developing a better understanding of the remaining non-linearities in our time- and position-sensitive detector, thereby pushing our energy resolution to new limits. This progress allowed him to lead the way on a very comprehensive study of O_2 predissociation. Because Dave hails (most recently) from Stanford, I have one parting message for him that I hope he will remember and cherish: GO BEARS!

To the newest member of the FRBM team, Eric Ross, I wish you well. There is no doubt that with hard work you will carry on the tradition of success that has been established thus far.

I owe thanks to several former Neumark Group members who I did not work directly with, but definitely contributed to my graduate school experience in one way or another.

Theofanis Kitsopolous continually demonstrated that hard work, dedication and perseverance in the laboratory eventually brings rewards; he has been an excellent role model in this regard. His instigation of impromptu discussions on any and all topics (particularly scientific in nature) is sorely missed. I wish Theo continued success at his faculty position in Greece.

As well as being a fine scientist, Alexandra Weaver managed to maintain a sense of perspective, having outside interests resembling those of a real person! Alex was also a very patient and kind person, whose genuine concern for the well-being of others was refreshing and inspirational.

Sharing an office with Steven Bradforth was good fun. Since we both instinctively sensed the tack the other would take in a discussion, and happily provided the opposing viewpoint as a matter of course, there was never a dull moment. Steve was also a storehouse of information about various theoretical and computational matters. He deserves recognition for his efforts as Sun system manager while in the group.

In accepting an appointment at UBC following a postdoctoral term in the Neumark Group, Dr. Irene Waller showed that it *is* possible to return to a faculty position in Canada. I wish her much success in her work there.

The current members of the Neumark Group (in addition to those I work most closely with, who were acknowledged above), also deserve mention. Don Arnold and I were new graduate students together in the Fall of '88, when we both joined the Neumark Group. In addition to his considerable success in the lab, Don breathed much needed life

into the softball exploits of the Neumark Group. I wish him and his co-workers, Eun Ha Kim, Cangshan Xu and Esther de Beer, every success in the future. Caroline Arnold has managed to maintain the success of the difficult threshold photoelectron experiment, as well as being a leading force in planning and executing many of the Group's social events. Caroline and her co-workers, Georg Reiser, Yuexing Zhoa, and Ivan Yourshaw, have my best wishes, as does Jeff Greenblatt, who recently joined the group to work on a planned fourth experiment.

Various staff members of the College of Chemistry have also contributed to make this project successful. The skill, dedication and responsiveness of the machine shop has been unrelenting, and all of the machinists and supervisors I have had the pleasure to work with have been extremely helpful and professional. Special thanks to George Weber, Chuck Souders, Eric Granlund, Larry Hull, Scott Bonilla, Bob Dunn and the rest of the machine shop staff. Instrument Repair, lead by Bob Cook, has been helpful, particularly in the building stages of the machine with advice on electrical wiring, etc. The Wood Shop, especially Norman Tom, also played a critical role during the building of the machine, provided sage advice, responsive service and plenty of Unistrut®!

The Neumark Group was especially lucky to gain the privilege of having Cheryn Gleibe as Dan's Administrative Assistant. Cheryn is exceptionally talented and her experience and intelligence allows her to cheerfully do the job of two people, in half the time!

Within the Lawrence Berkeley Laboratory Chemical Science Division we are fortunate to have Auben Winters as our effective advocate. Our connection to Lawrence Berkeley Laboratory has also allowed us to receive helpful advice at various times from Joe Katz, Oren Milgrome, and George Gabor.

Another resource to which we were fortunate to have access is Space Sciences Laboratory, a division of the UC Berkeley physics department. From them, we learned much about wedge-and-strip detectors. Without the help and advice of Dr. Oswald

Seigmund and Joe Stock, developing our time- and position-sensitive detector would have taken much longer.

I wish to thank my Preliminary Exam committee, which consisted of Profs. Yuan Lee (Chair), Gabor Somerjai, John Porter and Marjory Olmstead (Physics). My thesis committee, consisting of Profs. Daniel Neumark (Chair), Yuan Lee and Jeffery Reimer (Chemical Engineering) also deserve thanks for their efforts in this capacity.

I would not have considered setting my sights as high as Berkeley if not for the encouragement of many people. While working at the Notre Dame Radiation Laboratory as part of an undergraduate co-op program, I was under the supervision of Dr. P.K. Das. A frequent collaborator was a distinguished professor from India, Prof. M.V. George. Both of these gentlemen freely offered encouragement and knowledge, and without trying explicitly to do so, raised my opinion of science and scientists a great deal. Prof. John Hepburn at the University of Waterloo continued this trend, and introduced me to the exciting world of Chemical Physics.

I wish to thank NSERC (Canada) for a 1967 Science and Engineering Fellowship. The research described in this thesis was supported by the Director, Office of Basic Energy Sciences, Chemical Sciences Division of the U. S. Department of Energy under Contract No. DE-AC03-76SF00098. Additional support was provided by the National Science Foundation under Grant No. CHE 91-08145.

I have much to be thankful for away from the laboratory as well. I am very lucky to have both the strong support for my professional goals as well as the refreshing counterbalance to science that comes from my family. During my time here at Berkeley my wife, Lejla, and I have had two children. Because of the sacrifices they have made for me during my graduate career, especially during the writing of this thesis, it is to these three that this thesis is dedicated.

Preface

This thesis is an account of the photodissociation studies carried out using the Fast Radical Beam Machine to characterize the dissociative electronic states of several reactive free radicals.

Chapter 1 gives a general overview of the experimental program, including the motivation for our work, the approach that is used and the knowledge that is gained. The general technique of photofragment translational spectroscopy is introduced and the adaptations present in our experiments are then discussed. The principles of applying negative ion photodetachment as the source of our free radicals are also examined.

Chapter 2 describes the Fast Radical Beam Machine at length, as this thesis is the first detailed account of its construction and operation. Included is a description of a fast high voltage switch, which was designed and implemented as part of the construction process. This switch is the subject of a publication having Dr. Robert Continetti as principal author, appearing in Rev. Sci. Instrum. **63**, 1840 (1992). In addition, a promising new source of negative ions, based on a pulsed discharge-free jet expansion is outlined within this chapter.

Chapter 3 provides the details of our data analysis procedures. Included is a short description of Monte Carlo forward convolution. This method of analysis is applied in time-of-flight distribution analysis of the type found in Chapter 5, and has been used as a consistency check of the more direct inversion of data from our dissociation dynamics experiments involving the time and position-sensitive detector. This direct inversion process is the subject of the remainder of the chapter, with a step-by-step description of the techniques involved in the analysis of experiments such as those described in Chapter 6 (see below).

In Chapter 4, examples of experimental results obtained in our studies of O_2 and N_3 are presented as an illustration of the capabilities of the machine. The O_2 data shown is

part of a data set which resulted in a letter to appear in Chemical Physics Letters and a full article to be submitted to the Journal of Chemical Physics, both of which will have Dr. David Leahy as the principal author. The N_3 data have been published previously with Dr. Robert Continetti as the principal author; the relevant articles can be found in Chem. Phys. Lett. **182**, 406 (1992) and J. Chem. Phys. **99**, 2616 (1993).

Chapter 5 describes experiments characterizing the NCO free radical. The photofragment time-of-flight spectrum is analyzed to reveal the occurence of spin-forbidden dissociation below the energetic threshold for spin-allowed dissociation. It is then shown that above the threshold for spin-allowed dissociation, the spin-allowed channel dominates. From the determination of this threshold, we are able to directly deduce a heat of formation for the NCO radical. These experiments resulted in the first full publication from this machine; published in J. Chem. Phys. **97**, 4937 (1992).

Chapter 6 contains an account of our work on the nitromethyl radical CH_2NO_2. This radical dissociates in a manner completely different from that which might be expected based on the UV photodissociation behavior of nitromethane, CH_3NO_2. We observe the existence of two major dissociation channels, and propose excited-state mechanisms that are consistent with the experimental observations. This work was discussed in preliminary form in SPIE Proceedings **1858**, 49 (1993) where the principal author was Dr. David Leahy, but a more detailed treatment, following Chapter 6, will appear in the December 1st 1993 issue of the Journal of Chemical Physics.

Appendices A and B contain the Control and Data Aquisition programs FRBM and TPS.

Chapter 1

Fast Radical Beam Photofragment Translational Spectroscopy – An Introduction

In recent years there has been a growing interest in the characterization of reactive free radicals. These species have long attracted attention[1] because of their universally acknowledged role as important intermediates in a variety of critical processes, such as many of those found in combustion or atmospheric chemistry. However, both the inability to easily generate such species purely and their high reactivity once made have historically hindered their study. For this reason free radicals are, in general, much less well characterized than stable, closed shell molecules.

Consequently, the thermodynamics and photochemistry of radicals is often ill-defined. Many radicals have large uncertainties associated with even the most basic thermochemical information, such as their heats of formation. These quantities are vitally important if we are to fully understand and eventually model processes involving these radicals.

Further motivation for the investigation of the free radicals lies in the fact that the open shell nature of free radicals will result in a higher density of low lying electronic states. Because of the presence of these low lying electronic states we can expect a relatively rich spectroscopy and photochemistry for these species. This increased state density will also result in a higher probability of surface crossings following photoexcitation, which will often result in complex dissociation dynamics. The dynamics involved in such dissociation processes will be directly reflected in the nature of the resulting product final state distributions. In addition, this regime where strong interaction amongst various excited states is common can typically be accessed with relative ease in free radicals (the excitation energies required are usually amenable to visible or UV laser excitation). In this regard free radicals may represent model systems for study of these interactions.

In cases where the difficulties associated with generating a suitable sample of free radicals have been overcome, spectroscopic techniques such as laser induced fluorescence (LIF), multiphoton ionization (MPI),[2] and absorption spectroscopy have enjoyed success in the study of the bound states and molecular structure of free radicals.[3]

Dissociative states of free radicals, however, have received much less attention. A brief review of work completed thus far in this area is given in the next section. In the remainder of the chapter, our technique, fast beam photofragment translational spectroscopy, is outlined, followed by a consideration of what can be learned from photodissociation spectroscopy and dynamics experiments. The final section of the chapter discusses the method by which we produce our reactive free radicals for subsequent photodissociation, namely, photodetachment of the corresponding negative ion, and goes into some detail about the characteristics of this process relevant to our application.

1.1 Background

Lee and co-workers have completed a variety of photofragment translational spectroscopy studies involving the photodissociation of radicals. Specifically, they have studied the primary photodissociation of CCl_3, ClO_2, and NO_3 radicals. The secondary dissociation of the CCH and HS radicals, observed following the primary dissociation of C_2H_2 and H_2S, respectively, also received attention, although secondary dissociation experiments in general suffer from ill-defined initial state preparation of the radical via the primary dissociation process. Briefly, Hintsa et al.[4] used photolytic generation of CCl_3 from the precursor CCl_4, followed by thermalization in a buffer gas and then molecular beam expansion. This allowed the study of CCl_3 photodissociation, to form CCl_2 + Cl. Davis and Lee[5] observed a mode specific effect in the photodissociation of ClO_2; where at roughly the same photolysis energy concerted elimination of O_2 is promoted by both bending excitation and combinations of symmetric and antisymmetric stretches.

Additionally, Davis et al.[6] directed a study at the photodissociation of the NO_3 radical. Two separate dissociation channels were observed, forming NO + O_2 and NO_2 + O products. A heat of formation was found for the NO_3 radical, by measuring the O-NO_2 bond dissociation energy, which is in good agreement with a recent value determined by Weaver et al.[7] . The height of the potential barrier involved in the $NO_3 \rightarrow NO + O_2$ channel was also determined. Wodtke and Lee[8] used 193 nm light to photodissociate C_2H_2, observing secondary dissociation of the CCH in the process. Three electronic levels of C_2 are populated as a result of dissociation, with the lower $^1\Sigma_g$ and $^3\Pi_u$ electronic states having a greater amount of vibrational excitation than the $^1\Pi_u$ electronic state. Secondary HS fragmentation was observed in a photodissociation study of H_2S by Continetti et al.,[9] providing an improved heat of formation for the HS radical.

Coombe and co-workers[10] performed photolysis experiments on NCO at 193 nm, forming CN ($X\ ^2\Sigma^+$) + O (^3P) and monitoring the laser-induced fluorescence of the CN fragment.

Houston and co-workers[11] have thoroughly investigated HCO/DCO dissociation, observing interesting effects on predissociation lifetime, photofragment recoil distribution and CO product distributions brought about by pumping various K' levels of the upper $\tilde{A}(^2A'')$ state. These observations confirmed a dynamical curve crossing model developed by Dixon[12] to describe the $\tilde{A}(^2A'')$–$\tilde{X}(^2A')$ Renner-Teller interaction.[13]

Ng and co-workers have investigated the dissociation of the HS[14] and CH_3S[15] radicals. They determine the S(^3P)/S(^1D) branching ratios and the fine structure distribution for the S($^3P_{j=2,1,0}$) photoproducts following 193 nm photolysis for both radicals through 2+1 resonance enhanced multiphoton ionization of the S atoms.

Dagdigian and co-workers[16] have observed the production of ND from the predissociation of the DNF radical. This molecule is subject to Renner-Teller interaction, between the $\tilde{A}(^2A')$ and $\tilde{X}(^2A'')$ states, just as is found for HCO (the state ordering is reversed here, however). Radiative lifetimes for various $\tilde{A}(^2A')$ excited levels are found,

and considerable state-mixing is proposed to explain the photofragment excitation spectra, which also depend significantly on which ND rotational level is probed.

As can be seen by the preceding summary, clever methods exist through which dissociative states of free radicals can be investigated. However, with the notable exception of Lee's photofragment spectroscopy experiments, which use mass selected time-of-flight following electron bombardment ionization, these techniques lack broad generality, as they depend on the formation of a photoproduct that can be probed via a state-selective technique such as MPI or LIF. In addition, the limited number of studies involving reactive free radicals underscores the fact that a truly general source of reactive free radicals remains elusive.

It was the major goal of this project to develop a general technique for probing both the spectroscopy and the dynamics of dissociative electronic states, suitable for the study of reactive free radicals. To do this, we have developed a novel method of generating a pure, internally cold beam of free radicals for subsequent photodissociation. The spectroscopy of the dissociative electronic states in a given radical is then mapped out by measuring the total photofragment flux as a function of excitation energy. To investigate the dynamics of a particular dissociating system, we utilize a coincidence version of photofragment translational spectroscopy that is adapted to our fast beam conditions (see below). This allows us to obtain detailed information about the disposal of energy within the various degrees of freedom in the photofragments. By exploiting the relationship between the product state distribution of the photofragments and the molecular potential energy surfaces sampled during the dissociation, we can learn about the potential energy surface(s) involved in the dissociation process.

1.2 Fast Radical Beam Photofragment Translational Spectroscopy

Photofragment translational spectroscopy is an extremely general, powerful tool for investigating molecular photodissociation dynamics.[17,18] The original form of

photofragment translational spectroscopy, demonstrated by Wilson[19] in the late 60's, involved the dissociation of a molecule of interest, followed by the measurement of the distribution of times taken for a particular type of photofragment to travel a well-defined distance to a detector, all in a collision-free environment. Since then, substantial improvements in both laser photolysis sources and the original experimental design[17] have been made. In addition, variations of the basic experiment have appeared, perhaps most notably the ion-imaging technique of Chandler and Houston[20] and the H atom time-of-flight method developed by Welge and Ashfold.[21]

Our experiment involves quite a different application of photofragment translational spectroscopy. It can be summarized as follows:

$$AB^- \xrightarrow{h\nu_1} AB + e^- \xrightarrow{h\nu_2} A + B. \qquad (1.1)$$

In Eqn. (1.1), as is true throughout this chapter, AB refers to a generic free radical, not necessarily a diatomic, and therefore the products A and B may be polyatomic.

Due to their open shell nature, free radicals typically have positive electron affinities. Our experiment utilizes this property in order to generate a pure, well-characterized packet of free radicals via photodetachment of the corresponding mass-selected negative ion. As an added advantage, the high (5–8 keV) laboratory frame kinetic energy imparted to the radicals during their acceleration prior to detachment allows both the efficient *collection* and *detection* of photofragments, since they must simply strike the face of a microchannel plate (MCP) detector.

Two types of photodissociation experiments can be carried out within this general scheme, as alluded to in the previous section. The photofragment flux can be measured as a function of dissociation photon energy, $h\nu_2$, to map out the dissociative transitions of the free radical. Following this, the dissociation dynamics at a given photon energy can be investigated using a coincidence time- and position-sensitive detector,[22,23] which allows

the determination of the photofragment translational energy distribution and energy-dependent angular distributions.

The version of photofragment translational spectroscopy employed in our experiment differs from Wilson's original method in two major ways, which are ultimately a consequence of the kinematic scheme we use for investigating free radicals. Since free radicals are generated by photodetaching an electron from the negative ion corresponding to the radical of interest, we have the ability to produce the free radicals following acceleration of the precursor negative ions to a high laboratory kinetic energy (resulting in the formation of a 'fast beam'). This fast beam approach is similar to the method pioneered by Los' group,[22,24] and also used by Helm and Cosby,[25] for studying dissociative states of small molecules via dissociative charge exchange between a positive ion and metal vapor.

High center-of-mass laboratory kinetic energy allows: (1) *efficient (>50 %)*[26] *detection of photofragments* through simple collision with the face of a microchannel plate detector, eliminating the need for ionization or other final steps to permit detection, and (2) *high collection efficiency*, a necessary component of any coincidence detection scheme. If the combined collection and detection probability for a given single fragment is given by the fraction f, the probability of a given coincidence event being detected will be only f^2. However, because in our experiment the kinetic energy release of the fragments is relatively small compared to the large laboratory kinetic energy, the photofragments typically do not scatter far from the beam axis during the flight time prior to detection, and thus both fragments impinge on opposite halves of the same on-axis microchannel plate detector. In this configuration nearly all of the recoil angle distribution can be accepted at once, which compares favorably with most other methods. When using our time- and position-sensitive detector, we determine the distance between the two fragments and their orientation at the detector face, in addition to the difference in their time-of-flight. This directly yields the magnitude and direction of the center-of-mass velocity vector for a

single dissociation event in *three dimensions*. The resolution of our detection scheme is considered more fully in Chapter 2, however, the coincidence aspect of our technique is beneficial in this regard (see § 2.7.6.5). As a result, our translational energy resolution ($\Delta E/E_T \approx 0.7\%$)[27] also compares favorably with other forms of photofragment translational spectroscopy.

Fast beam photofragment translational spectroscopy is quite a general method for the study of free radical photodissociation. Although photodetachment of a fast, internally cold, mass-selected negative ion beam is a not a highly efficient means to produce free radicals, this has not proven to be a limitation. In particular, the coincidence detection scheme processes only one dissociation event per laser shot; for this reason copious radical production would not benefit us.

Two factors prevent our experiment from being completely universal once the radical beam has been generated. The accuracy of our kinetic energy release measurement requires that the lifetime of the dissociating state not be exceedingly long, since dissociation must occur within a small fraction of the total flight time between the photodissociation region and the detector. For example, N_3 at a beam energy of 8 keV has a center-of-mass flight time (using our standard 1 m flight length) of ~5μs. Provided the upper state dissociation lifetime is < 30 ns, no noticeable degradation of resolution will result. These possible lifetime effects can be investigated by performing the experiment with various beam energies or flight lengths. The lifetime 'restriction' is quite reasonable, and has not been an issue thus far.

The second, more strict requirement for any radical system that is to be studied is that dissociation must occur to give products of mass ratio less than approximately four to one. Higher mass ratios would result in unfavorable kinematics: either the light fragment would recoil far enough that it missed the detector, or the heavy fragment would not scatter out of the beam axis enough to avoid the beam block designed to prevent

undissociated radicals from being detected. This effectively precludes investigation of systems where a hydrogen atom is lost, at least in the current design of the instrument.

In the following section, a general overview of the spectroscopic and dynamical information we gain from our photodissociation experiments is given, and the relation to the potential energy surfaces involved in the dissociation is developed.

1.3 Photodissociation Spectroscopy and Dynamics

When embarking on the investigation of a given free radical, we initially wish to determine the locations of the dissociative electronic states. Once an electronic transition has occurred[28] that places a radical in an excited state at an energy above the lowest dissociation limit, dissociation will occur if a facile pathway exists. This dissociation process may be either direct or involve predissociation; the nature of the dissociation process can be often be discerned through the spectroscopy of the dissociative excited state. This subject is now briefly considered.

1.3.1 The Spectroscopy of Dissociative Excited States

By their very nature, conventional spectroscopic studies of free radicals provide the most information about bound ro-vibrational levels within particular electronic states. Knowledge of dissociative states of free radicals is limited to the occasional reported observation of broad absorption continua, broadened absorption lines, or the breaking off of signal in certain regions of emission spectra. From many of these studies, particularly those involving absorption,[29] it is even difficult to accurately determine the onset of dissociation.

As mentioned above, our first goal is to determine the spectroscopy of the dissociative electronic states to be investigated. We thus acquire the total photodissociation cross section spectrum of the radical by measuring the fragment flux as a function of dissociation photon energy. The exact details of our experimental method

are given in Chapter 2, but these spectra effectively map out the dissociative electronic states of the radical in the photon energy range that we use. When such information already exists, we are guided by previously reported observations of dissociative states. Two types of dissociation processes, resulting in very different photodissociation cross section spectra, can occur.

Direct dissociation involves excitation to a purely repulsive region in one coordinate of an electronic surface. Depending on the presence or absence of active vibrational modes perpendicular to the dissociation coordinate, and the rate of dissociation, direct dissociation may contain some broad structure associated with these bound modes. In any case, the excited molecule immediately begins to form products as it distorts irreversibly along the dissociation coordinate.

Predissociation, an example of which is illustrated by Figure 1.1,[30] involves excitation to a particular combination of vibrational levels of an excited electronic state. Depending on the type of predissociation occurring, this system may subsequently couple to a directly repulsive electronic state. Alternately, the system might undergo intramolecular vibrational redistribution (IVR) until enough energy to cause dissociation appears in the vibrational degree of freedom that is the dissociation coordinate (this mechanism often follows internal conversion from the initially excited state to the ground electronic state). The strength of the interaction with the dissociative state, or the rate of IVR, determines the lifetime of the initially prepared states, and correspondingly, the absorption linewidths associated with transitions to these states (the lifetime broadening is, in some sense, a manifestation of the Heisenberg Uncertainty Principle). These absorption lines will be observed as structure in our photodissociation cross section spectra.

Figure 1.1 A diagram showing a predissociative system, and the structure which can appear in the total cross section spectrum of such a system. As with all of the diagrams showing potential energy curves, the radical AB is not restricted to be a diatomic, so fragment A and/or B may represent polyatomic molecules.

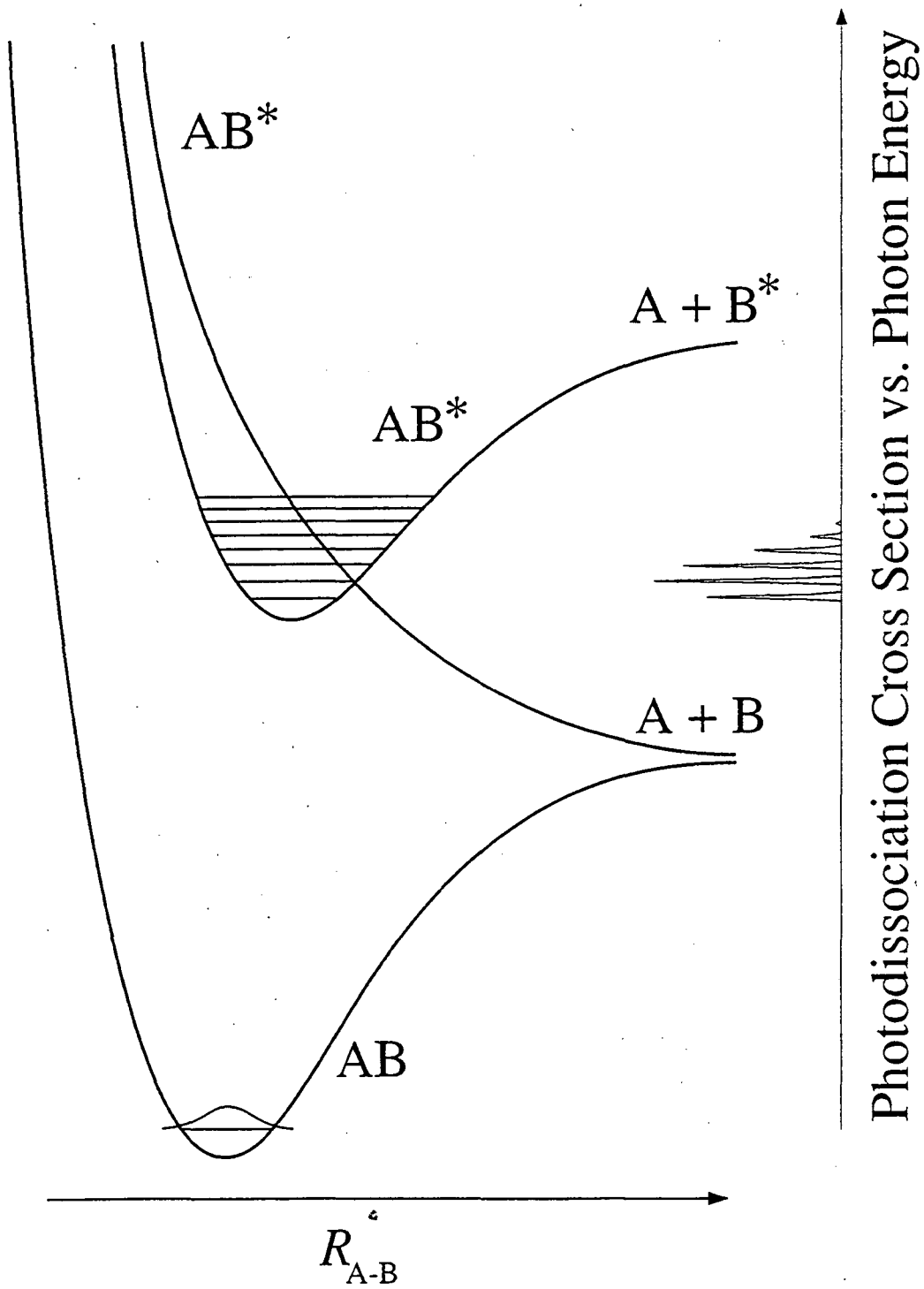

AB*
A + B*
AB*
A + B
AB
$R_{A\text{-}B}$
Photodissociation Cross Section vs. Photon Energy

Figure 1.1

Unstructured photodissociation cross section spectra therefore imply either an extremely fast predissociation process or excitation to a directly repulsive state (discussed above).

It is important to remember that the photodissociation cross section measurements that are obtained in this experiment are what is known as 'action spectra'. Action spectra contain signal that is the result of a particular process (in our case dissociation) which occurs following absorption of a photon. However, quite often the quantum yield for the process that is monitored is not unity, and may vary with wavelength. For example, fluorescence may effectively compete with dissociation at certain photon energies when long-lived predissociative excited states are involved. Therefore, in general, the intensities found in photodissociation cross section spectra must not be equated with the absorption intensity.

Once the spectroscopy of a particular dissociative state has been determined, it is then of interest to determine several key aspects of the dissociation that is occurring, such as: the identity and amount of each product formed; the product translational, electronic, vibrational and rotational (resolution permitting) energies; the angular distribution of the products; and the dissociation lifetimes. A discussion of the extraction of this information from our photodissociation dynamics experiments follows.

1.3.2 Photodissociation Dynamics

The quantities of interest in a photodissociation dynamics experiment are the true photofragment center-of-mass translational energy and angular distributions. For a one-photon dissociation the form of the photofragment energy and angular distribution $\mathcal{P}(E_T,\theta)$ can be written as

$$\mathcal{P}(E_T,\theta) = P(E_T)[1 + \beta(E_T)P_2(\cos\theta)] \qquad (1.2)$$

where θ is the angle between the photofragment recoil vector and the electric vector of the dissociation laser, $P_2(\cos\theta)$ is the second Legendre polynomial, $\beta(E_T)$ is the (energy-

dependent) anisotropy parameter, and $P(E_T)$ is the angle-integrated kinetic energy distribution. The significance of the anisotropy parameter, which describes the form of the recoil angular distribution, will be discussed below. At this point we will examine the relevance of knowing the translational energy distribution.

With reference to Figure 1.2, it can be seen that Eqn. (1.3) describes the balance of energy in a photodissociation experiment.

$$E_{INT,AB} + E_{h\nu} = E_{INT,A} + E_{INT,B} + E_T + D_0 \qquad (1.3)$$

Here $E_{h\nu}$ represents the photon energy, E_T the translational energy (kinetic energy release) and $E_{INT,AB}$, $E_{INT,A}$, and $E_{INT,B}$ are the internal energies of the parent radical and the two photofragments, respectively. As will be described, in our experiment great effort is expended to insure $E_{INT,AB} \approx 0$, and for the subsequent discussion it will be assumed to be zero. D_0 represents the energy required to form the dissociation products in their ground states from ground state radicals at 0 K.

As mentioned earlier, the energetics of free radicals are often somewhat uncertain; for example, large error bars in free radical heats of formation are quite common. However, products from the dissociation of such a radical usually have more well-defined energetics (i.e., the heat of formation of AB might be fairly uncertain, while the heats of formation of A and B could be among the most precisely known.) Because of this, photodissociation studies provide a method with which to investigate the energetics of free radicals. As Eqn. (1.4) shows,

---

Figure 1.2 The energetics involved in a dissociation process. Note the energy partitioning which occurs between the internal degrees of freedom and the relative translation of the product fragments. On the right is a hypothetical translational energy distribution that might be appropriate for dissociation of a triatomic, with a diatomic and an atom as products. The diatomic is meant to have a long vibrational progression excited, with a rotational contour giving shape to the vibrational peaks.

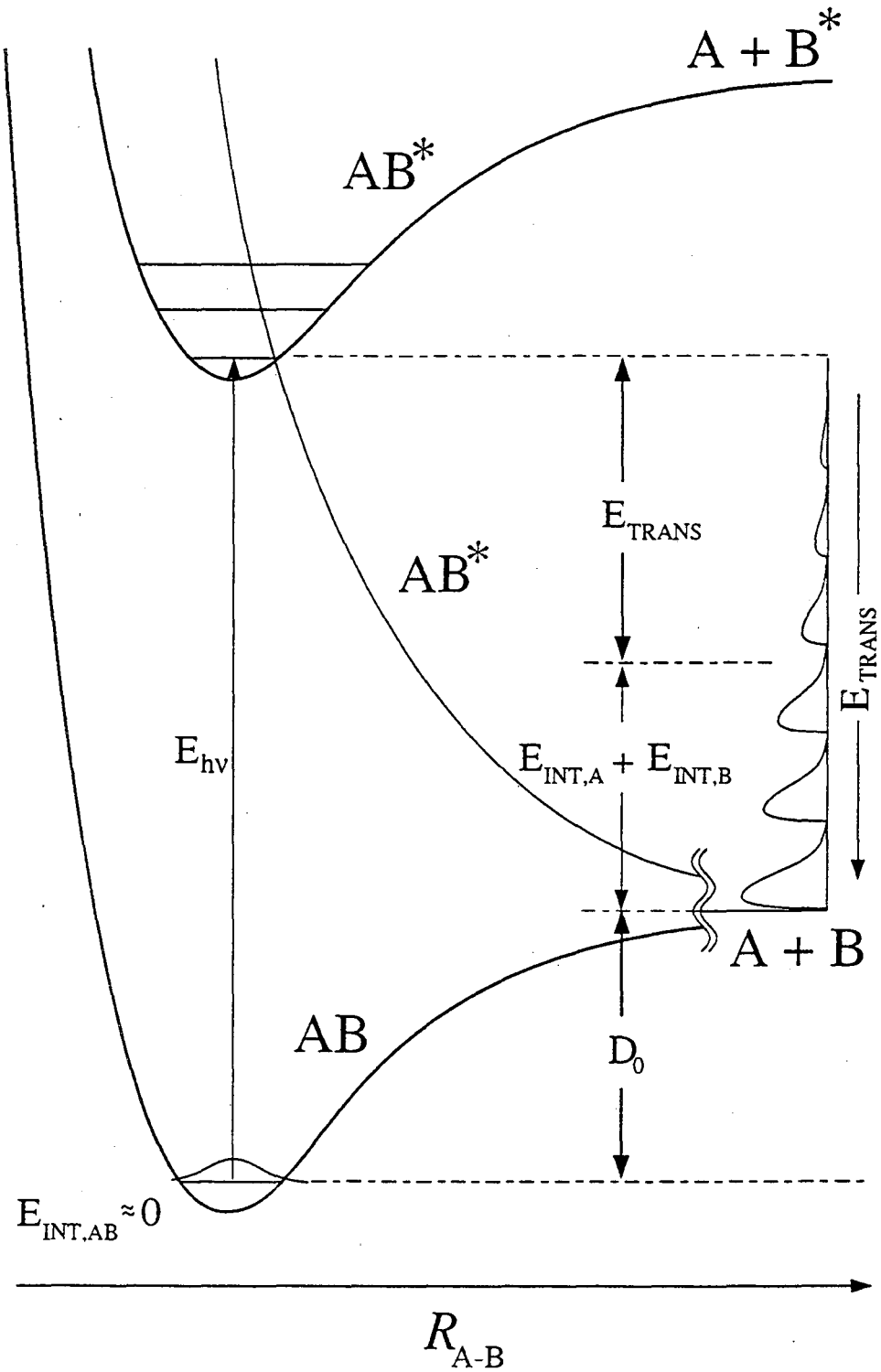


Figure 1.2

$$\Delta_f H^0_{0K}(\mathrm{AB}) = \Delta_f H^0_{0K}(\mathrm{A}) + \Delta_f H^0_{0K}(\mathrm{B}) - D_0(\mathrm{A-B}) \qquad (1.4)$$

the quantity linking the heats of formation values is the bond dissociation energy, D_0, for the bond that is broken.

By measuring the kinetic energy release distribution for a given process, we can discern the maximum kinetic energy that is released. If we assume this kinetic energy release corresponds to the production of photofragments with no internal energy, a upper limit of the bond dissociation energy, D_0, can be obtained via Eqn. (1.5):

$$D_0 \leq E_{h\nu} - E_{T,MAX}. \qquad (1.5)$$

Then, via Eqn. (1.4), a lower limit of the heat of formation for the radical of interest is found. A variation of this method is used in Chapter 4, where it is determined that the N_3 radical is 0.05 kcal/mol unstable with respect to ground state N_2 + N, and in Chapter 5 to determine a value for the heat of formation of NCO, which until our study had been generally overestimated by >5 kcal/mol.[31,32]

From the translational energy distributions measured in our experiments one can infer the internal energies of the products. Rearranging Eqn. (1.3) we get

$$E_{AVAIL} = E_{INT,A} + E_{INT,B} + E_T = E_{h\nu} - D_0, \qquad (1.6)$$

where we have defined the energy available for distribution amongst the product translational and internal energy degrees of freedom as the photon energy less the bond dissociation energy, D_0, determined above (or known previously). Therefore, the internal energy distribution of the products is simply the available energy minus the translational energy distribution. We now have detailed knowledge of the energy disposal in the dissociation under investigation. This partitioning of energy into translation and the various internal degrees of freedom reflects the dynamics of the dissociation process. As will be seen in the next section, the observed dynamics relates directly to the nature of the potential energy surface(s) involved in the dissociation.

1.3.3 Excited State Potential Energy Surfaces

The translational energy distributions of the products allow insight into the characteristics of the excited state potential energy surfaces involved in the dissociation process. For example, if excitation occurs to a directly repulsive surface, roughly half of the available energy is channeled in the relative translation of the products. In contrast, a simple bond rupture occurring on the electronic ground state with no exit channel barrier will result in very little kinetic energy release; the translational energy distribution will peak very close to zero (most of the available energy remains in the internal degrees of freedom). This behavior is identical whether the original excitation occurred within the ground electronic state, via infrared multiphoton photodissociation, or to an excited electronic state which undergoes internal conversion to vibrationally excited levels of the ground state. Molecular elimination channels have a large barrier in addition to any exothermicity. This results in a large fraction of the barrier being released into translation as the closed shell product repels the remaining fragment on the way down the exit channel of the potential energy surface.

When the product fragments are small, there is a limited number of internal degrees of freedom that may be populated. In favorable cases (relatively few vibrational modes present or active in the products) our translational energy resolution is sufficient to determine populations of the product vibrational levels, and we can estimate the extent of the rotational excitation by the shape of the rotational contour (i.e., we can determine the value of the most probable rotational quantum number, N). These considerations allow us to comment in even greater detail on the dissociation dynamics of such a radical. For example, a radical that predissociates following an electronic transition from a linear ground state to a linear excited state, yet produces a highly rotationally excited photofragment, must sample bent geometries as it dissociates.

As can be seen from the above, the translational energy distribution allows a great deal of insight into the nature of the potential energy surfaces involved in the dissociation.

One last source of dynamical information is available to us, and is discussed in the following section.

1.3.4 Photofragment Recoil Angular Distributions

As mentioned at the beginning of this section, in addition to the masses of the photofragments and the kinetic energy released in the dissociation process, our experiment also reveals the angle of the photofragment recoil with respect to the electric field vector $\bar{E}$ of the linearly polarized laser light (see Figure 1.3). This additional information allows further insight into the nature of the dissociative state under consideration,[33] since recoil angle distributions contain dynamical information, and can identify the symmetry of an upper state in a dissociative transition.

For a one photon allowed transition the probability is proportional to $\left|\bar{\mu}\cdot\bar{E}\right|^2$, or $\mu^2E^2\cdot\cos^2\theta$, where $\bar{\mu}$ is the transition moment vector, $\bar{E}$ is the electric field vector of the polarized light, and θ is the angle subtended by the two vectors (note that the transition probability is azimuthally symmetric about the electric field vector). In a linear molecule, for example, the transition moment is restricted to lying either along or perpendicular to the molecular axis. Of course, the transition moment has a well-defined set of possible orientations in other symmetry point groups as well.

If dissociation is prompt, that is, recoil occurs before rotation of the molecular frame, the photofragment recoil angular distribution will reflect the initial distribution of the molecular axes on excitation. For example, in the case of a linear molecule[34] having a transition moment parallel to the molecular axis, a prompt dissociation would result in a $\cos^2\theta$ distribution of photofragment recoil vectors, while if the transition moment were perpendicular to the molecular axis, a prompt dissociation would result in a $\sin^2\theta$ distribution of photofragment recoil vectors.

Figure 1.3 A diagram showing the quantities determined in our experiment.

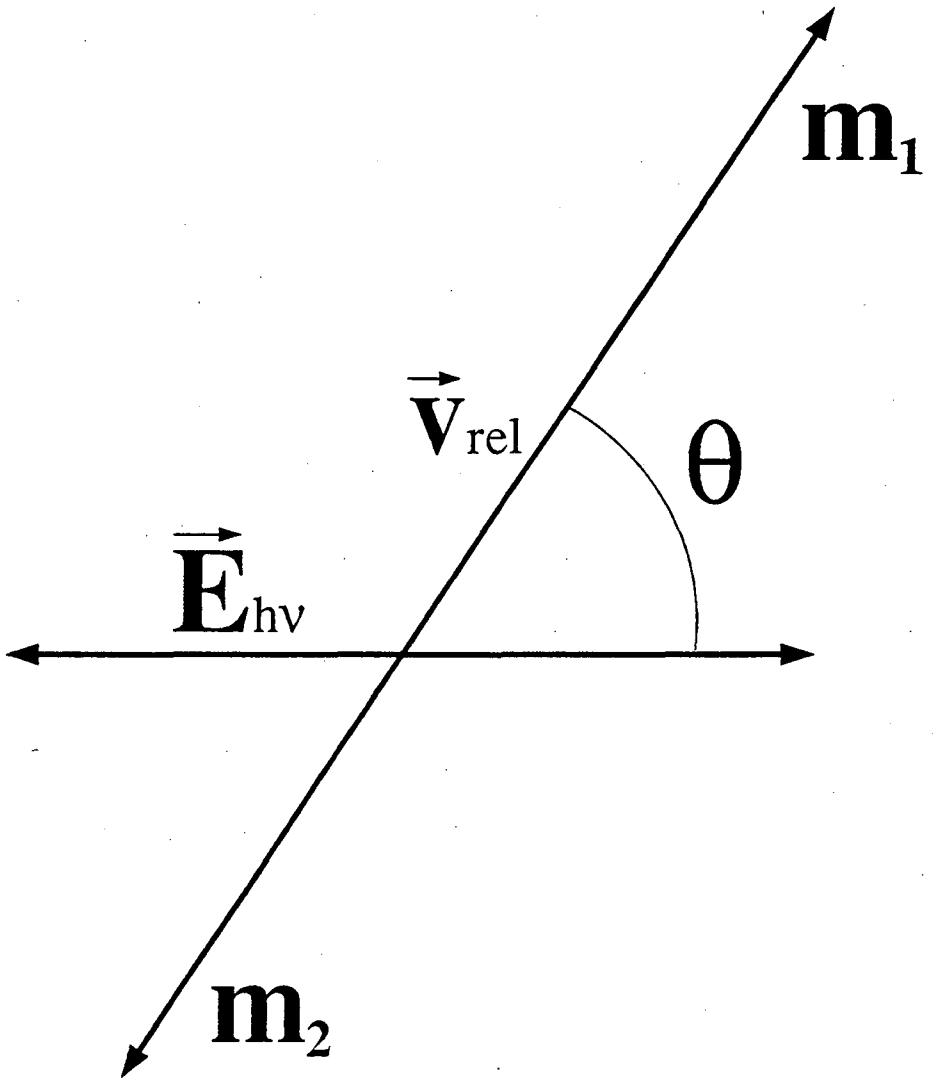


Figure 1.3

If, however, the excited state has a lifetime on the order of the period of a molecular rotation, the photofragment recoil angular distribution will appear more isotropic because of rotational averaging. Thus, the photofragment angular distribution is sensitive to both the electronic symmetry and the lifetime with respect to dissociation of the upper state.

A standard way of expressing the anisotropy in the recoil angular distribution has been developed by Zare.[33] This method expresses the probability of recoil at an angle theta with respect to the electric field vector $\vec{E}$ of the polarized dissociation light as

$$I(\theta) = \frac{1}{4\pi}[1 + \beta \cdot P_2(\cos\theta)], \tag{1.7}$$

where $P_2(\cos\theta)$ is the second Legendre polynomial in $\cos\theta$

$$P_2(\cos\theta) = \frac{3\cos^2\theta - 1}{2}. \tag{1.8}$$

β is referred to as the anisotropy parameter, and it ranges from +2 to –1. A $\cos^2\theta$ distribution corresponds to β parameter of +2, a $\sin^2\theta$ distribution corresponds to a β parameter of –1, and an isotropic distribution corresponds to a β parameter of 0.

In our experiment, we have the ability not only to determine a β parameter for a given dissociative transition, but to determine β as a function of photofragment kinetic energy release from this transition. $\beta(E_T)$ is potentially much more informative. For example, a change in dissociation pathway when forming a given product in a different electronic state may be signaled by a difference in the anisotropy parameter associated with the kinetic energy release values characteristic of each channel.

1.4 Photodetachment as a Source of Free Radicals

The main challenge in photodissociation studies of reactive free radicals is preparation of a pure, internally cold sample of sufficiently high number density.

Traditional sources of free radicals usually cause a stable, closed-shell, precursor molecule to dissociate, forming the reactive free radical as a product. Such sources use techniques such as flash photolysis, in which photodissociation is used to produce the radical of interest, and electric or microwave discharges, which also result in dissociation. These types of sources suffer from two severe difficulties for most applications. First, they are not very selective, producing many by-products, often mixed with unconverted precursor. This mixture will complicate matters when an experiment is subsequently performed: the identity of the signal carrier may be ambiguous. The second difficulty is a consequence of the environment present in these energetic sources. The processes at work in these sources form radicals by adding enough energy to the precursor to break a chemical bond. Unfortunately, energy is generally not deposited into the precursor in a specific manner, and the resulting radicals are formed with a great deal of internal energy. On rare occasions this can be an advantage, i.e., when the spectroscopy of highly vibrationally and rotationally excited levels of the radical is to be investigated. However, in general this is highly undesirable. In particular, if the energetics of a particular radical are to be investigated, one would prefer as small an internal energy distribution as possible in the newly formed radical. This provides more well-defined initial conditions, allowing a more accurate accounting of the energy in the experiment that is subsequently performed.

There have been recent attempts elsewhere to construct free radical sources which typically couple the methods mentioned above to a free jet expansion, thereby obtaining a cold, if often impure, source of free radicals.

Photolytic free radial generation with subsequent free jet expansion is one such method which has undergone development by a number of research groups in the past ten to 15 years. As mentioned in § 1.1, this technique has been used in some of the previous photofragmentation studies of free radicals. The general method is explained in a review by Miller[35] who has successfully applied it to the spectroscopic study of many radicals under free jet expansion conditions.

As will be discussed more thoroughly in § 2.3.5, pulsed discharges have also been coupled to pulsed free jet expansions in an effort to form free radicals. In particular, more exotic species, with no obvious photolytic precursor, can often be made in this manner.

Another promising alternative radical source has been developed by Chen and co-workers. They have developed a source[36] that works quite well for selected cases of radicals.[37] They employ a flash pyrolysis source, having a heated tip on the end of a pulsed free jet expansion. By synthesizing precursors with carefully chosen substituents, nearly quantitative yields of the desired free radical can be formed in the heated section, and internal energy present in these radicals is then removed during the molecular beam expansion. This technique is limited, however, by the ability to identify and then synthesize a precursor which dissociates quantitatively at the appropriate bond on undergoing flash pyrolysis.

Our approach to the difficult problem of preparing reactive free radicals with the goals of highest purity and minimal internal energy is to photodetach the negative ion corresponding to the radical of interest (specifically, the ion having the same chemical structure, but with one extra electron). Since negative ions can be mass selected, an immense benefit of this method is that the radical of interest can be generated exclusively via photodetachment, completely free of impurities (including unconverted precursor). In addition, prior to detachment the ion can be accelerated to high laboratory kinetic energy, permitting uncomplicated detection of the eventual photofragments; the detection efficiency of microchannel plates for neutral particles with > 1 keV kinetic energy is > 50%.[26]

The possible advantages of this approach make it seem very attractive. However, it is necessary to have the capability of producing sufficient quantities of the negative ion corresponding to the radical of interest. Because of their open-shell nature, free radicals tend to have sizable positive electron affinities, particularly if they contain one or more electronegative atom, and/or can benefit from resonance stabilization of the negative

charge. This allows negative ion photodetachment to be a very general technique. The method by which we make internally cold negative ions involves crossing an electron beam with the high density region of a pulsed free jet expansion. The details of this source are discussed in § 2.3 and subsections therein. The important consideration for the discussion that follows, however, is that we have the ability to make a sufficient number density of many types of negative ions, typically populating the ground vibrational level and lowest rotational states following expansion.

1.4.1 Photodetachment Considerations

The electronic selection rules for photodetachment simply require a one-electron process (i.e., the neutral must be one that can be made by the removal of one electron from the negative ion). For this reason, the ground state of the radical is always an allowed final state in the detachment of an electron from the corresponding negative ion when the anion is closed-shell, and very often an allowed final state otherwise.

The proper photon energy is dictated by two factors. Contrary to the photoionization behavior of neutrals[38] (where the threshold cross section behaves like a step function), the photodetachment cross-section for negative ions always rises as some function of the energy just above threshold.[39,40] For this reason, it is beneficial to be as far above threshold as is practical when photodetaching electrons from negative ions.

However, the second factor that is important in choosing the photodetachment photon energy is the extent of any geometry change on going from the negative ion to the neutral. While the full derivation of all considerations involved in photoelectron spectroscopy will not be presented here, one key characteristic of this process must be appreciated. The photodetachment process occurs on a time scale short with respect to vibrational motion. The vibrational population of the radicals formed by photodetachment of a vibrationally cold negative ion is thus determined chiefly by the Franck-Condon overlap between the v"=0 level of the negative ion and the various energetically accessible

vibrational levels of the radical. When determining Franck-Condon factors it is common to assume the electronic dipole term is invariant within a given electronic state. The relative intensities of the vibrational levels are therefore computed using the relation

$$I_{v'} \propto \left| \left\langle \Psi_{v'}^{0}(R) | \Psi_{v''=0}^{-}(R) \right\rangle \right|^2, \tag{1.9}$$

where $\Psi_{v''=0}^{-}(R)$ is the vibrational wavefunction of the ground vibrational level of the negative ion, and $\Psi_{v'}^{0}(R)$ are the vibrational wavefunctions of the neutral radical. In pursuing the goal of preparing internally cold radicals via photodetachment of an electron from a negative ion two general cases are possible, as shown schematically in Figure 1.4 and discussed below.

If the change in geometry on photodetachment is minimal, as might be expected for the removal of a nonbonding electron, then the Franck-Condon factors for $\Delta v \neq 0$ transitions will be quite small. In the limit of virtually no geometry change and very small changes in frequency, the Franck Condon factors will be such that the ground vibrational level of the radical will be the only vibrational level populated from the ground vibrational level of the anion. This will hold true even when the detachment photon energy is well above threshold. The first panel in Figure 1.4 illustrates this case, showing how in this situation, it is wise to take advantage of the larger photodetachment cross section expected further above threshold by using as energetic a detachment photon as is practical.

If, on the other hand, the change in geometry on detachment is substantial, the Franck-Condon factors will be large for transitions to excited radical vibrational levels in the one or more modes which most closely resemble the movement of the nuclei. As a result, if the photon energy is far above photodetachment threshold a manifold of radical

---

Figure 1.4 A diagram showing the two different cases of photodetachment with respect to geometry change between the negative ion and the radical. See text.

No Geometry Change
upon Photodetachment
X
hν
X⁻
ΔQ
Electron Kinetic Energy
Some Geometry Change
upon Photodetachment
hν₁
X*
hν₂
X⁻
ΔQ
Electron Kinetic Energy

Figure 1.4

vibrational levels will be populated. As the second panel of Figure 1.4 shows, there is nevertheless a strategy by which radicals can be formed exclusively in the ground vibrational level, even in the case of substantial geometry change. This strategy entails limiting the detachment photon energy to the threshold energy plus an amount just less than the least energetic active vibrational mode (as shown by $h\nu_2$ in Figure 1.4).

In short, judicious choice of photodetachment photon energy will typically allow photodetachment yield to be maximized while resulting in no additional excitation of the neutral radical which is formed.

Occasionally, the Franck-Condon factors for the 0-0 transition between the negative ion and the neutral radical will be unfavorable, and the photoelectron spectrum will show only a broad progression in one or more modes, with no evidence for the origin. In this case, it must be decided if uncertainty in the internal energy of the radicals prior to photodissociation is acceptable. Since there is no shortage of systems that are completely amenable to our photodetachment scheme, this case has yet to be explored. However, the diminished ability to form free radicals in a well-defined initial state is expected to reduce the clarity of subsequent photofragment translational spectroscopy results in a certain number of cases, since the total energy within a given dissociating radical is somewhat unclear. An exception to this expectation are systems that have a structured total photodissociation cross section spectrum (see § 4.1 and § 4.2). In this case, the initial energy found in the radical on photoexcitation will still be well-defined, even if a variety of vibrational levels in the ground state of the radical are populated via photodetachment.

1.4.2 Examples

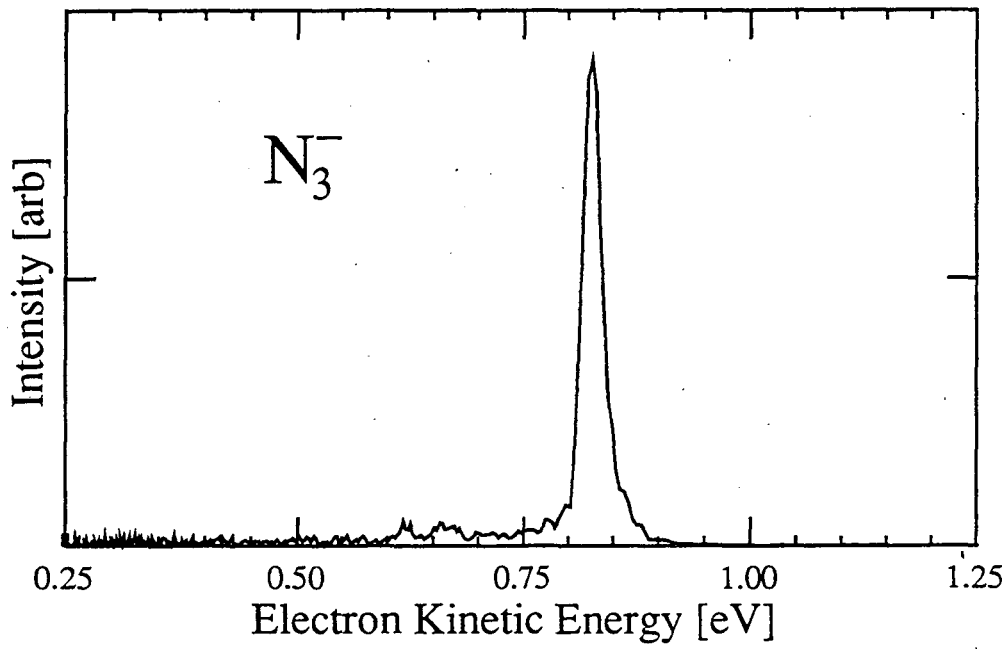
Examples of the two photodetachment cases outlined above are now given. The first example illustrates the case involving a negative ion which has very little geometry change upon photodetachment of an electron. N_3^- is a 'textbook example' of this case because of the non-bonding nature of the electron that is detached to form the ground

state N_3 radical. Rotationally resolved spectra of the N_3 radical have been obtained,[41] and more recently vibration-rotation spectra were obtained for N_3^-.[42] These spectra show that both the radical and negative ion are linear, with a very small geometry change upon detachment.

The photoelectron spectrum of N_3^-, shown in the top panel of Figure 1.5, was recorded[43] with a separate fixed-frequency negative ion photoelectron apparatus[44] using 355 nm (3.49 eV) photons. The lone feature in the spectrum corresponds mainly to the 000←000 (often abbreviated as 0-0) transition between the negative ion and the neutral free radical. The minor contribution of sequence bands involving the relatively low frequency bending mode (i.e., 010←010, 020←020, etc.) is suggested by the width of this peak, and eventually confirmed in the total photodissociation cross section spectrum (see § 4.3 and Figure 4.6).

The photoelectron spectrum indicates that virtually no vibrational excitation of the neutral will be occur on photodetachment of the ground vibrational level of the negative ion, even with a photon energy well above the photodetachment threshold. In preparing N_3 it was typical to photodetach N_3^- using the peak of the PTP dye curve, at 343 nm. This corresponds to a photon energy of 3.62 eV, well above the measured 2.68 eV electron affinity.[43] The benefit of detaching this far above threshold is enhanced photodetachment cross section, as described above. Detachment efficiencies of up to 80%

Figure 1.5 Negative ion photoelectron spectra illustrating the two cases with respect to geometry change between the negative ion and the neutral radical.
Top panel: The 355 nm (3.49 eV) photoelectron spectrum of N_3^-.[43] The photon energy used for detachment to form N_3 in the present experiment is ~0.94 eV above the detachment threshold.
Bottom panel: The 266 nm (4.66 eV) photoelectron spectrum of NCO^-.[45] In contrast to the strategy employed in the case of N_3^- photodetachment, the photon energy used for detachment to form NCO in the present experiment is only ~0.05 eV above the detachment threshold, as indicated by the arrow.


N_3^-
Intensity [arb]
0.25
0.50
0.75
1.00
1.25
Electron Kinetic Energy [eV]

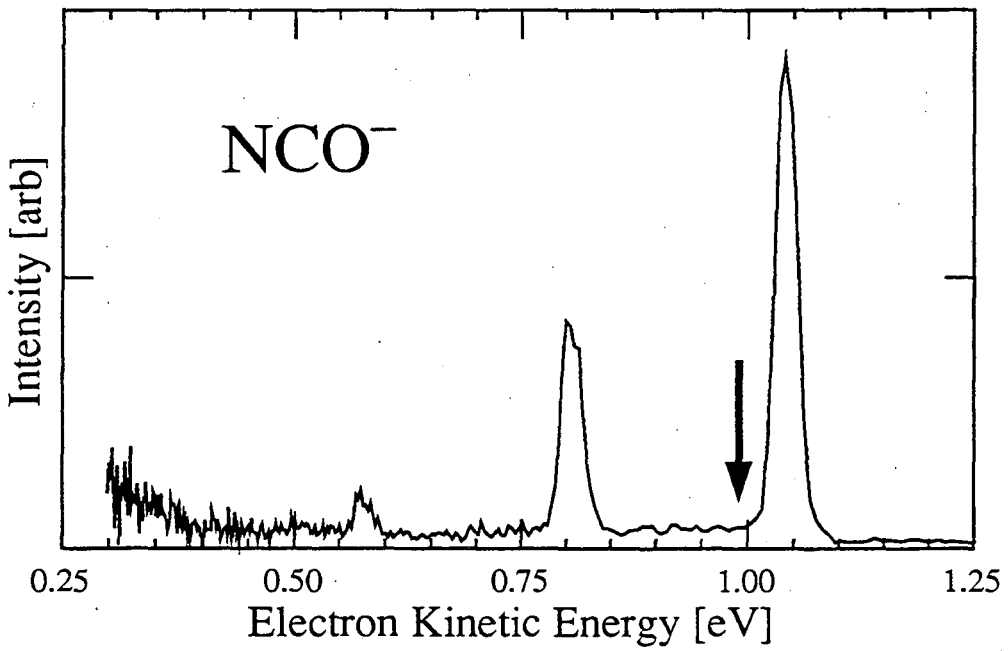

NCO^-
Intensity [arb]
0.25
0.50
0.75
1.00
1.25
Electron Kinetic Energy [eV]

Figure 1.5

could be observed for N_3^- with fresh excimer and dye lasers.

An second photodetachment example illustrates the case where geometry change does occur upon photodetachment. The 266 nm (4.66 eV) photoelectron spectrum of NCO^- reported by Bradforth et al.,[45] reproduced in the bottom panel of Figure 1.5, shows a progression in the ν_3 mode. In order to avoid vibrational excitation in the radicals formed via photodetachment, the photon energy must be limited to the electron affinity (determined by Bradforth et al. to be 3.609 eV) plus an amount just less than the ν_3 fundamental (~1950 cm^{-1}). In our photodissociation studies of NCO, to be described in chapter 5, we therefore use 339 nm (3.66 eV) photons to detach the electron from NCO^-. While slightly more energetic photons could have conceivably been used without exceeding our self-imposed detachment photon energy limit, the maximum for the product of the dye curve and the detachment cross section was found to be at 339 nm.

1.5 References

[1] G. Hertzberg, *The Spectra and Structure of Simple Free Radicals*, (Cornell University Press, Ithica, NY, 1971); J. K. Kochi, Ed. *Free Radicals*, (Wiley, New York, 1973) Vols. I and II.; J. M. Hay, *Reactive Free Radicals*, (Academic Press, London, 1974).

[2] For a recent review see M. N. R. Ashfold, S. G. Clement, J. D. Howe, and C. M. Western, J. Chem. Soc. Faraday Trans. **89**, 1153 (1993).

[3] A fairly lengthy article entitled 'Spectra and Structure of Gaseous Organic Free Radicals: A Status Report', S. C. Foster and T. A. Miller, J. Phys. Chem. 93, 5986 (1989) attests to the recent progress made in this area.

[4] E. J. Hintsa, X. Zhao, W. M. Jackson, W. B. Miller, A. M. Wodtke, and Y. T. Lee, J. Phys. Chem. **95**, 2799 (1991).

[5] H. F. Davis and Y. T. Lee, J. Phys. Chem. **96**, 5681 (1992).

[6] H. F. Davis, B. Kim, H. S. Johnston, and Y. T. Lee, J. Phys. Chem. **97**, 2172 (1993).

[7] A. Weaver, D. W. Arnold, S. E. Bradforth, and D. M. Neumark, J. Chem. Phys. **94**, 1740 (1991).

[8] A. M. Wodtke and Y. T. Lee, J. Chem. Phys. **89**, 4744 (1988).

[9] R. E. Continetti, B. A. Balko, and Y. T. Lee, Chem Phys. Lett. **182**, 400 (1991).

[10] X. Liu and R. D. Coombe, J. Chem. Phys. **91**, 7534 (1989).

[11] S. H. Kable, J. C. Loison, and P. H. Houston, J. Chem. Phys., **92**, 6332 (1990); J. C. Loison, S. H. Kable, P. L. Houston, and I. Burak, J. Chem. Phys. **94**, 1796 (1991); S. H. Kable, J. C. Loison, D. W. Neyer, P. H. Houston, I. Burak, and R. N. Dixon, J. Phys. Chem., **95**, 8013 (1991); D. W. Neyer, S. H. Kable, J. C. Loison, P. H. Houston, I. Burak, and E. M. Goldfield, J. Chem. Phys., **97**, 9036 (1992). D. W. Neyer, X. Luo, P. H. Houston, and I. Burak, J. Chem. Phys., **98**, 5095 (1993).

[12] R. N. Dixon, Mol. Phys. **54**, 333 (1985).

[13] For reviews of the Renner-Teller effect, see Ch. Jungen and A. J. Merer in *Modern Spectroscopy: Modern Research*, K. N. Rao, Ed., (Academic, New York, 1976), Vol. 2, Chapter 2; J. M. Brown and F. Jørgensen, Adv. Chem. Phys. **52**, 117 (1983).

[14] S. Nourbakhsh, K. Norwood, H.-M. Yin, C.-L. Liao, and C. Y. Ng. J. Chem. Phys. **95**, 946 (1991); C.-W. Hsu, C.-L. Liao, Z.-X. Ma, P. J. H. Tjossem, and C. Y. Ng, Chem. Phys. Lett. **199**, 78 (1992).

[15] S. Nourbakhsh, C.-L. Liao, and C. Y. Ng. J. Chem. Phys. **92**, 6587 (1990); C.-W. Hsu, C.-L. Liao, Z.-X. Ma, P. J. H. Tjossem, and C. Y. Ng, J. Chem. Phys. **97**, 6283 (1992).

[16] J. Chen and P. J. Dagdigian, J. Chem. Phys. **98**, 3554 (1993).

[17] A. M. Wodtke and Y. T. Lee in *Molecular Photodissociation Dynamics*, M. N. R. Ashfold, J. E. Baggot, Eds. (Royal Society of Chemistry, London, 1987), p. 31.

[18] For a recent review, see M. N. R. Ashfold, I. R. Lambert, D. H. Mordaunt, G. P. Morley, and C. M. Western, J. Phys. Chem. **96**, 2938 (1992).

[19] G. E. Busch, J. F. Cornelius, R. T. Mahoney, R. I. Morse, D. W. Schlosser, and K. R. Wilson, Rev. Sci. Instrum. **41**, 1066, (1970).

[20] D. W. Chandler and P. L. Houston, J. Chem. Phys. **87**, 1445, (1987); D. W. Chandler, J. M. Thoman, Jr., M. H. M. Janssen, D. H. Parker, Chem. Phys. Lett. **156**, 151 (1989); D. W. Chandler, M. H. M. Janssen, S. Stolte, R. N. Strickland, J. W. Thoman, Jr., D. H. Parker, J. Phys. Chem. **94**, 4839 (1990); D. P. Baldwin, M. A. Buntine, and D. W. Chandler, J. Chem. Phys. **93**, 6578 (1990); M. A. Buntine, D. P. Baldwin, R. N. Zare, and D. W. Chandler, J. Chem. Phys. **94**, (1991); M. H. M. Janssen, D. H. Parker, G. O. Sitz, S. Stolte, D. W. Chandler, J. Phys. Chem. **95**, 8807 (1991); M. A. Buntine, D. P. Baldwin, and D. W. Chandler, J. Chem. Phys. **96**, 5843 (1992); W. P. Hess, D. W. Chandler, and J. W. Thoman Jr., Chem. Phys. **163**, 277 (1992); T. Suzuki, V. P. Hradil, S. A. Hewitt, P. L. Houston, and B. J. Whitaker, Chem. Phys. Lett. **187**, 257 (1991); A. G. Suits, R. L. Miller, L. S. Bontuyan, and P. L. Houston, J. Chem. Soc. Faraday Trans. **89**, 1443, (1993); V. P. Hradil, T. Suzuki, S. A. Hewitt, P. L. Houston, and B. J. Whitaker, J. Chem. Phys. **99**, 4455 (1993).

[21] J. Biesner, L. Schnieder, G. Ahlers, X. Xie, K. H. Welge, M. N. R. Ashfold, and R. N. Dixon, J. Chem. Phys. **91**, 2901 (1989); X. Xie, L. Schnieder, H. Wallmeier, R. Boettner, K. H. Welge, and M. N. R. Ashfold, J. Chem. Phys. **92**, 1608 (1990); L. Schnieder, W. Meier, K. H. Welge, M. N. R. Ashfold, and C. M. Western, J. Chem. Phys. **92**, 7027 (1990); G. P. Morley, I. R. Lambert, M. N. R. Ashfold, K. N. Rosser, and C. M. Western, J. Chem. Phys. **97**, 3157 (1992); D. H. Mordaunt, I. R. Lambert, G. P. Morley, M. N. R. Ashfold, R. N. Dixon, C. M. Western, L. Schnieder, and K. H. Welge, J. Chem. Phys. **98**, 2054 (1993).

[22] D. P. DeBruijn and J. Los, Rev. Sci. Instrum. **53**, 1020 (1982).

[23] R. E. Continetti, D. R. Cyr, D. L. Osborn, D. J. Leahy, and D. M. Neumark, J. Chem. Phys. **99**, 2616 (1993).

[24] L. D. A. Siebbeles, J. M. Schins. W. J. Van der Zande, J. Los, and M. Glass-Maujean, Phys. Rev. A **44**, 343 (1991), and references therein.

[25] H. Helm and P. C. Cosby, J. Chem. Phys. **86**, 6813 (1987); H. Helm and P. C. Cosby, J. Chem. Phys. **90**, 4208 (1989).

[26] See Chapter 1 of *Microchannel Plate Report*, H. Kersten, Ed., (FOM-Instituut voor Atoom- en Molecuulfysica, Amsterdam, 1982), p. 32.

[27] D. J. Leahy, D. R. Cyr, D. L. Osborn and D. M. Neumark, Chem. Phys. Lett. (in press).

[28] This process is subject to selection rules, which must be satisfied for absorption to occur. These rules determine the spectroscopy associated with a given radical, and derive from the consideration of the quantum states and symmetries of the initial and final states in a given transition. See, for example, the discussion of this subject in reference 29.

[29] G. Herzberg, *Molecular Spectra and Molecular Structure, III. Electronic Spectra and Electronic Structure of Polyatomic Molecules*, (Van Nostrand, New York, 1966), pp. 457-458.

[30] A word of caution must be extended concerning the potential energy diagrams used to illustrate various processes. The two-dimensional nature of the printed page is insufficient to fully represent the

multidimensional nature of polyatomic molecular potential energy surfaces. For this reason, the diagrams cannot be taken as any more than two dimensional 'cartoons' of a multidimensional system.

[31] M. W. Chase, Jr., C. A. Davies, J. R. Downey, Jr., D. J. Frurip, R. A. McDonald, and A. N. Syverud, *JANAF Thermochemical Tables*, 3rd. ed. (American Chemical Society and American Institute of Physics, New York, 1986).

[32] K.-Y. Du and D. W. Setser, Chem. Phys. Lett. **153**, 393 (1988).

[33] R. N. Zare, Mol. Photochem. **4**, 1 (1972).

[34] The theory of the anisotropy of the flux of molecular fragments produced by polarized light is analysed for arbitrary molecules by: S.-C. Yang and R. Bersohn, J. Chem. Phys. **61**, 4400 (1974).

[35] T. A. Miller, Science **223**, 545 (1984).

[36] D. W. Kohn, H. Clauberg, and P. Chen, Rev. Sci Instrum. **63**, 4003 (1992).

[37] J. A. Blush, H. Clauberg, D. W. Kohn, D. W. Minsek, Xu Zhang, and P. Chen, Acc. Chem. Res. **25**, 385 (1992).

[38] J. Berkowitz, *Photoabsorption, Photoionization, and Photoelectron Spectroscopy*, (Academic Press, New York, 1979), p. 36.

[39] E. P. Wigner, Phys. Rev. **73**, 1002 (1948).

[40] K. J. Reed, A. H. Zimmerman, H. C. Anderson, and J. I. Brauman, J. Chem. Phys. **64**, 1368 (1976).

[41] A. E. Douglas and W. J. Jones, Can. J. Phys. **43**, 2216 (1965).

[42] M. Polak, M. Gruebele, and R. J. Saykally, J. Am. Chem. Soc. **109**, 2884 (1987).

[43] R. E. Continetti, D. R. Cyr, R. B. Metz and D. M. Neumark, Chem. Phys. Lett. **182**, 406 (1991).

[44] A. Weaver, Ph. D. Thesis, University of California, Berkeley, 1991.

[45] S. E. Bradforth, E. H. Kim, D. W. Arnold, and D. M. Neumark, J. Chem. Phys. **98**, 800 (1993).

Chapter 2

The Fast Radical Beam Machine

2.1 Introduction

The Fast Radical Beam Machine (FRBM) is the experimental apparatus constructed in our laboratory to undertake photodissociation studies of free radicals. The apparatus is built to enable the photodissociation of a fast beam of free radicals and subsequent detection of the resulting photofragments. Generation of the fast beam of free radicals is achieved through photodetachment of the corresponding negative ion.

The machine is a fairly complicated entity, as illustrated by a schematic diagram shown in Figure 2.1. To minimize its complexity, it is advantageous to describe the machine in sections. By way of an overview, the apparatus consists of a source region, where negative ions corresponding to the radicals of interest are generated, internally cooled in a supersonic expansion, and accelerated. Next is a region where the negative ions are steered and focused with electrostatic optics and mass selected via time-of-flight to the photodetachment region. Within the photodetachment region, a tunable pulsed laser is timed to photodetach the mass of interest, with the detached electrons collected to allow monitoring of the detachment efficiency. Undetached negative ions are deflected out of the beam axis onto a detector which allows the source performance to be monitored. In the following region the radicals formed by photodetachment are photodissociated and the product fragments detected.

Two types of photodissociation experiments are possible. In a total photodissociation cross section experiment, the total fragment signal is measured as a

---

Figure 2.1 This diagram shows the overall layout of the Fast Radical Beam Machine. Various components of the machine are labeled, as are the differential pumping regions. This diagram is roughly to scale; the total length of the machine is ~13 feet.

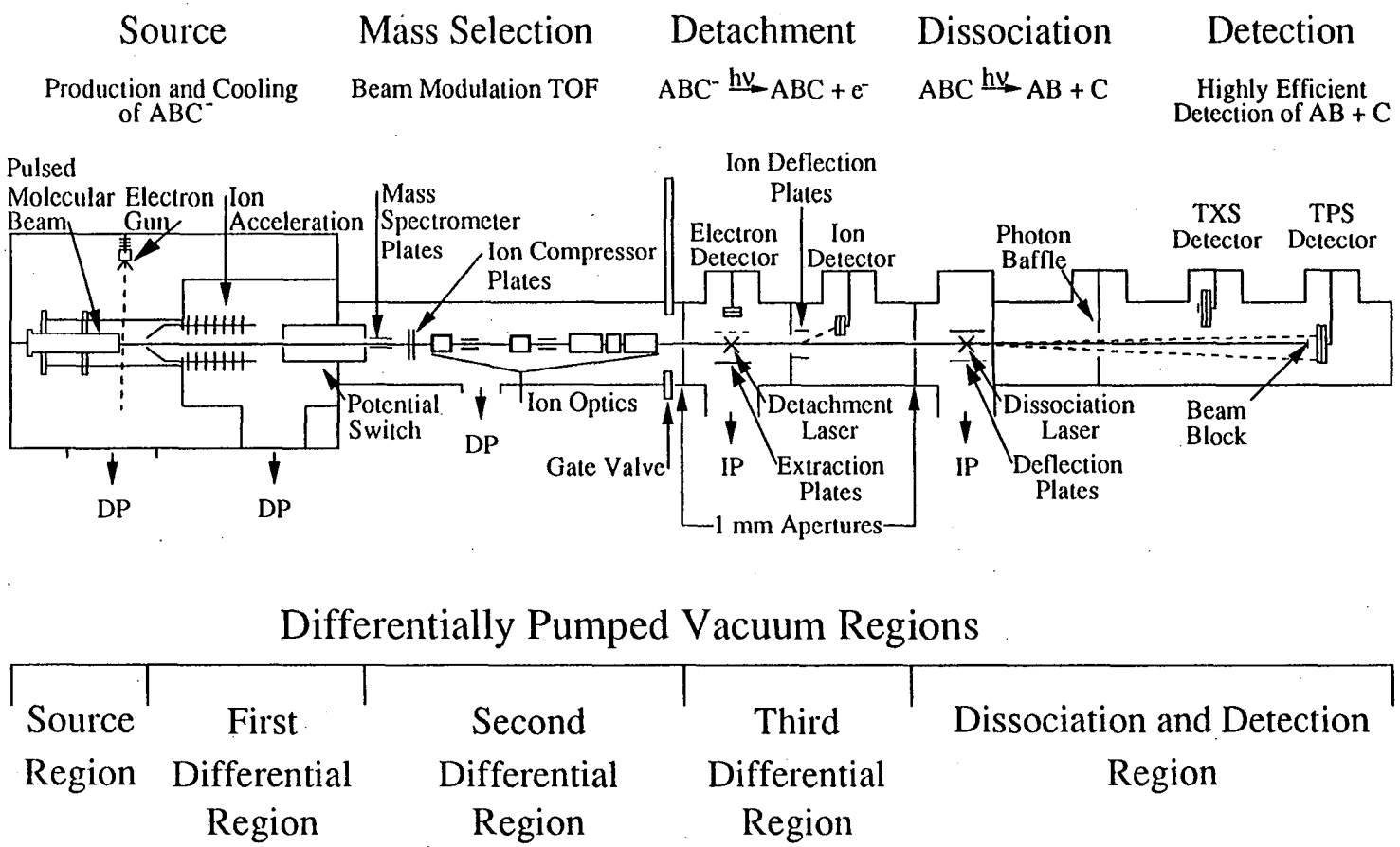

Source
Mass Selection
Detachment
Dissociation
Detection
Production and Cooling of ABC⁻
Beam Modulation TOF
ABC⁻ hν→ ABC + e⁻
ABC hν→ AB + C
Highly Efficient Detection of AB + C
Pulsed Molecular Beam
Electron Gun
Ion Acceleration
Mass Spectrometer Plates
Ion Compressor Plates
Ion Deflection Plates
Electron Detector
Ion Detector
Photon Baffle
TXS Detector
TPS Detector
Potential Switch
Ion Optics
Detachment Laser
Dissociation Laser
Beam Block
DP
DP
DP
Gate Valve
IP
Extraction Plates
IP
Deflection Plates
1 mm Apertures
Differentially Pumped Vacuum Regions
Source Region
First Differential Region
Second Differential Region
Third Differential Region
Dissociation and Detection Region

Figure 2.1

function of dissociation wavelength. To then perform a detailed investigation of the dissociation dynamics at a certain fixed wavelength, the kinetic energy and recoil angle distributions for each product channel are determined using a time- and position-sensitive detector.

In the rest of this chapter the instrument is discussed at length, as this is the first detailed account of its construction.

2.1 Vacuum Systems

The machine has five differentially pumped vacuum regions. They are referred to imaginatively as the source, first differential, second differential, third differential and detector regions, and are labeled in Figure 2.1. An electropneumatic gate valve delineates the boundary between the first three high vacuum regions (on the source side) and the two ultrahigh vacuum regions downstream (on the detector side).

On the source side of the gate valve, flanges are sealed with viton o-rings and vacuum is provided by diffusion pumps, backed by mechanical pumps. The source region is pumped by a 10", 2300 l/s diffusion pump (Edwards High Vacuum, Series 250 Diffstak) backed by a 40 l/s mechanical pump (Edwards E2M40). The first differential region, separated from the source region by the 3 mm aperture in the skimmer, is pumped by a 6", 805 l/s diffusion pump (Edwards High Vacuum, Series 160 Diffstak) backed by an 18 l/s mechanical pump (Edwards High Vacuum, E2M18). The second differential region, separated from the first differential region by a 4 mm aperture, is pumped by a 2", 173 l/s diffusion pump (Edwards High Vacuum, Series 63 Diffstak) backed by an 8 l/s mechanical pump (Edwards High Vacuum, E2M8). The diffusion pumps are charged with a polyphenyl ether fluid (Monsanto, Santovac 5®) because of its superior electrical conducting property. This avoids a problem that can occur when insulating varieties of diffusion pump oil are used — inner surfaces of the chamber, coated with a thin layer of

non-conducting oil can develop a potential and establish stray fields that affect ion production and/or transmission.

In contrast to the regions prior to the gate valve, the regions following the gate valve employ knife edge/copper gasket seals and relatively cleaner sources of vacuum having lower ultimate pressure. Immediately beyond the electropneumatic gate valve is a 1 mm aperture which facilitates differential pumping between the second and third differential regions. A second 1 mm aperture separates the third differential region from the detector region. The third differential and detector regions are pumped by 120 l/s and 220 l/s ion pumps (Perkin Elmer, Model D-I) respectively. These pumps require no backing, but must not be turned on when the pressure is above 10^{-4} Torr. Longer electrode life is expected if the high vacuum regions are brought to 10^{-5}-10^{-6} Torr prior to the ion pumps becoming operational. This is accomplished by temporarily uniting the third differential and detector regions and evacuating them via a 75 l/s turbomolecular pump (Varian, Model V-80) backed by an 8 l/s mechanical pump (Edwards High Vacuum, E2M8). The ion and turbomolecular pumps can pump in unison in the 10^{-5} to 10^{-7} Torr range, but once the pressure is below $1x10^{-7}$ Torr the turbomolecular pump has approached its ultimate pressure, and is best isolated from the machine.

Pressures in all five regions are measured by ion gauges, with foreline pressures determined using thermocouple gauges. Typical background pressures for the source, first differential, and second differential region are $1x10^{-7}$, $2.5x10^{-7}$, and $2x10^{-7}$ Torr, respectively. After pump down and bake-out of the ultrahigh vacuum region, background and operating pressures are in the low to mid 10^{-9} Torr range.

2.2 Interlock Circuits

The vacuum system and electronics are protected in case of equipment failure, abnormally high pressure in critical regions, and many types of operator error by an interlock system. The interlock system allows manual control as well as fail-safe

automatic control of the gate valve, all high voltage electronics and various vacuum equipment. A schematic of the circuit is shown in Figure 2.2 (a) and (b). All relays are shown in their fail-safe positions (i.e., the state that they revert to if the AC power to the interlock box failed). The orientation of the indicator lights was chosen such that green would be visible when conditions are as they should be *while running an experiment.* This means, for example, that when the electropneumatic gate valve separating the regions on the source side from the regions on the detector side is open, the green light is on. The idea was that if the bell sounds during an experiment, it will be easy to determine the status of the interlocks with a glance; in particular, which interlocks have tripped. For the most part, the symbols in Figure 2.2 can be interpreted with ease. It should be pointed out that the circle containing an alternating current symbol indicates A/C power that is fed into the interlock box directly from a separate circuit breaker on the rack panel. This power then flows out to the designated circuit after passing through a relay controlled by the various interlocks appropriate for the control of that circuit. Some details of the interlock circuits follow.

The interlock allows power to all three diffusion pumps to be turned off manually, while interrupting power automatically if their respective foreline pressures rise above a preset limit, usually ~0.2 – 0.5 Torr, or if there is an interruption in cooling water, as determined by a bi-metallic temperature sensor mounted on the diffusion pump cooling coils.

Both the first and third differential region ion gauge controllers have ion gauge high pressure trip points. The first differential region ion gauge controller was chosen to have this capability because it contains the bulk of components that are at the highest voltage in the source region. Of the two ultrahigh vacuum regions, the third differential

---

Figure 2.2 (a) and (b) The circuit diagram for the FRBM interlock box. See text for details.

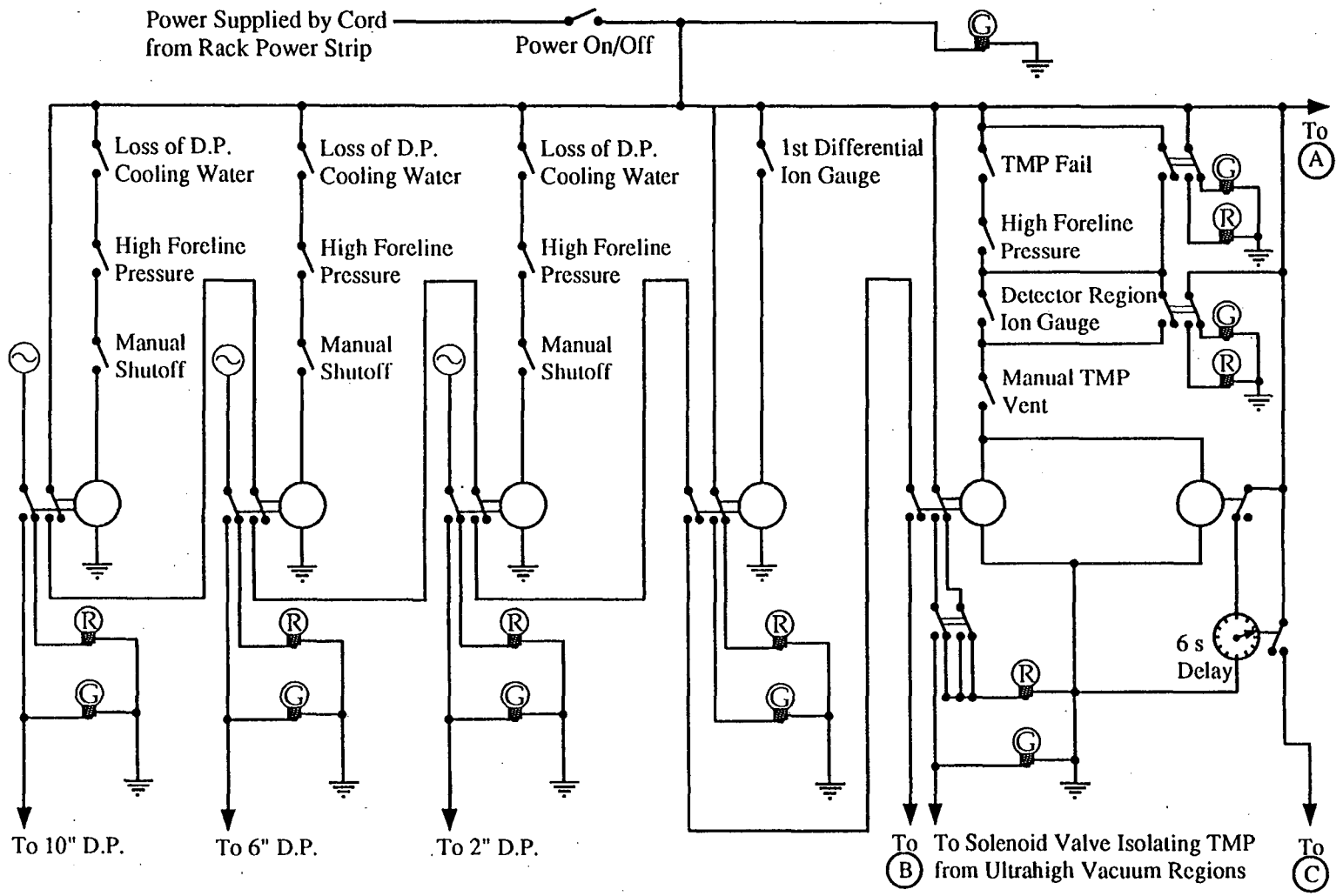

Power Supplied by Cord
from Rack Power Strip
Power On/Off
G
To
A
Loss of D.P.
Cooling Water
High Foreline
Pressure
Manual
Shutoff
Loss of D.P.
Cooling Water
High Foreline
Pressure
Manual
Shutoff
Loss of D.P.
Cooling Water
High Foreline
Pressure
Manual
Shutoff
1st Differential
Ion Gauge
TMP Fail
High Foreline
Pressure
Detector Region
Ion Gauge
Manual TMP
Vent
R
G
6 s
Delay
To 10" D.P.
To 6" D.P.
To 2" D.P.
To
B
To Solenoid Valve Isolating TMP
from Ultrahigh Vacuum Regions
To
C

Figure 2.2(a)

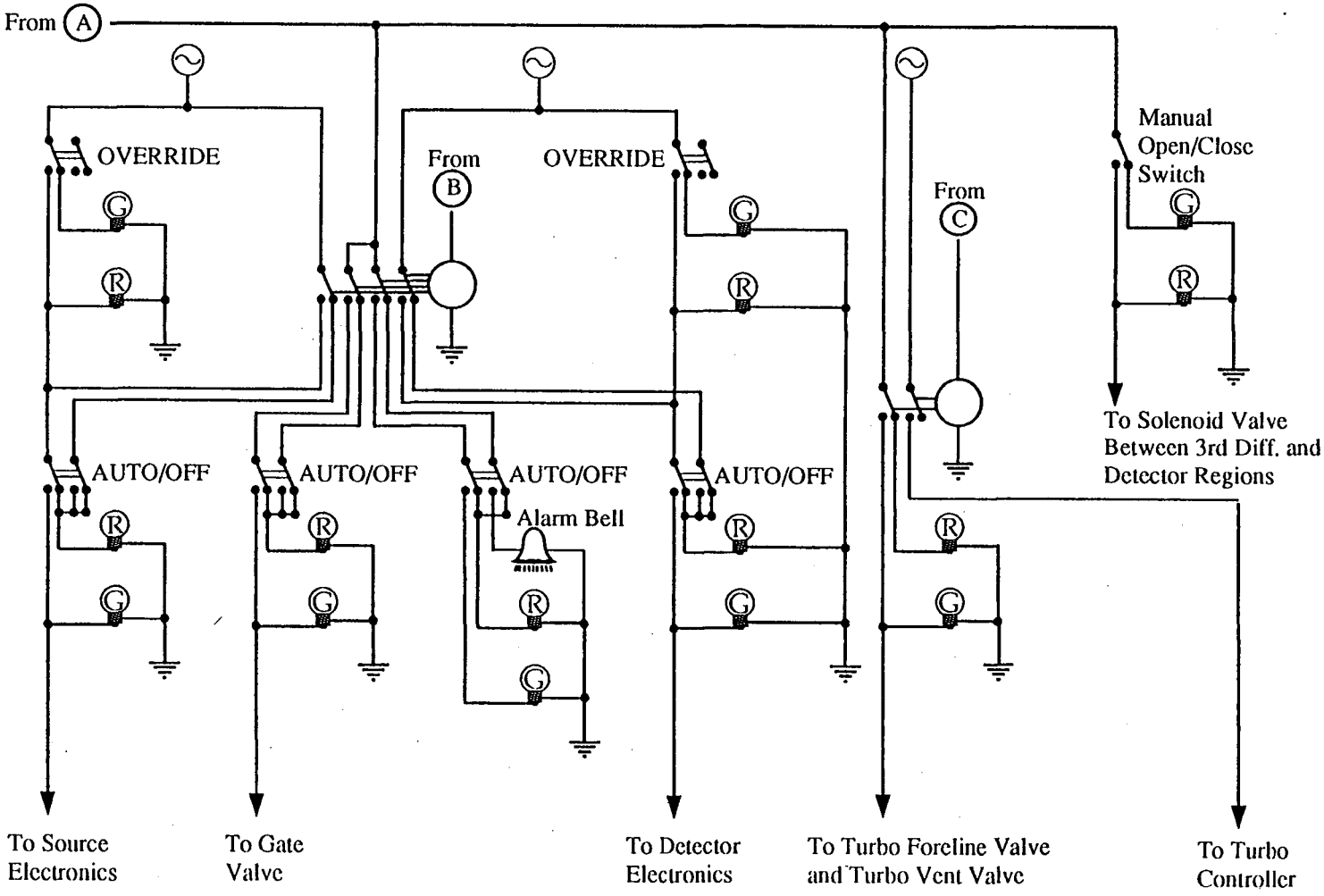

From A
OVERRIDE
From B
OVERRIDE
From C
Manual Open/Close Switch
G
R
AUTO/OFF
AUTO/OFF
AUTO/OFF
Alarm Bell
AUTO/OFF
To Solenoid Valve Between 3rd Diff. and Detector Regions
To Source Electronics
To Gate Valve
To Detector Electronics
To Turbo Foreline Valve and Turbo Vent Valve
To Turbo Controller

Figure 2.2(b)

region ion gauge controller was chosen because the detector ion gauge is often turned off while an experiment is in progress.

If any of the diffusion pumps or either of the ion gauges has their interlock trip, a number of things will occur immediately. The gate valve will close, the source and detection electronics plug strips will have their power interrupted, and an ear-piercing, bell will commence ringing. It should be stressed that for safe operation, it is essential that all high voltage power supplies for the source and detection region be plugged in to their interlocked power strips.

Several features exist in the form of front panel AUTO/OFF switches and OVERRIDE switches. As mentioned above, the diffusion pumps have AUTO/OFF switches; the user can definitely turn them off using this switch, but with the switch in the AUTO position all other conditions must be satisfied for the interlock to allow operation. The same is true for the source and detection electronics, and the gate valve. The alarm bell also has an AUTO/OFF switch, although of course the conditions for the operation of the alarm bell are somewhat opposed to most other components, in that it turns on when a problem exists.

OVERRIDE switches will allow the source and detection electronics to receive power despite high pressure or other interlocks being tripped, the sole exception being if the manual AUTO/OFF switch corresponding to the respective OVERRIDE switch is set to OFF. An OVERRIDE switch is also provided to specifically circumvent the effect of a high pressure reading from the detector region ion gauge.

Two other manual switches are present on the front panel. They control electropneumatic valves which can unite the third differential and the detector regions and open the turbomolecular pump (TMP) to these regions, respectively. The color of indicator light associated with these switches is consistent with our original belief that during an experiment the third differential region and the detector region would be isolated from each other (true), but that the TMP would be open to the detector region to

assist in pumping (false). In fact, as mentioned above, it is found that the ultimate pressure of the TMP is greater than the pressure we can attain in these regions without the assistance of the TMP. Therefore, contrary to the general rule of thumb stated above, under normal operations the indicator corresponding to the switch controlling the solenoid between the TMP and detection region is RED.

The TMP interlock section of this circuit was designed before we acquired the TMP, and has never been fully implemented. There is little incentive to do so because we do not have the TMP open to the machine except during pumpdown. Therefore, in practice the Turbo Fail and High Foreline Pressure sensors are not present in the circuit shown in Figure 2.2, and these 'switches' can be considered to be always closed.

To provide an fail-safe interlock for the TMP a very simple circuit was built and is housed separately. This interlock ensures that if the TMP failed, a valve located on the foreline to the pump would close, isolating the relatively 'dirty' foreline and mechanical pump from the TMP and the ultrahigh vacuum regions of the machine. A switch allows the bypass of this interlock during the initial stages of a pumpdown, where the foreline valve must be open to allow the vacuum chamber to be roughed out by the mechanical pump even when the TMP is not operational.

The ion pumps need not be interlocked, and the ultrahigh vacuum region is not vented in case of loss of power. Ion pumps will turn themselves off in a pressure excursion, and will not create a mess if power is lost. In fact, through 'gettering' action,[1] these pumps retain pumping ability for some time after being shut off (useful during power outages).

2.3 Source Region

Within the source region, we wish to generate negative ions corresponding to the radical of interest. It is usually highly desirable that these ions have very low internal energies, with most of the ions being in the ground vibrational level and the lowest

rotational states. The advantage of populating relatively few quantum states lies in the resulting narrow distribution of internal energies. This narrow distribution, if transferred to the free radical on photodetachment, will allow for elementary accounting of the total energy in the radical on excitation, and a very precise determination of the amount of energy partitioned into the internal degrees of freedom of the recoiling photofragments, via subtraction of the kinetic energy release from this total energy. As discussed in § 1.4, if negative ions are formed primarily in the ground vibrational level, photodetachment can usually be undertaken in such a way as to populate only the ground vibrational level of the radical.

The method which is typically used to generate negative ions in this experiment is the electron beam-free jet expansion technique developed by Lineberger and co-workers.[2,3] In this technique, the continuum flow region of a free jet expansion is crossed at 90 degrees by a 1 keV electron beam. The high energy electrons create lower energy electrons via electron-impact ionization of the molecules in the free jet:

$$M + e^-(\sim 1\ \text{keV}) \rightarrow M^+ + e^-(< 1\ \text{keV}) + e^-(> 0\ \text{eV}) \qquad (2.1)$$

The lower energy electrons are rapidly thermalized via inelastic collisions within the expansion and can then participate in the formation of negative ions. Dissociative attachment is the major pathway to negative ion production.[3] "Dissociative" may mean the breaking of a conventional chemical bond, or the "evaporation" of molecules that had been bound in a van der Waals cluster. Eqns. (2.2) and (2.3) give examples of both of these possibilities, while illustrating our method of producing the azide anion and the oxygen anion, respectively:

$$C_6H_5CH_2N_3 + e^- \rightarrow N_3^- + C_6H_5CH_2 \qquad (2.2)$$

$$O_2(O_2)_n + e^- \rightarrow O_2^-(O_2)_{n-m} + m\ O_2 \qquad (2.3)$$

A second route to negative ions that can be formed via proton abstraction involves formation of F^- via dissociative detachment of NF_3, followed by the abstraction of a

(relatively) acidic proton to make the desired negative ion and HF. This process is shown in Eqn. (2.4), illustrating our method of producing the nitromethyl anion:

$$NF_3 + e^- \rightarrow F^- + NF_2 \qquad (2.4\ a)$$

$$CH_3NO_2 + F^- \rightarrow CH_2NO_2^- + HF \qquad (2.4\ b)$$

At some point in the future, the laser vaporization source[4] that has been used with great success on the other two experiments in the Neumark Group will be adapted to our machine, enabling the production and study of various metallic and semiconductor clusters. The pulsed discharge source, discussed further in § 2.3.5, is also expected to prove very valuable in the future.

2.3.1 Source Chamber Design

The source chamber was designed to maintain maximum flexibility without compromising on integrity of construction. The floor of the source chamber, to which the large diffusion pumps are bolted, and the downstream wall, to which the remainder of the machine is attached, are fabricated from 0.75" stainless steel for high strength. Flexibility in the design is achieved by having the remaining frame made from 3" wide, 0.5" thick strips of stainless steel, with all of the pieces inside-welded vacuum-tight along the edges. The chamber walls consist of 1" aluminum plate bolted on to this skeleton, sealed with o-ring gaskets. In this way, the large diffusion pumps can be hung from the floor of the chamber and the interface to the remainder of the machine is robust, while any of the remaining sides or the top can be easily removed for wide-open access to the interior of the source chamber. It is also possible to quickly substitute alternate chamber walls machined for specialized applications, though this option has not been exercised to this point.

2.3.2 Pulsed Molecular Beam Valve

A supersonic free jet expansion is used so that negative ions that are formed will undergo internal cooling. The conversion of internal energy to translational energy of molecules during a series of collisions in a free jet expansion is well established, and documented extensively elsewhere.[5] Pulsed lasers are used in both detaching the negative ions and to cause dissociation of the radical because of their high peak power. It is therefore logical to employ a pulsed free jet expansion. In exchange for the more complex operation of a pulsed versus continuous nozzle, the instantaneous number density in a pulsed expansion is much higher, concurrent with the more modest vacuum pumping speed requirements and low background pressures characteristic of a pulsed molecular beam.

Originally, we experimented with using a standard solenoid-driven pulsed valve (General Valve Corporation, Series 9) to produce our pulsed molecular beam. This arrangement worked relatively well, but was subject to certain problems. Solenoid varieties of pulsed valves draw the poppet back from where it sits covering the faceplate by generating a magnetic field which acts on an armature attached to the poppet. The magnetic field results from the flow of current through a coil surrounding the armature. This flow of current dissipates power and, particularly at high repetition rates, causes excessive heating of the pulsed valve. In addition, the poppet that gave the best results was made of Kel-F and was easily scored, resulting in a leak that could only be repaired by the replacement of the poppet. Even if the poppet did not become badly scored, it often became compressed over time so that it would require adjustment. The adjustments that were necessary to restore proper operation, principally the faceplate tightness, needed to be made outside the machine. This meant venting at least once (but often more that once) in order to adjust the valve. Lastly, the armature of the solenoid appeared to lose its magnetic properties over time, and needed to be replaced periodically.

The solenoid valve was soon replaced by a Trickl-type[6] piezoelectric valve at the urging of Dr. Bob Continetti, who had used this design in his past work[7] in the laboratory of Prof. Y.T. Lee. There was a noticeable improvement in performance using the new pulsed valve, principally in the intensity and full width at half maximum of the gas pulse, as measured using a Fast Ion Gauge (the FIG is discussed in § 2.3.3). In contrast to the General Valve, the piezoelectric valve dissipates very little power while running, and thus does not become warm/hot. Additionally, the piezoelectric valve can achieve 'choke flow', where the flat-topped shape of the gas pulse indicates that the faceplate orifice is fully open and the pulse is limited by the conductance through the orifice.

Another important benefit of the piezoelectric pulsed valve design is the ability to perform dynamic adjustments on the valve while it is in operation in a test configuration. This configuration consists of the valve mounted on the exterior of a flange so that it seals a hole which opens to the interior of the source region. The back of the valve casing is removed so adjustments to the poppet depth and carrier nuts can be made as the valve performance is monitored using the FIG. Once the valve is adjusted optimally, it can be reassembled in its case and the mounted inside the machine. An extremely long period between adjustments or replacement of parts can be expected. Since the beginning of the "Piezoelectric Era" we have not considered a return to the General Valve.

A short description of the construction of the piezoelectric pulsed valve that is currently in use follows. The variations in design that are necessary for using gases that swell or damage flexible o-ring material are also given. Lastly, the driving circuit for the piezoelectric valve is shown and briefly outlined.

A schematic showing the cross section of the piezoelectric pulsed valve is shown in Figure 2.3. This pulsed valve is designed to rapidly and completely open the small (~375 μm) hole in the faceplate. A gas at some backing pressure, (typically 10-40 PSIG) rushes into the source chamber through this hole for a short period of time (~100-200 μs) before the orifice is sealed again. The size of the faceplate orifice can be varied by the

substitution of alternate faceplates. The faceplate is normally sealed by a 2-002 viton o-ring mounted in a groove cut on the end of the aluminum poppet. The poppet is mounted perpendicular to the faceplate through the center of a ceramic disk (Physik Instrumente, U.S. Distributor: Polytec Optronics, Part No. P-910.160) which has piezoelectric properties.[8] The rapid opening of the faceplate orifice is accomplished by applying high voltage to one side of the disk, which causes it to flex like a drum head. The induced translation is up to 100 microns at 1000 V, drawing the poppet and o-ring back away from the faceplate.

In cases where gases are present that will swell or deform the flexible o-ring material that normally makes the seal with the faceplate, the standard configuration becomes unreliable and a different poppet and faceplate combination is used. The alternate poppet is constructed using a Kel-F ball tip screwed onto and aluminum shaft, which is again mounted onto the piezoelectric crystal. The Kel-F ball tip seals against a conical seat in a modified faceplate. Kel-F is a much stiffer material, which resists swelling much better than viton (minor swelling may still occur). However, the stiffness of the Kel-F ball means that if it is scored or deformed, it will leak. Kel-F tips also require very precise adjustments to achieve a proper seal against the faceplate. To its credit, the Kel-F ball tip allows certain gases to be present in the expansion that could not be used otherwise, but is no match for the convenience and reliability of the viton o-ring design.

The circuit that is used to drive the disk translator was built from circuit diagrams supplied by the group of Hanna Riesler at USC, and is shown in Figure 2.4. The main components of this circuit are two 1 kV MOSFETs which act as switches. They are

---

Figure 2.3 This schematic is a cross section of the piezoelectric pulsed valve used in the source. (1) Stainless steel body. (2) High voltage connector. (3) Stainless steel faceplate with 0.015" orifice. (4) Piezoelectric translator disk. (5) Aluminum poppet affixed to disk with lock nut/screw assembly to adjust tension of o-ring seal against faceplate. (6) Carrier o-ring used to damp disk.

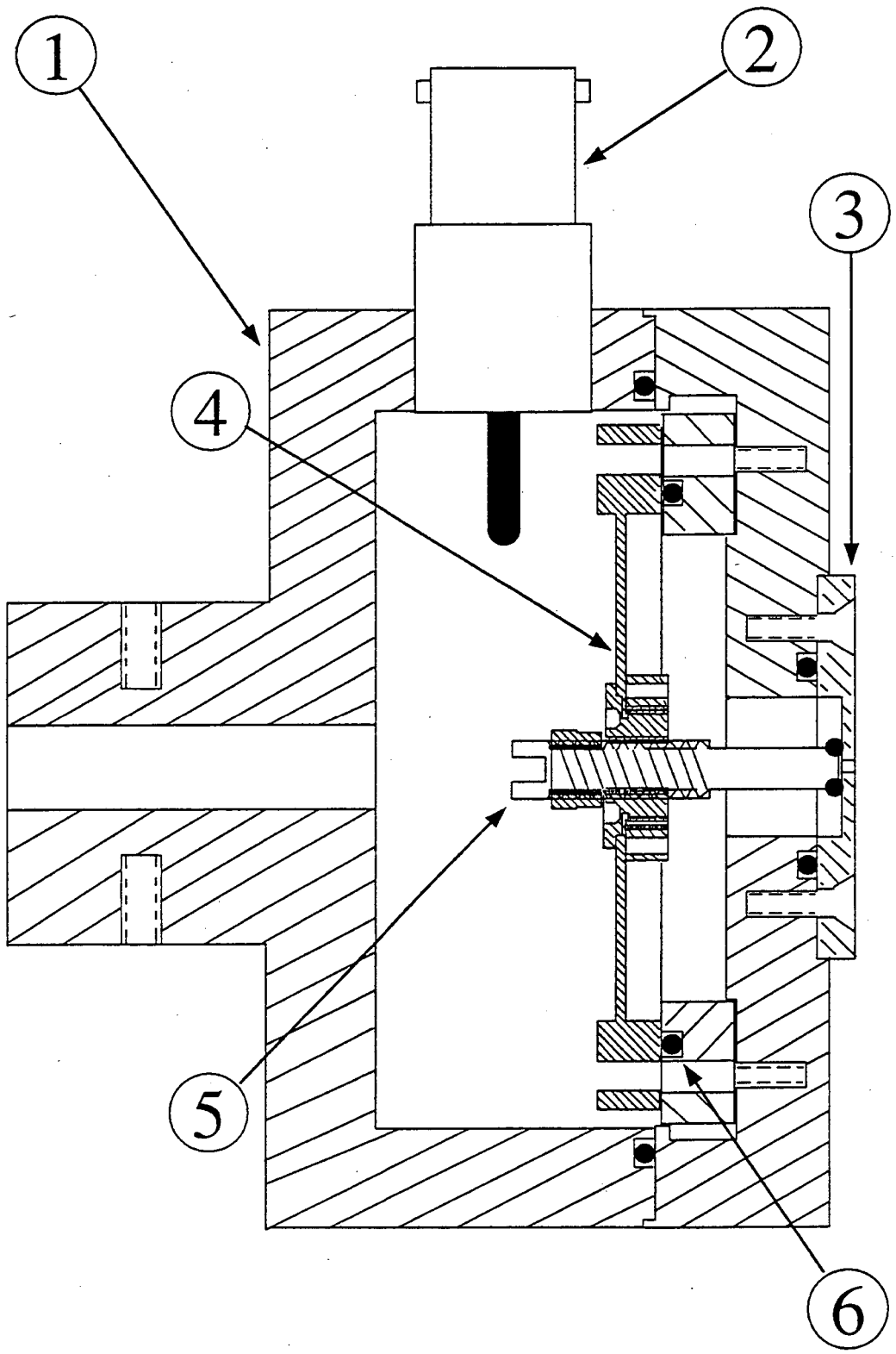


Figure 2.3

arranged in a push-pull configuration, with one MOSFET switching a connection between negative high voltage and the piezoelectric disk, and the other MOSFET switching a connection between the disk and ground. Normally, the connection to ground is present, but when triggered, the driver opens this switch, isolating the crystal from ground. Simultaneously the other switch closes, allowing the disk to be brought to the negative high voltage. At the end of a user-set delay, both switches revert to their normal state. A critical factor in driving the disk is that the fall- and rise-times of the potential that is applied to the piezoelectric disk must have a minimum RC time constant of ~50 μs. This results in a pulse shape resembling an inverted shark fin. The natural frequency of the disk is 2.5 kHz,[8b] and, while this can be safely exceeded to some degree, the disk will crack if a high voltage step function pulse is used.

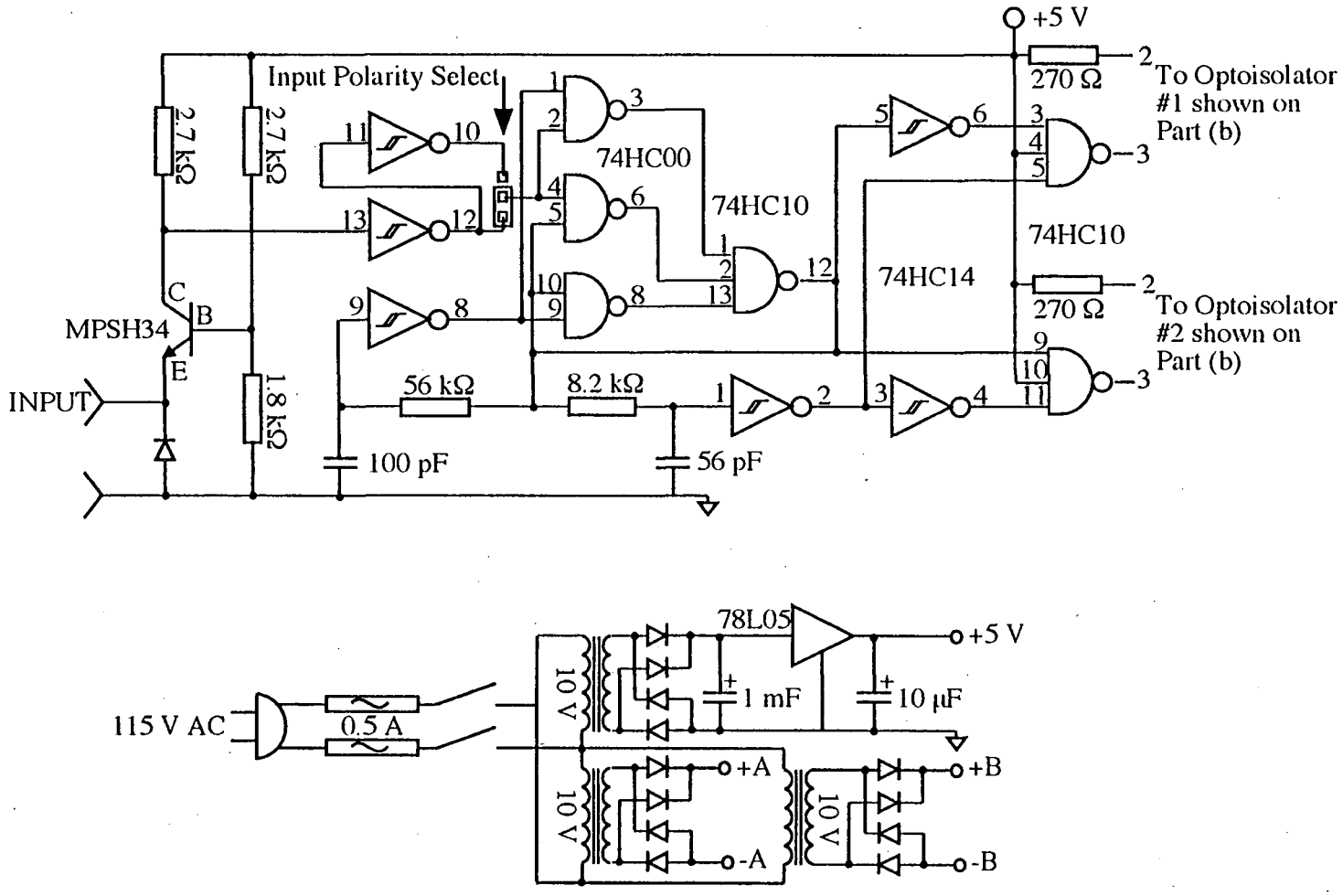
2.3.3 Fast Ion Gauge

A fast ion gauge (FIG) is used to monitor the on-axis temporal profile of the pulsed free jet expansion. By placing the FIG in front of the free jet expansion the local pressure as a function of time can be determined. This allows the performance of the pulsed valve to be monitored, an especially valuable asset while the valve is being adjusted. The FIG thus allows the narrowest, most intense temporal profiles to be obtained in a straightforward manner.

The FIG is home-built in our laboratory by Dr. Robert Continetti, based on a design described by Giese and Gentry[9] and his previous experience while a graduate student.[7] It operates on the same principles as an ordinary Bayard-Alpert ion gauge, the difference being a more compact size and the ability for fast time response. Relative pressure is determined by ionizing nearby molecules, collecting the ions and measuring the

---

Figure 2.4 (a) and (b) The circuit diagram for the piezoelectric pulsed valve driver circuit.



Signal Transformers: ST2-10

Figure 2.4 (a)

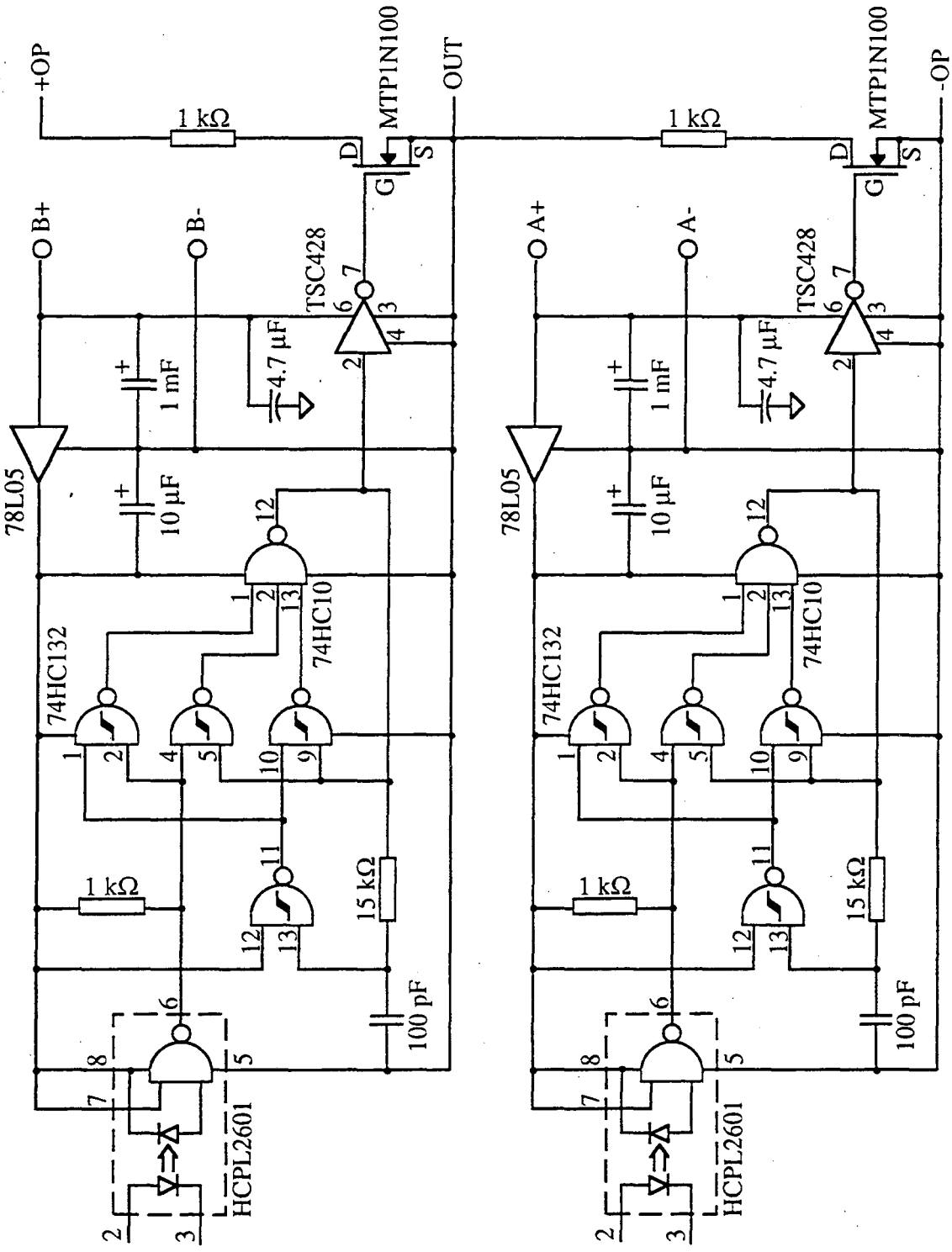

+OP
1 kΩ
MTP1N100
OUT
B+
B-
TSC428
4.7 µF
1 mF
78L05
10 µF
74HC132
74HC10
1 kΩ
15 kΩ
100 pF
HCPL2601
A+
A-
-OP

Figure 2.4 (b)

current, which is taken to be proportional to the pressure.

Figure 2.5 shows the geometry of the FIG components. A pair of 2.6 cm long, 1 mm diameter tungsten filaments are connected to the leads of a constant voltage/current power supply (Power Designs, Model 3650R). When a ~4 V potential is across the filaments the resultant current heats the resistive material enough that thermionic emission occurs. This emission of electrons causes the ionization of nearby molecules. The filament power supply is biased at +10 V by a second power supply (Acopian, Model A060NT12). The emission current of the filaments corresponds to the current drawn from this float power supply in order to maintain the +10 V bias on the filament power supply. An emission current of 10 μA is typically used. A grid, made from a spring and thus having a coil shape, is biased at ~ +200 V with respect to ground using a third power supply (Acopian, Model U300Y20). This cylindrical coil encircles the length of a thin wire which is nominally a ground potential and acts as a collector of the positive ions. The collector leads to the input of the electrometer circuit that provides a time-resolved voltage reading corresponding to the temporal evolution of the local pressure.

The electrometer circuit that provides time-resolved pressure measurement is also shown in Figure 2.5. The pressure-dependent current on the collector causes a set of two LF 357J operational amplifiers connected in series to produce an inverted signal corresponding to the temporal profile of the local pressure in the pulsed beam. The FIG output typically has 100 mV of high frequency noise superimposed on a 5-10 V signal. This is an unavoidable consequence of such high amplification by the two operational amplifiers, but is hardly noticeable on the larger scales required to view the full signal.

Care must be taken to keep the filament emission current constant, because ion current at the collector will reflect any fluctuation or long-term drift. Other problems with

---

Figure 2.5 A diagram of the fast ionization gauge geometry and electrometer circuit. The location in the circuit of the microammeter used to monitor the emission current of the filament is denoted by a circle surrounding an "A".

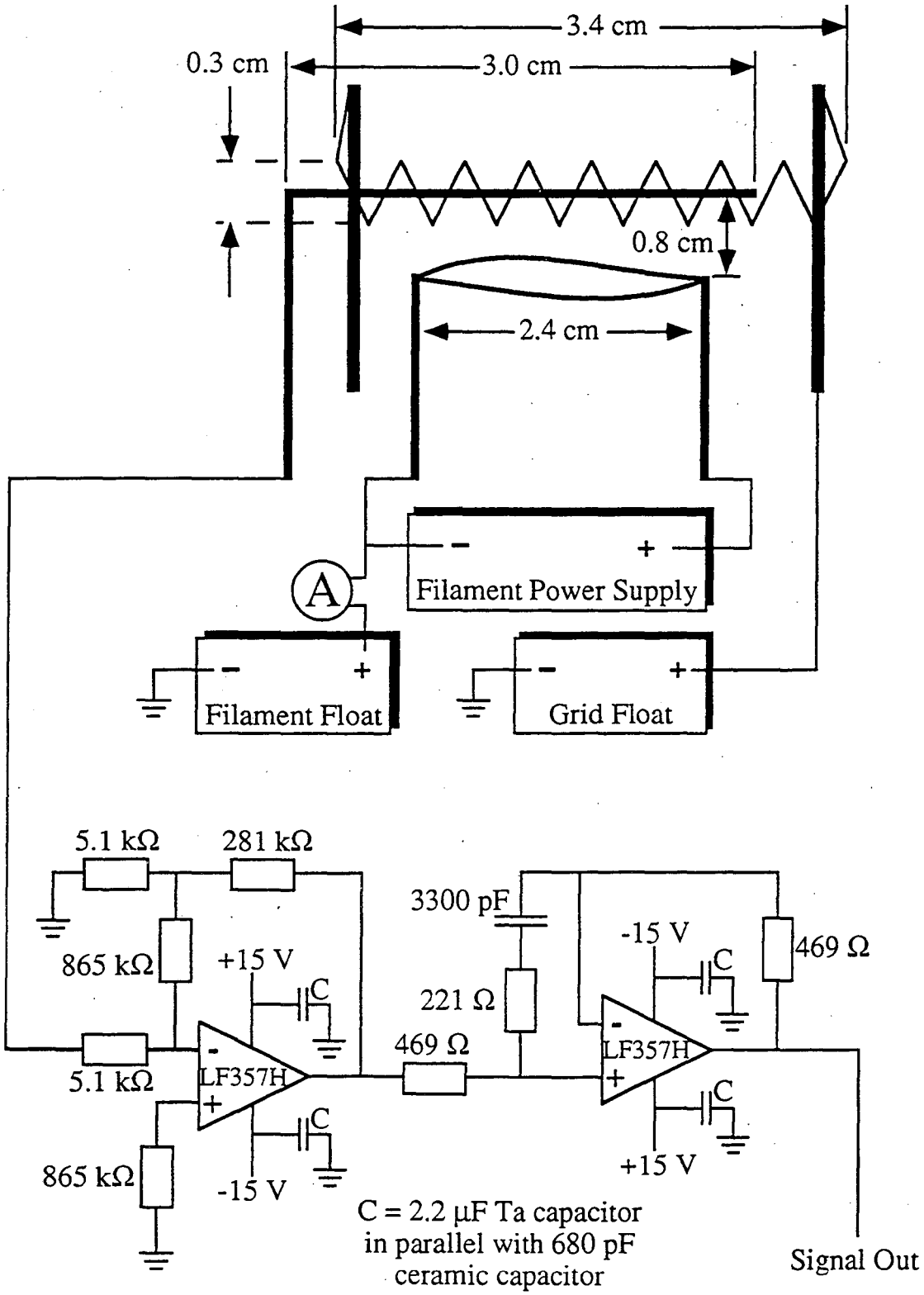

3.4 cm
3.0 cm
0.3 cm
0.8 cm
2.4 cm
A
Filament Power Supply
Filament Float
Grid Float
5.1 kΩ
281 kΩ
865 kΩ
+15 V
C
5.1 kΩ
LF357H
865 kΩ
-15 V
3300 pF
221 Ω
469 Ω
-15 V
469 Ω
LF357H
+15 V
C = 2.2 μF Ta capacitor
in parallel with 680 pF
ceramic capacitor
Signal Out

Figure 2.5

reproducibility and comparability of results stem from the exact location of the FIG with respect to the nozzle orifice. The velocity spread present in the gas pulse serves to broaden the pulse width as it propagates. Therefore, a given pulse will always have a wider temporal profile and lower peak intensity when measured further from the nozzle. Lastly, different gases will have different propensities to ionize under these conditions and therefore give different signal intensities at similar number densities. This also hinders meaningful comparison under different conditions.

2.3.4 Electron Beam

The 1000 eV continuous electron beam that crosses the high pressure region of the molecular beam expansion is generated in an identical manner to previous Neumark Group experiments.[4,10] These implementations in turn owe much to Lineberger and co-workers,[11] who were the originators of the basic design. The electron beam is generated using a Tektronix oscilloscope electron gun assembly. To adapt this electron gun for operation in the relatively high pressure environment of our source region, an iridium filament coated with thorium oxide is used as a thermionic source of electrons. These electrons are then extracted through an anode, accelerated to ~1 keV, focused by an einzel lens and steered by a set of deflectors. The electron beam is collected by a Faraday cup, located on the opposite side of the chamber, which is kept at 9 V relative to ground and is connected to a microammeter. Maximum output is measured to be > 500 μA, but quite often only a reading of 100 μA or less results in optimum conditions for producing negative ions of interest. The microammeter readings are almost certainly underestimation of the true electron beam current during an experiment however, because in normal operating conditions part of the electron beam is intercepted by the faceplate of the pulsed valve and, to a lesser extent, scattered by the molecular beam expansion, and will not be collected on the Faraday cup.

The electronics associated with the electron gun are displayed in Figure 2.6. The electron energy and einzel focus element power supplies are two 0-2000V (Kepco, Model APH2000M), which are kept at 1000 V and between 600-950 V respectively. Only the electron energy supply draws current, the amount of which is equal to the total emission of the filament. The filament power supply is a constant voltage/current supply (Kepco, Model ATE15-15M) which must provide between 4.5 and 8 Amperes at low voltage. The anode and deflector supplies are home-built units containing 150 and 60 VDC supplies (Acopian, Model A0150NT05 and A060NT12, respectively). Because the anode potential and the filament potential need to be close to the electron energy, these power supplies are 'floated' at the electron energy potential. A 1:1 isolation transformer[12] is used to accomplish this, and the anode potential and filament current power supplies are kept in a Plexiglas enclosure for safety.

2.3.5 Pulsed Discharge Source

Feasibility studies have been carried out on a new negative ion source that will serve as an alternative to our usual electron beam-free jet expansion source. This new source involves a pulsed discharge through a mixture of precursor molecules seeded in a carrier gas just prior to undergoing a free jet expansion. It shares certain characteristics of sources used by Bondybey and co-workers,[13] Ohshima and Endo,[14] Sharpe and Johnson,[15] and Bramble and Hamilton.[16] These workers have generally used this type of source to make and cool transient species, typically radicals or van der Waals complexes involving free radicals. With the exception of the recent short description of C_2^- production by Bondybey and co-workers,[13] very little effort has been expended characterizing the negatively charged species formed with such a source.

The pulsed discharge source is expected to be a powerful tool for making negative

---

Figure 2.6 A circuit diagram and schematic of the electron gun.

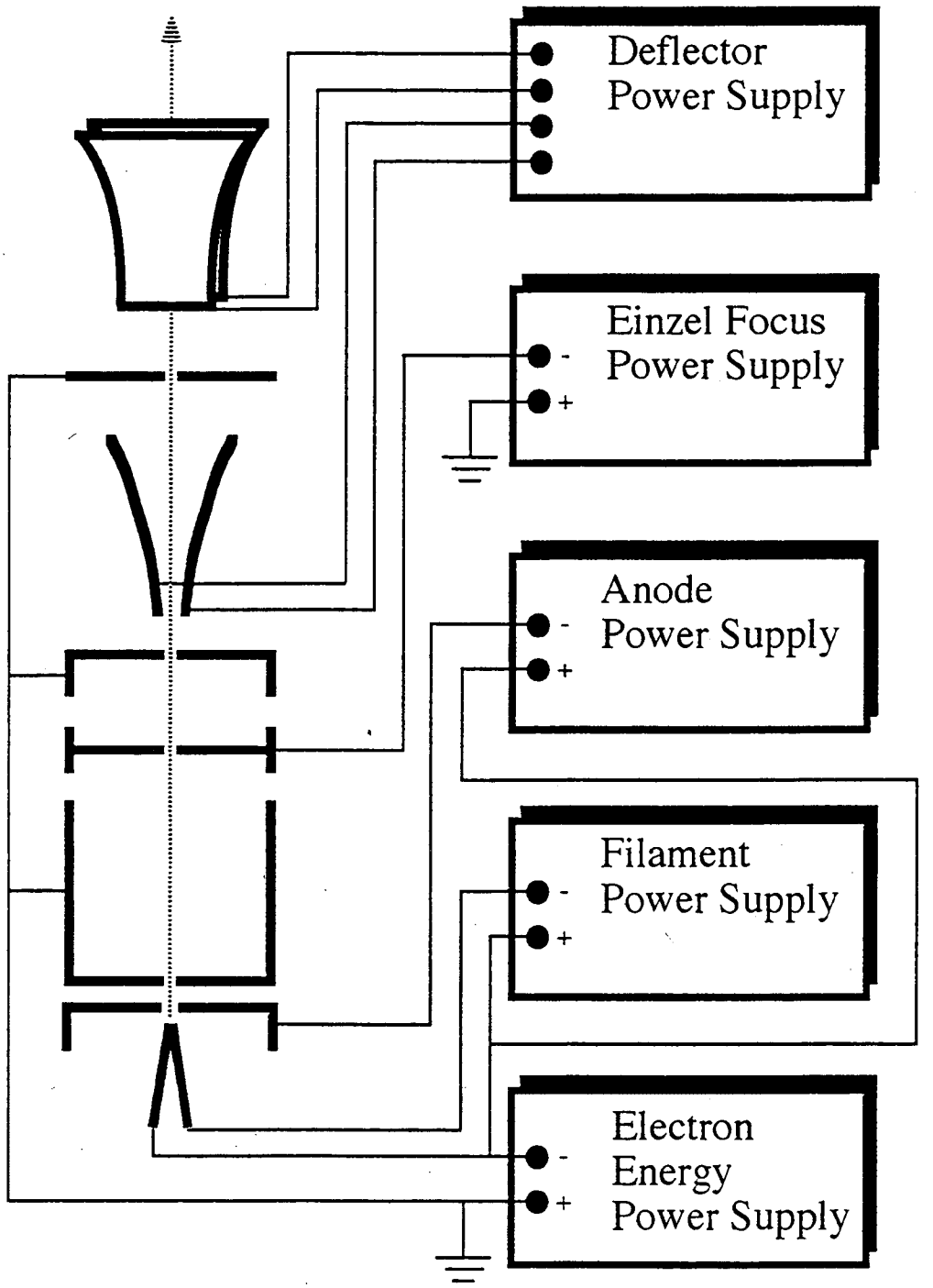

Deflector
Power Supply
Einzel Focus
Power Supply
-
+
Anode
Power Supply
-
+
Filament
Power Supply
-
+
Electron
Energy
Power Supply
-
+

Figure 2.6

ions. The combination of the presence of large numbers of radicals, typically having positive electron affinities, and the electrons associated with the discharge plasma seems to be an attractive one for negative ion formation. In addition, as researchers have shown in the production of radicals, the negative ions that may be produced do not necessarily have to bear close resemblance to the precursor molecule(s). This is typically a requirement for the electron gun/molecular beam source currently in use, and it is easy to imagine cases where this is restrictive.

We therefore can hope to expand our arsenal of negative ions, forming species through rearrangements and breakdown of larger molecules in the relatively violent environment of the discharge. As an example, it is not clear how to use the current source to make CCN^-. Since Bondybey and co-workers were able to generate CNN using a mixture of CS_2, N_2, and Ar in a pulsed discharge source,[13] it is hoped to be able to make CCN^- simply by using a mixture of HCCH, N_2, and Ar in such a source.

In certain instances the pulsed discharge source may also be able to produce larger amounts of negative ions that can be made with sources currently in use. Recently, Baker and co-workers[17] studied C_3 by passing neat CO through a pulsed discharge. Carbon clusters have been studied within the Neumark group[18] but were formed using laser ablation. While quite effective, this technique does necessitate a Nd:YAG laser, introducing both repetition rate and financial limitations.

The pulsed discharge source also offers the possibility (thus far not pursued) of vaporizing normally solid material, without the use of a laser.

The design of the pulsed discharge source consists of a pair of electrodes having a channel through them, mounted directly onto the end of the pulsed valve. The version tested thus far,[19] shown in Figure 2.7, uses a modified pulsed valve faceplate as one electrode. This electrode remains at ground, since the body of the pulsed valve is grounded. The faceplate has a 0.015" orifice with a flat face on the pulsed valve side to provide a good seal for the o-ring and a beveled face on the discharge side to match the

0.080" diameter channel in the insulator. The insulator is 4 mm thick and consists of a Teflon disk with a 0.25" inner diameter hole in the center, into which a 0.25" outer diameter alumina (Al_2O_3) tube, having an 0.080" diameter channel through the center, is inserted.

During operation a second stainless steel electrode, fixed to the front of the insulator, is pulsed to a potential above that where breakdown occurs. This electrode is designed to accept inserts that can be used to easily vary the channel length and width, allowing the mixing and expansion conditions to be varied. The inserts are generally fabricated from stainless steel, but can be made from any conductive material. For example, smaller C_n^- clusters, particularly C_2^-, have been observed from this source when the electrode insert was composed of graphite.

By pulsing the electrode potential, two advantages are gained. First, the breakdown can be made to occur in the middle of the expansion, for a relatively brief time, which may assist in cooling the subsequently formed species. Second, the ability to quickly remove the potential from the electrode allows the negative ions that are formed to undergo the subsequent expansion unperturbed by an electric field associated with the

---

Figure 2.7 A cross section of the pulsed discharge source attachments which affix to the pulsed valve in place of the usual faceplate. The top half of this drawing shows a section that is at 45° relative to the bottom half. (1) Stainless steel electrode which functions as faceplate to which the poppet makes an o-ring seal. (2) Alumina (Al_2O_3) cylinder which serves as the inert section separating the two electrodes in which the discharge plasma is created. (3) Teflon spacers. The first helps isolate the two electrodes as well as providing tapped holes for the outer electrode assembly to be fastened. The second, disk shaped spacer covers the steel screws used to fasten the faceplate and first spacer to the pulsed valve body. (4) High voltage electrode insert. This piece, which may be fabricated of any conducting material, has the final orifice through which the free jet expansion occurs. (5) Stainless steel high voltage electrode holder.

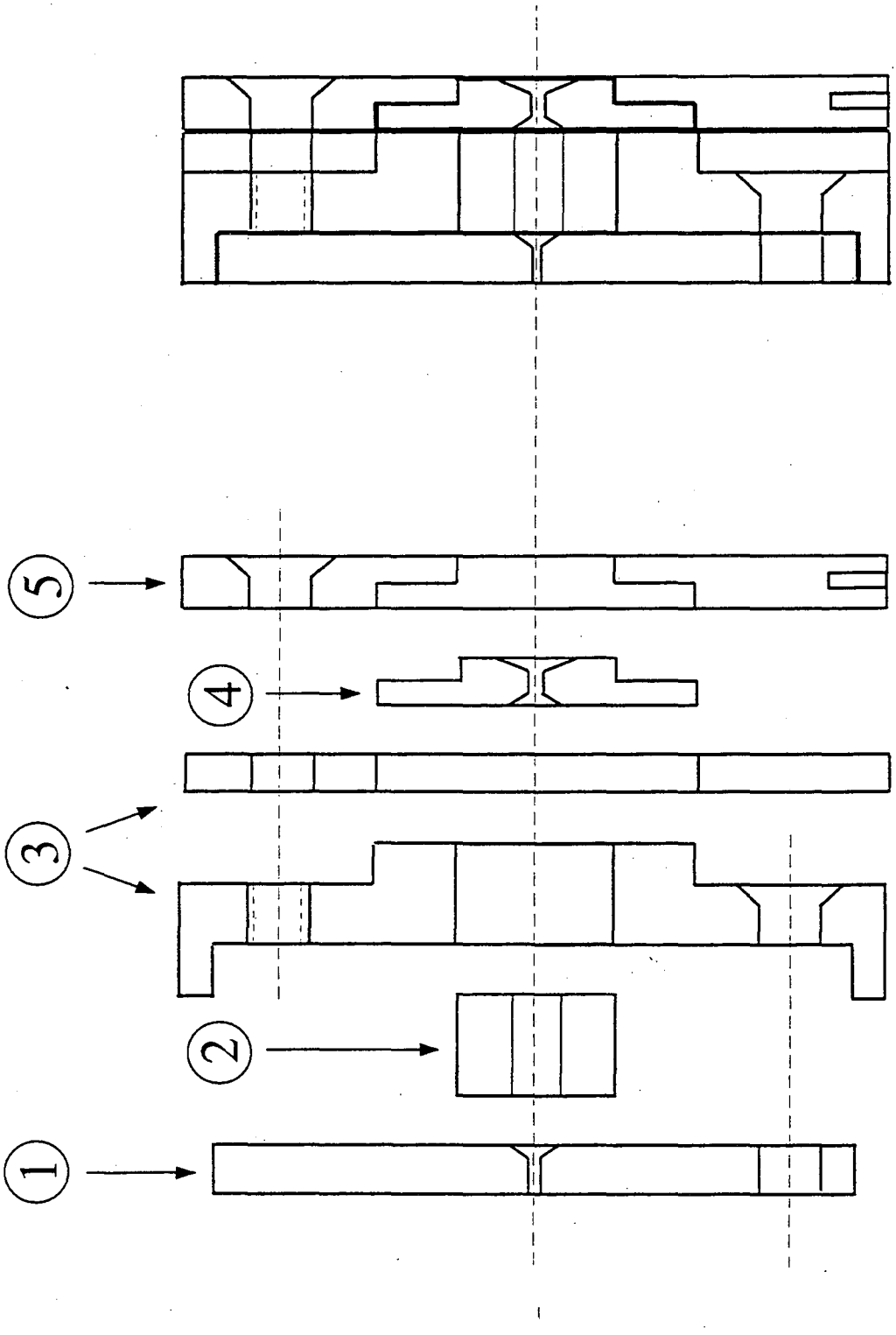


Figure 2.7

electrode. Negative-going pulses of variable duration (1-100 μs) with about a 800 ns fall time and similar rise time are provided by a pulsed, 'square wave', high voltage power supply (Velonex High Power Pulse Generator, Model 350 with V-1097 direct output plug-in module) connected to a very simple circuit, illustrated in Figure 2.8. This particular power supply requires a 200 Ω load to form properly shaped pulses, this resistance value is approached in the circuit shown in Figure 2.8 if breakdown occurs. Otherwise, the load will be 250 Ω, which is completely acceptable. It should be emphasized that non-inductive (denoted NI in Figure 2.8) resistors able to dissipate high power must be used in this circuit. As an illustration, the instantaneous current flowing through 250 Ω when 1000 V is applied is 4 A. This corresponds to a dissipated power of 4000 W if the applied potential were continuous. At our typical duty cycle of (60 Hz × 40 μs =) 0.24% the power dissipated is still 9.6 W. The theoretical limitation of the circuit actually lies with the single 1500 Ω resistor, rated at 10 W. When breakdown occurs and there is a potential of 1000 V across this resistor, nearly 5 W will have to be dissipated under the conditions of the above example. Raising the repetition rate to 100 Hz, or significantly lengthening the discharge time will cause the power dissipated to approach the rating of this resistor. The necessity for these conditions is unlikely, however.

The voltage that is required for a stable and productive discharge depends on many factors, including: instantaneous pressure at the point in the expansion where the discharge is to occur, the specific gases that are present in the expansion, the materials and detailed shapes of each electrode, the inter-electrode spacing etc. In short, because of the wide number of variables, some amount of trial and error is expected in the production of every new species.

The utility of the pulsed discharge source in making negative ions has been the

---

Figure 2.8 A schematic of the pulsed discharge source circuit and a typical discharge oscilloscope trace.

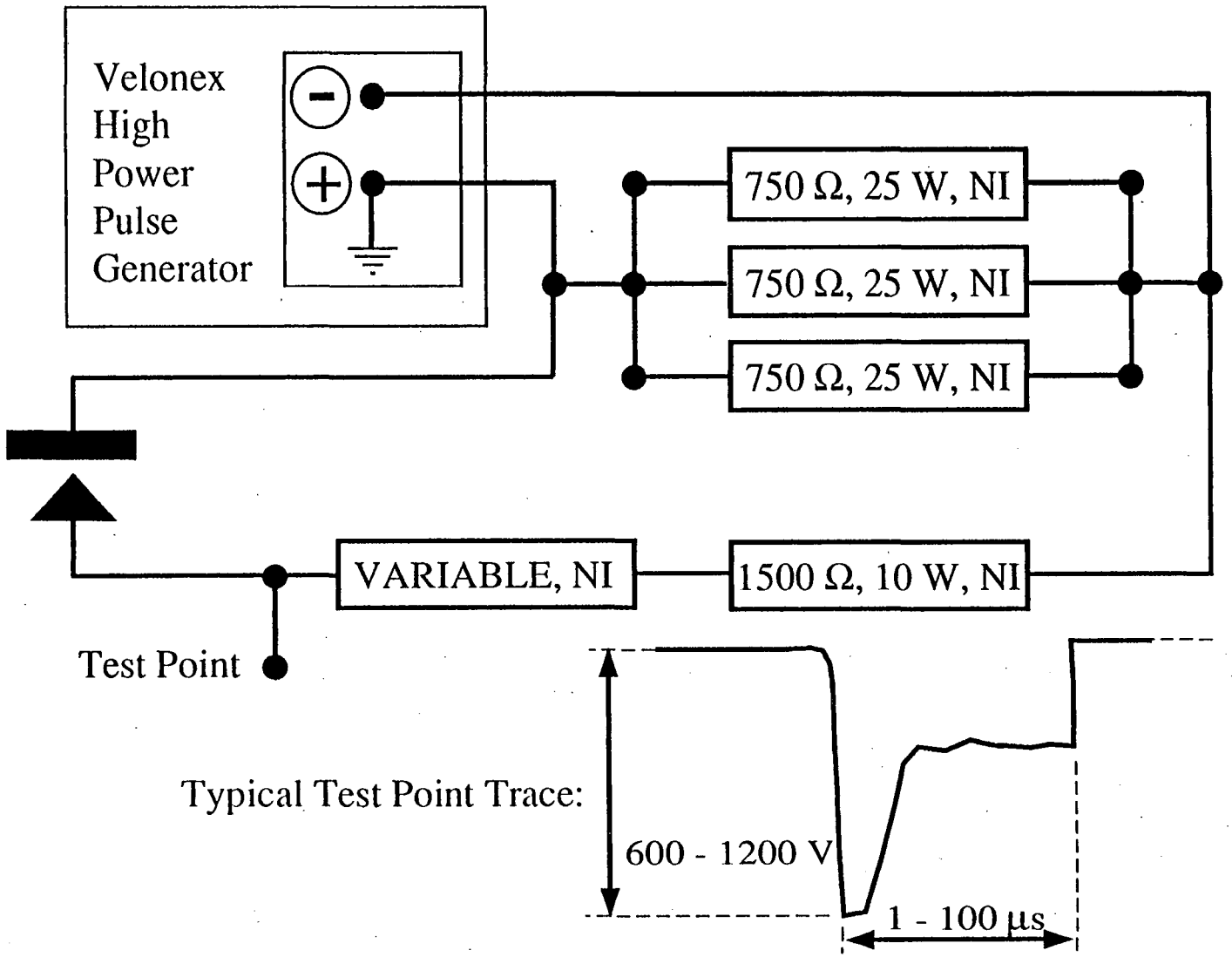

Velonex
High
Power
Pulse
Generator
750 Ω, 25 W, NI
750 Ω, 25 W, NI
750 Ω, 25 W, NI
VARIABLE, NI
1500 Ω, 10 W, NI
Test Point
Typical Test Point Trace:
600 - 1200 V
1 - 100 µs

Figure 2.8

subject of a preliminary investigation using the fast radical beam machine. One critical factor that has been found to be important for satisfactory operation is a stable discharge. A stable discharge can be assured in our case by 'seeding' the discharge; using the electron gun to produce charged species near the front electrode. We have found (quite by accident) that by training a very weak (> 10 μA) electron beam on the exit of the discharge electrode enhances the stability of the discharge enormously. Positive ions accelerated towards the negative high voltage of the outer electrode are thought to be the key to this effect. In the absence of such seeding, the initiation of a discharge is a random process, and therefore an inherently unstable one. By increasing the potential difference between the electrodes the discharge stability can also be improved, but this action can have other effects and may not be desirable in all cases.

One aspect of the discharge circuit that was not fully investigated was the effect of varying the value of the resistor located in series with the discharge (see Figure 2.8). This resistor has the effect of limiting the current flowing in the discharge. This is a very critical factor, effectively determining the type of discharge that is formed. The high currents (~0.5 A, instantaneous current) found in the discharges we have been using likely created a very violent arc-type discharge. Evidence that this is so can be found in the mass spectra obtained under these conditions, with very small fragments dominating. In fact, when a Teflon disk was used as the inter-electrode insulating spacer, so that the walls of the insulator channel were composed of Teflon, the most abundant negative ion species present on operation of the pulsed discharge source was F^-! This is a reflection both of the electronegativity of fluorine and of the violent conditions during the discharge. While these conditions may be desirable in some instances (perhaps the evaporation of solids might then be possible), more delicate negative ions are not likely to survive. In addition, the ability of the free jet expansion to cool such a tremendously hot distribution of ions may be diminished, resulting in vibrationally and rotationally excited ions being produced.

In summary, the pulsed discharge source shows great promise as an alternate method for making certain types of negative ion species. It should be investigated further with the goal of optimizing yields of desired negative ions while maintaining low internal energy distributions following free jet expansion.

2.4 First Differential Region

Within the first differential region, illustrated in Figure 2.9, two important steps in forming a fast ion beam occur. It is here that acceleration to the appropriate beam energy occurs. Subsequently, the ion beam is rereferenced so that we avoid floating either the source or detection regions.

2.4.1 Ion Acceleration

Negative ion acceleration takes place in the first differential region, with the initial effects of the acceleration field felt by the ions as soon as they pass through the skimmer. Ions are accelerated through a potential that is typically 5 to 8 kV. The lower and higher limits are usually dictated by kinematics; the center-of-mass velocity of the radical beam with ideally be chosen such that the largest fraction of coincidence photofragmentation events will hit the active area of the detector face. Achieving this goal involves some trial and error, and compromise between the fraction of events where one (or both) of the fragments impinge on the blocking strip and the fraction where one (or both) of the fragments recoils far enough from the center of mass to miss the detector. The absolute lower limit for acceleration potential depends mostly on the decreasing efficiency of detection of neutral particles with a small amount of laboratory frame kinetic energy; the guideline enforced in this experiment is to use an acceleration voltage high enough so that

Figure 2.9 A cross section of the first differential region. Shown are the skimmer, the acceleration plates and the metal cylinder used for rereferencing.

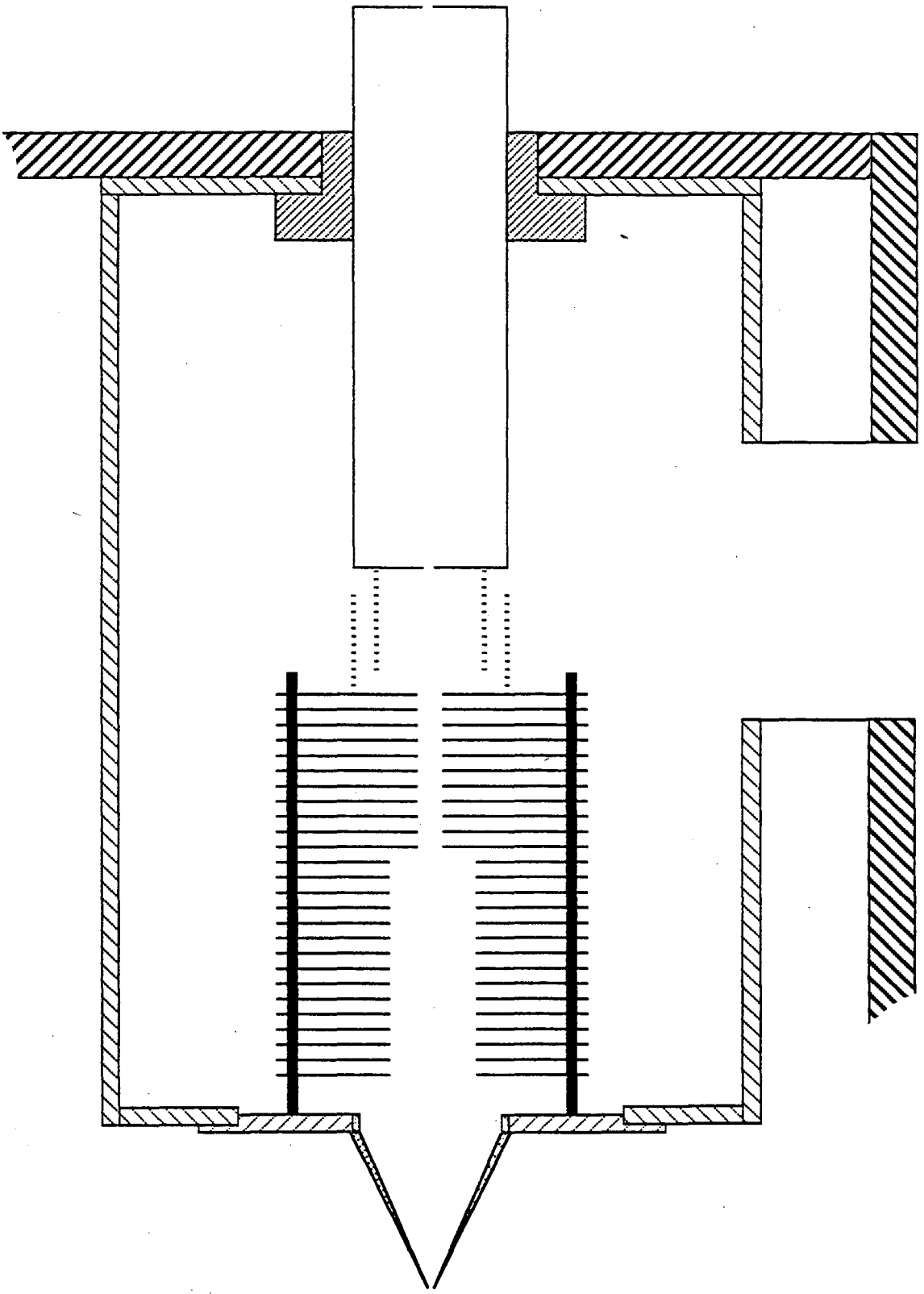


Figure 2.9

all anticipated and/or observed fragment masses will possess greater than 1 keV laboratory frame kinetic energy. The absolute upper limit for acceleration potential is determined chiefly by the limits of the acceleration power supply used and the high voltage capacity of our 'potential switch', to be discussed below.

Figure 2.9 shows the arrangement of the acceleration stack within the first differential region, while Figure 2.10 shows the circuit used to apply the proper potentials to the stack, and the location within the circuit of the 'potential switch', which is to be discussed in the following section. A stack of 26 plates, 6 inches in outer diameter are used to accelerate and gently focus the ions. The first 16 plates are joined through 698 kΩ resistors and have a 2 inch inner diameter. Additional focusing is achieved by having the final ten plates in an decelerating einzel configuration,[20] with their inner diameter reduced to 0.5 inch. The first three and last four plates of these ten are at the voltage corresponding to the final ion beam energy, while the middle three plates are at some variable, though lesser, voltage. The acceleration region einzel lens allows optimization of the focusing effects due to acceleration, and the voltage on the middle three plates in the einzel lens is adjusted to allow the largest number of ions to be detected downstream.

2.4.2 Rereferencing the Ion Beam Potential

After the acceleration process the ions have gained kinetic energy equal to the product of the acceleration potential and the elementary charge on the ion. At this point,

---

Figure 2.10 A schematic of the acceleration stack, showing the acceleration einzel lens and the location of the 'potential switch' within the overall circuit. Details of the 'potential switch' circuit are given in Figure 2.11. Note that the geometry of these components relative to one another is better depicted in Figure 2.9.

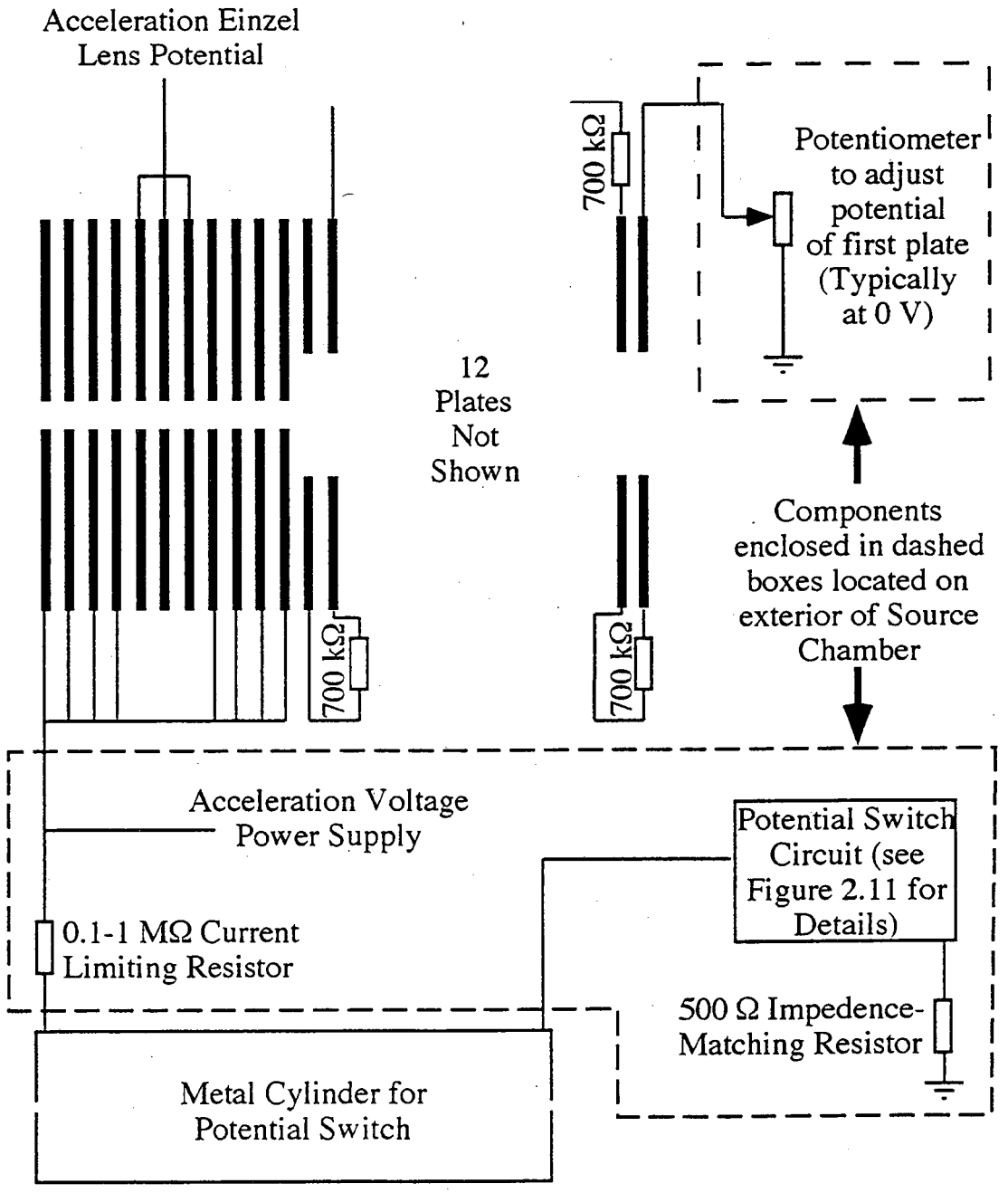

Acceleration Einzel
Lens Potential
700 kΩ
Potentiometer
to adjust
potential
of first plate
(Typically
at 0 V)
12
Plates
Not
Shown
700 kΩ
700 kΩ
Components
enclosed in dashed
boxes located on
exterior of Source
Chamber
Acceleration Voltage
Power Supply
Potential Switch
Circuit (see
Figure 2.11 for
Details)
0.1-1 MΩ Current
Limiting Resistor
500 Ω Impedence-
Matching Resistor
Metal Cylinder for
Potential Switch

Figure 2.10

the designers of many ion beam experiments face a dilemma. For effective beam transport, either the source or the detector section of the apparatus has to float, maintaining a potential difference corresponding to the ion beam energy. In our case this situation would result in many obvious inconveniences.

Pulsed ion beam experiments have been shown by Johnson and co-workers[21] to be amenable to another solution known as rereferencing. However, to our knowledge prior examples where rereferencing has been used were not at the high (4-10 keV) beam energies that are typical for the experiments discussed here.

Rereferencing involves allowing ions to enter an isolated metal cylinder that is initially at the acceleration potential. While the ions are in the cylinder, the voltage is rapidly brought to ground potential. Ions which enter the cylinder before the potential falls, and exit while the potential is at ground, leave the cylinder with virtually the same kinetic energy as they entered, but their kinetic energy has been 'rereferenced'. These ions now possess kinetic energy equal to the beam energy with respect to ground potential, rather than the acceleration potential.

Although perhaps initially non-intuitive, rereferencing works because the ions within the cylinder cross no field lines, and experience no gradient while the potential falls uniformly around them. The fall time of the voltage on the cylinder must be fast so that only a small fraction of the ions inside the cylinder when the potential drop begins will exit prior to the completion of the voltage drop. The specific details of the application of this technique to our apparatus are discussed next, with details of the switching circuit that allows us to satisfy the demanding switching requirements posed by this technique given at the end of this section.

On exiting the acceleration plates, ions enter an isolated 25 cm long, 8 cm diameter, hollow, stainless steel cylinder having 3 mm apertures at the center of each end. This cylinder is normally at the same potential as the final acceleration plate, and is pumped out through mesh that covers portions of the end extending back into the first

differential region. The aperture on the end of the cylinder extending into the second differential serves as the differential pumping aperture between the first and second differential regions. There is a ground shield surrounding the end of the cylinder that extends towards the detector. This ground shield will confine the potential on the cylinder, preventing any possible perturbations of the detached electrons by electric fields from the cylinder.

At some experimentally optimized delay relative to the triggering of the pulsed valve (ca. 280-380 μs, depending on the mass of the desired ions), the potential of this cylinder is taken from the ion beam potential to very close to ground in ~150 ns. When the potential of the cylinder reaches a minimum, all the ions that subsequently emerge from the cylinder will maintain a kinetic energy approximately equal to the full acceleration energy. Only the ions within the cylinder during the rereferencing process will proceed down the machine towards the detector. In this way, the rereferencing process effectively selects a section of ions from a longer stream emanating from the source.[22]

Evidence of this selection can be observed in a time-of-flight spectrum of ions that emerge from the potential switch, even with the mass spectrometer apparatus disabled. Since different masses travel at different speeds if they possess the same kinetic energy, it is possible to obtain a very low resolution mass spectrum by using the potential switch to select an initial bunch of ions, which then separates according to mass. A spike on the short-time side of the broad time-of-flight peak for ions mass selected in this manner is thought to come from ions that leave the cylinder before the cylinder potential hits the minimum. These ions enjoy a comparative 'head start', but are handicapped by having a lower kinetic energy. They therefore pile up, arriving simultaneously with the earliest ions of proper kinetic energy. Thus, although it is quite tempting to optimize this spike for use in an experiment, it must be avoided. Since the ions in this spike have a large kinetic energy spread, they will diverge as they continue down the machine.

2.4.2.1 Details of the 'Potential Switch' Rereferencing Circuit

For several months we flirted with an unsatisfactory circuit for quickly dropping the voltage on the metal cylinder from the acceleration voltage to ground, which involved capacitively coupling the output of a pulsed high voltage power supply. It was decided that this method involved too much shot-to-shot jitter in beam energy and a somewhat unreliable power supply; in short, a new scheme was needed.

The new generation of potential switch[23] was assembled and tested primarily by Dr. Robert Continetti following a suggestion from George Gabor and advice from Oren Milgrome, both of whom are associated with Lawrence Berkeley Laboratory. It involves an inexpensive yet reliable fast-switching metal-oxide semiconductor field-effect transistor (MOSFET) circuit to function as a switch, quickly closing to complete a connection from the cylinder to ground when triggered. Although power MOSFETs capable of rapidly switch up to 1 kV are available, switching higher voltages demands the use of schemes in which MOSFETs are implemented in series. The design utilized in the original MOSFET potential switch to rapidly switch up to 8 kV is based on a series of 10 transformer-isolated, varistor-bypassed, 1 kV power MOSFETs. A key to this approach is the use of metal-oxide varistors in parallel with each MOSFET in order to clamp the voltage across a given MOSFET to less than the rated voltage. Metal-oxide varistors display a nonlinear resistance as a function of voltage, having extremely high resistance below their characteristic turn-on voltage, while becoming quite conductive if the turn-on voltage is exceeded. Because of these characteristics they serve as rugged, fast-acting surge suppressers.[24] Their presence in series with the power MOSFETs allows the requirement for accurate synchronization of the MOSFET triggering to be relaxed, greatly simplifying this aspect of the design.

The current version of this circuit was recently assembled by David Osborn, closely following the original circuit design. However, it contains more recently available components and boasts both an upper voltage limit of up to ~15 kV (using a maximum of

16 MOSFET stages) and a modular design, with MOSFET stages on plug-in circuit board cards. This affords the ability to use a minimum number of stages for the voltage to be switched to ground, which has a slight minimizing effect on fall time. It is also very convenient for when the rare but inevitable component failure occurs.

As can be seen in Figure 2.11, the circuit consists of up to 16 identical varistor-bypassed 1 kV MOSFET stages (Harris, RFP4N100) connected in series, with one end of the chain connected to the cylinder and the other end at ground. The metal oxide varistors used are rated at 920 VDC (Panasonic ZNR 20K102U). The MOSFETs are simultaneously triggered in groups of up to five each through transmission line transformers by high speed FET drivers (Teledyne, TC4422CPA). Each transmission line transformer was made by wrapping 17 turns of RG 316/U coaxial cable around a ferrite toroid (Fair-Rite #59-77-001601), using the outer conductor of the coaxial cable as the primary and the inner conductor as the secondary of the transformer.[25] An optoisolator (Hewlett-Packard, HCPL-2601) is used to prevent damage to the external pulse generator supplying the trigger pulse in the event of a failure.

A current-limiting resistor is placed between the acceleration voltage power supply and the cylinder. This is necessary since the ON resistance of the power MOSFETs is quite low (~3.5 Ω per MOSFET); without this resistor there would effectively be a short circuit from the high voltage power supply to ground. This would not only quickly destroy the MOSFETs but exceed the current rating of the high voltage power supply. The current-limiting resistor value should be large enough to allow the cylinder to closely approach ground. There is effectively a voltage divider determining the potential on the cylinder, with the current limiting resistor dropping most of the potential, but the combined resistance of the MOSFETs accounting for a small potential drop as well. The

Figure 2.11 A schematic of the 'potential switch' circuit used to rereference the ion beam by bringing a metal cylinder from as high as 15 kV down to ground potential in <150 ns.

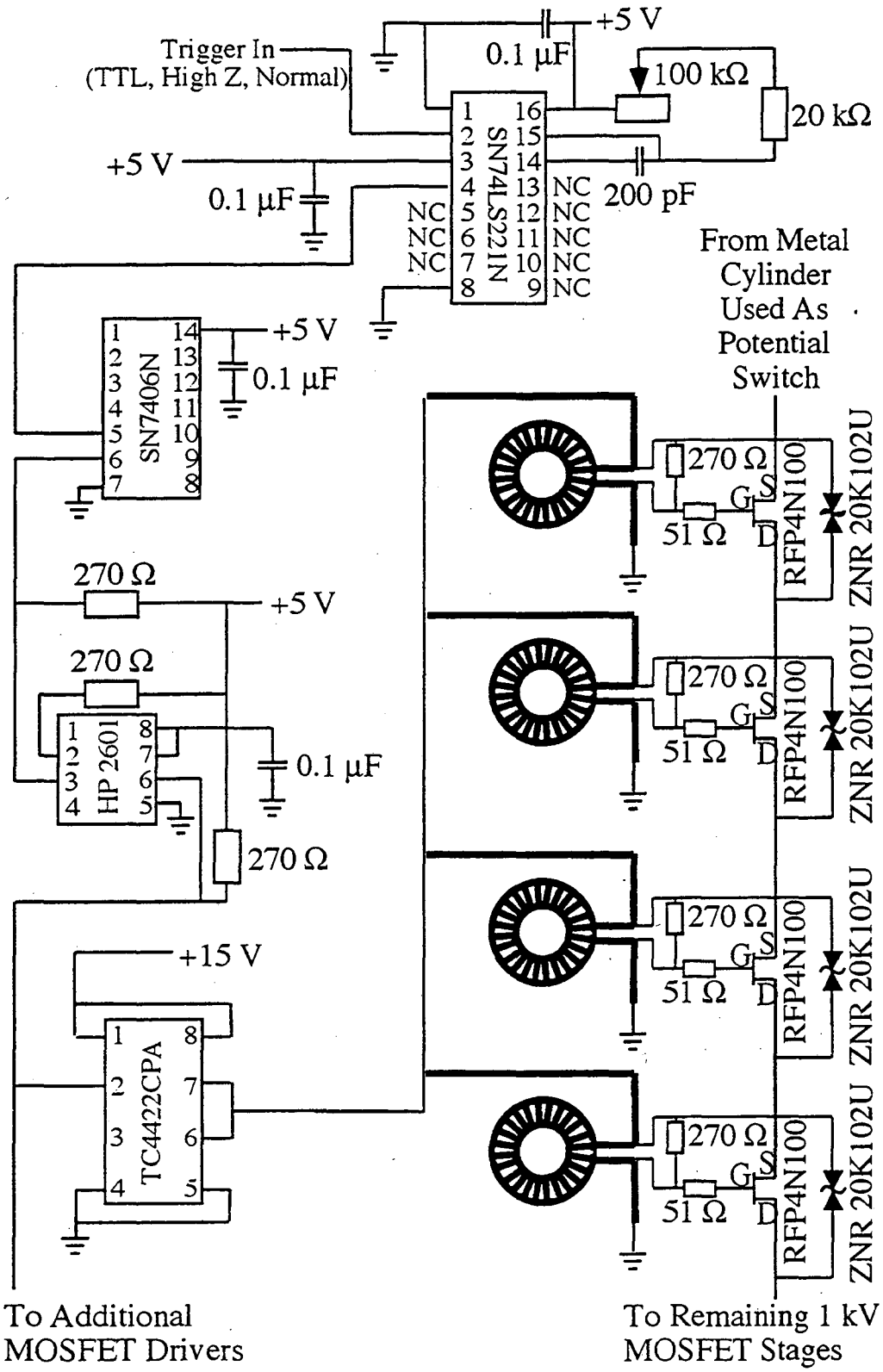

Trigger In
(TTL, High Z, Normal)
+5 V
0.1 µF
100 kΩ
20 kΩ
200 pF
SN74LS221N
NC
SN7406N
HP 2601
TC4422CPA
+15 V
270 Ω
51 Ω
RFP4N100
ZNR 20K102U
From Metal
Cylinder
Used As
Potential
Switch
To Additional
MOSFET Drivers
To Remaining 1 kV
MOSFET Stages

Figure 2.11

current limiting resistor value cannot be too large, however, because the cylinder must charge back up to the acceleration potential between consecutive iterations. Charging of the cylinder occurs when the MOSFET switch opens and current trickling through the current limiting resistor can no longer go to ground. Instead, this current brings the potential of the metal cylinder up to the acceleration voltage.

A final concern is that the current limiting resistor be of appropriate power rating. At least a 2 Watt rating has been found to be necessary, in agreement with quick calculations (i.e., assuming 10 kV acceleration voltage, 100 kΩ resistor, and a 60 Hz × 20 μs = 2×10^{-3} duty cycle, one gets 1.2 Watts as the power dissipated).

Since our application requires only that the falling edge be fast, the rising edge is described by the RC time constant $R_{lim}C_{cyl}$, or 10 μs, for a 100 kΩ current limiting resistor and a 100 pF cylinder. Were it necessary to have two fast edges, it would be possible to use two MOSFET switches in a push-pull version of this circuit. This type of circuit is well suited to fast, high voltage switching applications, and should be considered in all cases where these characteristics are desired.

2.5 Second Differential Region

The second differential region contains a beam modulation time-of-flight mass spectrometer, ion beam deflectors and various focusing optics. The geometries of these components and their locations within this region are illustrated in Figure 2.12. These components, discussed below, provide mass selection of a specific ion and the ability to

Figure 2.12 A diagram illustrating the various components present within the second differential region. (1) Metal cylinder used for rereferencing surrounded by a ground shield. (2) Beam modulation deflection plates. (3) Ion compressor plates. (4) Ion beam vertical and horizontal deflection plates. (5) Einzel lens. (6) Gate valve. (7) 1 mm aperture that serves as a differential pumping aperture in addition to the beam modulation time-of-flight mass spectrometer slit.

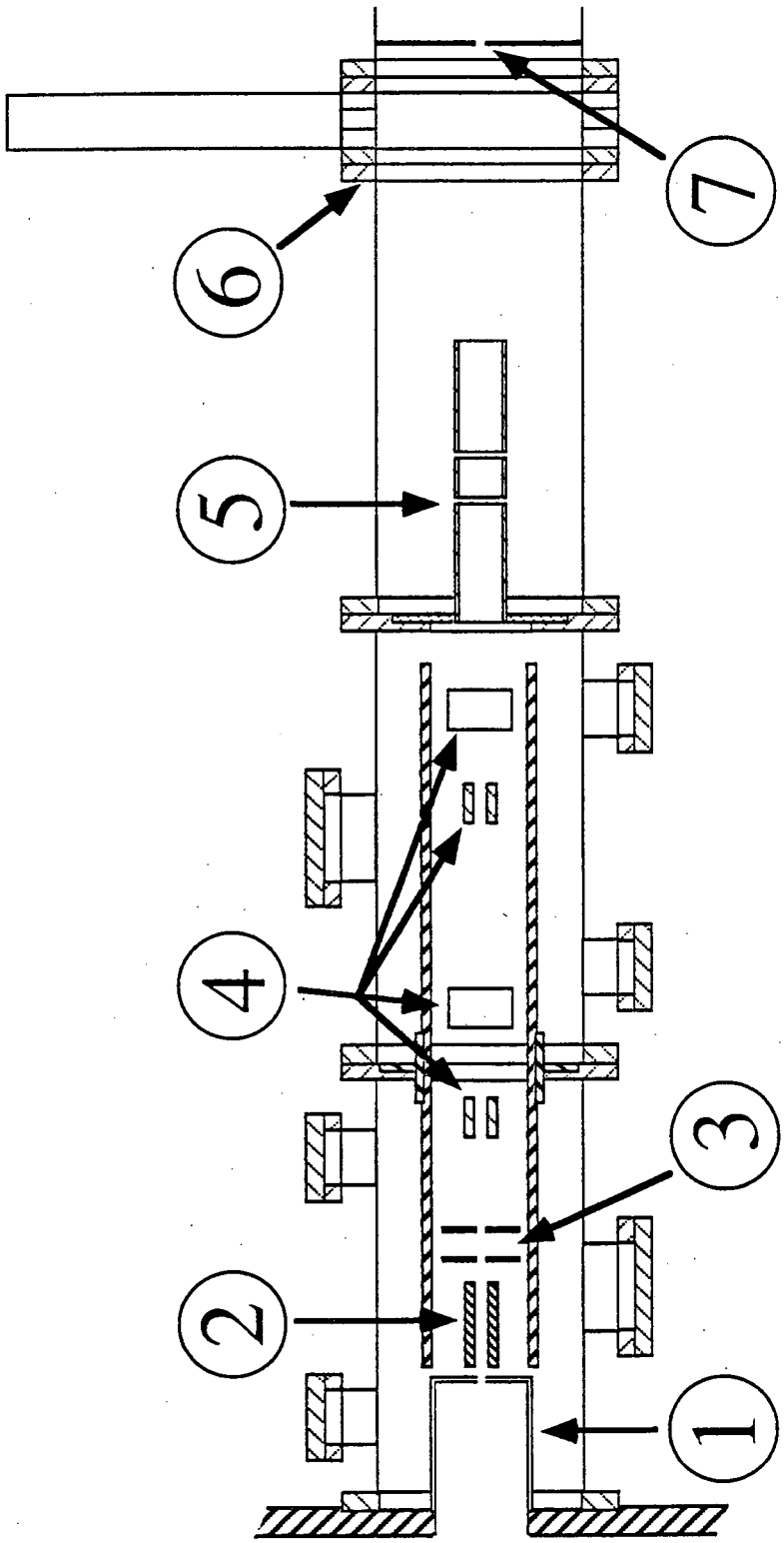


Figure 2.12

steer and focus the ions as necessary, including the compression of an ion packet along the beam axis.

2.5.1 Beam Modulation Mass Spectrometer

The time-of-flight (TOF) mass spectrometer used in this experiment is a collinear beam modulation type, first described by Bakker.[26] The Bakker-type beam modulation mass spectrometer works by in effect selecting a compact bunch of ions, which will then separate spatially and temporally according to mass because their equal kinetic energies imply different, mass-dependent velocities. The selection process is accomplished simply by rapidly reversing the electric field between a pair of plates whose faces are perpendicular to the ion beam axis. The beam modulation deflection plates are the first components located in the second differential region, [labeled (2) in Figure 2.12]. Only the ions which experience an equal but opposite amount of acceleration from the initial field and the reversed field (i.e. those ions very near the center of the plates when the field is reversed) will continue along parallel to the beam axis and be admitted through the initial 1 mm defining aperture 0.86 m from the center of the plates [labeled (7) in Figure 2.12].

Figure 2.13 shows the electronic circuit that is used to generate the rapidly reversing electric field. This circuit reverses the field between the two mass spectrometer plates by maintaining a voltage V_0 on one plate until the arrival of a trigger pulse, at which point the voltage on this plate goes to ground in < 20 ns. The other plate is held at a continuous potential of $V_0/2$, as determined by a voltage divider formed by the two 10 kΩ resistors shown in Figure 2.13.

The beam modulation mass spectrometer has the advantage of introducing only a negligible kinetic spread in the ion beam energy. The commonplace Wiley-McClaren[27]

Figure 2.13 A schematic of the beam modulation mass spectrometer circuit.

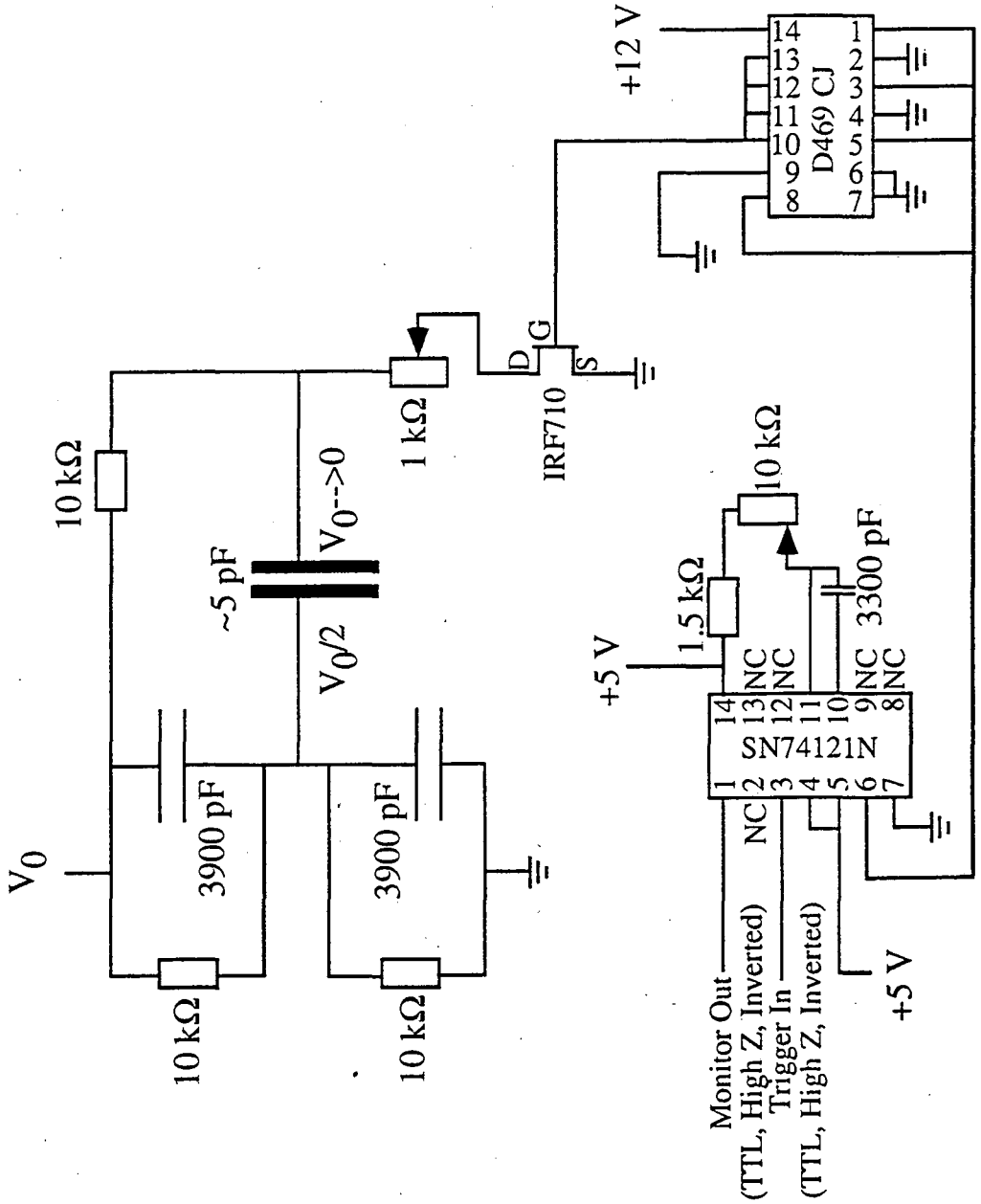

+12 V
D469 CJ
1 kΩ
IRF710
D
G
S
10 kΩ
~5 pF
V0-->0
V0/2
V0
3900 pF
3900 pF
10 kΩ
10 kΩ
+5 V
1.5 kΩ
10 kΩ
3300 pF
SN74121N
NC
NC
NC
NC
Monitor Out
(TTL, High Z, Inverted)
Trigger In
(TTL, High Z, Inverted)
+5 V

Figure 2.13

type space-focusing technique was not used in this experiment because it, on the other hand, introduces a significant kinetic energy spread. Maintaining a compact kinetic energy distribution within the negative ions is an important and desirable characteristic for our dynamics experiments. Any increase in the spread of the beam energy, and therefore velocity, results in the formation of a packet of ions that is divergent along the beam axis. If the kinetic energy distribution of the ions in the beam is broad enough, it will be harder to overlap with the detachment and dissociation laser pulses. In addition, a wider kinetic energy distribution in the ions transfers directly to the radicals formed in detachment. For reasons to be addressed in § 2.7.6.5, this ultimately results in an increased uncertainty in the measurement of photodissociation kinetic energy release, and to a minor, extent, the recoil vector angle.

Bakker has shown[26] that the theoretical resolving power is given by:

$$R = \frac{L^2 V_0}{2DU(B+S)} \tag{2.5}$$

This formula is appropriate when the rise time of the square wave is short compared to the transit time between the plates, which itself is short compared with the field-free drift region, as is the case in our apparatus. Here, L is the length of the field-free drift region, V_0 the potential difference of the step function applied to the plates, D the separation of the deflection plates, U the ion beam acceleration voltage, B the beam width and S the defining aperture width. The values in Eqn. (2.5) that are fixed in our machine are as follows: D = 1 cm, B = 0.4 cm, S = 0.1 cm. In terms of L, the initial 1 mm beam defining aperture is located 86 cm from the center of the beam modulation plates. However, for the full experiment the ultimate mass spectrometer resolution depends on a second 1 mm aperture located 153 cm from the center of the beam modulation plates. This effectively results in a factor of $(153/86)^2 = 3.17$ higher resolution, since Eqn. (2.5) shows that the resolving power is proportional to the square of the drift region length.

V_0 and U are variables for any given system, but a typical value for U is 6000 V. In this situation the resolving power past the first 1 mm aperture is roughly ~1.2 × V_0. Therefore, if 100 V is used, the mass resolution will be ~120 amu. Typically, however, the minimum voltage is used to resolve a given mass to charge ratio, and these values of V_0 are closer to 40 V. The values appropriate at the first aperture are given because resolution effects of the mass spectrometer voltage are most conveniently monitored using the ion time-of-flight measured at the ion detector. This detector is located after the first 1 mm aperture but before the second.

As can be seen from Eqn. 2.5, resolution is independent of ion mass. It can also be shown[26] that there is no mass-dependent discrimination, i.e., all masses are transmitted with equal efficiency by the beam modulation mass spectrometer. These last two favorable characteristics round out the list of reasons the beam modulation time-of-flight mass spectrometer is well suited for our experiment.

Some idea of the ion burst duration can be obtained from an additional formula given by Bakker,[26] again appropriate when the rise time of the square wave is short compared to the transit time between the plates, which itself is short compared with the field-free drift region:

$$\Delta t = \frac{(B+S)D\sqrt{\frac{2Um}{e}}}{2V_0 L}. \qquad (2.6)$$

When typical values of B = 0.4 cm, S = 0.1 cm, D = 1 cm U = 6000 V, V_0 = 100 V and L = 86.6 cm are used, the ion burst duration for mass 32 amu, corresponding to the O_2^- anion, will be ~18 ns. This illustrates how temporally compact the ion packet is. Under these conditions the size of the packet along the beam axis will be ~3 mm if the ion compressor, to be described next, is not used.

2.5.2 Ion Compressor

The ion compressor can be used to focus a group of ions in the dimension parallel to the ion beam axis. The focal point is usually chosen to be midway between the laser interaction regions to improve overlap with the output of both lasers, thus improving overall signal intensity.

Figure 2.12 shows the location of the ion compression components. The ion compressor consists of two 7.5 cm outer diameter, 0.5 cm inner diameter plates spaced 2 cm from each other, concentric with, and perpendicular to the ion beam. These plates are located immediately following the mass spectrometer plates. Ion compression involves the application of a pulsed negative potential to the rear plate with the front plate held at ground, while the negative ions to be focused are completely between the two plates. In this way, ions closest to the back of the packet are accelerated through the largest potential, gain the most kinetic energy, and will therefore eventually overtake negative ions that were closer to the front plate when the potential was switched on.

Figure 2.14 shows the electronic circuit that is used to generate the pulsed electric field used in the operation of the ion compressor. This circuit is extremely similar to the beam modulation mass spectrometer circuit, with the exception that the front plate is held at ground potential, rather than at half the step function voltage of the other plate.

Use of the ion compressor has no adverse effects on a total cross section experiment, and is recommended without reservation in that instance. Ion compression will, however, degrade the kinetic energy release resolution in the dynamics experiments for the same reason a Wiley-McClaren design mass spectrometer would, by increasing the spread of the ion beam energy. The increased uncertainty in the quantity will ultimately blur the determination of the photodissociation kinetic energy release distribution.

Figure 2.14 A schematic of the ion compressor circuit.

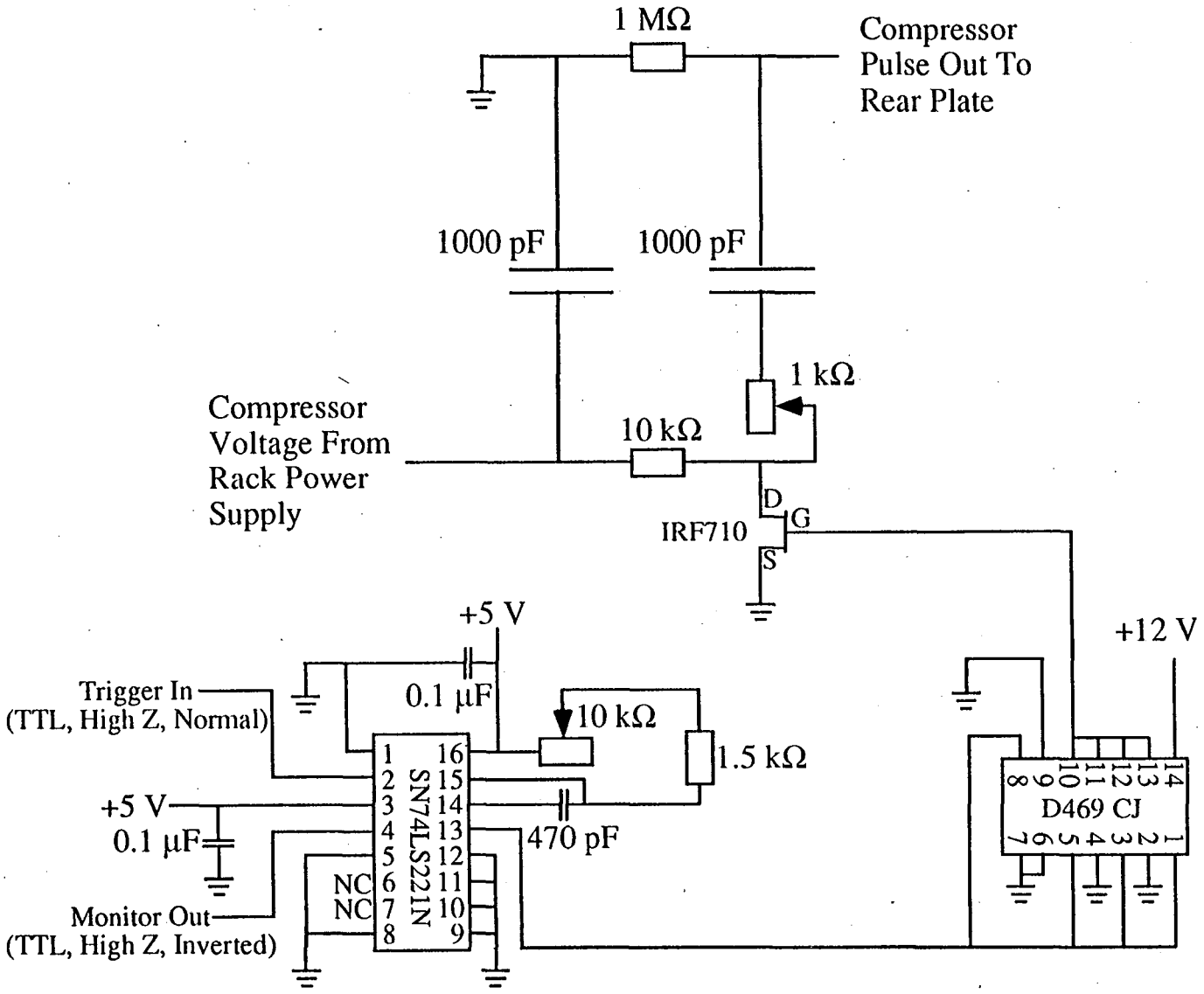

1 MΩ
Compressor
Pulse Out To
Rear Plate
1000 pF
1000 pF
1 kΩ
Compressor
Voltage From
Rack Power
Supply
10 kΩ
D
G
IRF710
S
+5 V
+12 V
Trigger In
(TTL, High Z, Normal)
0.1 µF
10 kΩ
1.5 kΩ
+5 V
0.1 µF
SN74LS221N
470 pF
NC
NC
Monitor Out
(TTL, High Z, Inverted)
D469 CJ

Figure 2.14

2.5.3 Ion Optics

The ion optics, save the acceleration einzel lens, are found in the second differential region. As depicted in Figure 2.12, they follow the mass spectrometer plates and ion compression plates and consist simply of two sets of vertical and horizontal deflector plates and a second einzel lens. The deflectors allow a slight redirection of the ion beam to maximize ion and neutral transmission through the two 1 mm collimation holes downstream. Each set of deflection plates are 1" long in the direction of the beam axis and have a 1" separation, symmetric about the beam axis. They are mounted on a rail which allows their position along the beam axis to be easily adjusted when accessed through side flanges. Quite often, however, it is the case that these deflector plates are kept at ground, their influence unnecessary. When they are required to be on at any significant voltage it is usually a clear sign that something else is amiss.

An einzel lens comprised of three cylindrical elements follows these deflectors. It is generally operated in the deceleration mode, with the central element at a negative potential and the first and last elements at ground. The voltage used is mass- and beam energy-dependent, but is typically around -700 V. Higher voltages can improve throughput of ions by focusing the ion beam more tightly between the two 1 mm apertures. However, this results in a larger degree of divergence at the fragment detector(s). This can be a problem. For example, the radical beam diameter may grow larger than the 3 mm radical beam blocking strip on the total photodissociation cross section detector. When this happens, background from undissociated radicals will contaminate the photofragment signal.

2.6 Third Differential Region

The third differential region is illustrated in Figure 2.15. Within this region the ion, and then radical, beam undergoes its most rigorous collimation, with 1 mm apertures forming the entrance and exit to this region. Here the ions having the appropriate mass

are photodetached, and the photoelectrons generated in this process are collected. In addition, determination of the negative ion species present in the beam can be undertaken simultaneously via time-of-flight mass spectroscopy. This ability assists in determining the intensity and identity of a given negative ion, particularly useful when initially generating a negative ion that had previously not been produced using this apparatus.

2.6.1 Photodetachment

Photodetachment is carried out using the output of a tunable excimer-pumped dye laser that produces 20 ns, ~20-50 mJ pulses at wavelengths extending from ~335 nm to >700 nm (Lambda Physik LPX 210i and FL3002, respectively). The laser output is gently focused by a 1 m lens, with the photodetachment volume located about 0.3 m from the lens. The laser is timed to intercept the negative ions that have a mass corresponding to the radical of interest.

The detachment wavelength is judiciously chosen so as to form radicals in the ground vibrational state, according to the principles discussed in § 1.4. Detachment efficiencies can approach 80%; with high laser power and optimal timing, this efficiency can be observed in the depletion of the ion mass of interest or, in relative terms, by the size of the detached electron signal.

2.6.2 Laser Timing Jitter

Timing jitter is a serious concern in this experiment. The dye laser temporal profiles for both the detachment and dissociation laser systems (as the laser systems are

---

Figure 2.15 A diagram of the third differential region. (1) Gate valve. (2) Initial 1 mm aperture. (3) Photodetachment volume straddled by electron extraction plates. (4) Detached electron detector surrounded by grounded mesh. (5) Undetached ion deflector. (6) Ion detector, shown in the fully retracted position. (7) Final 1 mm aperture.

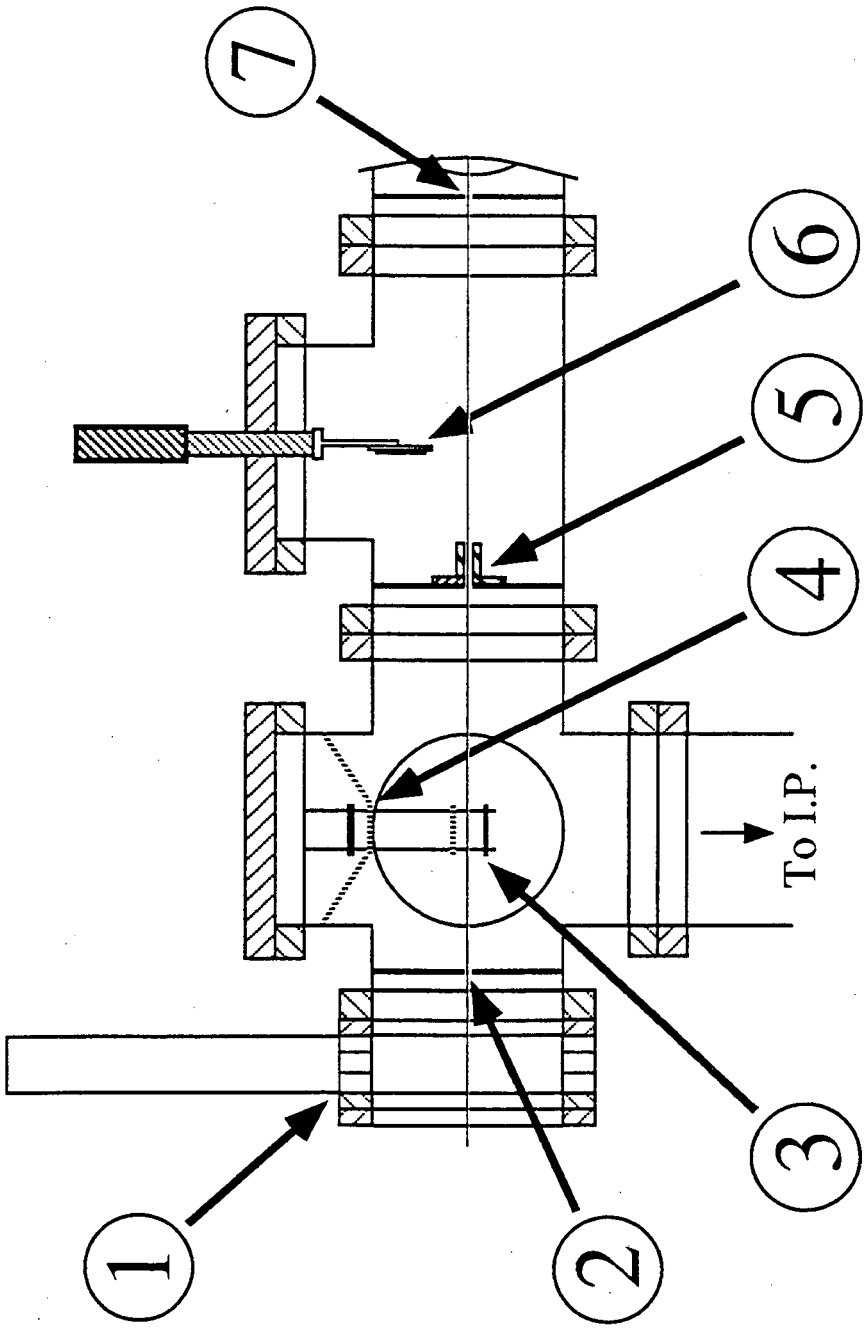


Figure 2.15

identical) are determined chiefly by that of the excimer pump lasers. The most intense portion of the laser pulse is confined to ~28 ns. The same packet of ions, and, following detachment, radicals, must be intercepted by both detachment and dissociation laser pulses in order for radical photodissociation to occur. Adding to the complexity of this task is the fact that, depending on the ion mass, beam energy and mass spectrometer voltage, the ion burst duration is on the order of ~18 ns, as shown in § 2.5.1. The excimer laser specification for timing jitter is 4 ns, which is acceptable. However, as the thyratron (the high voltage, high current switch that controls the discharge current) in an excimer laser ages, this specification can be exceeded, sometimes dramatically.

Fast photodiodes are employed to monitor the timing jitter of both the detachment and dissociation lasers. A photodiode circuit is shown in Figure 2.16. The key component of this circuit is a quartzface (for transmission of UV light) photodiode (Motorola, Model MRD510) reverse-biased by a 90 V potential. A low-intensity back-reflection from the output of the dye laser that is to be monitored is made to fall on the face of the photodiode. The response time of the photodiode is approximately 1 ns, sufficient to follow the temporal profile of the dye laser pulse. The output of the photodiodes can be observed using the oscilloscope.

Once the optimal timing for a given laser has been established, monitoring the temporal characteristics of the laser pulses with fast photodiodes can serve two purposes. The photodiode output can be monitored for excessive shot-to-shot timing jitter, as well as long-term drift in laser timing. Adjustment of the thyratron reservoir heater is usually required to correct shot-to-shot timing jitter. Long-term drift is usually the result of thermal variations as the excimer lasers run, except when the gas fill or the thyratron is near the end of its useful life. It is therefore mostly observed as a shortening in the delay between trigger pulse and the excimer laser firing during the initial period of operation,

---

Figure 2.16 A fast photodiode circuit diagram.

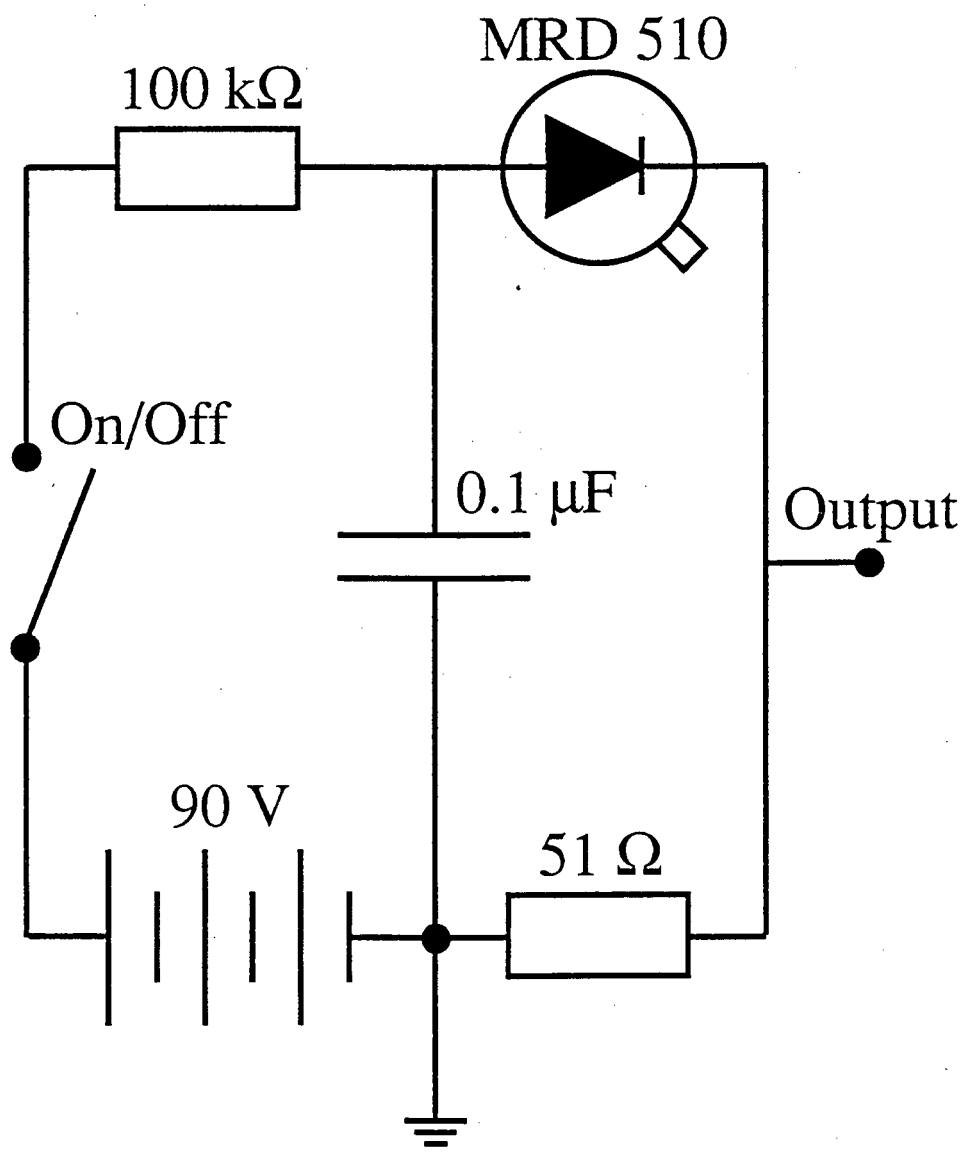


Figure 2.16

prior to thermal equilibrium within the excimer laser. As the gas fill becomes old, timing drift in the opposite direction is sometimes observed. Small adjustments in trigger pulse timing can be made periodically after consulting the photodiode output to correct for long-term drift.

2.6.3 Detached Electron Detection

Detection of detached electrons provides a measure directly proportional to the number of radicals produced in the detachment process. This information is useful for evaluating and adjusting the spatial and temporal overlap of the detachment laser with the negative ion precursor of interest. It is also a critical for proper normalization of total photodissociation cross section spectra.

The gross features of the detached electron detector assembly can be discerned in Figure 2.15. Detached electron detection is accomplished by using a small, pulsed, electric field to accelerate detached electrons perpendicular to the beam axis, up into a final acceleration field which is just prior to the face of a microchannel plate detector assembly.

Microchannel plate detectors serve as large area electron multipliers. The plates themselves are typically thin (1-2 mm) circular disks available in various diameters. All microchannel plate detectors on this machine, with the exception of the time-and position-sensitive detector, use a set of two microchannel plates (Galileo Electro-Optics Corporation), of diameter 25 mm or 40 mm. The detached electron detector uses 25 mm diameter microchannel plates. They are made of a semiconducting glass material and consist of close-packed, regularly spaced channels that are about 10 μm in diameter and 12 μm from center to center, at an angle of 13° with respect to the parallel surfaces. The material is highly resistive and a high potential can be placed across them. A photon, neutral particle, ion or electron of sufficient energy will cause electron(s) to be emitted by the microchannel plate at the point of impact. Microchannel plates then function in a

similar manner to a photomultiplier. The primary electrons are accelerated in the electric field perpendicular to the surface, down into one or more of the channels. Because these channels are at a slight but significant angle away from being perpendicular to the plane of the microchannel plate, the electrons soon collide with the wall of the channel with sufficient energy to cause additional electrons to be released. This cascading effect continues, and gains of up to 10^3-10^4 per plate can be achieved. It is estimated that the set of two microchannel plates used in the detached electron detector obtain a gain of 10^6.

The 'chevron mounting' configuration is used in all of the microchannel plate detectors on this machine (save the time- and position-sensitive detector, whose three plates form an analogous 'Z-stack'), in which the angles at which the channels are oriented is reversed between plates. This reduces ion feedback, the process where positive ions are formed through electron bombardment and accelerated back in the opposite direction, eventually causing spurious counts.

The details of the overall detector assembly and its application to the detection of detached electrons will now be given. A set of three threaded rods screwed into the top flange of a six-way cross supports all components of the detector. Extraction plates hang down from the detector, mounted near the end of these threaded rods. The extraction plates are oriented horizontally about 1" apart, equidistant from, and parallel to the ion beam axis. The upper extraction 'plate' is made using a high (~90%) transmittance steel screen spot-welded to a steel ring. A +6 V pulse is applied to this upper plate just as detachment occurs. Detached electrons are accelerated through this mesh and on towards the detector by this extraction pulse. The extraction potential is weak compared to the beam potential and pulsed on only just before detachment has occurred (at the earliest) so that the trajectory of the negative ions is not affected.

The detached electron detector is mounted in a perpendicular orientation to the beam axis. A grounded grid made from high transmittance steel mesh spot-welded to a steel ring is supported by the threaded rods, concentric with both the outer diameter of the

microchannel plates and the extraction plates, and located ~1" from the front of the first microchannel plate. An acceleration voltage is set up between this grid and the microchannel plates by biasing the front of the first microchannel plate at ~+300 V. This enables the detached electrons (possessing on the order of 3 V kinetic energy from the extraction pulse) to gain enough kinetic energy to be detected.

When struck by an energetic electron anywhere on the front of the first microchannel plate, ~10^6 electrons will cascade from the back of the final microchannel plate. These electrons are accelerated across a final 100 V potential difference onto a contiguous steel disk that serves as an anode. This signal is capacitively coupled out to a BNC feedthrough, providing a fast, 5 ns wide, 10-40 mV negative-going peak.

The circuit which is used in the operation of the electron detector is shown in Figure 2.17. This circuit consists of a resistor network that ensures an equal potential drop across the two MCPs, with a 100 V Zener diode ensuring a constant drift region potential between the back of the last MCP and the anode. The output is AC coupled from the anode, which is typically at 2150 V, through a 1000 pF capacitor and out through a BNC connector welded in the flange on which the detector is mounted. The detached electron signal is then typically fed directly into a fast preamplifier (EG&G Ortec, Model 9301). All circuit components (resistor network, zener diode) are located in a small aluminum box mounted on the exterior of a six-conductor feedthrough welded in the flange directly above the detector. This allows easy access should a component fail and avoids component out-gassing concerns which are problematic in the high vacuum environment.

As briefly mentioned above, during total photodissociation cross section scans, the detached electron signal is collected for the purpose of normalizing the observed photofragment signal. In the limit where saturation of dissociative radical transitions is

---

Figure 2.17 The circuit diagram for the detached electron detector.

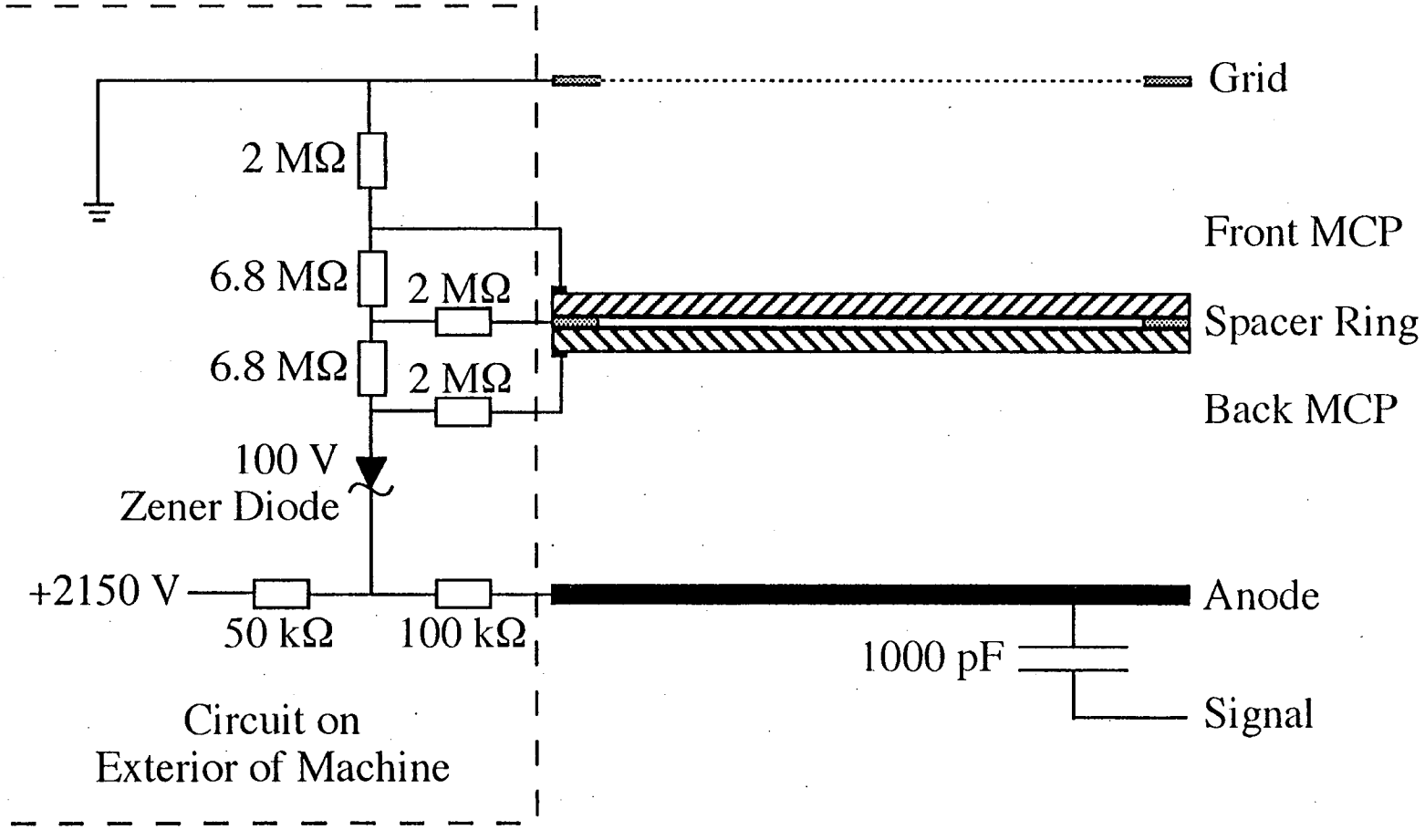

Grid
2 MΩ
Front MCP
6.8 MΩ
2 MΩ
Spacer Ring
6.8 MΩ
2 MΩ
Back MCP
100 V
Zener Diode
+2150 V
50 kΩ
100 kΩ
Anode
1000 pF
Signal
Circuit on
Exterior of Machine

Figure 2.17

not important, for a given photon energy the number of photofragments should be proportional to the relative number of radicals present in the photodissociation volume. This, in turn, should be proportional to the number of detached electrons.

In contrast, time- and position-sensitive experiments do not need this type of normalization. The detached electron signal is still useful, however, as a gauge of photodetachment efficiency. Any fall-off indicates problems with negative ion production, detachment laser power or detachment laser timing.

2.6.4 Ion Detection

It is of interest to be able to detect the negative ions in the beam for a number of reasons. Most importantly, we can carefully monitor the source performance while collecting data,[28] making adjustments if necessary to maintain a robust ion beam. Additionally, we can often get a very good idea of the ion depletion due to photodetachment, which at times can be >75%, by detuning the detachment laser in time and observing the increased intensity of the ion being detached. Lastly, the time-of-flight mass spectrum that is readily obtained with this detector is invaluable during the initial process of investigating the production of a new ion for the study of the corresponding radical.

The ion detector is mounted on a linear motion feedthrough manipulator (Perkin-Elmer Vacuum Products, Cat. No. 281-6200). Using this manipulator it can easily be raised completely out of the beam or lowered so that it is centered on the beam axis. As illustrated in Figure 2.15 the raised position is typically used since this allows any radicals made by photodetachment to pass; this configuration is necessary during a photodissociation experiment. If the ion detector is in the raised position ion detection is accomplished by deflecting all undetached ions remaining in the beam up out of the beam axis and onto the face of the ion detector. The ion deflection plates are 'hidden' from the photodetachment region so that the static positive potential present on the upper plate

does not perturb the trajectory of the ions prior to photodetachment. A steel ring welded onto the inner surface near the entrance of the tee housing the ion detector has been machined to be concentric with the beam axis. On this ring is mounted a 3" wide thin steel plate, keyed into a groove to insure concentricity for the ~3 mm (0.125") aperture at its center. To minimize the thickness of the hole, the side away from the source is countersunk and expands out to 1 cm. On this same side, two deflection plates are mounted, spaced 1 cm apart. The steel deflection plate mounted below the aperture is fastened directly on the shielding plate, thereby remaining at ground potential at all times. The upper deflection plate is mounted on Vespel®, a machinable plastic with good high vacuum out-gassing properties, which is then fastened to the shielding plate. This deflection plate is then kept at a potential in the vicinity of 1500 V, with the proper value depending primarily on the ion beam energy and the exact vertical position of the detector. Of course, if the detector is in the beam axis, or it is desirable for the ions to be allowed to proceed to subsequent regions of the machine, the deflection plate is grounded.

As Figure 2.18 shows, the components and circuitry for the ion detector are quite similar to that of the detached electron detector. However, it is not necessary to accelerate the ions towards the microchannel plates since they have a kinetic energy determined by the ion beam energy. Therefore the acceleration grid used in the detached electron detector is not present in the ion detector.

The circuit is also has some minor differences. The first relates to the fact that the acceleration grid is not present. Because of this, the total voltage applied to the detector has to be only 1850 V. A second difference is that this detector must be able to translate freely upwards and downwards. It was decided that for this reason all of the voltage dividing resistors should be within the machine, to avoid many long connections that might foul and short circuit upon translation. To minimize outgassing problems, special glass

Figure 2.18 The circuit diagram for the ion detector.

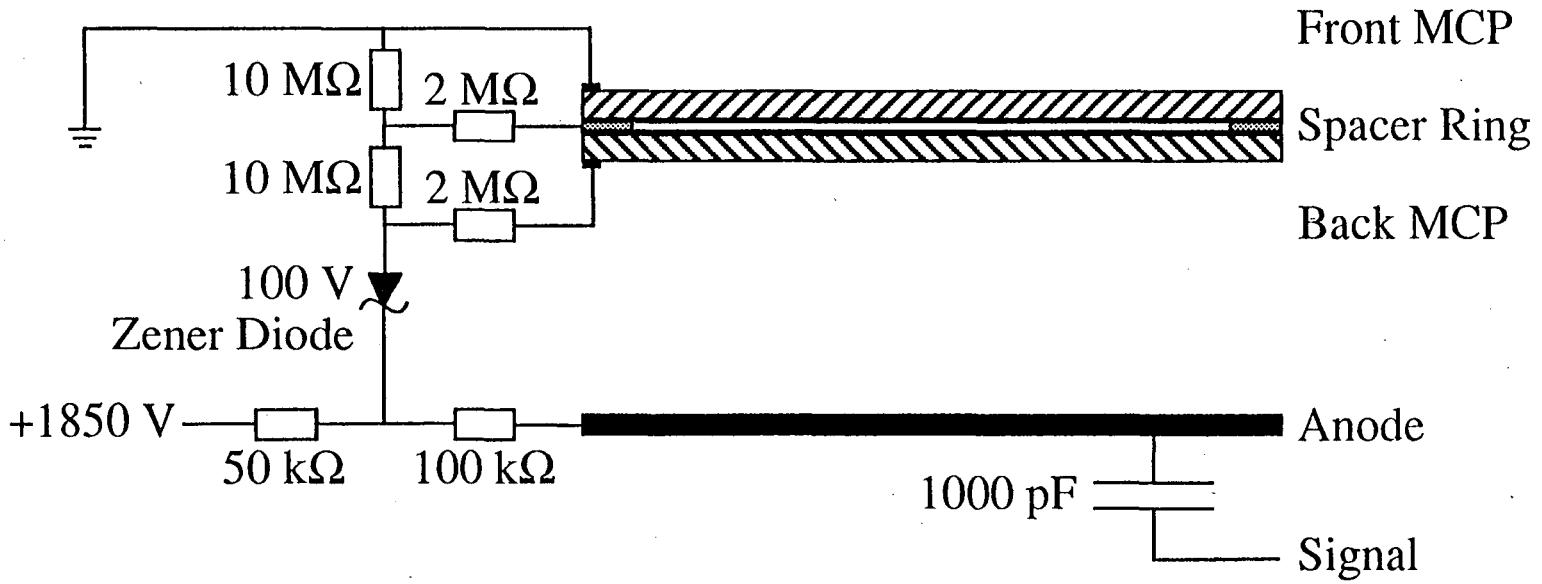

Front MCP
10 MΩ
2 MΩ
Spacer Ring
10 MΩ
2 MΩ
Back MCP
100 V
Zener Diode
+1850 V
50 kΩ
100 kΩ
Anode
1000 pF
Signal

Figure 2.18

resistors (K & M Electronics) were used.

2.6.4.1 Ion Time-of-Flight Mass Spectroscopy: Monitoring Source Output

The ion detector is used for investigating the possibility of making new (for this apparatus) negative ion species. The key to determining what negative ions are being produced is the following time-of-flight to mass conversion:

$$m_i = m_0 + ct_i^2. \tag{2.7}$$

By determining the time-of-flight t_i for two ions of known mass m_i, one can determine the constants m_0 and c using elementary algebra. The constant m_0 should be close to zero if the times-of-flight are measured relative to the mass spectrometer trigger pulse, while the constant c will vary with beam energy. For a variety of reasons it is best to have the detector lowered to intersect the beam axis for these measurements. In addition, the ion compressor should be turned off and the mass spectrometer voltage raised to narrow the time-of-flight mass peaks.

While observation of the ion times-of-flight on the oscilloscope can be informative, it is also possible to record mass spectra for posterity by feeding the ion detector signal along with an appropriately timed trigger pulse into a transient recorder (DSP, Model 2001AS). The output of this module is fed into an averaging memory module (DSP, Model 4101), and the summed and averaged signal can be read out to the PC/AT laboratory computer through an interface to the CAMAC crate controller (DSP, Model 6001). Time-of-flight mass spectrometer data acquisition is a main menu option in the FRBM program (found in Appendix I).

2.7 Photodissociation and Detection Region

Within this final region, photodissociation of radicals is undertaken followed by detection of the photofragments produced in this process. This region is illustrated in Figure 2.19.

2.7.1 Photodissociation

We customarily frequency-double the output of a second excimer-pumped dye laser system (Lambda Physik LPX-210i and Lambda Physik FL-3002, respectively) to achieve the photon energies necessary for our radical dissociation studies. We use a frequency doubling unit (Interactive Radiation, Inc. Model AT-II), equipped with either BBO and KDP crystals (as appropriate), to allow the production of doubled light from as low as ~206 nm up to the lowest wavelength where the fundamental of the dye laser may be used (~340 nm). Frequency doubling is a nonlinear optical process dependent upon the second order polarizability of the doubling medium, and will not be discussed further here.[29] The fundamental can be separated from the frequency doubled output, if desired, by the use of an in-line four-prism assembly which spatially separates the two colors and transmits only the frequency doubled light. In addition, the frequency-doubled output is highly polarized in the horizontal plane. This orientation can be preserved, resulting in polarization of the $\vec{E}$ vector of the laser light parallel to the radical beam axis. Alternately, a periscope arrangement of prisms may be used to rotate the polarization to the vertical orientation, perpendicular to the radical beam axis. This ability is useful in cases where a very thorough investigation of the recoil angle distribution is desired.

2.7.2 Acquiring Photodissociation Laser Timing

A previously mentioned, the timing of the photodetachment laser can be optimized by inspection of photoelectron signal, or even depletion of ion signal. However, due to the relative paucity of radicals as compared with ions, and the therefore even smaller fragment signal levels, finding the proper timing of the dissociation laser system could be

---

Figure 2.19 A diagram of the dissociation and detection region. (1) Initial 1 mm aperture. (2) Pulsed deflection plates. (3) Photon baffle. (4) Total photodissociation cross section detector, shown raised out of the beam axis. (5) Time- and position-sensitive detector.

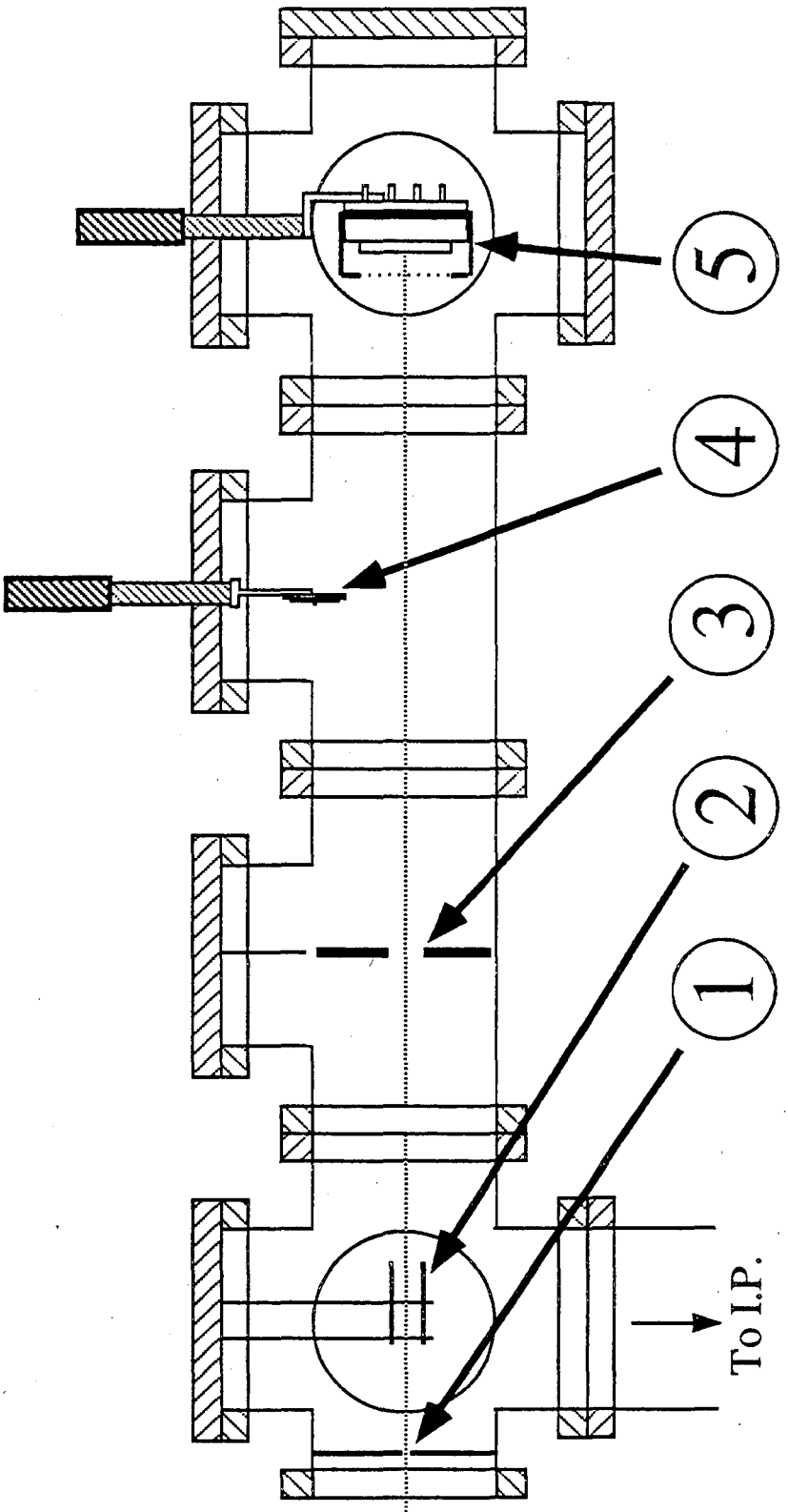


Figure 2.19

extremely difficult. Timing can be calculated to perhaps better than 0.5 μs, but the uncertainty in the size of the delay between the excimer laser trigger pulse and the actual firing of the laser is fairly large, as well as being dependent on the operating temperature and discharge voltage of the laser. The required accuracy of timing is on the order of 20 ns, as this is roughly the duration of the laser pulse. The solution to this difficult problem lies in the moderately clever use of a set of deflection plates suspended from the flange above the dissociation volume [see label (2) in Figure 2.19].

The proper dissociation laser timing can be determined by initially allowing ions to propagate through the dissociation volume and impinge on the face of the fragment microchannel plate detector, which has been positioned just off-center to allow the ions to pass by the blocking strip. Of course, this implies that the detachment laser is not in operation, the ion deflector is grounded and the ion detector is grounded and up out of the beam axis. After establishing a respectable signal of ions on the fragment detector, the time of flight of the mass of interest is noted. A pulsed field is then applied to one of the deflection plates straddling the photodissociation volume just early enough to deflect all the ions so that they miss the detector. These plates are extended towards the photofragment detectors by ~5 cm, primarily to increase the amount of time an applied potential can act on the ions, thereby reducing the required strength of the field. However, so that the deflector plates do not perturb the ion beam prior to reaching the photodissociation volume, it is essential that the potential is pulsed on only after the ions have passed this volume. By applying a relatively large potential ions can still be easily made to miss the detector. Invariably the wavelength that is to be used to photodissociate the radicals will be sufficient to detach the ions. Therefore, by not using the detachment laser system and by allowing the ions to continue along undeflected in to the dissociation region, one will be able to detach the ions using the dissociation laser system *once the laser timing is correct.* The resulting neutrals will not be deflected by the pulsed electric

field and a clear signal will be present on the neutral detector at the time-of-flight of the corresponding ion.

The usefulness of the preceding method is that it is easier to see the relatively plentiful radicals formed by detaching a large number of negative ions, than the much fewer number of photofragments formed by the dissociation of radicals previously formed by photodetachment. This is particularly true if it is not yet known at which wavelengths the radical in question will dissociate, a scenario quite common with predissociating systems. In this case wavelength scanning concurrent with the laser timing search would be necessary. This method does have the requirement that the electron affinity of the radical must be less than the relevant bond dissociation energy. This situation has always been the case in the experiments thus far and is generally true.

2.7.3 Monitoring Photodissociation Laser Power

In order to have the capability to normalize to the intensity of photodissociation laser during total photodissociation cross section measurements, a pyroelectric detector is used to measure laser power. The pyroelectric detector (Molectron, Model P1-61) circuit is shown in Figure 2.20. The active area of this detector is quite small. For this reason, care must be taken that the laser beam does not 'walk' during a laser scan, which would change the relative amount of the beam that falls on the face of the detector, and thus the apparent intensity. To minimize this effect, as well as to attenuate the high laser intensity, ground glass diffusers are mounted in front of the pyroelectric detector.

2.7.4 Laser Light Baffles

Scattered photons must be controlled in our experiment, both in the photodetachment region and in the photodissociation region. The detached electron

---

Figure 2.20 A schematic of the Molectron pyroelectric detector circuit.

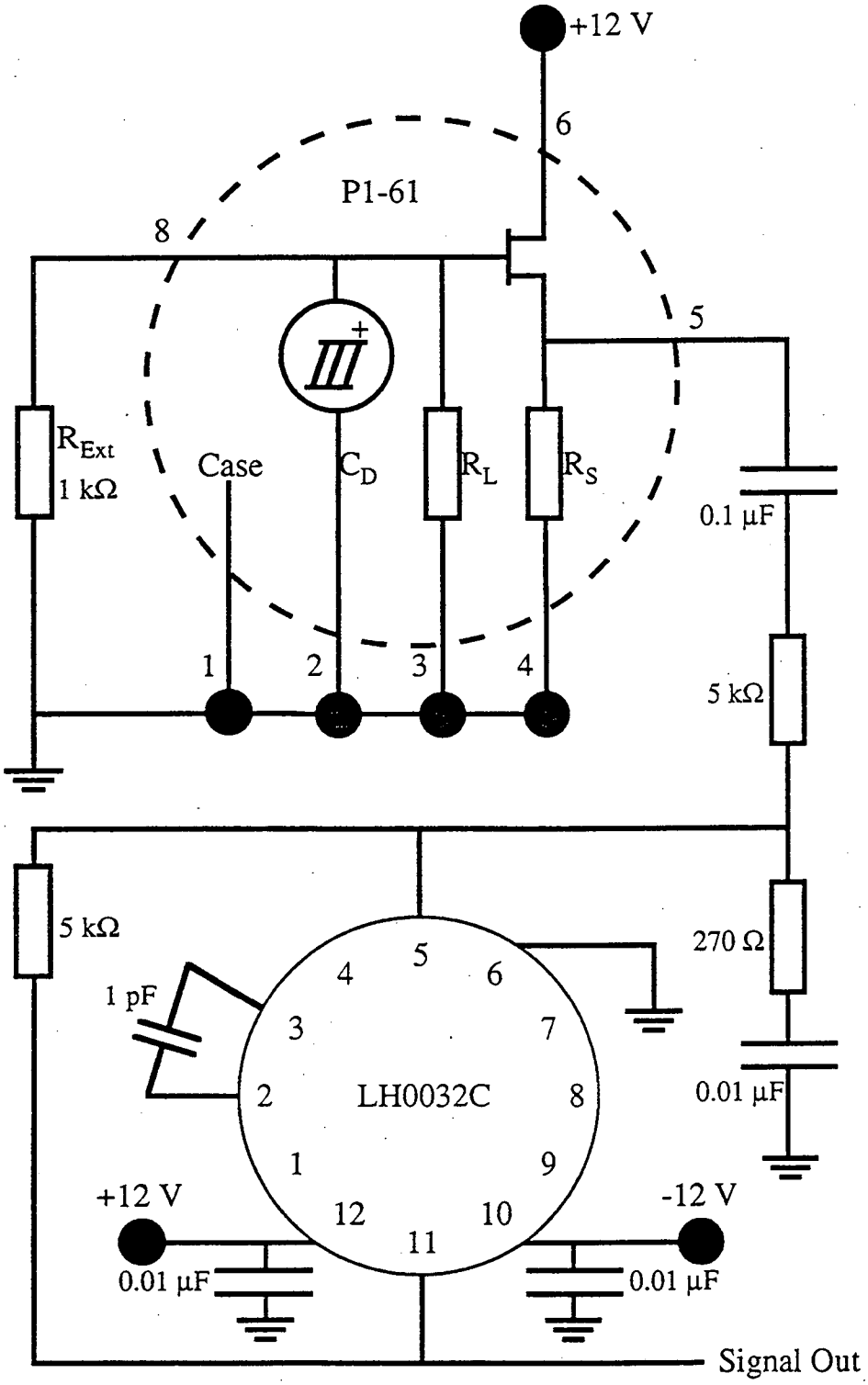

+12 V
6
P1-61
8
5
R_{Ext}
1 kΩ
Case
C_D
R_L
R_S
0.1 µF
1
2
3
4
5 kΩ
5 kΩ
270 Ω
1 pF
LH0032C
0.01 µF
+12 V
-12 V
0.01 µF
0.01 µF
Signal Out

Figure 2.20

detector can register photons as well as photoelectrons from scattered light hitting the inner surfaces of the machine; both will interfere with the accurate appraisal of detached electron signal. In the photodissociation region stray photons are a particular problem when we acquire time- and position-sensitive data. Photons, which reach the detector a few microseconds prior to the arrival of fragments, can cause the detector electronics to be unable to recover to baseline in time to register the fragments properly.

To address these concerns, we have installed anodized aluminum light baffles on the input and output flanges of both the detachment and dissociation regions. In addition, approximately one third of the way to the time- and position-sensitive detector we have installed an anodized aluminum disk that is very nearly flush with the inner walls of the flight tube [see label (3) in Figure 2.19]. This disk has a hole in the center which subtends a slightly greater solid angle than the detector face, preventing photons from reaching the detector by indirect routes (i.e., by bouncing off the wall). These precautions are quite successful, eliminating up to 90% of spurious signal caused by scattered photons.

2.7.5 Total Photodissociation Cross Section Detection

In order to measure the photofragment yield as a function of photon energy, giving the total (integrated over all product states) photodissociation cross section, fragments are detected using a MCP detector as the photodissociation laser wavelength is varied. This detector is located 0.67 m from the photodissociation volume and has an active area that is 40 mm in diameter. The detector assembly has a 3 mm wide blocking strip that is mounted in front of the active area and extends completely across the center of the face. This blocking strip insures that undissociated radicals will not strike the detector. The same type of linear motion feedthrough manipulator is used to mount this detector assembly as is used to mount the ion detector assembly. Prior to operation, the detector is lowered down from a fully retracted position (as it is shown in Figure 2.19) until the 3 mm blocking strip is centered on the beam axis.

The detector construction is identical to that of the ion detector, described previously, save the presence of the 3 mm blocking strip. In contrast to the time-and position-sensitive detector to be described below, these detectors are characterized by their contiguous anode and thus uncomplicated output. When struck by a fragment with high laboratory-frame kinetic energy anywhere on the active area of the microchannel plate face, the detection efficiency is roughly 50 %.[30] The resulting cloud of electrons produced by the microchannel plates is collected on the anode, and a fast-rising, 5 ns wide, 10-40 mV negative polarity peak is produced. This pulse is AC coupled, using a 1000 pF capacitor, from the anode which is at 1850 V. It is then transmitted using a shielded cable through a BNC connector welded in the flange. The signal is then fed directly into an fast preamplifier with a factor of 10 gain (EG&G Ortec, Model 9301), and from there sent to the gated input of a charge sensitive analog-to-digital converter (ADC, LeCroy, 2249SG). The ADC also receives input from the detached electron detector and the pyroelectric detector (Molectron P1-61, discussed in § 2.7.3) whose output is proportional to the photodissociation laser power. These measurements can be used to normalize the photofragment signal for variations in these quantities. Lastly, there are provisions for the ADC to collect signal from an iodine absorption cell to be used in subsequent wavelength calibration.

When a total (photodissociation) cross section (TXS) scan is taken, a wavelength range and step size is chosen that will result in usually no more than 200 wavelength steps at 100 shots per point. At 60 Hz, this results in a 6 minute scan. The reason for this limited scan length is to guard against the effects of possible instabilities in the source. Counting statistics are improved by repeating these scans until satisfactory signal-to-noise is achieved. In practice, this can mean up to one thousand or more shots for each point. If the ions are unstable, a greater number of quicker scans can be taken by reducing one of the two above mentioned parameters.

Total cross section data acquisition is controlled by the program FRBM, which can be found in Appendix A.

2.7.6 Time- and Position-Sensitive Detector

Detailed dynamics of dissociation events undertaken at a specific dissociation photon energy are obtained using a time- and position-sensitive detector. The detector allows the simultaneous two-dimensional position detection of both fragments from a given photodissociation event, with excellent timing resolution (typically < 0.5 ns) in the relative time-of-flight of the two fragments. This in turn provides a direct measurement of the mass ratio, recoil angle and kinetic energy release for each dissociation event.

Time- and position-sensitive detectors have previously found use in a variety of experiments involving the dissociation of metastable[31] or photoexcited species.[32] Typically, however, the detector design incorporated one-dimensional (radial) position sensing. Two advantages are gained by determining the two-dimensional positions of each fragment rather than merely the radial displacements. First, it is necessary to measure fragment positions in two dimensions to obtain recoil angular distributions in experiments where the electric field vector of the dissociation laser is perpendicular to the beam axis, since the photofragments will now longer have azimuthal symmetry about the beam axis. A second advantage is realized in discrimination against false coincidences (the detection of fragments from uncorrelated events), since

$$\frac{x_2}{x_1} = \frac{y_2}{y_1} \tag{2.8}$$

must hold true due to conservation of momentum if the two photofragments originated from the same photodissociation event. Detectors with two-dimensional position sensitivity have been developed for a variety of electron spectroscopies[33] and recently, for charge transfer studies of high energy ion beams at surfaces.[34]

Because both fragments from a dissociation event are detected in coincidence, important advantages are gained over single particle detection. Single particle detection can be used to yield significant amounts of information,[35] however, the coincidence technique allows the unambiguous determination of the masses of the fragments and the center-of-mass dynamical quantities for each dissociation event. Resolution is also enhanced by a coincidence measurement, as will be discussed in § 2.7.6.5.

Position sensing is accomplished using the method of charge division, mounting two anodes with three conductors each, whose relative areas vary with position, behind a single set of microchannel plates. Other methods were considered but determined to be not ideal for our purpose. Resistive anodes and CCD-based imaging techniques have slower time responses, and are thus unable to determine the relative time-of-flight of the two photofragments to suitable precision. Multi-anode array detection with multi-event capabilities would be ideal, but extremely complex and exorbitant.

The following subsections deal with various aspects of the time- and position-sensitive detector in some detail.

2.7.6.1 Detector Body

Following the design used by the UC Berkeley Space Sciences Laboratory, we have assembled the time- and position-sensitive detector as shown in Figure 2.21. The

---

Figure 2.21 A schematic of the TPS detector assembly. (1) Kovar/alumina brazed body assembly. (2) 8 mm wide blocking strip. (3) Spring-loaded ring assembly for holding microchannel plate Z-stack in place. (4) Z-stack of image-quality microchannel plates. (5) Electron drift region with resistive coating on the inner wall. (6) Fused silica anode substrate with photoetched copper anode. (7) Feedthroughs for output signals. For the detection of neutral particles, the front end of the detector assembly is held at –4.15 kV, giving microchannel plate voltages of ≈ 1300 V/plate, and a drift region voltage of 230 V over 8 mm.

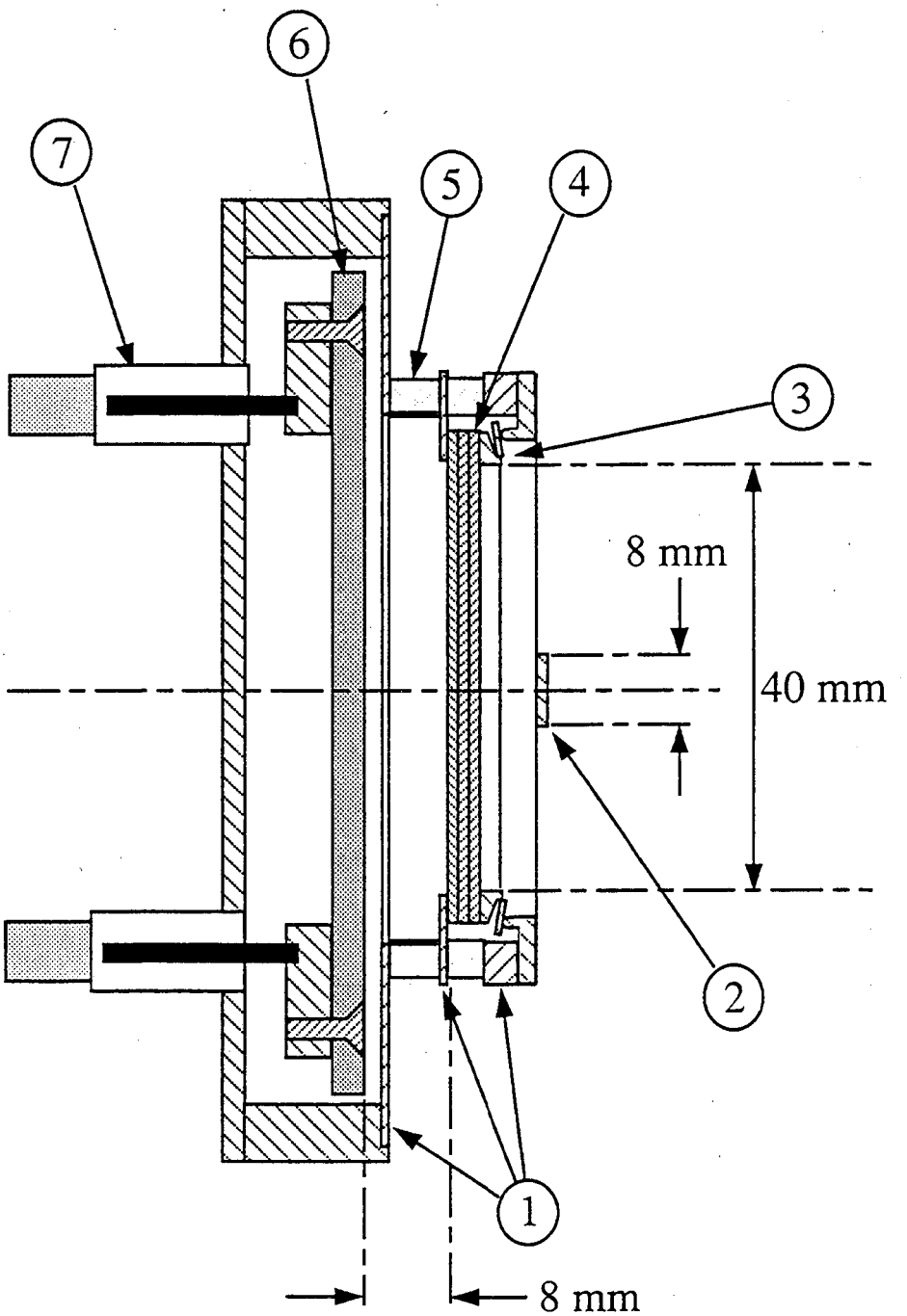


Figure 2.21

detector body consists of a microchannel plate holder, a resistively coated ceramic (alumina) drift region and an anode mount. The microchannel plate holding section is constructed of Kovar rings brazed on both ends of a ceramic cylinder whose ends have been nickel coated. Both rings have tabs so that leads for the high voltage potentials may be spot-welded on, and the thin inner ring extends inward to form a shelf for the microchannel plates to sit on. The ceramic cylinder provides excellent insulation, isolating the MCPs. The thicker outer ring has threaded holes in it to allow a retaining ring to be screwed in, tightening down on an annular spring which in turn holds the microchannel plates down. The 8 mm wide blocking strip is also keyed into the retaining ring.

The microchannel plates that are used in this detector are a resistance-matched triplet set of soft edged, double thickness, Z-stacked microchannel plates (Philips, G12/46-DT 13-13-13). They have a 12 micron channel diameter, a 13 degree bias angle and a 46 mm plate diameter.

Immediately below the microchannel plate holder is the drift region, an 8 mm gap between the anode and the back of the MCPs. In order to control the amount of lateral spreading of the electron cloud, and thus the anode area that the electron cloud covers, the potential applied across this region can be varied using the voltage dividing scheme shown in Figure 2.22. The walls of this drift region are made of resistively-coated ceramic (MacroMetallics) so that the field lines will vary smoothly from the top to the bottom of the drift region. The total resistance from top to bottom is ~1 GΩ.

The anode is mounted on seven pads, one for each of the three conductors plus a ground connection. Flat head machine screws are countersunk into the copper-covered fused silica anode, and provide a connection to the pads. These pads are in turn welded to pins that lead to SMA connectors, which extend out the back of the housing. The

Figure 2.22 The voltage divider used to establish the potential across the MCPs and the drift region within the time- and position-sensitive detector.

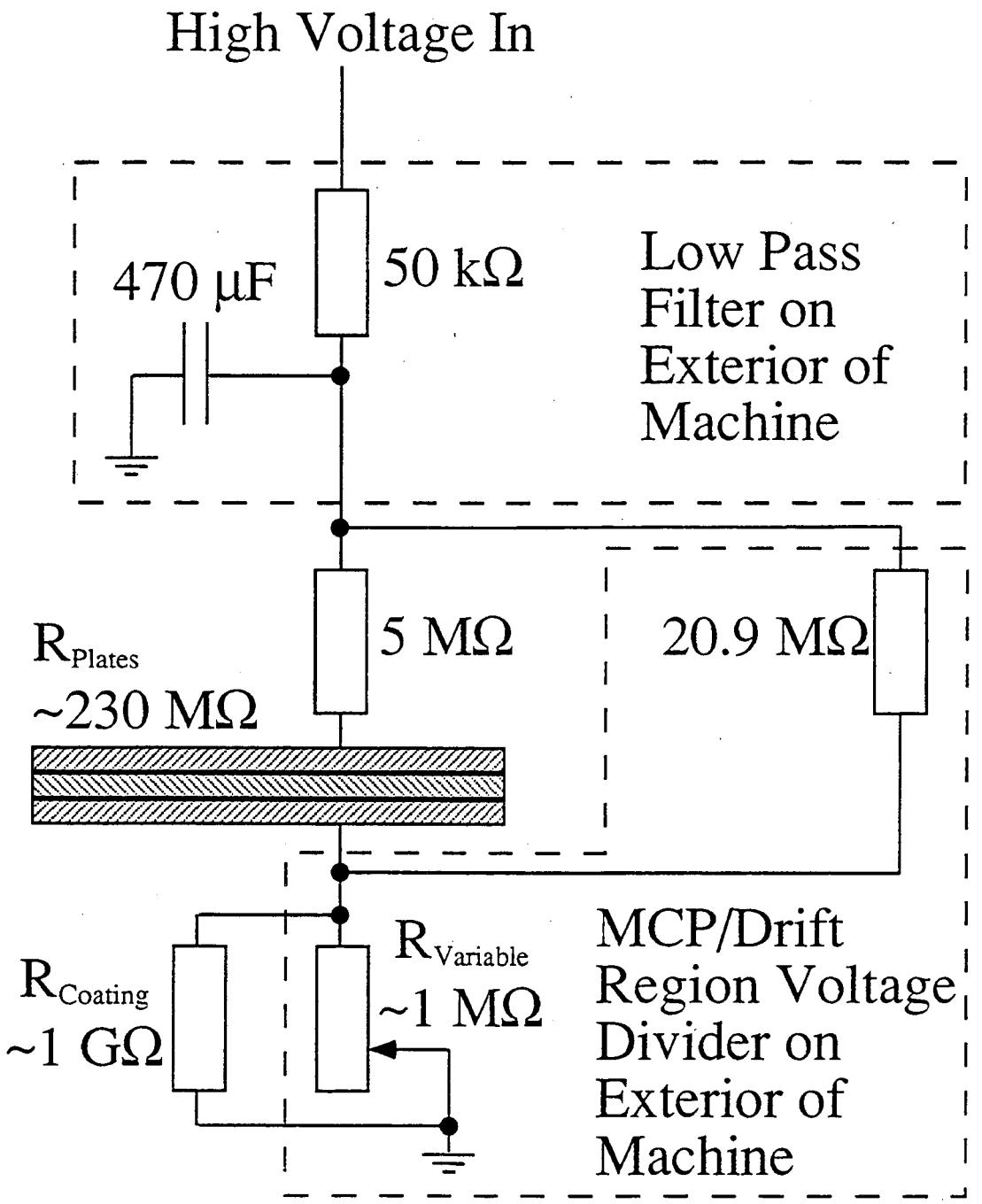

High Voltage In
470 μF
50 kΩ
Low Pass
Filter on
Exterior of
Machine
5 MΩ
20.9 MΩ
R Plates
~230 MΩ
R Variable
~1 MΩ
R Coating
~1 GΩ
MCP/Drift
Region Voltage
Divider on
Exterior of
Machine

Figure 2.22

connectors couple the signal to lengths of RG-316/U Teflon-dielectric coaxial cable (chosen chiefly for it's low capacitance) which in turn are connected to BNC feedthroughs on the vacuum chamber mounting flange.

Mounted on the anode assembly and surrounding all of the other sections, save the face of the MCPs themselves, is a magnetic shield made from annealed stainless steel 416, which guards against the influence of magnetic fields on the electrons produced by the MCPs. Clamped to this magnetic shield is an assembly that supports a 90 % transmittance screen in front of the detector, to reduce the background caused by stray positive ions. In contrast to the other microchannel plate detectors used in this experiment, the anode is held at ground for ease of coupling the signal out to the preamplifiers, while the front of the MCPs are at ~ –4.15 kV. Without this shield, positive ions formed in the ion pumps or the detector region ion gauge will be attracted by this large negative potential and will contribute to spurious background signal. Because the experiment is pulsed, and the detection electronics are gated on for only a few μs per laser shot, these background counts would rarely register. Even when they do register, they will be disregarded since they will not meet our coincidence discrimination criteria. If they are too plentiful, however, they will begin to disrupt detection of valid events by causing pile-up. For this reason, we accept a slight diminution of signal which accompanies the installation of the shielding mesh. Additionally, we often turn off the detector region ion gauge during an experiment. These measures cut down drastically on positive ion background.

2.7.6.2 Wedge-and-Strip Anode

Our detector possesses position sensitivity because of a specially designed anode placed behind our stack of microchannel plates. This anode is flat, and is comprised of two separate semicircular sections, each of which has a wedge-and-strip pattern of conductors. A simplified schematic of this pattern is shown in Figure 2.23. It is enlarged and, for the sake of clarity, involves five periods per half rather than 32, which is the

number typically used in the anodes installed and used in the detector. It is comprised of interlocking wedges, which taper along their length in the x coordinate, and strips, which vary in width linearly with the y coordinate. A third conductor zigzags between the wedges and strips and fills the remaining area of the anode pattern. Charge division between the three conductors will depend on the area of each conductor, which varies with position. The position of a fragment in the x and y directions is nominally given by the following relations

$$x_i \propto \frac{Q_{wi}}{Q_{wi}+Q_{si}+Q_{zi}} \qquad \text{and} \qquad y_i \propto \frac{Q_{si}}{Q_{wi}+Q_{si}+Q_{zi}}. \quad (2.9)$$

The pattern which has been used most extensively has a period of 0.8 mm, with minimum and maximum wedge fractions of 0.04 and 0.40 and minimum and maximum strip fractions of 0.06 and 0.40. Insulating borders 40 μm thick separate each conductor.

The substrate on which the anode pattern is fabricated is a custom made fused silica disk (S & S Optical), 2.9 inches in diameter and 3 mm thick. Fused silica is chosen for its low dielectric constant, and consequent small capacitance. Countersunk holes are machined near the outer diameter to allow mounting of the anode and electrical connections to the conductors via the flathead screws which hold the anode in place. This disk then has a 6 μm thick layer of copper vapor-deposited onto its surface (Scientific Coatings). The wedge-and-strip pattern is photoetched onto this copper surface.

The pattern is constructed using a modified version of computer code obtained from the UC Berkeley Space Sciences Laboratory. Gerber (plotter format) output from

---

Figure 2.23 A wedge-and-strip anode pattern, enlarged and simplified for clarity. The wedges are hatched, the strips have a vertical striped pattern, while the solid line bordering conductors represents a thin insulating gap. W1, S1, Z1 and W2, S2, Z2 measure the charges for the upper and lower halves of the detector, respectively. The actual patterns typically have 32 periods per side, rather than five as shown here, and a diameter of 54 mm.

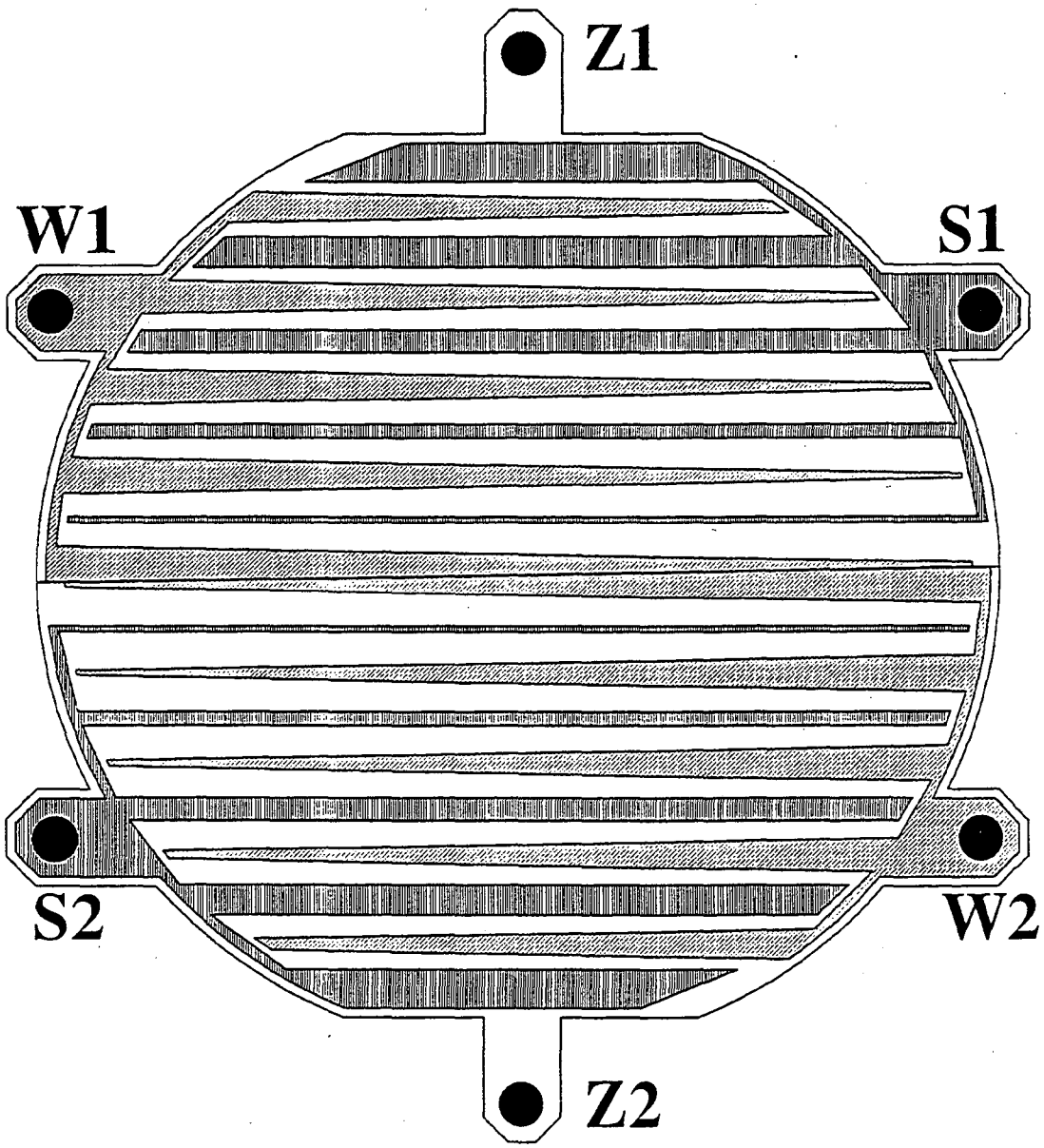


Figure 2.23

this program is transmitted to a facility (Image Technology Inc.) where a full-scale photomask is made that is seven times larger than the eventual photomask used in the fabrication. Photoreduction of the image is then carried out, with both positives and negatives made of the actual size photomask.

At Space Sciences Laboratory, the wedge-and-strip pattern is then photoetched into the vapor deposited copper using a large mercury lamp to illuminate the treated copper surface through the photomask. Copper is completely and selectively removed where an insulating region is desired, forming conductors separated from each other by an insulating strip of bare fused silica.

2.7.6.3 Detection Electronics

To process the charge fraction and timing information, we must accomplish two separate tasks: determine the charge fractions on each of the three conductors for each half of the anode, and determine the difference in time-of-flight of the two photofragments. A schematic of the data acquisition system that is used for this purpose is given in Figure 2.24.

To accurately determine the charge fractions, the signal from each conductor is fed directly into a charge sensitive preamplifier (Canberra, Model 2004). We have made modifications to the input stages of these units for improved performance in their role in this experiment. Specifically, the input of these preamplifiers is manufactured with the ability to float many kilovolts above the rest of the circuit. This ability is not only unnecessary, certain aspects of the input circuitry were found to be detrimental to our application. Figure 2.25 shows a circuit diagram of the input stage prior to any modifications, followed by the circuit diagram after modifications. Since we do not float our detector, the high voltage input connector is fitted with a grounding plug. The removal of the 100 MΩ resistor was necessary to allow adequate draining of the accumulated charge on the anode conductors to ground between events, particularly at

high count rates. The sole purpose of the 1000 pF capacitor to ground originally located between the two resistors on the HV input line was to condition the high voltage power by serving as a low pass filter. It is therefore removed. The last change that was made involves the capacitor that couples charge from the conductor to the charge sensitive amplifier circuit. The value of this capacitor was raised from 1000 pF to 0.01 μF. The major effects of these modifications are a decrease in inter-conductor cross-talk, thought to occur because the capacitive charge division that occurs between the preamplifier input capacitor on one hand and the neighboring conductors on the other hand is altered to increase charge flowing to the preamplifier, and decrease charge coupling to neighboring conductors. A slight gain in signal is observed because the increased capacitor value allows a lower frequency cut-off than was the case previously.

The output of the charge sensitive preamps is a fast-rising pulse with a relatively long fall-time; the amplitude of the pulse is proportional to the amount of charge detected by the preamplifier. The preamplifier output is sent to a spectroscopy amplifier (EG&G Ortec, Model 575A) which forms an amplified, shaped peak whose amplitude is

---

Figure 2.24 Schematic of the data acquisition system. W1,S1,Z1 and W2,S2,Z2, charge signals from the three conductors on both halves of the wedge-and-strip anode; QA, Canberra 2004 charge sensitive preamplifiers; FA, ComLinear CLC 100 fast timing amplifiers, AC coupled to the strip conductor from each half of the anode to provide fast timing pulses; SA, Ortec 575A spectroscopy amplifiers; CFD, Tennelec TC455 constant fraction discriminator; DDG, Precision Instruments 9650 digital delay generator; TAC, Canberra 2145 TAC/SCA time-to-amplitude converter; ADC, Ortec 413A analog-to-digital converter; CAMAC, Kinetic Systems CAMAC crate interfaced with a DSP Model 6001 CAMAC crate controller to a 386-based PC/AT clone. Externally supplied gating allows the ADC's and the TAC to accept data only when photofragments are expected.

Figure 2.25 Schematic of the Canberra 2004 input stage before, and after modification to remove the components deemed unnecessary and detrimental to the present application. See text.

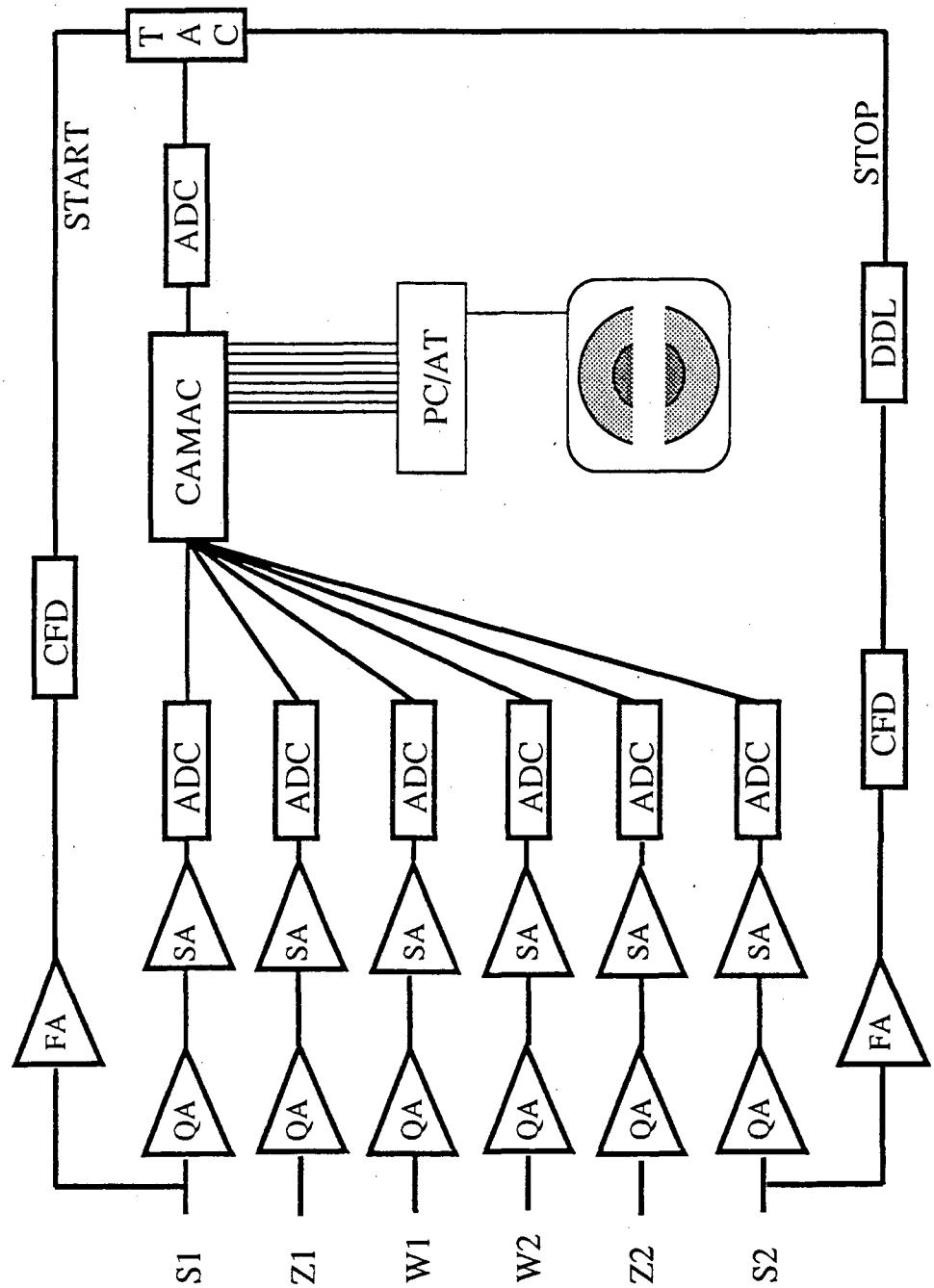

T A C
START
STOP
ADC
CAMAC
PC/AT
DDL
CFD
CFD
ADC
ADC
ADC
ADC
ADC
ADC
SA
SA
SA
SA
SA
SA
FA
FA
QA
QA
QA
QA
QA
QA
S1
Z1
W1
W2
Z2
S2

Figure 2.24

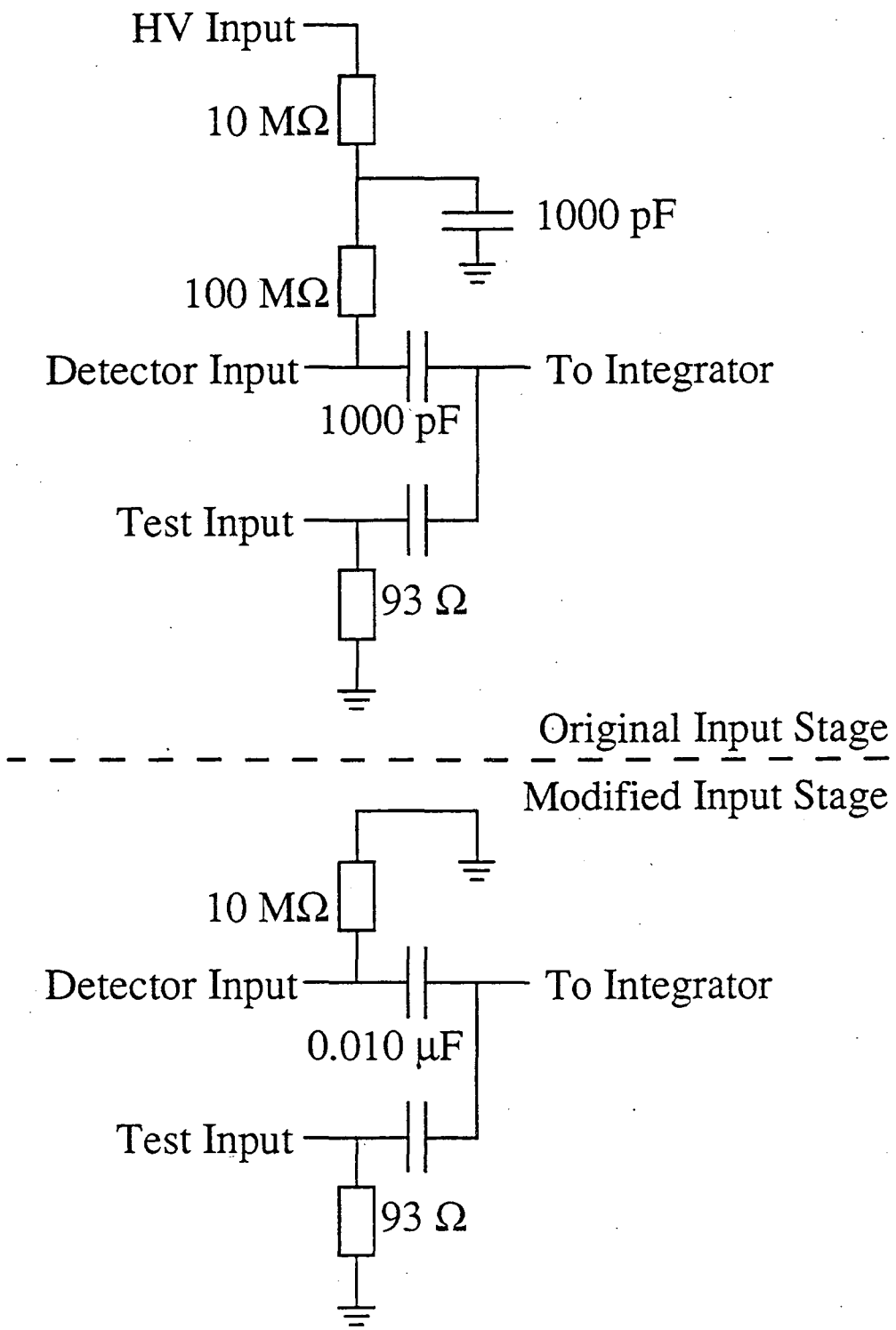

HV Input
10 MΩ
1000 pF
100 MΩ
Detector Input
To Integrator
1000 pF
Test Input
93 Ω
Original Input Stage
Modified Input Stage
10 MΩ
Detector Input
To Integrator
0.010 μF
Test Input
93 Ω

Figure 2.25

proportional to the amplitude of the rising edge of the preamplifier output. The shaping time constant can be 0.5, 1.5 or 3.0 μs, but is in our case dictated by two competing factors. A more precise output is obtained when the shaping time constant is longer. However, at the photodissociation photon energies we typically use, scattered photons are occasionally detected at the time- and position-sensitive detector. If scattered photons are being registered, it is best to use the smallest shaping time constant, thus returning the output of the spectroscopy amplifier to baseline promptly (the difference in time of flight for the photons and the fragments is only 4–8 μs).

The six spectroscopy amplifier outputs are fed into channels 1-3 of two 13-bit quad analog-to-digital converters (ADCs, EG&G Ortec, Model AD413). These are gated using a home-built gating circuit to accept signal only for a few microseconds during the period when photofragments are expected. The ADCs also have a hardware discriminator that can be set at any value up to 0.5 V, causing pulses of smaller amplitude to be rejected. This prevents line noise or spurious events from causing the ADC channels to be busy.

The difference in time-of-flight of the two fragments is measured by first splitting off, on each anode, a portion of the strip conductor signal through a 2 pF capacitor to the input of a 500 MHz video amplifier (ComLinear Corp., Model CLC100). These negative going signals, having been amplified by 10 times, are then routed into two channels of a quad constant fraction discriminator (CFD, Tennelec, Model TC455). The CFD is typically set up to run in the 50 % mode, with the fixed delay being provided by 0.5 ns cables plugged into the front panel. A threshold is set for each channel with the knowledge that the output of the CLC100's is biased at ~0.019 V. Thus a threshold setting of –5 mV actually cuts out all peaks less negative than ~ –25 mV. In the case of a real event, where both the upper and lower anode register signal, the discriminator then feeds the upper timing pulse to the START input of a time-to-amplitude converter (TAC, Canberra, Model 2145 TAC/SCA). The lower timing pulse is sent through a long delay cable or is used as the external trigger for a digital delay generator (DDG, Precision

Instruments, Model 9650) that is set to provide a delay of greater duration than the greatest difference in time-of-flight possible for the system being studied. This ensures that the lower timing pulse, after being subjected to this fixed delay, always follows the upper timing pulse and can be reliably used as a STOP pulse for the TAC. The TAC is gated on for only a few μs per laser shot using a gate delayed to the appropriate time by a gate generator (EG&G Ortec, Model 416). This gate generator is triggered by the synch-out pulse of the dissociation excimer laser.

The experiment is controlled by a PC/AT clone interfaced to the CAMAC crate where the ADC's are housed. The program used for data acquisition is called TPS (an acronym for Time- and Position-Sensitive), which can be found in Appendix B.

2.7.6.4 Calibration

Calibration of the time- and position sensitive detector involves both calibration of the electronics, so that a reliable correlation between signal level and charge fraction (or time) is established, and the calibration of coefficients within the algorithm that converts the measured charge fractions to positions.

The level of amplification of the charge collected on each conductor is determined by the individual amplification settings on the spectroscopy amplifiers. Making this amplification uniform amongst all of the conductors is then a matter of simulating the arrival of a specific amount of charge at each conductor by capacitively coupling the falling edge of the square wave output of a pulse generator (Hewlett-Packard, Model 8012B) through a 1 pF capacitor. This charge is injected just outside the feedthrough associated with each anode conductor in turn while the spectroscopy amplifiers for that conductor are adjusted to give an identical reading into the same ADC channel. Using the same ADC channel accounts for possible differences in ADC sensitivities (to be determined below). The zigzag conductors are adjusted to read half of the value returned by the wedge and strip conductors for the same test pulse. This convention is followed

because on average, the charge fraction on the zigzag conductor is twice as much as on either the wedge or the strip conductors due to the larger relative area of the zigzag conductor. The ADC value given for the zigzag conductors is multiplied by two immediately after being read by the computer during data collection.

Calibration of the ADC channels used in charge fraction measurement is accomplished by noting the ADC output for a range of spectroscopy amplifier output levels from 1 to 9 V, as measured using the oscilloscope. The dependence of ADC channel output on the height of the input pulse can then be determined using a linear least-squares fit. This information is stored and used to rescale the raw charge measurements during data analysis.

For timing measurement calibration the TAC output is fed into channel 4 of the ADC which measures the W1, S1 and Z1 output in channels 1–3. The ADC output is noted while injecting charge onto S1 and S2 at various relative delays. A TAC calibration curve can then be constructed and fit by a linear least-squares routine. When using the digital delay generator (DDG) to establish a variable delay, the intrinsic response time (approximately 40 ns) of the DDG[36] following external triggering must be added to the front read-out. The total delay includes the sum of contributions from this response time as well as the delay dialed into the delay generator. Alternately, a series of long BNC cables can be used to give a precise, if cumbersome delay line.

From this calibration curve, experimental TAC output is easily converted during data analysis to the apparent difference in time-of-flight for the two fragments (including known delay). Subtraction of the total known, fixed delay that is present to insure the upper fragment is always the START pulse and the lower fragment the STOP pulse can then be done. The signed difference in time of arrival, τ, will then be known to better than 0.02 % of full scale on the TAC. This translates to 0.2 ns on the 1 μs full scale TAC setting.

The relative amounts of charge on the three conductors of each anode are translated into positions in a slightly more complicated manner. The x and y positions on the detector face are roughly proportional to the fraction of total charge found on the wedge and strip conductors, respectively, as given in § 2.7.6.2. However, the full formulae for finding position from charge fractions includes empirical coefficients for these terms that are linear in charge fraction as well as other factors to correct various residual nonlinearities in the detector. The actual formulae used for the x and y coordinates of the upper (denoted "1") position are:

$$x_1 = x_0 + a_{w1} \cdot \left[f_{w1} + (ct_{wz1} - 1) \cdot f_{z1} + (q_{w1} - 1) \cdot f_{w1}^2 \right] - b_{w1} \qquad (2.10)$$

$$y_1 = y_0 + a_{s1} \cdot \left[f_{s1} + (ct_{sz1} - 1) \cdot f_{z1} + (q_{s1} - 1) \cdot f_{s1}^2 \right] - b_{s1}. \qquad (2.11)$$

The formulae for the position on the lower half of the detector are identical except for having "2" substituted everywhere for "1". The assumed position of the center of the radical beam axis on the detector face is denoted x_0, y_0. The fraction of the total charge found on the wedge conductor of the upper half of the detector is given by f_{w1}, i.e.,

$$f_{w1} = \frac{Q_{w1}}{Q_{w1} + Q_{s1} + Q_{z1}} = \frac{Q_{w1}}{Q_1}. \qquad (2.12)$$

A multiplicative and additive coefficient are given as a and b, respectively, with subscripts indicating whether they refer to the wedge or strip (i.e., a_{w1} or b_{s1}). Crosstalk between electrodes is possible given their extended shared borders. However, the wedge and the strip are always separated by the zigzag conductor. Since they are not adjacent at any point, crosstalk between them will be minimal and is not treated. The factors ct_{wz1} and ct_{sz1} represent an attempt to account for wedge-zigzag and strip-zigzag crosstalk, respectively. Lastly, a coefficient for a term quadratic in the total charge found on the wedge (for x coordinate) or strip (for y coordinate) is included to assist help in modeling residual nonlinearities in the detector position sensing. For example, the problem of 'edge

effect', where the apparent position of events near the outer edge of the detector face begins to curl inwards, cannot be accounted for satisfactorily in the absence of nonlinear terms. This problem is thought to be a consequence of a number of factors, mostly related to the finite extent of the anode pattern and the slightly inhomogeneous drift field potential near the edge of the detector body.

The coefficients needed in Eqns. (2.10) and (2.11) can be deduced by measuring the apparent position of UV photons from a mercury lamp which pass through a regular 4×2 mm rectangular grid pattern photoetched into a 0.002 inch thick nickel sheet placed on top of the first microchannel plate. The pattern consists of both 50 and 25 μm diameter holes, and a non-linear least squares fitting routine is used to determine the coefficients that result in the best fit between the observed pattern image and the knowledge of the true positions of the holes.

Analysis of the intensity profiles of individual imaged pinholes indicate that the spatial resolution is on the order of ~125 μm full-width-at-half-maximum in x and ~75 μm in y. The difference in x and y arises from the fact that both the wedges and strips have nearly exactly the same maximum and minimum charge fractions, but the dynamic range of the wedge pattern is spread over the entire diameter of the anode, while the dynamic range of the strip covers only a radius.

2.7.6.5 Translational Energy Resolution Considerations

Relatively high translational energy resolution can be achieved in our experiment ($\Delta E/E_T = 0.7\%$),[37,38] partially because of the coincidence scheme we employ. The coincidence requirement differs in principle from nearly all other translational photofragment spectroscopy experiments, where only one fragment is detected. In our experiment the distance between the two fragments at the detector face, and their difference in time-of-flight, are determined to a precision of ~140 μm and 0.5 ns, respectively. A simpler scheme, where the position of only one of the two fragments is

measured relative to the center of the detector, and the time-of-flight of this fragment determined relative to the dissociation laser pulse, is indeed possible. However, this method would be subject to uncertainty both in the distance measurement (due to the size of the radical beam at the detector, about 1 mm in our case), and in the flight time (the dissociation laser pulse is ~30 ns long), that would serve to decrease the energy (and recoil angle) resolution. The coincidence method is insensitive to these effects.

One factor which *could* affect the resolution of the coincidence method is the spread in the kinetic energy of the radical beam. However, $\Delta E/E_0$ is also quite small in our experiment, at 0.1%[38] it compares favorably to the typical (1-10%) energy spread of most neutral molecular beam photodissociation studies.

The only single fragment detection scheme which has better translational energy resolution is the hydrogen atom time-of-flight method developed by Welge and Ashfold,[39] which provides unparalleled energy resolution ($\Delta E/E_T = 0.3\%$). However, this technique, as it is now formulated, takes advantage of the relatively light mass and long-lived Rydberg states of hydrogen atoms, and is thus restricted to studying systems which lose a hydrogen atom upon photolysis. On the other hand, the ion-imaging technique developed by Chandler and Houston,[40] while providing quantum state selection of one of the photofragments, actually has fairly poor translational energy resolution (0.2-0.5 eV) for the reasons, given above, that our resolution would suffer if a coincidence measurement was not made.

As mentioned previously, no ionization is required for detection of the photofragments produced in our experiment. This avoids a final resolution consideration which adversely affects experimentalists utilizing an electron-impact ionization region to facilitate fragment detection. The ultimate resolution achieved in this manner is usually limited by the ratio $\Delta L/L$, where ΔL is the length of the ionization region and L is the flight length to the detector.

2.8 Experimental Timing Sequence

The experimental timing is accomplished using two digital delay generators (Stanford Research Systems, Model DDG-525). The timing sequence used for both total cross section and dissociation dynamics experiments are nearly identical. The internal clock of the first delay generator is used as the base for our experiments, and the repetition rate we most frequently chose is 60 Hz. The delay values given below are representative of those used for a molecule of mass ~42 amu, and beam energy of 8 keV — other conditions will, of course, cause these numbers to vary correspondingly.

- T_0 triggers the pulsed valve driving circuit
- A triggers the 'potential switch', typical delay : 310 μs from T_0.
- B triggers the mass spectrometer pulsing circuit, typical delay : 1.1 μs from A.
- C triggers the ion compressor pulsing circuit, typical delay : 260 ns from B.
- D triggers home-built gate generator circuits[41] which gate the detached electron, laser power, and iodine absorption cell (if used for wavelength calibration) signal collection. In addition, the square wave extraction pulse used in the detection of detached electrons is a product of a gate generator circuit.

The second SRS DG-525 is triggered using the $\overline{AB}$ output of the first, which has a rising edge at the B (mass spectrometer) delay. Its outputs are as follows:

- T_0' triggers the oscilloscope used to monitor the experiment
- A' triggers the detachment laser, typical delay 4.100 μs from B.
- B' triggers the dissociation laser, typical delay 7.300 μs from B.
- C' is used infrequently to trigger the pulsed deflection plate involved in the determination of photodissociation laser timing (see § 2.7.2). Otherwise it is unused.
- D' triggers the gate generator circuit which gates the detection electronics on when photofragment signal is expected; in this case a typical delay is 12.000 μs from B.

If the pulsed discharge source is used, delay A then controls the firing of the high power pulse generator that provides the discharge voltage/current. All other delays are then shifted down by one position until C' (which is typically not in use) is reached.

2.9 References

[1] *Ion Pump Operation*, Perkin-Elmer Vacuum Products, (Perkin-Elmer, Eden Prairie, MN, 1984).

[2] M. A. Johnson, M. L. Alexander, I. Hertel, and W. C. Lineberger, Chem. Phys. Lett. **12**, 285 (1984).

[3] M. A. Johnson and W. C. Lineberger, in *Techniques of Chemistry* **20**, 591 (1988), J. M. Farrar and W.H. Saunders, eds. (Wiley, New York, 1988)

[4] T. N. Kitsopolous, Ph. D. Thesis, UC Berkeley, 1991.

[5] See D. Miller, p. 14, *Atomic and Molecular Beam Methods, Vol. I*, G. Scoles, Ed., (Oxford University Press, New York, 1988).

[6] R. Proch and T. Trickl, Rev. Sci. Instrum. **60**, 713 (1989). See also reference 7.

[7] R. E. Continetti, Ph. D. Thesis (University of California, Berkeley, 1989).

[8] a) W. G. Cady, *Piezoelectricity*, (Dover, New York, 1964).
b) *Physik Instrumente Catalog*, Number 109, U.S. Edition.

[9] W. R. Gentry and C. F. Giese, Rev. Sci. Instrum. **49**, 595 (1978).

[10] A. Weaver, Ph. D. Thesis, UC Berkeley, 1991.

[11] M. A. Johnson, M. L. Alexander, I Hertel, and W. C. Lineberger, Chem. Phys. Lett. **12**, 285 (1984); M. A. Johnson and W. C. Lineberger, in *Techniques of Chemistry* **20**, 591 (1988), J. M. Farrar and W.H. Saunders, Eds. (Wiley, New York, 1988)

[12] See Reference 4 , pages 37-38.

[13] R. Schlachta, G. M. Lask, S. H. Tsay, and V. E. Bondybey, Chem. Phys. **155**, 267 (1991); A. Thoma, B. E. Wurfel, R. Schlachta, G. M. Lask, and V. E. Bondybey. J. Phys. Chem. **96**, 7231 (1992); B. E. Wurfel, A. Thoma, R. Schlachta, and V.E. Bondybey, Chem Phys. Lett. **190**, 119 (1992); G. Schallmoser, B. E. Wurfel, A. Thoma, N. Caspary, and V. E. Bondybey, Chem. Phys. Lett. **201**, 528 (1993).

[14] Y. Ohshima and Y. Endo, J. Mol. Spectro. **153**, 627 (1992); **159**, 458 (1993).

[15] S. Sharpe and P. Johnson, Chem. Phys. Lett. **107**, 35 (1984).

[16] S. K. Bramble and P. A. Hamilton, Chem. Phys. Lett. **170**, 107 (1990); Meas. Sci. Technol. **2**, 916 (1991)

[17] J. Baker, S. K. Bramble, and P. A. Hamilton, Chem. Phys. Lett. **213**, 297 (1993).

[18] T. N. Kitsopoulos, C. J. Chick, Y. Zhao, and D. M. Neumark, J. Chem. Phys. **95**, 5479 (1991); D. W. Arnold, S. E. Bradforth, T. N. Kitsopoulos, and D. M. Neumark, J. Chem. Phys. **95**, 8753 (1991); C. C. Arnold, Y. Zhao, T. N. Kitsopoulos, and D. M. Neumark, J. Chem. Phys. **97**, 6121 (1992).

[19] As this dissertation is written, additional configurations are also under investigation. D. L. Osborn and D. J. Leahy, unpublished work.

[20] Strictly speaking, einzel lenses are usually comprised of three cylindrical elements (as our einzel lens in the second differential region is constructed). An attempt to approach this configuration was made by decreasing the inner diameter and using 3 or 4 plates for each element in this arrangement.

[21] L. A. Posey, M. J. DeLuca, and M. A. Johnson, Chem. Phys. Lett. **131**, 170 (1986).

[22] This is particularly true in the case of the electron beam/free jet expansion source, where negative ions are made continuously for most of the of the ~200 μs expansion. When very short discharge times are used with the pulsed discharge source, the discharge duration can be less than the transit time of the cylinder for certain mass ions.

[23] R. E. Continetti, D. R. Cyr, and D. M. Neumark, Rev. Sci. Instrum. **63**, 1840 (1992).

[24] P. Horowitz and W. Hill, *The Art of Electronics*, 2nd ed. (Cambridge University, New York, 1989), p. 53.

[25] P. Horowitz and W. Hill, *The Art of Electronics*, 2nd ed. (Cambridge University, New York, 1989), p. 882.

[26] J. M. B. Bakker, J. Phys. E, **6**, 785 (1973); **7**, 364 (1974)

[27] W. C. Wiley and I. H. McClaren, Rev. Sci. Instrum. **26**, 1150 (1955).

[28] It is probably not a good idea to vary conditions too much during a total photodissociation cross section scan, due to normalization difficulties, but during dynamics experiments using time- and position-sensitive detection adjusting ion production variables is considered completely acceptable.

[29] M. Young, *Optics and Lasers*, 3rd. Rev. Ed. Spinger-Verlag, New York, 1986.

[30] See Chapter 1 of *Microchannel Plate Report*, H. Kersten, Ed., (FOM-Instituut voor Atoom- en Molecuulfysica, Amsterdam, 1982), p. 32.

[31] D. P. DeBruijn and J. Los, Rev. Sci. Instrum. **53**, 1020 (1982); L. D. A. Siebbeles, J. M. Schins. W. J. Van der Zande, J. Los, and M. Glass-Maujean, Phys. Rev. A **44**, 343 (1991), and references therein.

[32] H. Helm and P.C. Cosby, J. Chem. Phys. **86**, 6813 (1987).

[33] L. J. Richter and W. Ho, Rev. Sci. Instrum., **57**, 1469 (1986).

[34] J. M. Schins, R. Vrijen, W. J. Van der Zande and J. Los, Surf. Sci. **280**, 145 (1993).

[35] L. D. Gardner, M. M. Graff, and J. L. Kohl, Rev. Sci. Instrum. **57**, 177 (1986).

[36] This is determined by splitting a trigger signal and, keeping cable lengths the same, have one branch trigger the digital delay generator (while the output channel is set to zero delay) and then out to the oscilloscope. Meanwhile, have the other branch go directly to the oscilloscope. The relative delay can thus be measured directly.

[37] D. J. Leahy, D. R. Cyr, D. L. Osborn and D. M. Neumark, Chem. Phys. Lett. (in press).

[38] This ratio can be higher depending on experimental considerations such as beam energy and, in particular, whether the on-axis ion compressor (see § 2.5.2) is employed.

[39] J. Biesner, L. Schnieder, G. Ahlers, X. Xie, K. H. Welge, M. N. R. Ashfold, and R. N. Dixon, J. Chem. Phys. **91**, 2901 (1989); X. Xie, L. Schnieder, H. Wallmeier, R. Boettner, K. H. Welge, and M. N. R. Ashfold, J. Chem. Phys. **92**, 1608 (1990); L. Schnieder, W. Meier, K. H. Welge, M. N. R. Ashfold, and C. M. Western, J. Chem. Phys. **92**, 7027 (1990); G. P. Morley, I. R. Lambert, M. N. R. Ashfold, K. N. Rosser, and C. M. Western, J. Chem. Phys. **97**, 3157 (1992); D. H. Mordaunt, I. R. Lambert, G. P. Morley, M. N. R. Ashfold, R. N. Dixon, C. M. Western, L. Schnieder, and K. H. Welge, J. Chem. Phys. **98**, 2054 (1993).

[40] D. W. Chandler and P. L. Houston, J. Chem. Phys. **87**, 1445 (1987); D. W. Chandler J. M. Thoman Jr., M. H. M. Janssen, D. H. Parker, Chem. Phys. Lett. **156**, 151 (1989); D. W. Chandler, M. H. M. Janssen, S. Stolte, R. N. Strickland, J. W. Thoman, Jr., D. H. Parker, J. Phys. Chem. **94**, 4839 (1990); D. P. Baldwin, M. A. Buntine, and D. W. Chandler, J. Chem. Phys. **93**, 6578 (1990); M. A. Buntine, D. P. Baldwin, R. N. Zare, and D. W. Chandler, J. Chem. Phys. **94**, (1991); M. H. M. Janssen, D. H. Parker, G. O. Sitz, S. Stolte, D. W. Chandler, J. Phys. Chem. **95**, 8807 (1991); M. A. Buntine, D. P. Baldwin, and D. W. Chandler, J. Chem. Phys. **96**, 5843 (1992); W. P. Hess, D. W. Chandler, and J. W. Thoman Jr., Chem. Phys. **163**, 277 (1992); T. Suzuki, V. P. Hradil, S. A. Hewitt, P. L. Houston, and B. J. Whitaker, Chem. Phys. Lett. **187**, 257 (1991); A. G. Suits, R. L. Miller, L. S. Bontuyan, and P. L. Houston, J. Chem. Soc. Faraday Trans. **89**, 1443, (1993); V. P. Hradil, T. Suzuki, S. A. Hewitt, P. L. Houston, and B. J. Whitaker, J. Chem. Phys. **99**, 4455 (1993).

[41] See reference 4, page 55 for gate generator circuits of the same design.

Chapter 3

Data Analysis: Principles and Procedures

Because of improvements in the detection scheme used in the experimental apparatus, we have obtained information concerning the dynamics of free radical dissociation through two different techniques. In our initial studies of the photodissociation dynamics of both N_3 and NCO (the investigation of NCO is detailed in Chapter 5) a somewhat primitive method of probing dissociation dynamics was applied, namely, measuring the time-of-flight distributions of the photofragments. This approach was considered intermediate between having essentially no way of probing the dynamics of dissociation and the imminent implementation of the time- and position-sensitive detector, then under construction. When the time- and position-sensitive detector was fully developed, it immediately became the tool by which dynamical information concerning the dissociating radical under investigation was gleaned. Because Chapter 5 chronicles experiments which were completed before the new detector was completed, while the experiments in Chapter 6 took place after the new detector was put into service, the principles used to analyze both types of data will be discussed.

3.1 Time-of-Flight Distributions: Monte Carlo Forward Convolution

As mentioned above, the time- and position-sensitive detector was developed after the rest of the apparatus. In order to obtain some information about the dynamics of the dissociation processes under study, a much simpler, although admittedly somewhat less informative detection scheme was assembled. By using a 200 MHz transient digitizer to record and sum the time of arrival of all fragments from $1–4 \times 10^4$ laser shots, we were able to build up a time-of-flight distribution. This was typically done for both vertical and horizontal laser polarizations, in order to be able to deconvolve the photofragment angular distribution from the data unambiguously.

It must be acknowledged that this technique is somewhat limited, in that it is only applicable if the fragment masses are known, for reasons which will become apparent. For this reason, while it was quite successful when used to investigate simple systems such as N_3 and NCO, it would be inappropriate for a system with multiple and/or undetermined product channels.

The time-of-flight distribution is typically comprised of signal from both fragments of a given dissociation event. However, there is no upper detection limit of one event per laser shot as there is with the more sophisticated time- and position-sensitive detection scheme that is currently employed. There is also no coincidence requirement, that is, if only one fragment of a given dissociation is detected, whether because of geometric discrimination or detector efficiency, it is not rejected for want of a corresponding fragment. The wings of the fragment time-of-flight distribution will always be from the light fragment of dissociation events releasing the most kinetic energy parallel to the beam axis. Thus, assuming the masses of the fragments are known, the maximum kinetic energy release can be determined by consideration of the kinematics and the observed minimum and maximum flight times. The shape of the time-of-flight distribution is the result of a convolution of the kinetic energy release and the recoil angular distribution. This time-of-flight distribution can therefore be analyzed to give internal energy distributions of the products at a moderate level of detail, corresponding to the vibrational populations of all accessible product levels (the data typically does not justify determining rotational distributions), as well as an anisotropy parameter for events associated with each N atom electronic state.

The method of analysis used to obtain the best fit to the data for these parameters is known as Monte Carlo forward convolution. The program used in this procedure was adapted by Dr. Robert Continetti from the LabAvg code used to analyze D + H_2 reactive scattering data.[1] LabAvg itself was a modified version of importance-sampling Monte Carlo code originally written by Dr. R. B. Walker, which was based on a publication by

Dr. R. T. Pack[2] describing a rigorous treatment of the apparatus averaging of the discrete center-of-mass to laboratory transformation. What follows is a brief explanation of the Monte Carlo forward convolution method of analysis and description of how it applies to our experiment.

'Monte Carlo' refers to the use of random sampling of all of the microscopic experimental variables leading up to a dissociation event. Importance sampling, which allows a weighted random sampling for each variable, is achieved by using normalized probability functions to describe the various distributions of interest. For example, it can be shown experimentally that the ion (and thus the radical) beam energy distribution are well described by a Gaussian having a FWHM of ~0.1 %. Randomly sampling this weighted distribution will result in a very realistic treatment, in that energies closer to the mean will be sampled more often than those in the wings of the distribution.

This approach is applied to each variable in turn during an iteration, which itself consists of following an event from radical, to fragments, to the detector face, if the event is successful in propagating that far. Examples of quantities having a distribution that must be sampled during an iteration include: lab kinetic energy of the beam (mentioned above), beam divergence, photodissociation volume, dissociation photon energy, etc. These quantities can all be assigned realistic distributions from which the program samples.

The program keeps track of position and velocity information for all particles as each iteration proceeds; determining, for example, whether the radical trajectory will be successful in propagating through the beam-defining apertures, and whether the photofragments hit the active area of the detector face. A predicted photofragment recoil angular distribution and vibrational level population distribution (the latter implies the translational energy distribution through conservation of energy considerations, as seen in § 1.3.2) is sampled as part of the simulation for each iteration. The program continues to follow the fragments to the detector face, noting the time-of-flight, so that a simulated

time-of-flight distribution can be accumulated following approximately 1×10^4 iterations. This simulation can then be compared to the actual time-of-flight distribution from the experiment, and any deficiencies noted. A new recoil energy and angular distribution can then be constructed, and this used in another simulation.

It is this process of predicting, simulating, comparing simulation to data, and modifying the prediction to attempt to correct deficiencies that results in the name 'forward convolution'. By this iterative, often time-consuming process a recoil and angular distribution can eventually be found that fits the time-of-flight distributions from both polarizations and all flight lengths at a given photon energy.

3.2 Analysis of Time- and Position-Sensitive Detector Data

Once the time- and position-sensitive detector became operational a direct measurement of the mass ratio (useful in the case of multiple channels), kinetic energy release (KER) and recoil angle with respect to the direction of laser polarization could be made for each dissociation event through methods outlined below, and analysis of time-of-flight distributions became obsolete. The specific implementation of our analysis principles for time- and position-sensitive detector data has evolved to incorporate improvements every time we have undertaken the analysis process for a new system. Several improvements and options in the data analysis which have been thought out remain to be implemented due to lack of time and/or necessity (i.e., provision for analysis of data obtained with the laser electric field vector $\vec{E}$ to be pointing perpendicular to the detector face). In this section, a description of the analysis principles will be given along with an rough idea of their application.

As explained in § 2.7.6.2, the raw information retrieved from our detector is a measure of the amount of charge found to impinge on each of the three elements (wedge, strip, and zigzag), for each of the two halves of the detector face, as well as the output of a time-to-amplitude converter, which corresponds to the difference in time-of-flight for the

two fragments. This data is then converted using the algorithms given in § 2.7.6.4 to a two-dimensional position of each of the two photofragments on the detector face, and their relative time of arrival. In the remainder of this chapter, the conversion of this position and timing information to kinetic energy release (KER), recoil angle theta with respect to electric field vector $\vec{E}$ of the laser, and mass ratios will be discussed. This is followed by a discussion of the required normalization of the raw translational energy and recoil angle distributions, necessary to account for the presence of the beam block and finite radius of the detector. Lastly, an account is given of the method by which final determination of the translational energy distribution and energy dependent recoil anisotropy is made.

3.2.1 Derivation of the Kinematic Equations Involved in Determining the Center-of-Mass Recoil Angle, Kinetic Energy Release and Mass Fraction for a Dissociation Event

Each of the above title quantities can be determined from the position and relative time-of-flight data afforded by our detector. With reference to Figure 3.1, the equations which allow the these quantities to be found will be derived. Before beginning, a few definitions and identities will be put forth.

3.2.1.1 Preliminaries

The time-of-flight from center of the photodissociation volume to the detector face for the center-of-mass of the dissociating system is

$$t_0 = \frac{L}{|\vec{v}_0|}, \tag{3.1}$$

Figure 3.1 The relationship between the experimental observables (L,M,v_0,r_1,r_2,R,τ) and the desired quantities (m_1,m_2,θ,v_{rel}) is shown by this diagram.

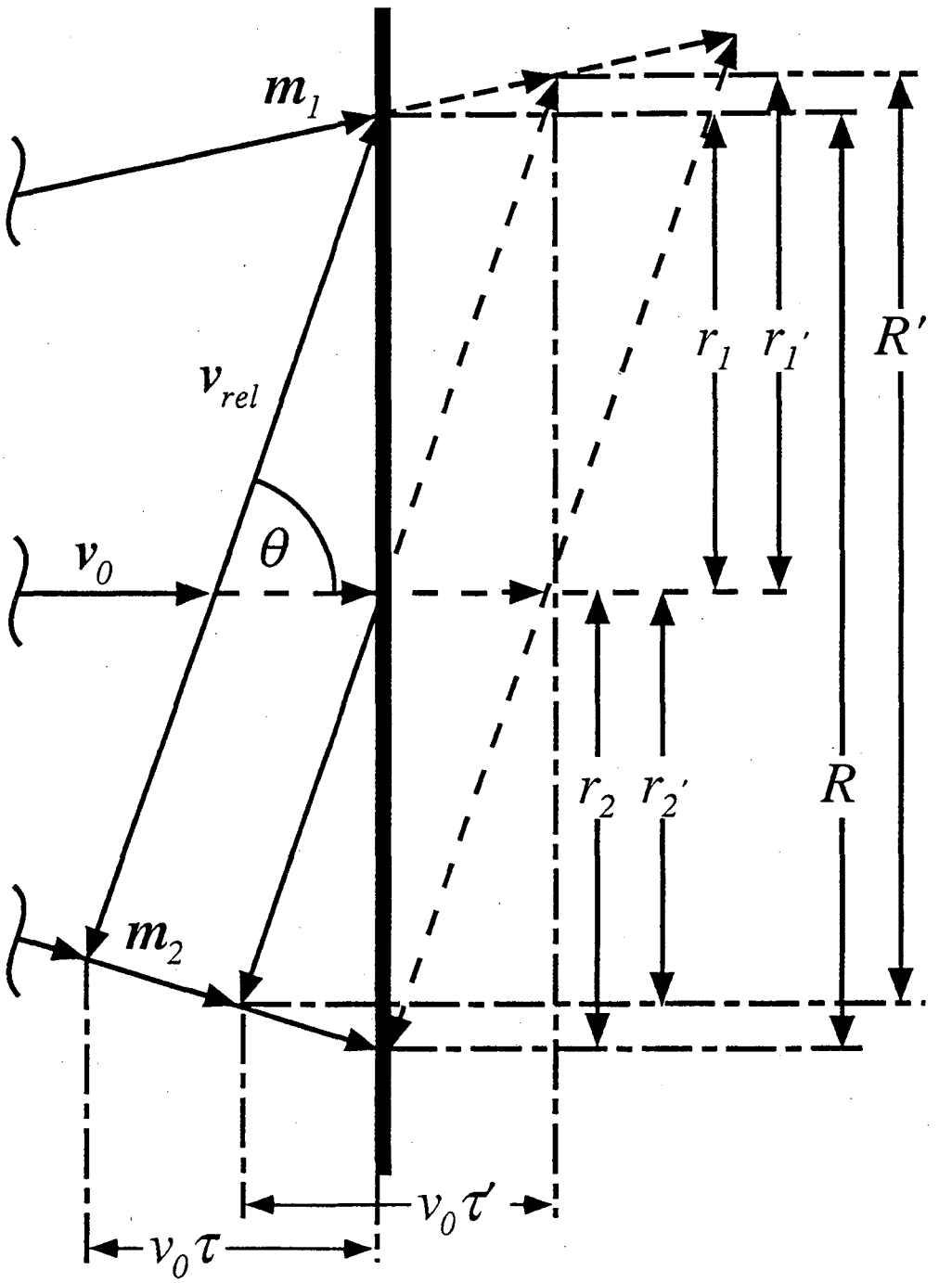


Figure 3.1

where L is the flight length from the center of the dissociation volume to the face of the detector, and $|\vec{v}_0|$ is the velocity of the center-of-mass of the dissociating system, which is determined by the mass of the parent radical M and the ion beam energy E_0, i.e.,

$$|\vec{v}_0| = \sqrt{\frac{2E_0}{M}}. \tag{3.2}$$

The relative velocity vector between the two fragments can be expressed as

$$|\vec{v}_{rel}| = \sqrt{\frac{2E_{KER}}{\mu}}. \tag{3.3}$$

where E_{KER} is the center-of-mass kinetic energy release in the dissociation event and μ is the reduced mass

$$\mu = \frac{m_1 \cdot m_2}{m_1 + m_2}. \tag{3.4}$$

The relative velocity vector is defined via the individual center-of-mass velocity vectors of the two fragments in the usual way:

$$\vec{v}_{rel} = \vec{v}_1 - \vec{v}_2 \tag{3.5}$$

We chose particle 1 to be the fragment which is found on the upper half of the detector, as shown in Figure 3.1. Each of v_1 and v_2 can be expressed as a sum of parallel and perpendicular components, i.e.,

$$\vec{v}_i = \vec{v}_i^{perp} + \vec{v}_i^{par} \tag{3.6}$$

Furthermore, we can express $\vec{v}_i^{par}$ and $\vec{v}_i^{perp}$ in terms of $|\vec{v}_{rel}|$ and θ. θ is defined relative to the direction of $\vec{E}$ (for the purposes of this derivation the electric field vector $\vec{E}$ of the polarized laser output is assumed to lie parallel to the radical beam axis, therefore θ is parallel to $|\vec{v}_0|$ and perpendicular to the detector face). $\vec{v}_i^{par}$ and $\vec{v}_i^{perp}$ are thus given by:

$$\vec{v}_1^{par} = \frac{m_2}{M}|\vec{v}_{rel}|\cdot\cos\theta \qquad \vec{v}_1^{perp} = \frac{m_2}{M}|\vec{v}_{rel}|\cdot\sin\theta$$

$$\vec{v}_2^{par} = -\frac{m_1}{M}|\vec{v}_{rel}|\cdot\cos\theta \qquad \vec{v}_2^{perp} = -\frac{m_1}{M}|\vec{v}_{rel}|\cdot\sin\theta \tag{3.7}$$

The time-of-flight for each particle can be expressed as

$$t_1 = \frac{L}{|\vec{v}_0| + \vec{v}_1^{par}} \qquad \text{and} \qquad t_2 = \frac{L}{|\vec{v}_0| + \vec{v}_2^{par}} \tag{3.8}$$

If τ is defined as positive if the upper fragment (particle 1) arrives first, then

$$\tau = t_2 - t_1 = \frac{L}{|\vec{v}_0| + \vec{v}_2^{par}} - \frac{L}{|\vec{v}_0| + \vec{v}_1^{par}}. \tag{3.9}$$

After substitution for the components of $\vec{v}_1$ and $\vec{v}_2$ which are parallel to the radical beam axis into Eqn. (3.9) an expression for $|\vec{v}_0|\tau$ can be found

$$|\vec{v}_0|\tau = L\left[\frac{\frac{|\vec{v}_{rel}|}{|\vec{v}_0|}\cdot\cos\theta}{1+\frac{m_2-m_1}{M}\cdot\frac{|\vec{v}_{rel}|}{|\vec{v}_0|}\cdot\cos\theta - \frac{m_1 m_2}{M}\left(\frac{|\vec{v}_{rel}|}{|\vec{v}_0|}\cdot\cos\theta\right)^2}\right]. \tag{3.10}$$

Now we turn our attention to finding an expression for $\vec{R}$, which is defined as the vector connecting the two positions of the fragments from a given dissociation event, from the lower to the upper half of the detector. This is almost (but not quite) equivalent to the two-dimensional projection of the three-dimensional relative velocity vector on the detector face. We begin by expressing $\vec{R}$ in terms of $\vec{r}_1$ and $\vec{r}_2$:

$$\vec{R} = \vec{r}_1 - \vec{r}_2 \tag{3.11}$$

Although, there is a relatively large uncertainty in the individual measurements of r_1 and r_2 (see § 3.2.1.4), most of the uncertainty is correlated, and is canceled if we are simply considering $\vec{R}$.

Now, r_1 and r_2 can be expressed as

$$\vec{r}_1 = \vec{v}_1^{\,perp} \cdot t_1 \qquad \text{and} \qquad \vec{r}_2 = \vec{v}_2^{\,perp} \cdot t_2 . \tag{3.12}$$

So

$$R = \left(\vec{v}_1^{\,perp} \cdot t_1\right) - \left(\vec{v}_2^{\,perp} \cdot t_2\right) \tag{3.13}$$

Since we already have expressions for t_1, t_2, and the perpendicular velocity components in terms of $\vec{v}_{rel}$, we substitute to obtain

$$R = L \cdot \left[\frac{\frac{m_2}{M} \cdot \frac{|\vec{v}_{rel}|}{|\vec{v}_0|} \cdot \sin\theta}{1 + \frac{m_2}{M} \cdot \frac{|\vec{v}_{rel}|}{|\vec{v}_0|} \cdot \cos\theta} + \frac{\frac{m_1}{M} \cdot \frac{|\vec{v}_{rel}|}{|\vec{v}_0|} \cdot \sin\theta}{1 - \frac{m_1}{M} \cdot \frac{|\vec{v}_{rel}|}{|\vec{v}_0|} \cdot \cos\theta} \right], \tag{3.14}$$

which can be rearranged to give

$$R = L \cdot \left[\frac{\frac{|\vec{v}_{rel}|}{|\vec{v}_0|} \cdot \sin\theta}{1 + \frac{m_2 - m_1}{M} \cdot \frac{|\vec{v}_{rel}|}{|\vec{v}_0|} \cdot \cos\theta - \frac{m_1 \cdot m_2}{M^2} \left(\frac{|\vec{v}_{rel}|}{|\vec{v}_0|} \cdot \cos\theta \right)^2} \right]. \tag{3.15}$$

3.2.1.2 Center-of-Mass Recoil Angle

If Eqn. (3.15) is compared with Eqn. (3.10) one will note similarities. In particular the denominator and portions of the numerator on both right hand sides are identical. Dividing Eqn. (3.15) by Eqn. (3.10) gives

$$\frac{R}{|\vec{v}_0|\tau} = \frac{\sin\theta}{\cos\theta}. \tag{3.16}$$

It is now apparent that θ can be determined from the measured quantities of R and τ, and knowledge of the radical beam velocity, $|\vec{v}_0|$. Rearranging Eqn. (3.16) to solve for θ gives

$$\theta = \tan^{-1}\left(\frac{R}{|\vec{v}_0|\tau}\right). \tag{3.17}$$

It should be emphasized that despite its simplicity, Eqn. (3.17) is exact.

3.2.1.3 Kinetic Energy Release

The second quantity that must be extracted from the data is the kinetic energy released in the dissociation or, synonymously, the center-of-mass translational energy of the dissociating system (E_{KER} and E_T are used interchangeably in our analysis and discussion). This quantity can be determined beginning with the ratio of the beam energy to the kinetic energy released in the dissociation process:

$$\frac{E_{KER}}{E_0} = \frac{\mu}{M}\cdot\frac{|\vec{v}_{rel}|^2}{|\vec{v}_0|^2} \tag{3.18}$$

Within this expression, only $|\vec{v}_{rel}|$ (and of course E_{KER}) is not known explicitly. However, the ratio of recoil velocity to beam velocity is proportional to the ratio of the recoil distance to the flight length. Unfortunately, unless θ = 90°, we do not measure the recoil distance of the fragments after propagating for exactly the flight time taken by the center-of-mass of the dissociating system. However, we can determine this quantity from the experimental observables R and τ.

Once again referring to Figure 3.1, if we call R' the separation of the fragments perpendicular to the radical beam axis after time t_0 (the center-of-mass flight time), and $|\vec{v}_0|\tau'$ the separation of the fragments parallel to the radical beam axis after time t_0, we can write

$$E_{KER} = E_0 \cdot \frac{m_1 \cdot m_2}{M^2} \cdot \frac{(R')^2 + (v_0\tau')^2}{L^2}. \tag{3.19}$$

Expressions for R' and $|\vec{v}_0|\tau'$ can be determined to be

$$R' = t_0 \cdot \vec{v}_{rel}^{\,perp} = \frac{L}{|\vec{v}_0|} \cdot |\vec{v}_{rel}| \cdot \sin\theta \tag{3.20}$$

$$v_0\tau' = t_0 \cdot \vec{v}_{rel}^{\,par} = \frac{L}{|\vec{v}_0|} \cdot |\vec{v}_{rel}| \cdot \cos\theta \tag{3.21}$$

Comparing these last Eqns. for R' and $|\vec{v}_0|\tau'$ to Eqns. (3.10) and (3.15) which give expressions for R and $|\vec{v}_0|\tau$, respectively, it can be seen that

$$R' = R \cdot \left[1 + \frac{m_2 - m_1}{M} \cdot \frac{|\vec{v}_{rel}|}{|\vec{v}_0|} \cdot \cos\theta - \frac{m_1 \cdot m_2}{M^2} \left(\frac{|\vec{v}_{rel}|}{|\vec{v}_0|} \cdot \cos\theta \right)^2 \right] \tag{3.22}$$

and similarly

$$|\vec{v}_0|\tau' = |\vec{v}_0|\tau \cdot \left[1 + \frac{m_2 - m_1}{M} \cdot \frac{|\vec{v}_{rel}|}{|\vec{v}_0|} \cdot \cos\theta - \frac{m_1 \cdot m_2}{M^2} \left(\frac{|\vec{v}_{rel}|}{|\vec{v}_0|} \cdot \cos\theta \right)^2 \right]. \tag{3.23}$$

Substituting for R' and $|\vec{v}_0|\tau'$ into Eqn. (3.19) gives

$$E_{KER} = E_0 \cdot \frac{m_1 \cdot m_2}{M^2} \cdot \frac{\left[(|\vec{v}_0|\tau)^2 + R^2 \right]}{L^2} \cdot \left[1 + \left(\frac{m_2 - m_1}{M} \right) \cdot \frac{|\vec{v}_{rel}|}{|\vec{v}_0|} \cdot \cos\theta - \left(\frac{m_1 \cdot m_2}{M^2} \right) \left(\frac{|\vec{v}_{rel}|}{|\vec{v}_0|} \cdot \cos\theta \right)^2 \right]^2 \tag{3.24}$$

We can now expand the final square-bracketed term to be a polynomial in $\frac{|\vec{v}_{rel}|}{|\vec{v}_0|} \cdot \cos\theta$. If we recognize that

$$\frac{|\vec{v}_{rel}|}{|\vec{v}_0|} = \sqrt{\frac{E_{KER}}{E_0} \cdot \frac{M}{\mu}}, \tag{3.25}$$

we can see that $\frac{|\vec{v}_{rel}|}{|\vec{v}_0|}$ is determined primarily by the square root of the energies involved.

At a typical beam energy of 6000 eV, standard recoil energies in the 0–5 eV range give a $\frac{|\vec{v}_{rel}|}{|\vec{v}_0|}$ ratio on the order of 0.01. Therefore, since $\frac{|\vec{v}_{rel}|}{|\vec{v}_0|} << 1$, we leave out terms higher than second order in $\frac{|\vec{v}_{rel}|}{|\vec{v}_0|} \cdot \cos\theta$ to obtain

$$E_{KER} = E_0 \cdot \frac{m_1 \cdot m_2}{M^2} \cdot \frac{\left[\left(|\vec{v}_0|\tau\right)^2 + R^2\right]}{L^2} \cdot \left[\begin{array}{l} 1 + 2\left(\frac{m_2 - m_1}{M}\right) \cdot \frac{|\vec{v}_{rel}|}{|\vec{v}_0|} \cdot \cos\theta + \\ \left[\frac{(m_2 - m_1)^2 - 2m_1 m_2}{M^2}\right]\left(\frac{|\vec{v}_{rel}|}{|\vec{v}_0|} \cdot \cos\theta\right)^{2^*} \end{array}\right] \tag{3.26}$$

At this point we need to eliminate references to $\frac{|\vec{v}_{rel}|}{|\vec{v}_0|}\cos\theta$ since we do not know this quantity exactly. We can, however, substitute the equivalent quantity as given by a rearranged Eqn. (3.10), shown here as Eqn. (3.27),

$$\frac{|\vec{v}_{rel}|}{|\vec{v}_0|} \cdot \cos\theta = \frac{|\vec{v}_0|\tau}{L}\left[1 + \frac{m_2 - m_1}{M} \cdot \frac{|\vec{v}_{rel}|}{|\vec{v}_0|} \cdot \cos\theta - \frac{m_1 m_2}{M}\left(\frac{|\vec{v}_{rel}|}{|\vec{v}_0|} \cdot \cos\theta\right)^2\right]. \tag{3.27}$$

Keeping only terms contributing at the second order level or less in $\frac{|\vec{v}_0|\tau}{L}$, which itself is on the order of $\frac{|\vec{v}_{rel}|}{|\vec{v}_0|}$, gives the result

$$E_{KER} = E_0 \cdot \frac{m_1 \cdot m_2}{M^2} \cdot \frac{\left[\left(|\vec{v}_0|\tau\right)^2 + R^2\right]}{L^2} \cdot \left[\begin{array}{l} 1 + 2\left(\frac{m_2 - m_1}{M}\right) \cdot \frac{|\vec{v}_0|\tau}{L} + \\ \left[\frac{3(m_2 - m_1)^2 - 2m_1 \cdot m_2}{M^2}\right]\left(\frac{|\vec{v}_0|\tau}{L}\right)^2 \end{array}\right], \tag{3.28}$$

which in almost all instances does not give a discernibly different result when the quadratic correction is neglected. Therefore the form of the equation we often use in practice is:

$$E_{KER} = E_0 \cdot \frac{m_1 \cdot m_2}{M^2} \cdot \frac{\left[\left(|\bar{v}_0|\tau\right)^2 + R^2\right]}{L^2} \cdot \left[1 + 2\left(\frac{m_2 - m_1}{M}\right) \cdot \frac{|\bar{v}_0|\tau}{L}\right]. \qquad (3.29)$$

3.2.1.4 Mass Fractions

It is sometimes the case that the radical under consideration is sufficiently simple that the identity of the dissociation products is evident. However, as larger and larger radicals are studied it will be critical to be able to identify the product channel(s) that are produced. Analysis of the kinetic energy release, for example, is precluded if the masses of the fragments involved is unknown. For mass ratio measurement, the position of each particle on the detector face relative to the average center of mass position of the radical beam (x_0, y_0) is required, and is given by $r_i = \sqrt{(x_i - x_0)^2 + (y_i - y_0)^2}$ (where i = 1,2).

The average center of mass position of the radical beam (x_0, y_0) must be determined from the data. This is a trivial procedure for cases where the masses of the products are known, since the center-of-mass coordinates of each event can be calculated in turn, summed and averaged. If the masses of the fragments are unknown, a slightly different procedure is used, which can be explained as follows: Imagine if, for all dissociation events, the positions of both fragments were joined by a straight line. By choosing a circle of diameter ~2 mm, and varying its location in the vicinity of where, experimentally, we know the position of the radical beam should be, we can eventually maximize the number of events whose lines intersect this circle. The position of the center of this circle is then taken to be (x_0, y_0), as well as the peak of a two dimensional Gaussian distribution describing the radical beam intensity, having a full width at half maximum of ~1 mm.

Knowing r_1 and r_2, albeit with some uncertainty, will allow the calculation of the mass fraction $\frac{m_1}{m_2}$ associated with a given event. Remembering that r_1 and r_2 can be expressed as

$$\vec{r}_1 = \vec{v}_1^{\,perp} \cdot t_1 \qquad \text{and} \qquad \vec{r}_2 = \vec{v}_2^{\,perp} \cdot t_2 \tag{3.30}$$

and that we have already developed expressions for $\vec{v}_1^{\,perp}$, $\vec{v}_2^{\,perp}$, t_1, and t_2, we can then set up the ratio

$$\frac{r_1}{r_2} = \frac{\frac{m_2}{M} \cdot \frac{|\vec{v}_{rel}|}{|\vec{v}_0|} \cdot \sin\theta \cdot \frac{L}{1 + \frac{m_2}{M} \cdot \frac{|\vec{v}_{rel}|}{|\vec{v}_0|} \cdot \cos\theta}}{\frac{m_1}{M} \cdot \frac{|\vec{v}_{rel}|}{|\vec{v}_0|} \cdot \sin\theta \cdot \frac{L}{1 - \frac{m_1}{M} \cdot \frac{|\vec{v}_{rel}|}{|\vec{v}_0|} \cdot \cos\theta}}. \tag{3.31}$$

After canceling terms, this reduces to

$$\frac{r_1}{r_2} = \frac{m_2}{m_1} \cdot \frac{1 - \frac{m_1}{M} \cdot \frac{|\vec{v}_{rel}|}{|\vec{v}_0|} \cdot \cos\theta}{1 + \frac{m_2}{M} \cdot \frac{|\vec{v}_{rel}|}{|\vec{v}_0|} \cdot \cos\theta}. \tag{3.32}$$

Because the $\frac{|\vec{v}_{rel}|}{|\vec{v}_0|}$ ratio is on the order of 0.01, and r_1 and r_2 have a relatively large uncertainty due to the reliance on the average (x_0, y_0), it is logical to invoke the approximation $\frac{1}{1+\delta} \approx 1 - \delta$, to obtain

$$\frac{r_1}{r_2} = \frac{m_2}{m_1} \cdot \left[1 - \frac{|\vec{v}_{rel}|}{|\vec{v}_0|} \cdot \cos\theta + \frac{m_1 m_2}{M} \cdot \left(\frac{|\vec{v}_{rel}|}{|\vec{v}_0|} \cdot \cos\theta \right)^2 \right]. \tag{3.33}$$

We now substitute the expression determined previously in Eqn. (3.27) for $\frac{|\vec{v}_{rel}|}{|\vec{v}_0|} \cdot \cos\theta$, keeping terms only to first order in $\frac{|\vec{v}_0|\tau}{L}$, to give

$$\frac{r_1}{r_2} = \frac{m_2}{m_1} \cdot \left[1 - \left(\frac{|\vec{v}_0|\tau}{L} \right) \right]. \tag{3.34}$$

Since we are interested in the mass ratio of the photofragments, we rearrange the equation once again to finally obtain

$$\frac{m_1}{m_2} = \frac{r_2}{r_1} \cdot \left[1 - \left(\frac{|\vec{v}_0|\tau}{L} \right) \right]. \tag{3.35}$$

Since we always know the total mass M of the radical that is being studied, it is a trivial matter to determine the actual masses from the mass fraction information. This expression is intuitively satisfying because if fragment 1 is detected first (τ is positive) it does not have as much time to recoil as fragment 2. Therefore the fraction $\frac{r_2}{r_1}$ will need a small downward correction, as the factor in the brackets provides. Of course, the situation is reversed when fragment 2 arrives at the detector first, but because τ is then negative by definition, the formula again provides the proper correction.

If *a priori* the identities of the fragment channel(s) produced are unknown, we must then examine the photofragment mass spectrum constructed from the data via Eqn. (3.35). Because we know the chemical composition in addition to the mass of the parent radical, there will be a limited number of mass combinations possible, and fewer still that are probable. Generally then, the determination of the identities of the photofragments will be straightforward.

However, mass ratio measurements are relatively imprecise, owing to the above-mentioned substitution of (x_0, y_0) for the actual location of the center-of-mass of each event; subsequently $m/\Delta m$ is on the order of ~ 20. Occasionally, therefore, this process will be more challenging. At least two methods of distinguishing between possible mass combinations may be helpful. Isotopic substitution can be a convenient approach, depending on the structure of the parent radical. The use of d_3-nitromethane in producing

the $CD_2NO_2^-$ ion *en route* to the photodissociation of CD_2NO_2 is illustrated in § 6.3. By examining the photofragment mass spectrum obtained for CD_2NO_2 and comparing it to that obtained for CH_2NO_2, the identities of two dissociation channels becomes unambiguous. Additionally, molecules sometimes contain an atom that has significant natural abundance at two or more masses (i.e., chlorine, bromine, sulfur, etc.). Since relatively high resolution mass selection of the parent radical occurs during the detachment process, it would be a trivial matter to study molecules containing each isotope in turn, if this would shed light on the identities of the photodissociation products.

3.2.2 Raw Translational Energy and Recoil Angle Distributions

As discussed in Chapter 1, the quantities of interest in a photodissociation dynamics experiment are the true photofragment center-of-mass translational energy and angular distributions. Once again, for a one-photon dissociation the form of the photofragment energy and angular distribution $\mathcal{P}(E_T,\theta)$ can be written as

$$\mathcal{P}(E_T,\theta) = P(E_T)[1 + \beta(E_T)P_2(\cos\theta)] \qquad (3.36)$$

where θ is the angle between the photofragment recoil vector and the electric vector of the dissociation laser, $P_2(\cos\theta)$ is the second Legendre polynomial, $\beta(E_T)$ is the (energy-dependent) anisotropy parameter, and $P(E_T)$ is the angle-integrated kinetic energy distribution.

Eqns. (3.17) and (3.29), originally derived by De Bruijn and Los,[3] (with the exception of an inconsistent sign convention) are used to obtain the center-of-mass kinetic (translational) energy release (E_{KER} or E_T) and fragment recoil angle θ, measured with respect to the electric field vector $\vec{E}$ of the laser, for each dissociation event. If the time- and position-sensitive detector were able to detect every dissociation event with equal probability, we would at this point be in possession of the actual translational energy and recoil angle distributions. However, due to the blocking strip across the center of the

detector and the finite radius of the detector face, some (well-defined) geometric discrimination occurs, which must be accounted for.

We begin by constructing the so-called raw translational energy and recoil angle distributions. For each product channel, we process a set of data from the time- and position-sensitive detector, consisting of a total of 50,000 to 100,000 events in most cases, into a two dimensional energy-angle array $I(E_T,\theta)$. This is done using energy steps comparable to the instrument resolution, and theta steps of one degree. Each element of this raw data array must then be divided by the relative acceptance of the detector for the energy-angle combination under consideration, in order to allow the final determination of the true translational energy distribution and energy dependent anisotropy parameters.

We now consider the determination of the detector acceptance function in some detail.

3.2.3 Normalization: The Detector Acceptance Function

Two geometric factors prevent the detection of all dissociation events. One factor is the presence of a blocking strip across the horizontal axis of the detector. This obstruction will always prevent the accumulation of the lowest KER coincidence events no matter what the recoil angle, and will block even large KER events if the recoil angle with respect to the radical beam axis is small. For example, the minimum kinetic energy release that can be detected for two fragments of masses 14 and 28 amu (which is the case in N_3 and NCO dissociation) is 0.20 eV (~1600 cm^{-1}, ~4.6 kcal/mol) if $\theta = 90°$, the detector is in the standard location (centered on the beam axis), the beam energy is 8 keV and the blocking strip is 8 mm wide. The second factor preventing certain dissociation events from being detected is the finite radius of the detector. This limitation will result in events which have one or both particles miss the active area of the detector face when both the kinetic energy release and/or the recoil angle with respect to the radical beam axis are large. Under the same conditions mentioned in the example above, the onset of field-

of-view cutoff occurs for events with purely transverse recoil (θ=90°) at 1.15 eV. After this point, more energetic dissociation recoils can still be detected, however the polar angular acceptance decreases as kinetic energy release becomes larger.

These characteristics of our ability to measure coincidence events translates into an energy and angular (both theta and phi) dependent 'acceptance function' for our detector. In order to determine the true photofragment energy and angular distributions from our data we must account for the effects of this detector acceptance function.

Although a Monte Carlo forward convolution approach to this problem is possible, a far more straightforward alternative involves the numerical modeling of the energy and angular dependence of the detector acceptance. This provides unbiased center-of-mass kinetic energy and angular distributions in a very direct manner, as will now be detailed.

We define the detector acceptance function $\mathcal{S}(E_T,\theta)$ such that:

$$\mathcal{P}(E_T,\theta) = \mathcal{N}(E_T,\theta)/\mathcal{S}(E_T,\theta) \tag{3.37}$$

The function $\mathcal{S}(E_T,\theta)$ is the probability of observing a coincidence event with values of kinetic energy release and recoil angle of E_T and θ respectively. While experimentally we can vary the polarization of the laser such that the electric field vector $\vec{E}$ is perpendicular or parallel to the radical beam axis, our default mode of operation is parallel to the radical beam axis, since the frequency doubled output of our dissociation laser is horizontally polarized. The following discussion will assume that this orientation is in use.

As discussed in § 1.3.4, the photofragment angular distribution is azimuthally symmetric about the electric field vector $\vec{E}$ of the dissociation laser. $\mathcal{S}(E_T,\theta)$ must be found by determining the range of azimuthal angles over which both fragments strike the active area of the detector, expressed as:

$$\mathcal{S}(E_T,\theta) = \frac{1}{2\pi}\int_0^{2\pi} D(E_T,\theta,\phi)\,d\phi. \tag{3.38}$$

In Eqn. (3.39), $D(E_T,\theta,\phi)$ is the doubly-differential detector acceptance function which gives the probability of detection of a dissociation event with kinetic energy release E_T, polar recoil angle θ, and azimuthal recoil angle ϕ. For a point-size radical beam, $D(E_T,\theta,\phi)$ is either 0 or1 (more realistic beam sizes are discussed below). In practice, the calculation of $\mathcal{S}(E_T,\theta)$ simply involves stepping through the full range of ϕ in steps of degrees, summing the evaluations of $D(E_T,\theta,\phi)$, and dividing by 360 to obtain the probability of observing dissociation events with a particular kinetic energy release E_T, and polar recoil angle θ. This is done for every E_T,θ combination, using energy and theta step sizes identical to those used when binning the raw data.

The best agreement with Monte Carlo simulations designed to check the accuracy of these results is obtained when the size and shape of the radical beam is taken into account in the calculation of the doubly differential acceptance function $D(E_T,\theta,\phi)$. This is accomplished by modeling the beam profile at the detector with a two-dimensional Gaussian with a 1 mm full width at half-maximum. The relative intensities of this two-dimensional Gaussian at each point of a 10×10 grid are used as coefficients for the overall detector acceptance function found for that point. After all 100 detector acceptance function calculations have been performed, normalized, and summed, a more realistic detector acceptance function is produced. With this modification, $D(E_T,\theta,\phi)$ takes on values between 0 and 1, representing what fraction of the radical beam profile gives rise to coincidences for kinetic energy release E_T and angles θ and ϕ.

3.2.4 Final Determination of the Angle-Integrated Translational Energy Distribution, $P(E_T)$, and the Energy Dependent Anisotropy Parameters, $\beta(E_T)$

Once $\mathcal{S}(E_T,\theta)$ has been determined, the raw energy and angular distributions can be converted into the (unnormalized) probability distribution $\mathcal{P}(E_T,\theta)$, using Eqn. (3.37). We then factor $\mathcal{P}(E_T,\theta)$ into the angle-integrated translational energy distribution, $P(E_T)$, and

an energy dependent recoil angular distribution, as in Eqn. (3.36). In practice, this entails first determining $\beta(E_T)$ values followed by the determination of $P(E_T)$.

Currently, $\beta(E_T)$ values are obtained for each kinetic energy interval from $E_i-\Delta E/2$ to $E_i+\Delta E/2$ where ΔE is an arbitrary, constant energy range large enough to provide good statistics for the bulk of the observed range of kinetic energy release. These intervals are typically ~10–15 times the typical kinetic energy bin size. The anisotropy parameter β for the kinetic energy release interval centered at the energy E_i is then found via summation of the normalized angular distribution data over the angles common to all of the kinetic energy bins in the range from $E_i-\Delta E/2$ to $E_i+\Delta E/2$, followed by a simple least squares fit to Eqn. (3.36).

The final step in this direct inversion analysis is to find the angle-integrated translational energy distribution $P(E_T)$ at the highest energy resolution. The values of $\beta(E_T)$ determined above are used in Eqn. (3.36) while $P(E_T)$ is found for energy intervals the size of the original bins (i.e., intervals on the order of the instrument resolution).

An improvement to this method is under development, to be implemented in the analysis of the next set of data. It involves using a flexible ΔE interval to allow the determination of $\beta(E_T)$ to have approximately the same uncertainty across the whole observed kinetic energy range. In addition, the simultaneous fitting of both $P(E_T)$ and a common $\beta(E_T)$ for the highest resolution energy intervals within the ΔE interval may result in a better overall determination of these quantities than the two step process we currently employ.

There are two major sources of uncertainty in the determination of $P(E_T)$. One source is simply associated with the nature of counting experiments, and is well modeled by Poisson statistics. The second arises from the uncertainty in the value we determine for $\beta(E_T)$. This uncertainty, especially when fixed ΔE intervals are used, can be negligible in some regions and dominate in others. The $\beta(E_T)$ uncertainty is especially large at high E_T where the polar acceptance grows small, thereby lowering both the angular range

contributing to the determination of β *and* the number of dissociation events detected. The uncertainty in $\beta(E_T)$ is locally correlated in the energy release spectrum (β is not expected to randomly jump from one energy bin to another), so the uncertainty in $\beta(E_T)$ is not added in quadrature to the statistical error. Instead, the effect of the uncertainty in $\beta(E_T)$ can be conveyed by using both the upper and lower uncertainty limits of $\beta(E_T)$ in separate determinations of $P(E_T)$, and comparing the results with the translational energy distribution found using the best-fit values.

Overall, the strengths of coincidence detection and direct inversion of the data are multiple. First, branching ratios between product channels (if applicable) are given by comparison of the integrated area under the angle integrated translational energy distribution for each channel. Second, we obtain the explicit energy dependence of the anisotropy parameter from our experiment, unlike most other photodissociation studies. Finally, determination of the angle-integrated translational energy distribution and the energy-dependent anisotropy parameter is through direct inversion of the data, avoiding an iterative and time-consuming forward convolution process.

3.3 References

[1] R. E. Continetti, Ph. D. Thesis, University of California, Berkeley, 1989.

[2] R. T. Pack, J. Chem. Phys. **81**, 1841 (1984).

[3] D. P. De Bruijn and J. Los, Rev. Sci. Instrum. 53, 1020 (1982).

Chapter 4

Photodissociation Studies of O_2 and N_3: Examples of FRBM Capabilities

Our photodissociation studies of the two systems, O_2 and N_3, are good illustrations of the capabilities of the Fast Radical Beam Machine. Both species have predissociative excited states, resulting in a structured total photodissociation cross section. The simplicity of O_2 allows its dissociation to be used as a calibrant and a measure of the ultimate resolution of the detector. On the other hand, N_3 is in many ways an ideal polyatomic radical system for investigation using this apparatus, and was chosen as the first radical to be studied using time- and position-sensitive detection. In the following four subsections the total photodissociation cross section experiments followed by the dissociation dynamics experiments for each of O_2 and N_3 are briefly summarized, with the emphasis on experimental results. A full account of both the O_2[1] and N_3[2] work is given elsewhere.

4.1 Photodissociation Cross Sections in the $B\,^3\Sigma_u^-(v') \leftarrow X\,^3\Sigma_g^-(v'')$

Schumann-Runge Bands of O_2

Total photodissociation cross sections of O_2 were originally obtained in limited spectral regions with the sole expectation of using the predissociation of the Schumann-Runge bands as an ideal means to calibrate (and determine the resolution of) the time- and position-sensitive detector, in a complementary manner to the calibration technique discussed in § 2.6.6.5. (A description of the calibration technique involving O_2 dissociation is included in the following section.) Fortunately, a relatively high kinetic energy release resolution was realized in the time- and position-sensitive experiments, and our important but narrow reasons for studying O_2 dissociation became expanded. This development prompted the measurement of the photodissociation cross section for all of the O_2 $B\,^3\Sigma_u^-$ state vibrational levels possible using our apparatus, from v'=0 to v'=11.[1]

Here, however, only the photodissociation cross section spectrum of the $B\,^3\Sigma_u^-(v'=7)\leftarrow X\,^3\Sigma_g^-(v''=4)$ band is discussed as a representative example.

The experimental conditions for O_2 photodissociation experiments are now briefly summarized. Neat O_2 at a stagnation pressure of 20 PSIG was used to form O_2^- using the electron beam-pulsed free jet expansion source. By operating the detachment laser at 480 nm (2.58 eV), we photodetached the electrons well above the 0.45 eV threshold for this process. Vibrationally excited O_2 $X\,^3\Sigma_g^-$ as well as O_2 $a\,^1\Delta_g$ are formed at this detachment wavelength, with the vibrational state distributions governed by the Franck-Condon factors between the O_2^- $X\,^2\Pi_g$ anion and the respective neutral states. As explained in § 1.4, we normally find it advantageous to avoid vibrational excitation of the radical by detaching near threshold. However, as the potential energy curves sketched in Figure 4.1 show, in studies of the Schumann-Runge $B\,^3\Sigma_u^-(v')\leftarrow X\,^3\Sigma_g^-(v'')$ bands, vibrational excitation in the ground electronic state of O_2 is an advantage as it allows subsequent excitation to the lower vibrational levels of the $B\,^3\Sigma_u^-$ state. In contrast, these lower vibrational levels of the $B\,^3\Sigma_u^-$ state have vanishing Franck-Condon factors from the ground vibrational level of the $X\,^3\Sigma_g^-$ state owing to the significant extension of equilibrium interatomic distance in the excited state.

Originally, O_2 total photodissociation cross section scans were undertaken using the detector primarily designated for that purpose, described in § 2.7.5. The total photodissociation cross section spectrum presented in Figure 4.2, however, was obtained using the TPS detector, after linking together the wedge, strip and zigzag output, and measuring only total photofragment flux. Figure 4.2 shows a total cross section spectrum

---

Figure 4.1 Schematic of the O_2 potential energy curves relevant to the study of predissociation following excitation of the Schumann-Runge $B\,^3\Sigma_u^-(v')\leftarrow X\,^3\Sigma_g^-(v'')$ bands.

Figure 4.2 The total photodissociation cross section of O_2 (0.0025 nm steps, corrected to vacuum wavelengths) in the region of the v'=7←v"=4 band.

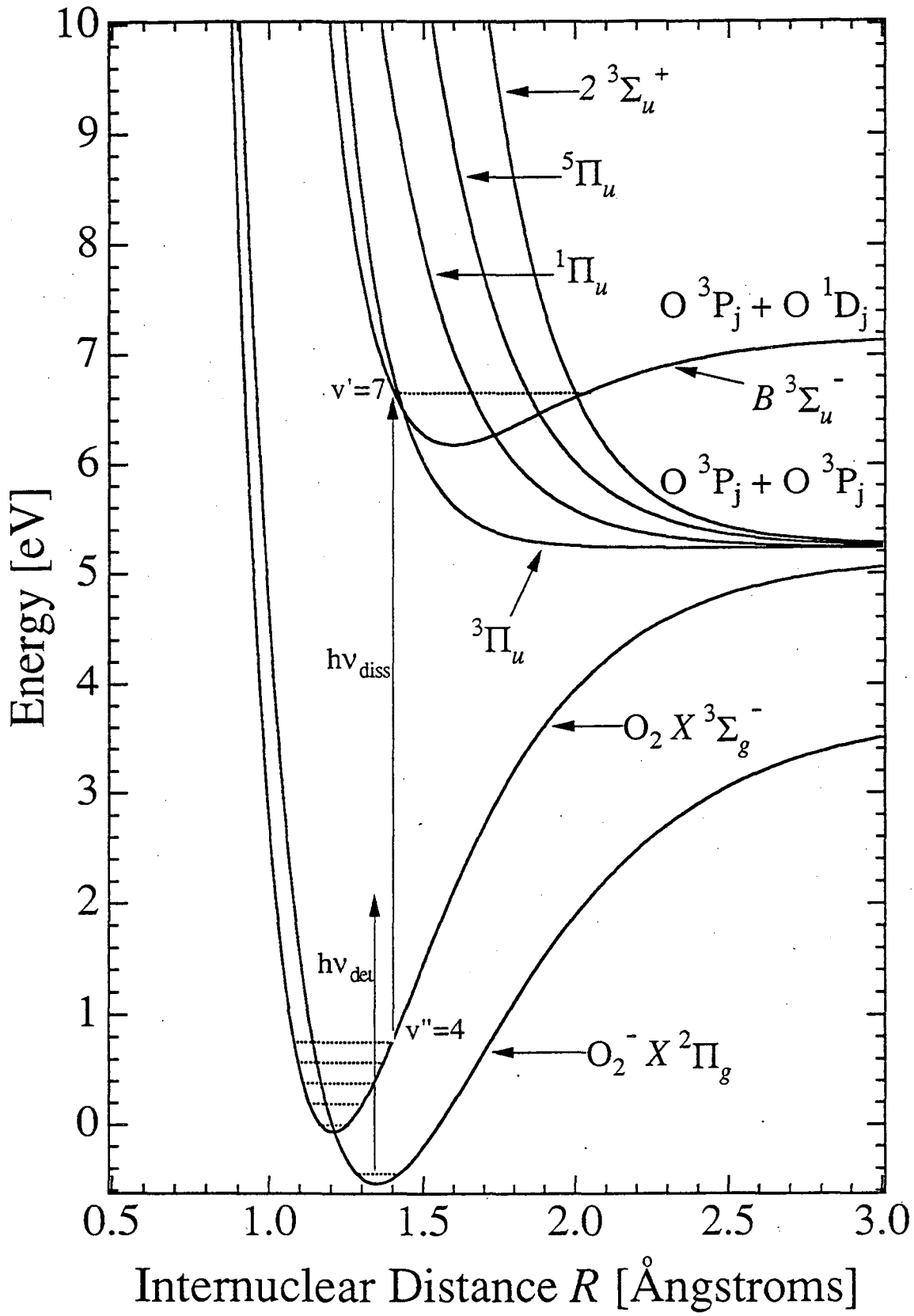

Energy [eV]
Internuclear Distance R [Ångstroms]
2 ³Σu+
⁵Πu
¹Πu
O ³Pj + O ¹Dj
B ³Σu-
O ³Pj + O ³Pj
³Πu
O2 X ³Σg-
O2- X ²Πg
v'=7
v''=4
hνdiss
hνdet

Figure 4.1

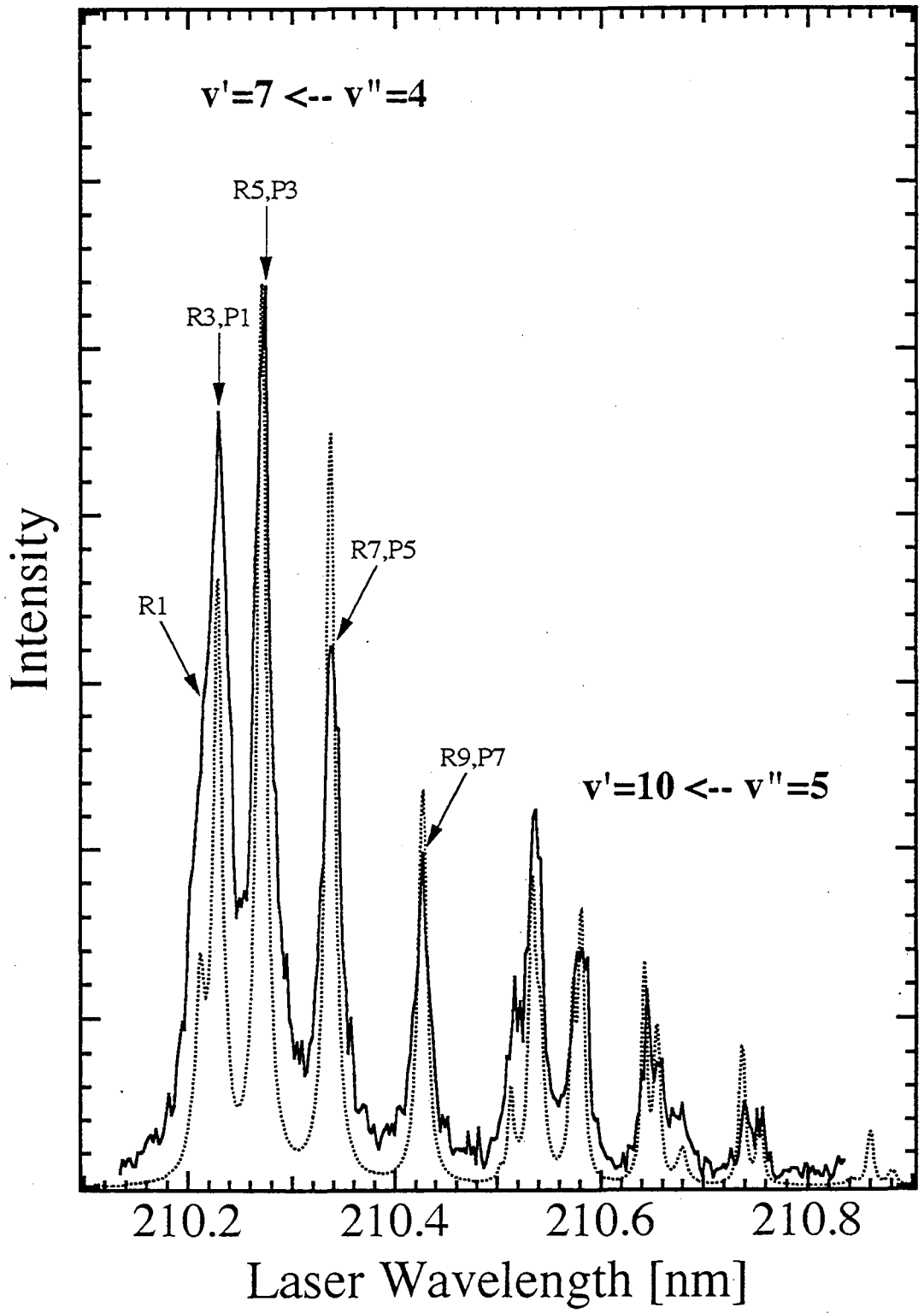

v'=7 <-- v"=4
R5,P3
R3,P1
R7,P5
R1
R9,P7
v'=10 <-- v"=5
Intensity
210.2
210.4
210.6
210.8
Laser Wavelength [nm]

Figure 4.2

in the region of the v'=7←v"=4 band along with a simulation that assumes a Boltzmann temperature of 50 K (offset for clarity). The photofragment signal intensity is normalized by detached electron signal and dissociation laser power. This spectrum contains the sum of signal from only 200 laser shots per point. Note that while (partial) rotational resolution is achieved, the natural linewidth is quite large for many of these transitions, obscuring spin-rotation fine structure. The vibrational bands within the $B\,^3\Sigma_u^- \leftarrow X\,^3\Sigma_g^-$ electronic transition are degraded to the red quite strongly, with the band-head occurring at the origin in most cases, and the v'=7←v"=4 transition is no exception. The rotational transitions are labeled above the appropriate peaks.

Although there is essentially no new spectroscopic information in this spectrum as the ground and excited levels involved have been thoroughly characterized in the past,[3] these total photodissociation cross section studies were still important for a number of reasons. As a test system for the experimental apparatus O_2 is quite convenient, both because of the ease of producing O_2^- and also because the Schumann-Runge bands predissociate, resulting in a structured total photodissociation cross section. However, the major purpose of investigating the total photodissociation cross section of O_2 became the mapping out of transitions to vibrational levels of the excited $B\,^3\Sigma_u^-$ state from v' = 0 to v' = 11. Once this was completed, we could calibrate our time- and position-sensitive detector and then use it to further study the dissociation dynamics of the Schumann-Runge bands.

4.2 TPS Detection of O_2 Dissociation:

Resolving Correlated Spin-Orbit States of the O $^3P_{j_1}$ + O $^3P_{j_2}$ Products

Because the products of O_2 dissociation are atoms they can possess no internal energy in the form of vibrational or rotational motion. Initially neglecting the relatively small spin-orbit splitting in the ^3P_j ground state of the oxygen atoms ($^3P_0$ is 28 meV, while $^3P_1$ is 19 meV above the ground state $^3P_2$), the kinetic energy release should resemble a

delta function. With reference to Figure 4.1, the recoil energy will be approximately 1.5 eV, equal to the difference in energy between the $B\,^3\Sigma_u^-$ state ro-vibrational level and the O (^3P) + O (^3P) product asymptote. The utility of this system as a calibrant stems from the fact that if many O_2 dissociation events are detected, the distribution of recoil angles will ensure that all regions of the detector face will receive coincidence fragments having recoil vectors with identical magnitudes. If the coefficients in the position algorithm are accurate across the whole detector face, and the electronics are calibrated correctly, then the translational energy distribution calculated from the measured positions and differences in time of flight will be the expected narrow delta function, centered at the proper recoil energy. In this way, the width of the observed translational energy distribution becomes a measure of the calibration of the equipment and algorithms involved in time- and position-sensing and, ultimately, a measure of the resolution.

This approach allows us to collect a large body of O_2 dissociation data and use it to find the best fitting empirical factors in the position algorithm via a least-squares routine. A compromise exists between using a sufficient number of parameters in the fitting expression and having a tractable fitting problem. The least-squares fit is typically prevented from achieving below ~30 meV resolution over the entire detector face due to the necessity of neglecting higher order empirical corrections.

However, with O_2 dissociation data the fit is locally quite good. If the detector face is divided up into 1 mm × 1 mm squares, a resolution of <10 meV is found for most squares with sufficient signal for good counting statistics. In the central portions of the upper and lower halves of the detector areas as large as 4 mm (in the y dimension) × 8 mm (in the x dimension) can exhibit this high resolution using one set of factors.

The use of this piece-wise high resolution can allow us to obtain 10 meV resolution over most of the detector face, in the following way. Best-fit factors are found for the position equations using all but the edges of the detector face, where we find the nonlinearities are too severe. The detector face is then divided up into 1 mm × 1 mm

squares. When the translational energy distributions are determined for each individual area, a correction can be applied to shift the mean kinetic energy release for each area to match the mean found with the best-fit factors applied to all regions. The translational energy distributions for all of the smaller regions can then be summed and the resulting resolution will be ~10 meV over the bulk of the detector face.

It should be mentioned, however, that this particular method of improving the overall resolution was most effective prior to recent modifications in the detector body design. Although our uncorrected resolution is now higher than before these modifications, this method presently results in less of an improvement in overall resolution. It is thought that perhaps the reason for the larger prior success was that it corrected primarily the same distortions ameliorated by the detector body modifications. The principal improvements in the detector body concerned reducing "edge effects", which are generally ascribed to inhomogeneities in the electric field near the edge of the detector body. These imperfections cause the acceleration of the electron cloud towards the anode to be skewed. This reason for this recent loss in efficiency will have to be investigated and confirmed, however, because it may indicate a change in importance among various resolution-degrading mechanisms, which should be addressed.

When examining O_2 dissociation data, and when the number of coincidence events permits, the highest resolution is obtained by selecting the region of the detector face where the position algorithm gives the best results; typically in the center of each half.

Once resolution on the order of 10 meV is achieved, however, the spin-orbit interaction that slightly splits the ground electronic state of the O atoms can no longer be neglected. Therefore we might expect to observe slightly different kinetic energy releases depending on the specific spin-orbit states formed by the two O atom products. As one example, careful measurement of the kinetic energy release distribution of the correlated O atoms following excitation to v'=7 in the $B\,^3\Sigma_u^-$ upper state allows the kinetic energy release corresponding to the correlated spin-orbit product states to be resolved, as shown

in Figure 4.3. Subsequent to this achievement, the correlated spin-orbit product state distributions of all of the $B\,^3\Sigma_u^-$ upper state vibrational levels from v' = 0 to v' = 11 have been determined.[1(b)]

Predissociation from the Schumann-Runge $B\,^3\Sigma_u^-(v')\leftarrow X\,^3\Sigma_g^-(v'')$ bands involves interaction of the $B\,^3\Sigma_u^-$ state with one of the four dissociative excited states of allowed symmetry known to intersect it, as shown in Figure 4.1. The pair of O atoms will correlate to one of the five symmetry- and parity-allowed correlated (j,j) spin-orbit energy asymptotes (the sixth possible combination of atomic limits, the (0,0) limit, is rigorously forbidden as it must correlate to a *gerade* molecular state). Relative populations of these correlated spin orbit states have never been determined before.[4] The systematic investigation of these distributions over the range of vibrational levels in the $B\,^3\Sigma_u^-$ upper state will help unravel the nature of the dissociative state(s) primarily responsible for the dissociation (i.e. which dissociative state(s) initially couple most strongly to the $B\,^3\Sigma_u^-$ state) as well as the subsequent interactions between various dissociative states as the interatomic distance grows.

Figure 4.4 shows the angular distribution of recoil vectors found in this experiment. Analysis of this angular distribution gives an anisotropy parameter β of 0.76 $\pm$ 0.03. While it is possible that this value is affected by residual saturation or power broadening, it is quite reproducible. An anisotropy parameter of 0.76 is also consistent

---

Figure 4.3 The translational energy release distribution $P(E_T)$ resulting from the predissociation of O_2 $B\,^3\Sigma_u^-$ (v'=7). The circles represent experimental data, and the solid line represents the results of a fit to these data. Every fourth point has an error bar, representing the 1σ uncertainty. The sticks denote the positions and intensities of the individual (j_1,j_2) limits as deduced from the fit.

Figure 4.4 The photofragment angular distribution $P(\theta)$. Also shown is the least squares fit to the experimental data, which yielded an anisotropy parameter of $\beta = 0.76 \pm 0.03$.

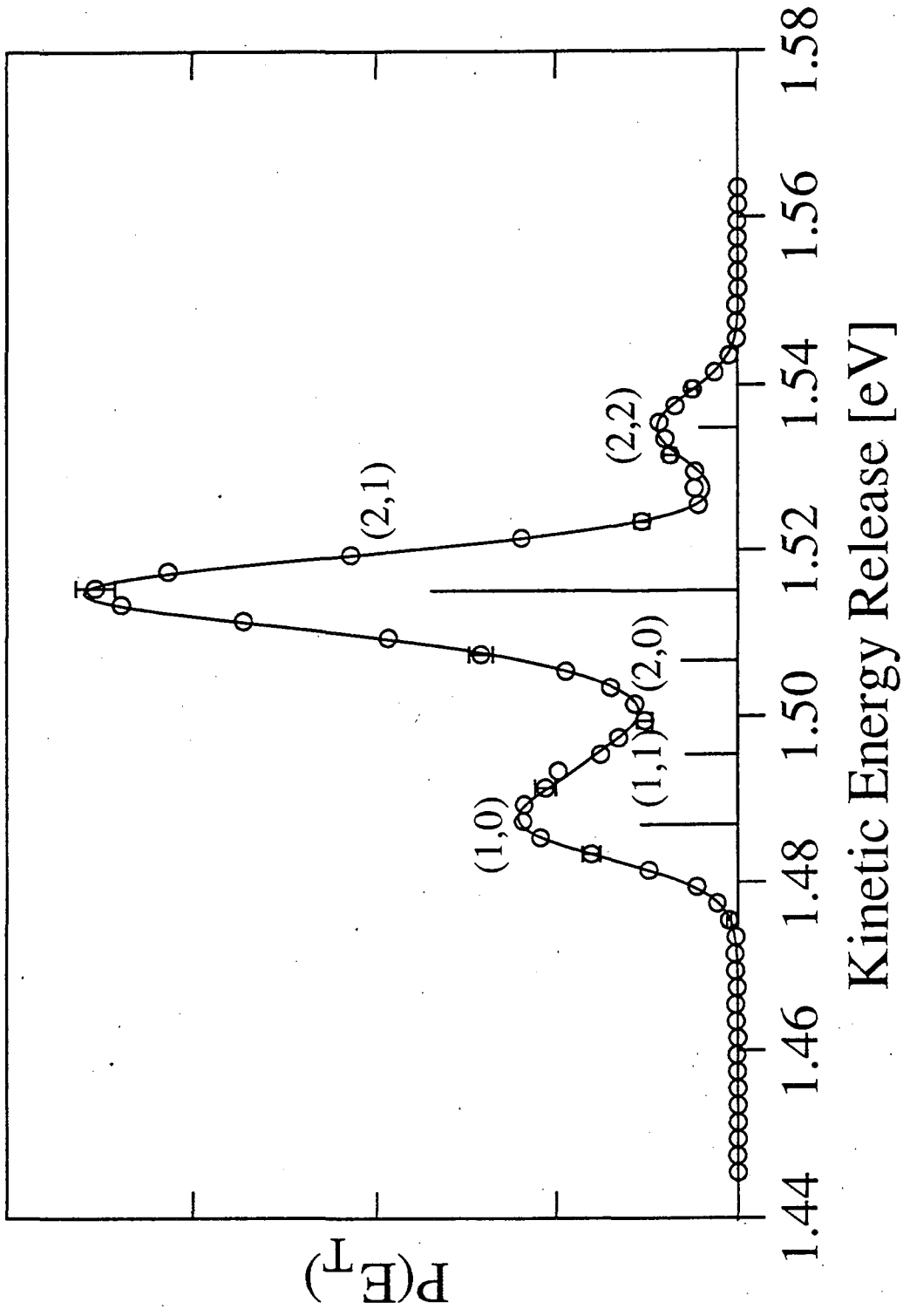

P(E_T)
1.44
1.46
1.48
1.50
1.52
1.54
1.56
1.58
Kinetic Energy Release [eV]
(1,0)
(1,1)
(2,0)
(2,1)
(2,2)

Figure 4.3

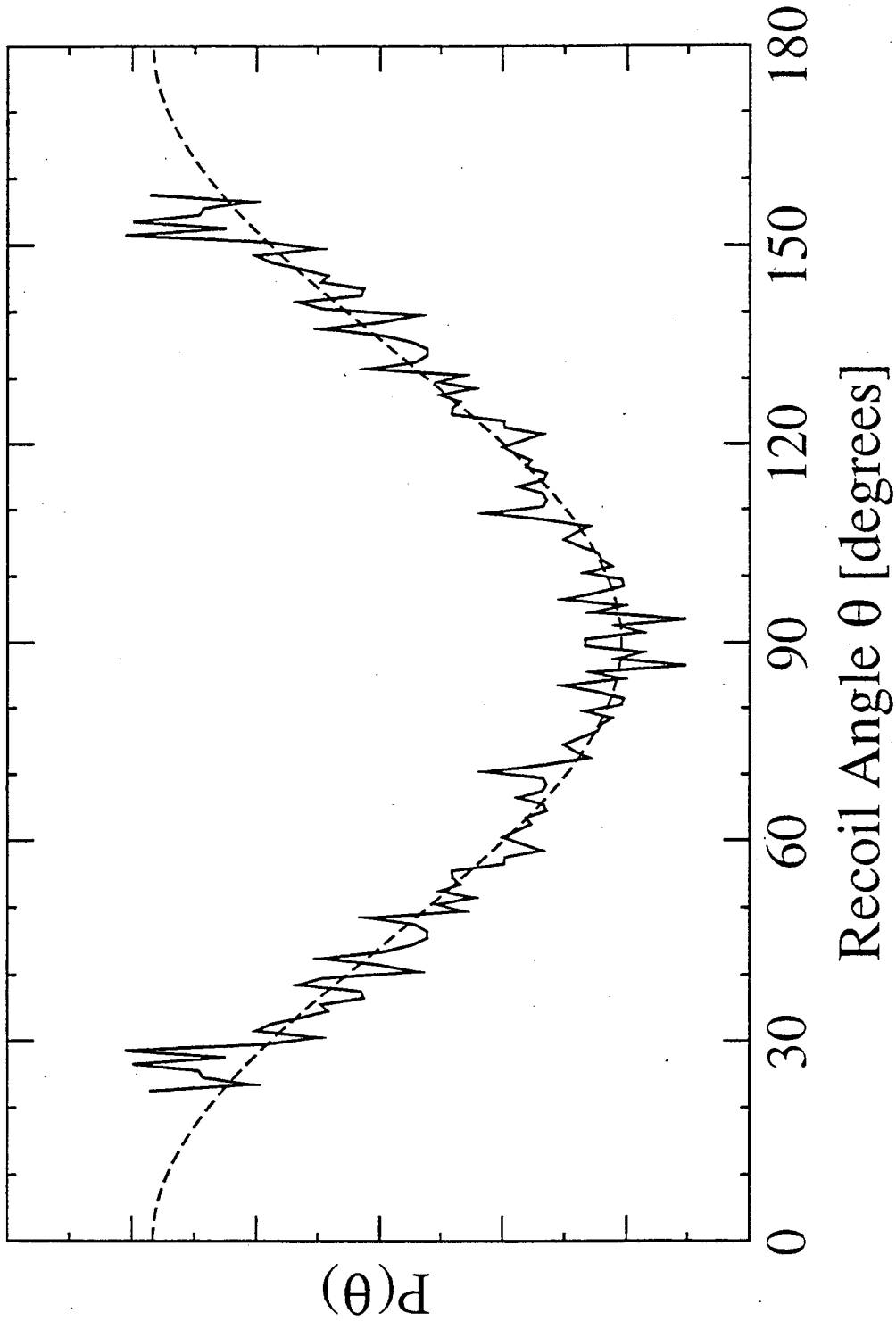


Figure 4.4

with the observed 2 cm^{-1} natural linewidth for the v'=7 level of the $B\,^3\Sigma_u^-$ state, since both measures imply a ~3 ps lifetime for the excited state. The rotation which occurs in this time (prior to dissociation) results in a reduction of the anisotropy parameter from the limiting value of +2 for a parallel transition (recall § 1.3.4).

4.3 Total Photodissociation Cross Sections of the N_3 Free Radical

The N_3 radical is, for our experiment, the prototypical polyatomic radical. The precursor anion, N_3^-, is easily made via dissociative attachment to benzyl azide ($C_6H_5CH_2N_3$). The detachment of this ion is discussed in § 1.4.2 as an example of a system where little geometry change occurs, resulting in $\Delta v=0$ transitions from the ion to the neutral. This in turn relaxes our detachment photon energy restrictions; we can thus detach using any convenient wavelength above threshold, which maximizes our detachment efficiency.

A diagram showing the energetics of the various N_3 electronic states in addition to dissociation product energy levels is shown in Figure 4.5. The N_3 radical was first studied spectroscopically in a series of flash photolysis absorption experiments.[5,6] These rotationally-resolved absorption studies identified the main absorption features at ~272 nm as arising from the $\tilde{B}\,^2\Sigma_u^+ \leftarrow \tilde{X}\,^2\Pi_g$ transition. They also showed that N_3 is a linear centrosymmetric molecule with little change in bond length on undergoing the $\tilde{B} \leftarrow \tilde{X}$ transition. Because the ground electronic state is degenerate and the geometry is linear, transitions originating from N_3 $\tilde{X}$ (ν_2=1) exhibit the expected Renner-Teller effect.[7,8] For our experiment, the relevant observation from these previous studies was the presence of diffuse bands reported at slightly shorter wavelength than the main $\tilde{B} \leftarrow \tilde{X}$ transitions. This strongly suggests predissociation is occurring from the upper states involved in these diffuse transitions. These bands were tentatively assigned to a $^2\Pi \leftarrow ^2\Pi$ electronic

---

Figure 4.5 The energetics of the N_3 system.

Energy [eV]
5
4
3
2
1
0
4.56
$N_3\,\tilde{B}\,^2\Sigma_u^+$
3.52
$N_2 + N(^2P)$
2.33
$N_2 + N(^2D)$
0
$N_3\,\tilde{X}\,^2\Pi_g$
−0.05
$N_2 + N(^4S)$

Figure 4.5

transition expected to be in this region.

The major motivation for obtaining the total photodissociation cross section of the N_3 radical was to once again map out and determine the extent of predissociation in these excited state(s). As will be shown below, the N_3 radical predissociates throughout the absorption bands due to the main $\tilde{B} \leftarrow \tilde{X}$ transitions in addition to the bands found at slightly higher energy.

N_2 or He at a pressure of 10 PSIG was bubbled through room temperature benzyl azide (Alfa) and this mixture was used to form N_3^- in the electron beam/pulsed free jet expansion source. A repetition rate of 50 Hz and a beam energy of 8 kV was used. The detachment excimer-pumped dye laser was operated at either 343 nm (3.62 eV) or 386 nm (3.21 eV), the peak of the dye curve intensity of PTP and BBQ (Exciton) respectively. This allowed the photodetachment of N_3^- well above the 2.68 eV threshold for this process. The frequency-doubled dissociation laser light had a 0.4 or 0.08 cm^{-1} bandwidth depending on whether an intracavity etalon was used, and was attenuated from ~2 mJ/pulse to 20-50 μJ/pulse to minimize power broadening in the rotationally resolved high-resolution spectra. Typically 300-500 laser shots per point were acquired and summed for total photodissociation cross section spectra. All other details are as described in § 2.7.5.

The low resolution survey scan of the N_3 predissociation spectrum is shown in Figure 4.6. The peaks are labeled according to the vibronic symmetries of the states involved. The two main peaks originate from the spin-orbit split $\tilde{X}\ ^2\Pi_g$ ground state to the excited $\tilde{B}\ ^2\Sigma_u^+$ state. Also labeled are sequence bands involving one quantum of bend excitation (and thus vibrational angular momentum) in both the ground and excited states. At first glance the vibrational distribution does appears not to be as cold as might be

---

Figure 4.6 Low resolution survey scan (0.05 nm steps, corrected to vacuum wavelengths) of N_3 photodissociation cross section. Transitions are labeled according to the vibronic symmetry of the states involved.

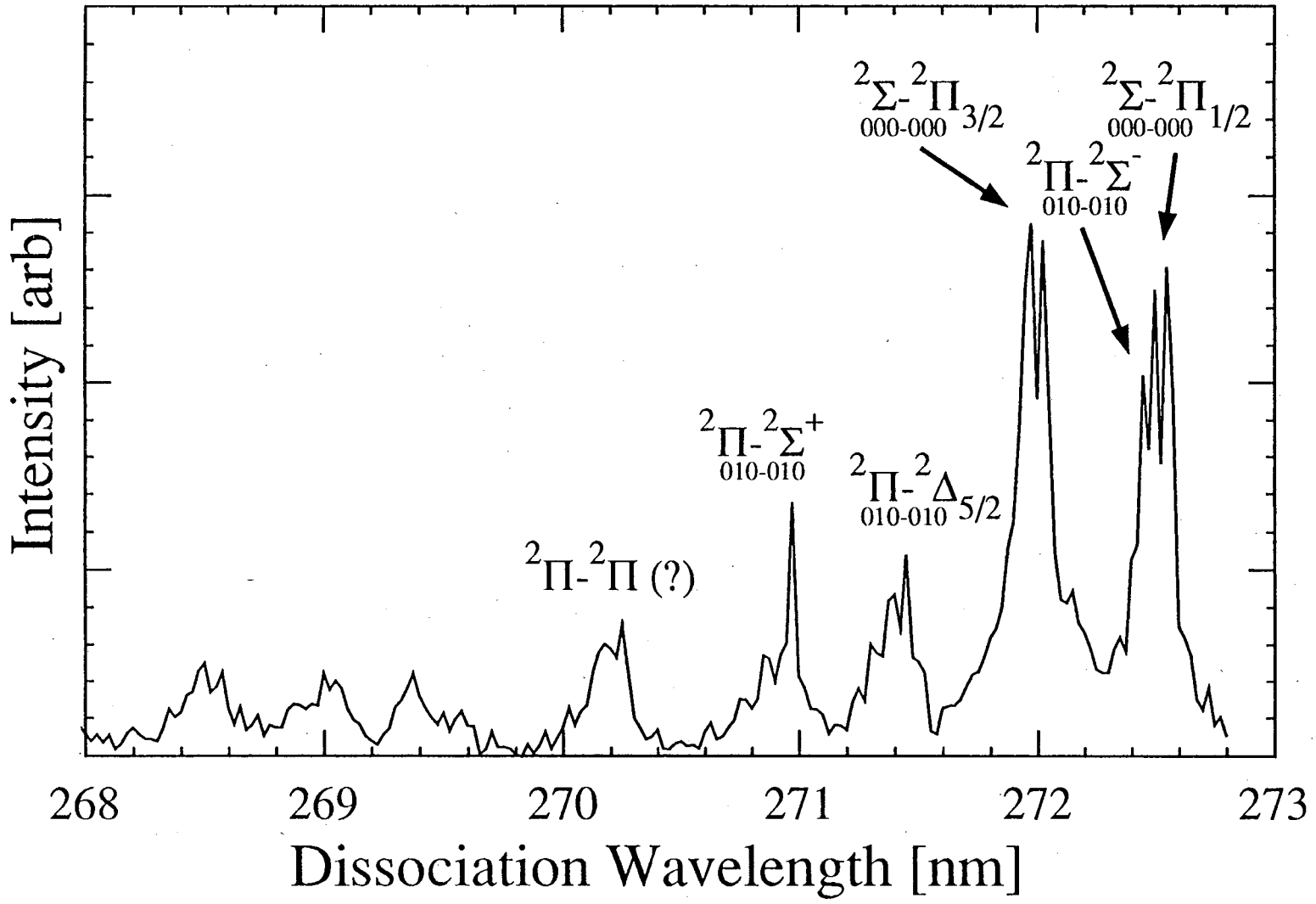

Intensity [arb]
Dissociation Wavelength [nm]
268
269
270
271
272
273
$^2\Pi$-$^2\Pi$ (?)
$^2\Pi$-$^2\Sigma^+$ 010-010
$^2\Pi$-$^2\Delta_{5/2}$ 010-010
$^2\Sigma$-$^2\Pi_{3/2}$ 000-000
$^2\Pi$-$^2\Sigma^-$ 010-010
$^2\Sigma$-$^2\Pi_{1/2}$ 000-000

Figure 4.6

expected. However, as will become apparent below, dynamical effects probably result in a greater quantum yield for dissociation from bend-excited upper state levels. In addition, no correction to the spectra for a possible variation in the branching ratios to the different N atom electronic states was made. A much larger fraction of those events which proceed to form the ground state (but spin-forbidden) N (^4S) atom channel will release enough kinetic energy so that the N atom recoils out past the detector than will the lowest energy spin-allowed N (^2D) atom channel. This will have an effect on the apparent photodissociation cross section intensity if the branching ratios to the different product channels vary with excited state vibrational level.

The exact nature of the bands below 270.5 nm remains unidentified. Whereas in the past there was some uncertainty due to the presence of other possible signal carriers, the mass selection feature of this experiment means they are now at least firmly associated with the N_3 radical.

The high resolution spectrum of the $\tilde{B}\ ^2\Sigma_u^+ \leftarrow \tilde{X}\ ^2\Pi_{3/2}$ sub-band is shown in Figure 4.7, accompanied by a simulation of the spectrum. The simulation was constructed using the $^2\Sigma \leftarrow ^2\Pi$ transition strengths for a diatomic molecule[9,10] and the ground and excited state constants of Douglas and Jones.[6] The peak intensities show the characteristic 2:1 nuclear spin statistics expected for N_3. The spectrum is well-characterized by a Boltzmann temperature of 50 K, confirming that rotational cooling is achieved in our source. The slightly diminished intensity of the middle portion of the spectrum relative to the simulation is likely the signature of residual saturation, although dynamical effects cannot be ruled out. Linewidths are instrument limited to 0.11 cm^{-1}, with the laser

---

Figure 4.7 Upper panel: High resolution etalon scan (0.0002 nm steps, corrected to vacuum wavelengths) of the $^2\Sigma_u^+ \leftarrow ^2\Pi_{3/2}$ sub-band origin. Rotational branches are labeled using combs.
Lower panel: Simulation of the spectrum using an assumed rotational temperature of 50 K.

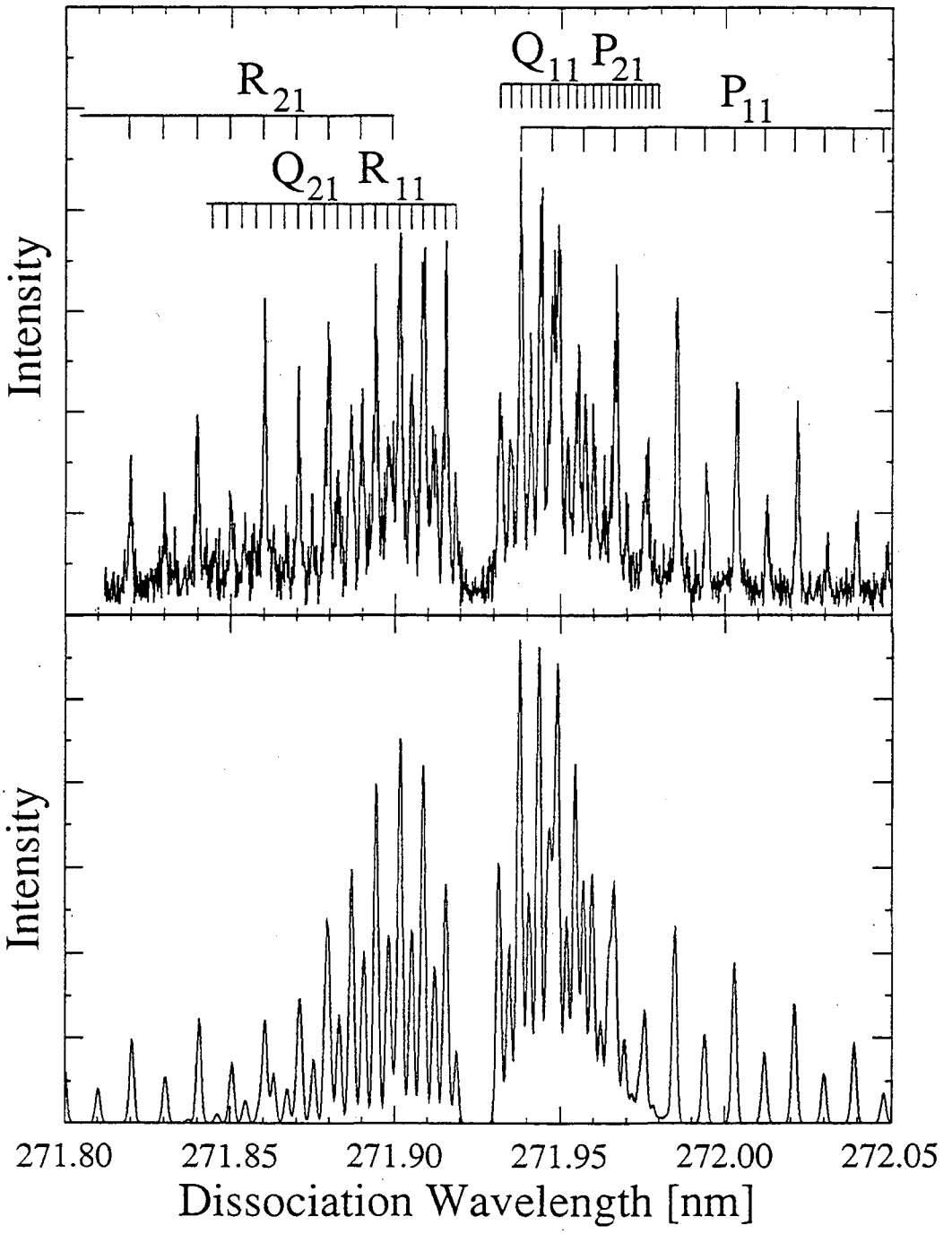

R21
Q11 P21
P11
Q21 R11
Intensity
Intensity
271.80
271.85
271.90
271.95
272.00
272.05
Dissociation Wavelength [nm]

Figure 4.7

bandwidth and Doppler broadening making the largest contributions to this value.

Once again, the primary result of the total photodissociation cross section studies of N_3 is not the spectroscopy, which was known at least as well from absorption studies, but rather the fact that predissociation occurs throughout the first electronic absorption band. This system is also a good test of the ultimate resolution of the apparatus for total photodissociation cross section experiments, because of the relatively narrow rotational linewidths. Armed with knowledge of the total photodissociation cross sections, it is possible to perform a detailed study of the translational energy and recoil angle distributions to determine the product electronic and vibrational branching ratios as a function of $\tilde{B}\ ^2\Sigma_u^+$ state vibrational excitation. This is the subject of the following section.

4.4 TPS Detection of N_3 Radical Dissociation:

'Mode Specific' Effects in the N Atom Electronic State Branching Ratio

The dissociation dynamics of the $\tilde{B}\ ^2\Sigma_u^+$ state of N_3 were then investigated using fast radical beam photodissociation coupled with the time- and position-sensitive detection technique described in previous chapters. Detailed photofragment kinetic energy and angular distributions were obtained after excitation to both the $(\nu_1,\nu_2,\nu_3) = (000)$ and (010) vibrational levels of the $\tilde{B}$ state. The branching ratio to the various N atom product channels varied quite strongly depending if one quantum of bend excitation was present in the upper state.

As was shown in the previous section, the N_3 radical undergoes predissociation following excitation to all accessible levels of the $\tilde{B}\ ^2\Sigma_u^+$ state. At the excitation energies required to access the $\tilde{B}\ ^2\Sigma_u^+$ state three different N atom electronic states can be produced in conjunction with ground electronic state N_2 (the first electronic excited state of N_2 is inaccessible at these energies). Motivation for this detailed study of the disposal of energy in the photodissociation products of the N_3 radical was threefold. First, it is of interest to determine the importance of the lowest energy, but spin-forbidden, $N(^4S) + N_2$

channel. Second, a comparison of dissociation product channel branching ratios depending on whether one quantum of vibration is present in the upper state may illuminate any dynamical effects as dissociation progresses. Finally, as explained in § 1.3.2, this technique results in a very direct measure of the bond dissociation energy from which an accurate heat of formation for the N_3 radical can be calculated.

Figure 4.8 presents the translational energy distribution of the photofragments formed upon excitation to the (000) level of the $\tilde{B}$ state, while Figure 4.9 presents the translational energy distribution of the photofragments formed upon excitation to the (010) level of the $\tilde{B}$ state. Both figures are labeled according to the best fit for the origins of the various electronic and vibrational levels. Both distributions share certain characteristics. The translational energy distributions extend out to much higher kinetic energy release than possible for the spin-allowed channels, indicating the presence of spin-forbidden products. The most intense portion of both distributions is within the lower vibrational levels of the first spin-allowed $N(^2D) + N_2$ channel. Both distributions indicate inverted and/or long vibrational distributions and peak rotational intensities well away from the vibrational origins. The most striking difference between the two translational energy distributions is the drastic change in the N atom (^2D):(^4S) branching ratio.

The anisotropy parameter that describes the photofragment recoil angular distributions following excitation to each of the (000) and (010) vibrational levels in the $\tilde{B}$

Figure 4.8 Center-of-mass translational energy distribution $P(E_T)$ obtained following excitation to the (000) level of the $\tilde{B}\ ^2\Sigma_u^+$ state.

Figure 4.9 Center-of-mass translational energy distribution $P(E_T)$ obtained following excitation to the (010) level of the $\tilde{B}\ ^2\Sigma_u^+$ state.

Figure 4.10 The anisotropy parameter $\beta(E_T)$ for N_3. The solid line with the diamonds shows the results for the (000) band while the dashed line with circles shows the (010) results. Error bars are 1σ. Vertical dashed arrows indicate the energetic thresholds of the channels involving ^2D and ^2P N atoms from the 000 level of the $\tilde{B}\ ^2\Sigma_u^+$ state.

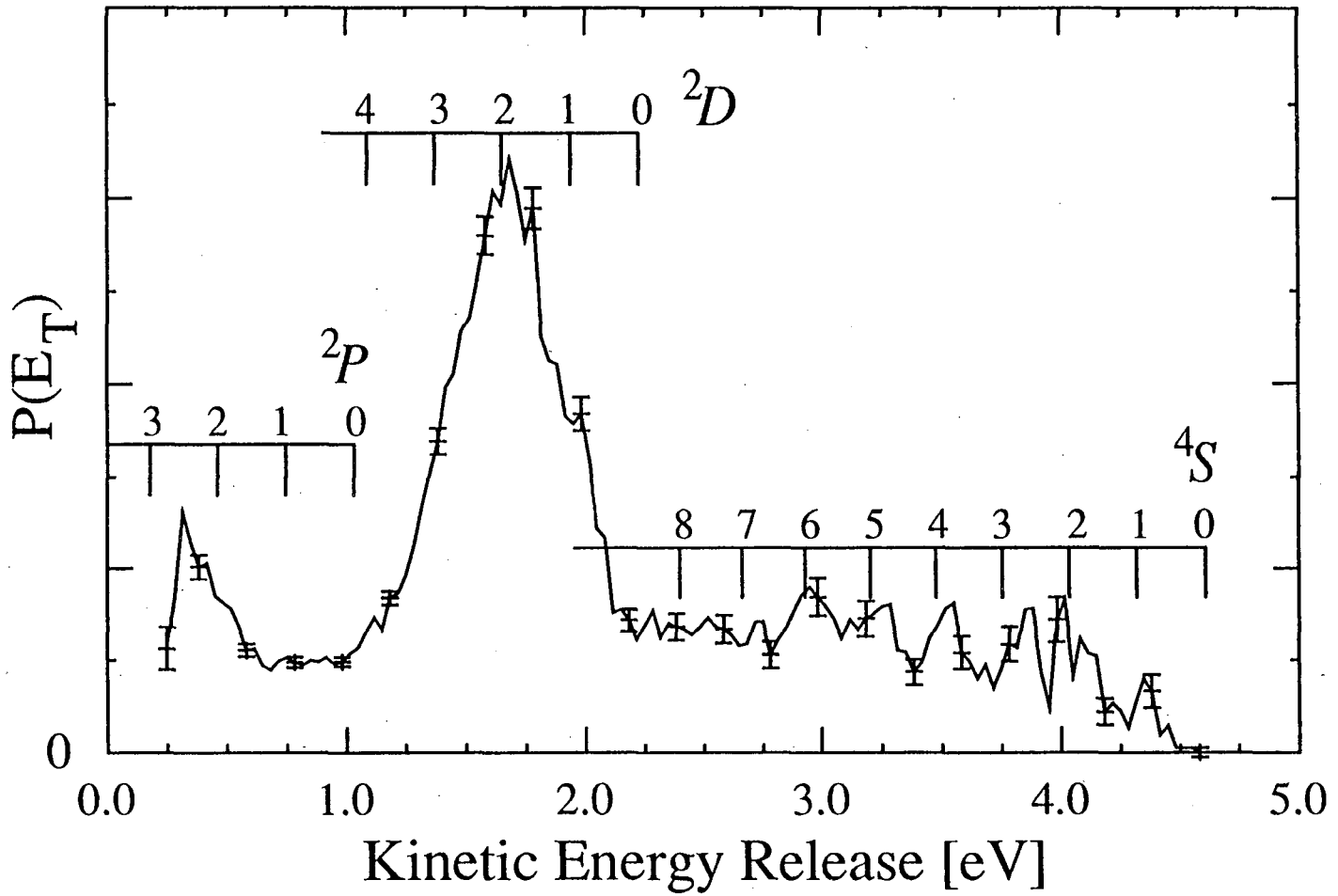

4 3 2 1 0 ^2D
^2P
3 2 1 0
^4S
8 7 6 5 4 3 2 1 0
P(E$_T$)
0
0.0
1.0
2.0
3.0
4.0
5.0
Kinetic Energy Release [eV]

Figure 4.8

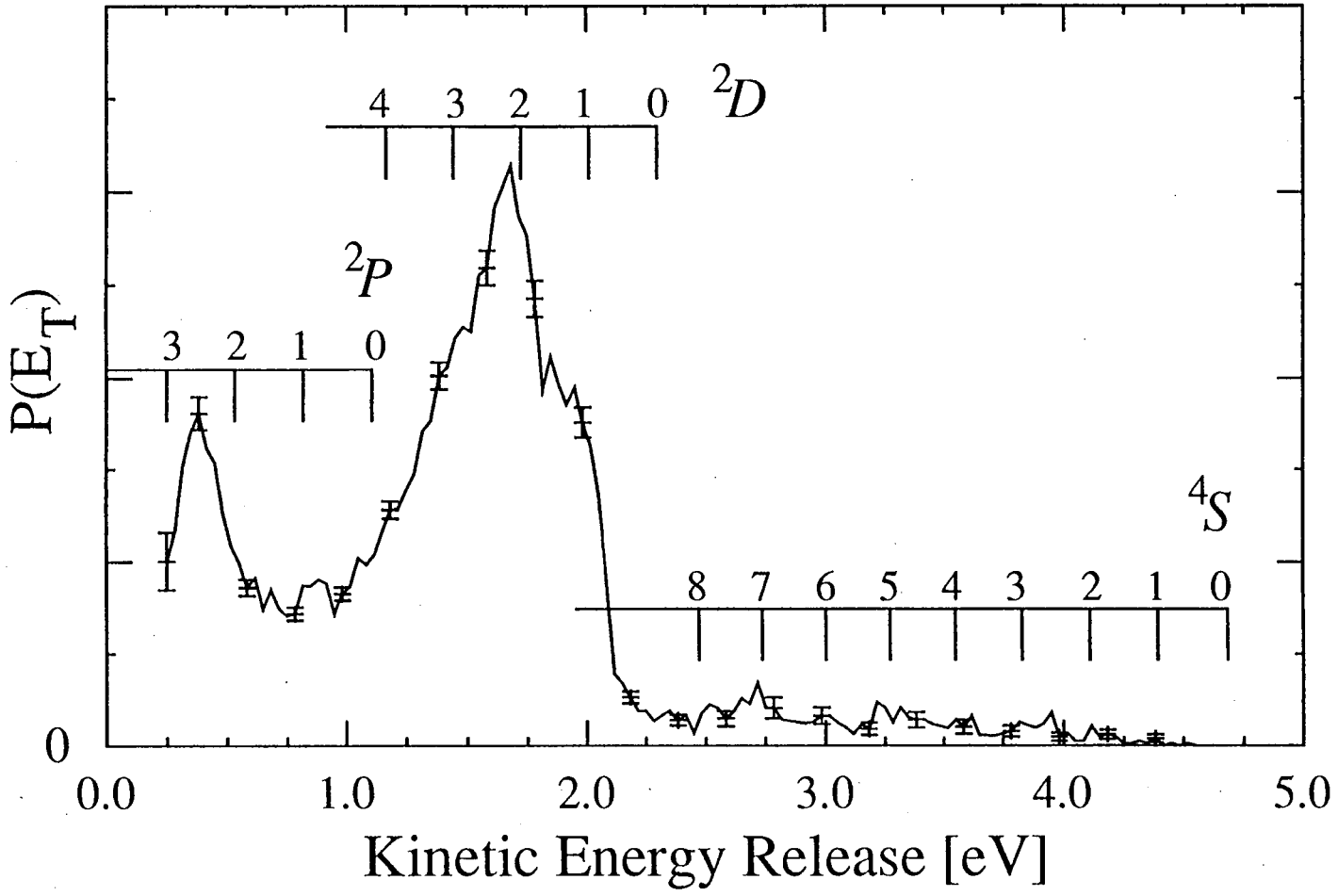

4 3 2 1 0
^2D
^2P
3 2 1 0
^4S
8 7 6 5 4 3 2 1 0
P(E$_T$)
0
0.0
1.0
2.0
3.0
4.0
5.0
Kinetic Energy Release [eV]

Figure 4.9

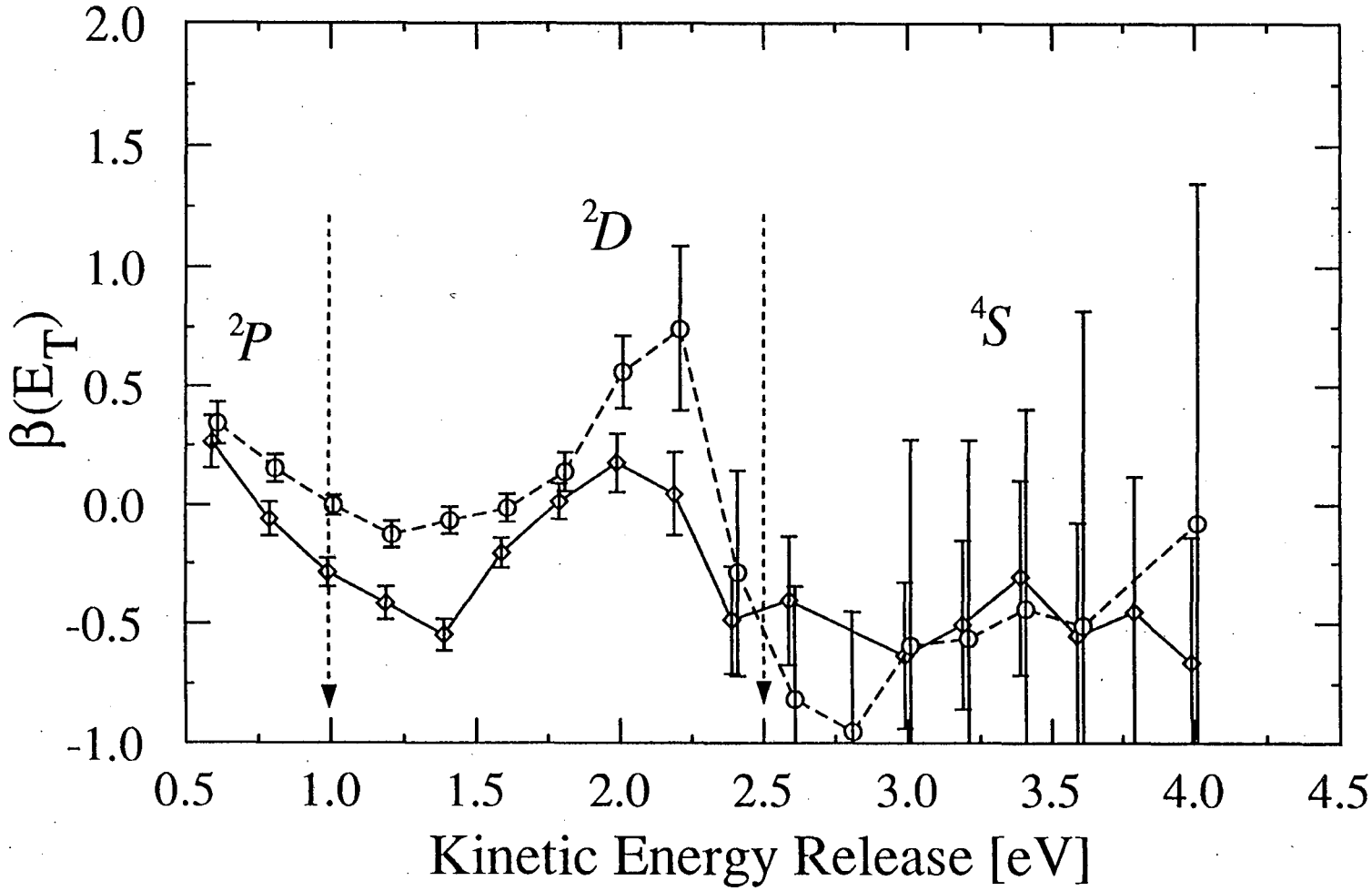

2.0
1.5
1.0
0.5
0.0
-0.5
-1.0
β(E_T)
²P
²D
⁴S
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
Kinetic Energy Release [eV]

Figure 4.10

state is plotted as a function of kinetic energy release in Figure 4.10. A change in the value of β can be observed if a comparison is made of the regions where each N atom electronic state dominates the translational energy distribution.

The conclusions which can be drawn about the dissociation of N_3 from these results are many. Briefly, it is immediately apparent from Figures 4.8 and 4.9 that N_3 dissociates to both spin-allowed and spin-forbidden products, despite the fact that at all times two spin-allowed channels are energetically accessible. As will be seen in Chapter 5, this is in contrast to the behavior of the NCO radical which dissociates predominantly to spin-allowed products once above the energetic threshold for their formation. There is, however, a dramatic change in the N atom (^2D):(^4S) branching ratio with the addition of one quantum of bending excitation in the upper state. This, coupled with the high rotational excitation of the N_2 products ($J_{peak}\approx32$) required to simulate the observed translational energy distributions, is strong evidence for the participation of bent geometries in the dissociation from the $\tilde{B}$ state. The values of the energy dependent anisotropy parameter are also largely consistent with a bent geometry during dissociation, particularly in the channel producing N (^2D) atoms where β becomes positive. For a perpendicular transition such as the $\tilde{B}\,^2\Sigma_u^+ \leftarrow \tilde{X}\,^2\Pi$ excitation studied here, a purely axial recoil would result in a negative anisotropy parameter.

The thermochemistry of N_3 is determined in a very direct way by these experiments, independently confirming that the radical is 0.05 kcal/mol endoergic with respect to N(^4S) + N_2, as reflected by the labels on Figure 4.5. This value is in good agreement with that determined by Brauman and co-workers after performing proton affinity studies[11] of N_3^- and electron affinity studies[12,13] of N_3.

4.5 References

[1] (a) D. J. Leahy, D. R. Cyr, D. L. Osborn and D. M. Neumark, Chem. Phys. Lett. (in press).
(b) Full details of all O_2 dissociation work will forthcoming in a full publication by D. J. Leahy, D. R. Cyr, D. L. Osborn and D. M. Neumark.

[2] R. E. Continetti, D. R. Cyr, R. B. Metz, and D. M. Neumark, Chem. Phys. Lett. **182**, 406 (1991); R. E. Continetti, D. R. Cyr, D. L. Osborn, D. J. Leahy, and D. M. Neumark, J. Chem. Phys. **99**, 2616 (1993).

[3] B. R. Lewis, J. H. Carver, T. I. Hobbs, D. G. McCoy, and H. P. F. Gies, J. Quant. Spectrosc. Rad. Trans. **20**, 191 (1978); **22**, 213 (1973); H. P F. Gies, S. T. Gibson, D. G. McCoy, A. J. Blake, and B. R. Lewis, *ibid*, **26**, 469 (1981); B. R. Lewis, L. Berzins, J. H. Carver, and S. T. Gibson, J. Quant. Spectrosc. Rad. Trans. **36**, 187 (1986); A. S-C. Cheung, K. Yoshino, D. E. Freeman, R. S. Friedman, A. Dalgarno, and W. H. Parkinson, J. Mol. Spectrosc. **134**, 362 (1989); A. S-C. Cheung, K. Yoshino, J. R. Esmond, S. S-L. Chiu, D. E. Freeman, and W. H. Parkinson, J. Chem. Phys. **92**, 842 (1990); S. S-L. Chiu, A. S-C. Cheung, K. Yoshino, J. R. Esmond, D. E. Freeman, and W. H. Parkinson, J. Chem. Phys. **93**, 5539 (1990).

[4] Indeed, only one other recent experiment has measured correlated fine-structure state distributions without a much larger spin-orbit interaction. This was done in the photodissociation of NO_2, by U. Robra, H. Zacharias, and K. H. Welge, Z. Phys. D **16**, 175 (1990).

[5] B. A. Thrush, Proc. R. Soc. London, Ser. A **235**, 143 (1956).

[6] A. E. Douglas and W. J. Jones, Can. J. Phys. **43**, 2216 (1965).

[7] J. A. Pople, Mol. Phys. **3**, 16 (1960).

[8] J. T. Hougen, J. Chem. Phys. **36**, 549 (1962).

[9] L. T. Earls, Phys. Rev. **48**, 423 (1935).

[10] I. Kovacs, *Rotational structure in the spectra of diatomic molecules*, (Elsevier, Amsterdam, 1969) pp. 125-131.

[11] M. J. Pellerite, R. L. Jackson, and J. I. Brauman, J. Phys. Chem. **85**, 1624 (1981).

[12] R. L. Jackson, M. J. Pellerite, and J. I. Brauman, J. Am. Chem. Soc. **103**, 1802 (1981).

[13] E. Illenberger, P. B. Comita, J. I. Brauman, H.-P. Fenzlaff, M. Heni, N. Heinrich, W. Koch, and G. Frunking, Ber. Bunsenges. Phys. Chem. **89**, 1026 (1985).

Chapter 5

Photodissociation of the NCO Radical

5.1 Introduction

During the past ten to 15 years, a wide variety of state-selected methods have been developed to study the photodissociation of stable, closed-shell molecules.[1] Ideally, these experiments enable one to map out the dissociative electronic states in the molecule, find the energy needed to break the dissociating chemical bond, and determine in detail the interactions between the departing photofragments. It is clearly of interest to extend the methods of photodissociation to the study of reactive open-shell radicals. At present, however, even the bond dissociation energies, let alone the detailed interactions among the excited electronic states, are poorly characterized in many of these species. Thus far, successful photodissociation experiments have been performed on only a handful of reactive free radicals,[2,3,4] largely because of the experimental difficulties involved in preparing these species free of contaminants and in reasonably well-defined quantum states. Recently, we have demonstrated that these problems can be overcome by preparing the radical of interest via photodetachment of a mass-selected negative ion beam.[5] This paper describes the application of this method to the photodissociation of the NCO radical.

The NCO radical has attracted considerable interest over the years both for its role in combustion chemistry and because of its spectroscopic complexity. NCO is believed to be an intermediate in the combustion of nitrogen containing compounds.[6] It has been observed in CH_4/N_2O flames[7] and is known to be the primary product of the $CN + O_2$ reaction.[8] The spectroscopy of the ground and excited states of NCO is also of interest. The electronic transitions from the ground $\tilde{X}\ ^2\Pi$ state to the low-lying $\tilde{A}\ ^2\Sigma^+$ and $\tilde{B}\ ^2\Pi$ excited states were first observed in 1958[9] and analyzed in detail by Dixon[10,11] in 1960;

the $\tilde{A}\ ^2\Sigma^+ \leftarrow \tilde{X}\ ^2\Pi$ and $\tilde{B}\ ^2\Pi \leftarrow \tilde{X}\ ^2\Pi$ origins are at 440 nm and 315 nm, respectively. Dixon's work on the $\tilde{A}\ ^2\Sigma^+ \leftarrow \tilde{X}\ ^2\Pi$ system was followed by several optical absorption[12,13] and laser-induced fluorescence[7a,7c,14,15,16,17] studies. The electronic ground state has been investigated using infrared,[12] electron paramagnetic resonance,[18] microwave[19] and far infrared laser magnetic resonance[20] spectroscopy. The detailed characterization of the NCO ground state and the $\tilde{A}\ ^2\Sigma^+ \leftarrow \tilde{X}\ ^2\Pi$ transition has in turn led to a better understanding of the CN + O_2 reaction; several groups have measured vibrational, rotational, and fine structure distributions of NCO produced by this reaction using laser-induced fluorescence of the $\tilde{A}\ ^2\Sigma^+ \leftarrow \tilde{X}\ ^2\Pi$ transition.[21]

The results presented here provide new insight into the $\tilde{B}\ ^2\Pi \leftarrow \tilde{X}\ ^2\Pi$ system of NCO which, in contrast to the $\tilde{A}\ ^2\Sigma^+ \leftarrow \tilde{X}\ ^2\Pi$ system, remains largely unassigned. This is due to Renner-Teller and Fermi resonance interactions within the $\tilde{B}\ ^2\Pi$ state, as well as extensive perturbations of this state by the nearby lower-lying $\tilde{A}\ ^2\Sigma^+$ state and predissociation to N + CO. In fact, one of the key unresolved issues with respect to the $\tilde{B}\ ^2\Pi$ state, and certainly the one which motivated the current study, is the identification of the onset of predissociation from the $\tilde{B}\ ^2\Pi$ state and the characterization of the dissociation products. Dixon[11] originally observed spectral broadening beginning at the transition to the $\nu_1 = 2, \nu_2 = 0, \nu_3 = 0$, or (200), level at 33,700 cm^{-1}. This broadening was attributed to predissociation resulting in the spin-allowed N(^2D) + CO products, thereby giving an upper bound of 40 kcal/mol for D_0(N-CO), the bond dissociation energy of NCO to form ground state (but spin-forbidden) N(^4S) + CO products. The corresponding heat of formation is $\Delta_f H^0_{298\,K}(NCO) = 45$ kcal/mol. [D_0(N-CO) = 85.3 kcal/mol $[=\Delta_f H^0_{298\,K}(N)+\Delta_f H^0_{298\,K}(CO)] - \Delta_f H^0_{298\,K}(NCO)$, see Discussion.] In a subsequent vacuum ultraviolet photolysis study of HNCO in which the threshold for production of the NCO $\tilde{A}\ ^2\Sigma^+$ state was measured, Okabe[22] determined $\Delta_f H^0_{298\,K}(NCO)$ to be 37 kcal/mol. This implies that predissociation to N(^2D) + CO can occur only above the (400) level of

the $\tilde{B}\ ^2\Pi$ state. Okabe's value for $\Delta_f H^0_{298\,K}(NCO)$ is consistent with that obtained by Coombe[2] in a recent study of NCO photodissociation at 193 nm. Crosley[23] found that the fluorescence lifetime of the $\tilde{B}\ ^2\Pi$ state was dramatically shorter for the (100) vibrational level ($\leq$ 10 ns) than for the (000) level (63 ns). This was interpreted to mean that the spin-allowed N(^2D) + CO threshold occurs between the (000) and (100) levels of the $\tilde{B}\ ^2\Pi$ state, yielding D_0(N-CO) $\leq$ 39 kcal/mol and $\Delta_f H^0_{298\,K}(NCO) \geq 48$ kcal/mol.

The observation that the CN + $O_2 \rightarrow$ NCO + O reaction occurs with no appreciable activation barrier[24] means that $\Delta_f H^0_{298\,K}(NCO) \leq 44$ kcal/mol, and this, in conjunction with the considerable NCO bending excitation produced by the CN + O_2 reaction[21b,c] lends support to Okabe's value for $\Delta_f H^0_{298\,K}(NCO)$. However, the extraction of $\Delta_f H^0_{298\,K}(NCO)$ from his measurement uses the heat of formation of HNCO, and this in turn requires an accurate value for $\Delta_f H^0_{298\,K}(NH)$, for which the spread in recently reported values is several kcal/mol. The most recent determination of $\Delta_f H^0_{298\,K}(HNCO)$ by Chandler,[25] when combined with Okabe's work, yields a revised value of 36.1 kcal/mol for $\Delta_f H^0_{298\,K}(NCO)$; this is the currently accepted literature value.[26] However, an independent determination of $\Delta_f H^0_{298\,K}(NCO)$ from a direct measurement of the N-CO bond dissociation energy is desirable; an accurate value is vital in modeling reaction schemes involving the NCO radical. In addition, one would like to understand the origin of the widely disparate values discussed above.

The NCO photodissociation study reported here addresses all of these issues. We have investigated the $\tilde{B}\ ^2\Pi \leftarrow \tilde{X}\ ^2\Pi$ electronic transition between 316 and 234 nm. The salient results are as follows. We show unambiguously that *all* of the vibrational levels of the NCO $\tilde{B}\ ^2\Pi$ state dissociate. In addition, translational energy measurements on the photofragments show that the spin-allowed N(^2D) + CO channel is accessible only for vibrational levels higher than the (600) level in the $\tilde{B}\ ^2\Pi$ state (greater than 20.3 kcal/mol above the electronic origin), while the lower vibrational levels undergo predissociation to

the spin-forbidden N(^4S) + CO channel. We believe that it is this spin-forbidden dissociation which led to the high values of $\Delta_f H^0_{298\,K}$(NCO) inferred by Dixon[11] and Crosley.[23] Finally, our photofragment kinetic energy release measurements near the N(^2D) + CO threshold yield a value of 30.5 ± 1 kcal/mol for $\Delta_f H^0_{298\,K}$(NCO).

5.2 Experimental Approach

Our experiments are based on the idea that, since free radicals typically have positive electron affinities, one can generate a very clean source of radicals by laser photodetachment of a mass-selected beam of the precursor negative ion. The resulting radical beam is then dissociated with a second laser. Since the initial kinetic energy of the ion beam is 8 keV, the fragments resulting from photodissociation of the radical have high kinetic energies in the laboratory frame of reference and can be directly detected with high (~50%) efficiency using a microchannel plate detector. The overall experiment in the case of NCO photodissociation is

$$NCO^- \xrightarrow{h\nu_1} NCO + e^- \xrightarrow{h\nu_2} N + CO.$$

The higher energy O + CN channel is not accessible at the dissociation wavelengths used in this study.

Two types of experiments can be performed. In photodissociation cross section measurements, we determine the total photofragment signal as a function of $h\nu_2$ and thereby map out the dissociative electronic transitions of the NCO radical. In the case of the $\tilde{B}\ ^2\Pi \leftarrow \tilde{X}\ ^2\Pi$ transition in NCO, one expects a highly structured spectrum since the $\tilde{B}\ ^2\Pi$ state undergoes predissociation rather than direct dissociation. Alternatively, at fixed $h\nu_2$, we can measure the photofragment time-of-flight (TOF) distribution at the

detector and determine the maximum kinetic energy release and (more approximately) the translational energy distribution of the fragments.

The fast radical beam photodissociation spectrometer used in this work is shown in Figure 2.1. The operation of this instrument is described in detail in Chapter 2, so consequently only the most pertinent features will be reviewed here as the details unique to the study of NCO are presented.

Negative ions are generated using the pulsed molecular beam/electron gun source developed by Lineberger and coworkers[27] and used in several of our other investigations.[28] To make NCO^-, neon (10 psig) is bubbled through benzyl isocyanate [Pfaltz & Bauer] at room temperature and the resulting gas mixture is expanded through a piezoelectric pulsed molecular beam valve[29] operating at a 50 Hz repetition rate. A 1 keV electron beam intersects the free jet expansion downstream of the valve orifice, forming NCO^- by dissociative attachment of an electron to the benzyl isocyanate. (A similar scheme was first used by Illenberger et al. to generate N_3^- from benzyl azide.[30]) Since the anions are formed in the continuum flow region of the free jet expansion, they experience significant vibrational and rotational cooling.

The ions pass through a skimmer and are accelerated to 8 keV. They then pass through a pulsed high voltage switch[31] which enables us to run an 8 keV ion beam through the instrument while maintaining the source and detector regions at ground potential. The ions are then mass-selected using a coaxial, beam-modulation TOF mass spectrometer;[32] this design induces a negligible kinetic energy spread in the ion beam, an important and desirable feature for the photofragment TOF measurements described below.

The ion beam passes through a 1 mm diameter aperture into the photodetachment region. Here, the ions are crossed by an excimer-pumped dye laser beam (Lambda Physik LPX 210i and FL3002, PTP dye from Exciton) timed to intercept only the ion mass of interest and operating at 339 nm (nearly the peak of the PTP dye curve). The

photodetachment wavelength is chosen to allow production of only the ground vibrational state of the neutral free radical, as discussed in § 1.4.2. The photoelectron spectrum[33] of NCO^- indicates that at 339 nm, no excited vibrational states of the neutral free radical will be formed from the ground vibrational state of the ion, so the vibrational distribution attained during the supersonic expansion is preserved. Photodetached electrons are collected and detected using a microchannel plate (MCP) detector; the resulting signal is stored for subsequent normalization of the photodissociation spectrum. Any undetached anions are deflected out of the beam, leaving a mass-selected, fast beam of neutral free radicals. These pass through another 1 mm collimating aperture and enter the photodissociation and detection region.

In this region, the radicals are crossed with the laser beam from a second excimer-pumped dye laser system, the output of which is doubled using the appropriate doubling crystal. The dyes used to cover the broad range of dissociation wavelengths (from 632 to 468 nm in the fundamental wavelength) were R640, R6G, R610, R590, C540A, C503 and C480 (Exciton). Laser pulse energies were typically 1-2 mJ after frequency-doubling, and the linewidth of the doubled light was 0.4 cm^{-1}. This is what determines the resolution of our spectra; the Doppler broadening in our experiment is only 0.02 cm^{-1}. Depending on the arrangement of right angle prisms used to direct the laser beam into the spectrometer, the laser polarization direction could be made either parallel or perpendicular to the radical beam axis.

The resulting photofragments are detected by a 40 mm diameter MCP which can be placed either 67.8 or 101.1 cm downstream of the photodissociation region. Any neutrals that strike this detector have sufficient laboratory kinetic energy to be detected with high efficiency. The center of this detector is shielded by a 3.0 mm wide beam block to prevent undissociated NCO radicals from impinging on the detector face, while the

N + CO fragments typically have enough center-of-mass translational energy to miss the beam block but not so much energy that they miss the detector entirely.
In photodissociation cross section measurements, the photofragment yield is determined as a function of the dissociation dye laser wavelength. At each wavelength, the MCP signal is measured with an analog-to-digital converter interfaced to a microcomputer. A step size of 0.02 nm, with 500-1000 shots per point is typically used to gather a survey scan, which provides sufficient resolution for the observation of rotational contours of NCO bands. Where strong photodissociation signal is observed, scans were taken with a step size of 0.001 nm and a laser pulse energy of 100 μJ to minimize power broadening.

In photofragment TOF experiments, the photodissociation laser wavelength is fixed while the photofragment MCP signal is collected and summed over 1–4 × 10^4 laser shots by a 200 MHz LeCroy transient digitizer interfaced to a microcomputer. TOF spectra were taken with both parallel and perpendicular laser polarization orientations at several wavelengths in order to determine the photofragment energy and angular distribution. For comparison, TOF spectra of the undissociated parent radical were obtained by blocking the dissociation laser and translating the MCP detector so that the radical beam missed the beam block and hit the detector.

5.3 Results

Figure 5.1 shows the compilation of our photodissociation cross section measurements (0.02 nm step size) on the $\tilde{B}\ ^2\Pi \leftarrow \tilde{X}\ ^2\Pi$ system over a total range of 82 nm, from 316 to 234 nm. These measurements cover a wider wavelength range than

---

Figure 5.1 Compilation of survey scans (0.02 nm step size) of the NCO $\tilde{B}\ ^2\Pi \leftarrow \tilde{X}\ ^2\Pi$ photodissociation cross section. Progressions in the ν_1 vibration and the ν_3 vibration are indicated. The spectrum shows that the lowest levels of the $\tilde{B}\ ^2\Pi$ state undergo predissociation. Asterisks (*) illustrate dissociation wavelengths where TOF data presented below were obtained.

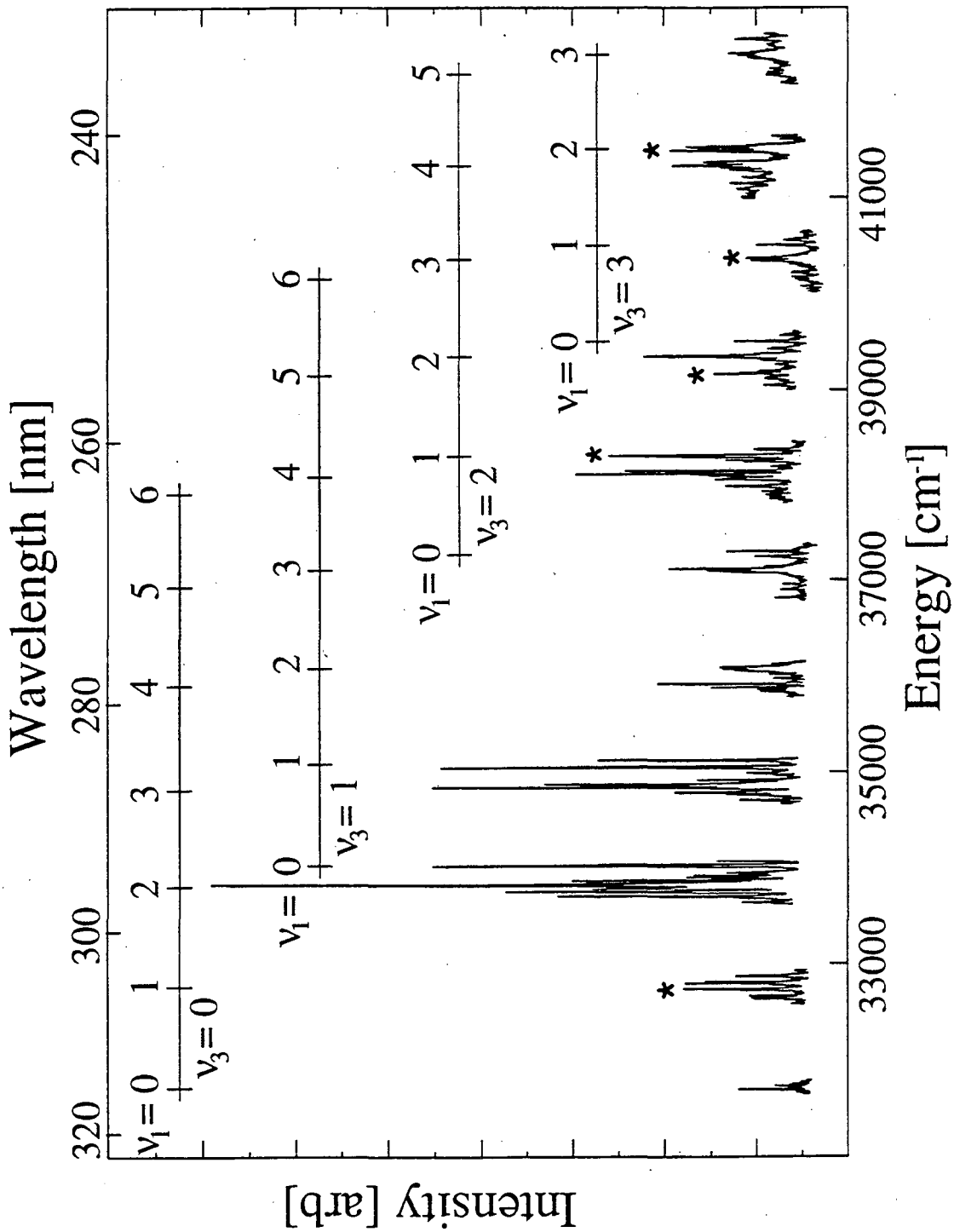

Wavelength [nm]
320
300
280
260
240
Energy [cm-1]
33000
35000
37000
39000
41000
Intensity [arb]
ν1= 0
ν3= 0
ν3= 1
ν3= 2
ν3= 3
0
1
2
3
4
5
6

Figure 5.1

Dixon's original work[11] and are at higher resolution than the matrix isolation spectra of Milligan and Jacox.[12] Preliminary scans, at a lower resolution than Figure 5.1, across the entire wavelength range enabled us to concentrate further scans in the regions where predissociation signal was observed. Therefore, the apparent gaps in our survey spectrum are regions that were found not to contain peaks of significant intensity. The intensity of the signal presented here is not normalized to laser power, since at the laser fluence customarily used for these survey scans (100-200 mJ/cm^2) the transitions could have been saturated.

To the extent that the spectrum in Figure 5.1 can be assigned, it appears to consist of progressions in the ν_1 and ν_3 vibrational modes of the $\tilde{B}\ ^2\Pi$ state with fundamental frequencies of approximately 1047 cm^{-1} and 2303 cm^{-1}, respectively. (These values were obtained in Dixon's study.[11]) The assignments for the location of the various ($\nu_1 0 \nu_3$) vibrational levels of the $\tilde{B}\ ^2\Pi$ state are indicated in Figure 5.1. An enlargement of the spectral region in Figure 5.1 containing the (100)←(000) band is shown in Figure 5.2. Figure 5.3 shows a higher resolution scan (laser step size = 0.001 nm) of the sub-bands labeled (d) and (e) in Figure 5.2; the more intense sub-band (d) is the main $^2\Pi_{3/2}$-$^2\Pi_{3/2}$ transition, which was one of the two sub-bands analyzed by Dixon in his original work on this system. Line widths in Figure 5.3 are instrument limited to ≡ 0.4 cm^{-1} FWHM, corresponding to the laser resolution.

Figure 5.2 Survey scan in the region of the $\tilde{B}\ ^2\Pi$ (100)←$\tilde{X}\ ^2\Pi$ (000) band. Labels a) through f) indicate individual sub-bands; b) and d) are the previously assigned (ref. 11) $^2\Pi_{1/2}$←$^2\Pi_{1/2}$ and $^2\Pi_{3/2}$←$^2\Pi_{3/2}$ sub-bands, while the other peaks have been tentatively assigned here as (020)←(000) sub-bands: a) μ-$^2\Pi_{1/2}$←$^2\Pi_{1/2}$, c) μ-$^2\Pi_{3/2}$←$^2\Pi_{3/2}$, e) κ-$^2\Pi_{1/2}$←$^2\Pi_{1/2}$, and f) κ-$^2\Pi_{3/2}$←$^2\Pi_{3/2}$. These transitions are shown schematically in Figure 5.6.

Figure 5.3 Finer scan (0.001nm step size) of sub-bands d) and e) in Figure 5.2. See Figure 5.6 for assignments. The P, Q and R branches in the two sub-bands are labeled.

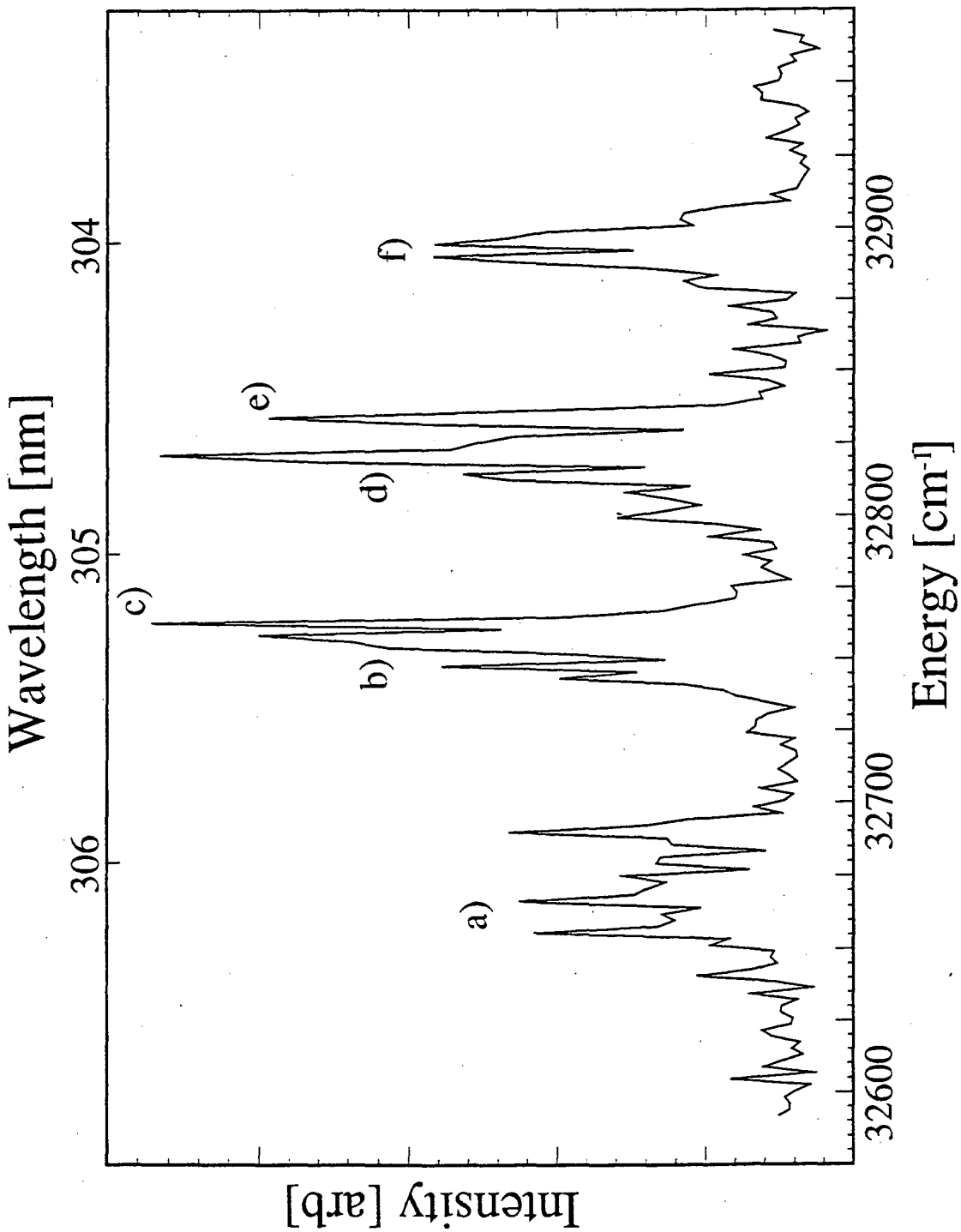

Wavelength [nm]
306
305
304
Energy [cm-1]
32600
32700
32800
32900
Intensity [arb]
a)
b)
c)
d)
e)
f)

Figure 5.2

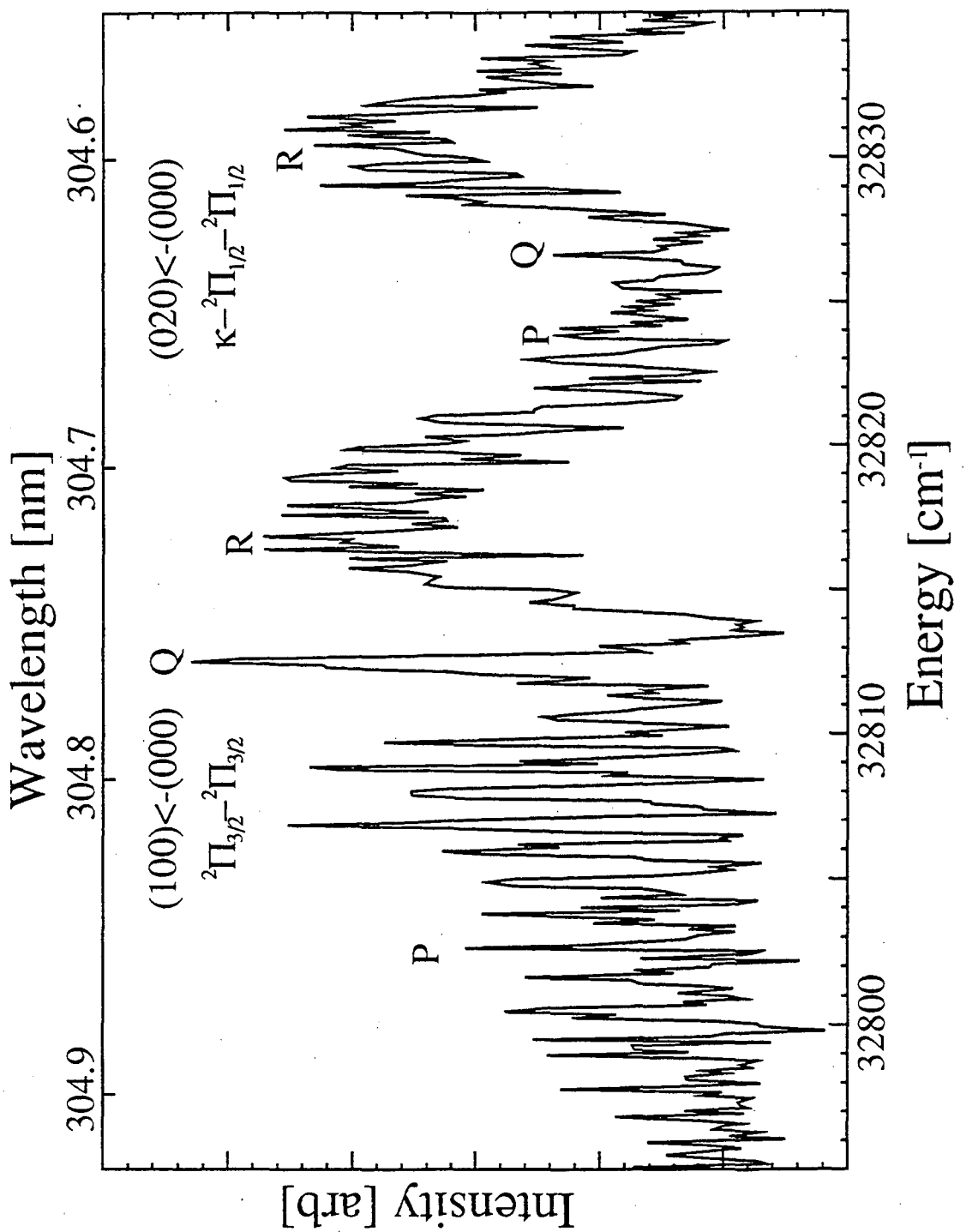

Wavelength [nm]
304.9
304.8
304.7
304.6
Energy [cm-1]
32800
32810
32820
32830
Intensity [arb]
(100)<-(000)
2Π3/2–2Π3/2
P
Q
R
(020)<-(000)
κ–2Π1/2–2Π1/2
P
Q
R

Figure 5.3

Since we observe signal *only when NCO dissociates* during the flight time between the photodissociation volume and the photofragment detector (about 5.27 μs for a 101.1 cm flight length), the spectrum in Figure 5.1 shows that the $\tilde{B}\ ^2\Pi$ state predissociates over the entire $\tilde{B}\ ^2\Pi \leftarrow \tilde{X}\ ^2\Pi$ system. Predissociation is observed even from the $\tilde{B}$ state (000) level, although the intensity of the photofragment signal was considerably less than for the bands further to the blue. This differs from the $\tilde{B}\ ^2\Pi \leftarrow \tilde{X}\ ^2\Pi$ absorption spectrum, in which transitions to the (000) level appear to be at least as intense as the transitions to higher levels. The implications of these observations for the NCO dissociation dynamics and energetics are discussed below.

Photofragment TOF spectra were taken at dissociation wavelengths of 305.23 nm, 260.69 nm, 255.20 nm, 247.58 nm, and 240.83 nm with the detector 101.1 cm from the photodissociation volume and the laser polarization such that the electric field vector $\vec{E}$ was parallel to the direction of the radical beam propagation. These wavelengths correspond to the peaks marked with an (*) in Figure 5.1. The five experimental spectra are displayed in Figure 5.4. The narrow peak shown in Figure 5.4 b) is a typical undissociated radical beam TOF profile, which was obtained to allow evaluation of the energy spread in the undissociated radical beam for incorporation into simulations of the dissociation spectra. In addition, the TOF spectra at 260.69 nm, 255.20 nm, and 247.58 nm were taken with the detector 67.8 cm from the photodissociation volume with the laser polarization parallel and perpendicular to the radical beam axis. These are shown in Figure 5.5. The spectra obtained with the two orthogonal laser polarizations are very

---

Figure 5.4 TOF data (circles) obtained with a flight length of 101.1 cm at five photodissociation wavelengths as indicated. Various simulations (solid, dashed lines) are discussed in the text.

Figure 5.5 TOF data (circles) obtained with a flight length of 67.8 cm at three wavelengths, with the dissociation laser output polarized both parallel and perpendicular to the ion beam axis as indicated. Various simulations (solid, dashed lines) are discussed in the text.

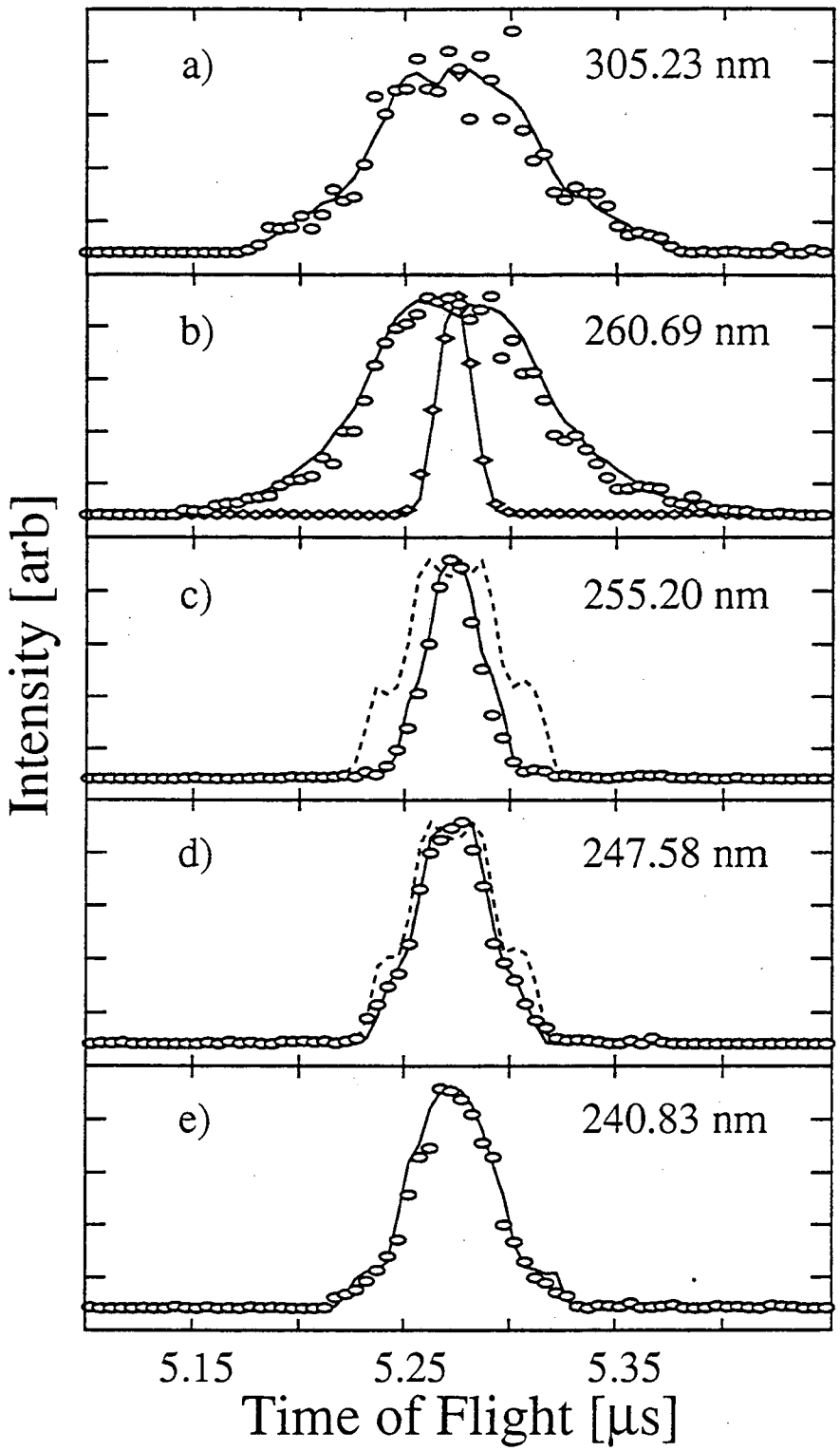

a)
305.23 nm
b)
260.69 nm
c)
255.20 nm
d)
247.58 nm
e)
240.83 nm
Intensity [arb]
5.15
5.25
5.35
Time of Flight [μs]

Figure 5.4

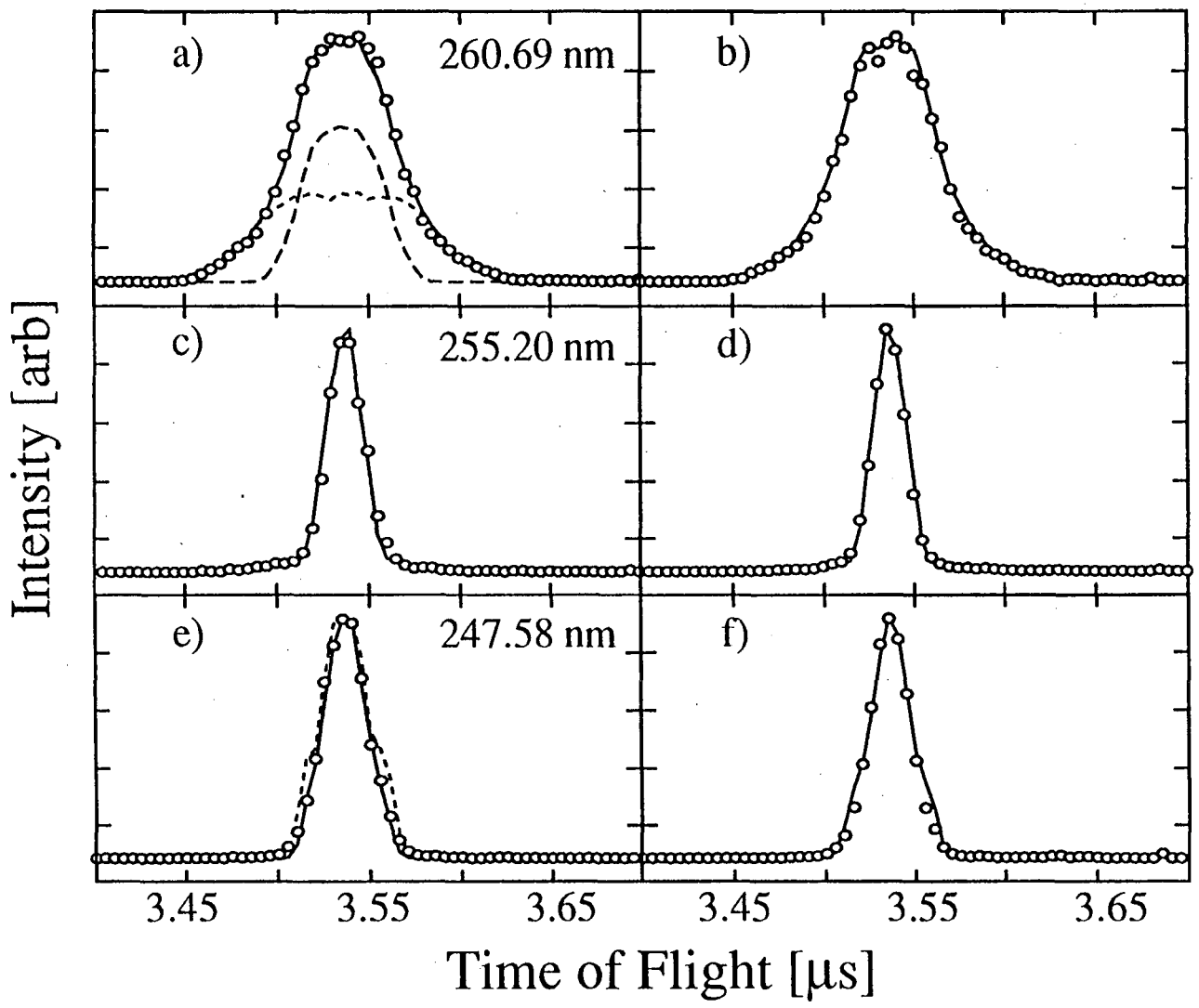

a)
260.69 nm
b)
c)
255.20 nm
d)
e)
247.58 nm
f)
Intensity [arb]
3.45
3.55
3.65
3.45
3.55
3.65
Time of Flight [μs]

Figure 5.5

similar, implying a nearly isotropic photofragment angular distribution. The most striking result seen in both figures is the abrupt narrowing of the TOF distribution between 260.69 and 255.20 nm, which suggests that a new dissociation channel opens in that wavelength interval.

5.4 Analysis

5.4.1 NCO Spectroscopy

The $\tilde{B}\ ^2\Pi \leftarrow \tilde{X}\ ^2\Pi$ system in NCO has proven notoriously difficult to assign. The $\tilde{X}\ ^2\Pi$ state vibrational energy level structure is complicated by Renner-Teller coupling and Fermi resonances between the ν_1 and ν_2 modes. These effects are also expected in the $\tilde{B}\ ^2\Pi$ state along with perturbations from the $\tilde{A}\ ^2\Sigma^+$ state. In Dixon's original spectrum,[11] only the (100)←(000) band had enough extended rotational structure and was sufficiently unperturbed to permit any rotational analysis; this led to an identification of the main $^2\Pi_{1/2} \leftarrow ^2\Pi_{1/2}$ and $^2\Pi_{3/2} \leftarrow ^2\Pi_{3/2}$ sub-bands. A rotational analysis of the (000)←(000) band was obtained only recently in a double resonance experiment.[34] Dixon found that the bands to the blue of the (100)←(000) band were broadened, presumably by predissociation, preventing any rotational analysis. Although these higher energy bands can be tentatively assigned to progressions in the ν_1 and ν_3 modes of the $\tilde{B}\ ^2\Pi$ state, the detailed vibrational structure within each band seen in Figure 5.1 and in Reference 11 remains largely unexplained. In this section we consider what additional information can be gleaned about the spectroscopy of the $\tilde{B}\ ^2\Pi \leftarrow \tilde{X}\ ^2\Pi$ transition from our photodissociation cross section measurements.

The rotationally-resolved spectrum in Figure 5.3 is a section of the (100)←(000) band. The main P, Q and R branches noted are from the previously assigned $^2\Pi_{3/2} \leftarrow ^2\Pi_{3/2}$ sub-band. A second, less intense transition is also shown just to the blue of this main sub-band, and the P branch associated with this transition extends into the R branch head of

the main sub-band. In comparison to Dixon's spectrum, the intensity distribution of the more intense band in Figure 5.3 reflects a substantially lower rotational temperature for the NCO radicals. The R and P branches in Figure 5.3 fall off in intensity at considerably lower J values. In addition, we observe a strong Q branch, while none was seen in his spectrum; in a $\Pi \leftarrow \Pi$ transition the intensity of the Q branch falls off rapidly with increasing J. The lower rotational temperature in our experiment is not too surprising, because our radicals are generated by photodetachment of a jet-cooled anion beam as opposed to flash photolysis. Simulations of our spectrum indicate a rotational temperature of about 50 K.

In the absence of perturbations or sequence bands, the (100)←(000) band should consist of only two sub-bands (the aforementioned $^2\Pi_{1/2} \leftarrow ^2\Pi_{1/2}$ and $^2\Pi_{3/2} \leftarrow ^2\Pi_{3/2}$ transitions) whose origins are separated approximately by $|A'-A''| = 18.9$ cm^{-1}, where $A' = -76.6$ cm^{-1} and $A'' = -95.5$ cm^{-1} are the spin-orbit coupling constants in the $\tilde{B}\ ^2\Pi$ and $\tilde{X}\ ^2\Pi$ states of NCO, respectively.[13,34] Instead, Figure 5.2 shows four well-separated sub-bands labeled (a), (b), (d), and (f), with approximate origins at 32,665 cm^{-1}, 32,750 cm^{-1}, 32,814 cm^{-1}, and 32,890 cm^{-1}, respectively. In addition, as mentioned above, there is an additional sub-band just to the blue of the main $^2\Pi_{3/2} \leftarrow ^2\Pi_{3/2}$ band at 32,829 cm^{-1} (labeled (e)), and a higher resolution scan of the main $^2\Pi_{1/2} \leftarrow ^2\Pi_{1/2}$ sub-band shows a similar sub-band at 32,760 cm^{-1} (labeled (c) in Figure 5.2). Thus, the (100)← (000) band consists of at least six sub-bands; only two of these have been assigned: (b) to the main $^2\Pi_{3/2} \leftarrow ^2\Pi_{3/2}$ sub-band and (d) to the $^2\Pi_{1/2} \leftarrow ^2\Pi_{1/2}$ sub-band. However, the splitting between the origins of sub-bands (b) and (d) is 64 cm^{-1}, rather than 19 cm^{-1} as expected from the difference of spin-orbit coupling constants. Thus, even this relatively well-understood band in the $\tilde{B}\ ^2\Pi \leftarrow \tilde{X}\ ^2\Pi$ transition is quite complex, and we wish to consider possible explanations for its appearance.

Previous studies of the electronic spectroscopy of BO_2 and CO_2^+ provide a wealth of information which can be used to interpret the spectrum of the isoelectronic NCO radical. The $\tilde{A}\ ^2\Pi_u \leftarrow \tilde{X}\ ^2\Pi_g$ absorption system in BO_2 was first seen at high resolution by Johns,[35] while the emission spectroscopy of the $\tilde{A}\ ^2\Pi_u \rightarrow \tilde{X}\ ^2\Pi_g$ system in CO_2^+ has been extensively studied by Rostas and co-workers.[36] Both the (100)←(000) band in BO_2 and the (000)→(100) band in CO_2^+ exhibit several vibrational features in addition to the two expected sub-bands. In both cases, there are sub-bands due to Fermi resonances between the (100) and (020) levels present; $\nu_1 \approx 2\ \nu_2$ in the $\tilde{A}\ ^2\Pi$ state of BO_2 and the $\tilde{X}\ ^2\Pi$ state of CO_2^+. The (020) levels are further split by the combination of Renner-Teller and spin-orbit interactions. The theoretical treatment for this combination of effects has been worked out in a series of papers by Pople[37] and Hougen.[38]

While a (000)-(020) band in a transition between two $^2\Pi$ electronic states is normally very weak, the (020) levels of Π vibronic symmetry borrow intensity from the nearby (100) levels (which also have Π vibronic symmetry). As discussed by Johns[35] and Larcher et al.,[36b] this interaction leads to as many as six sub-bands in what is nominally a (100)←(000) band, since the μ and $\kappa\ ^2\Pi_P$ ($P = 1/2,\ 3/2$, where $P=|\pm\Omega\pm l|$) states associated with the (020) level each borrow intensity from the $^2\Pi_\Omega$ state associated with the (100) level (see Figure 5.6). In addition, because of energy level shifts associated with the Fermi resonance, the splitting between the (100) $^2\Pi_{1/2}$ and $^2\Pi_{3/2}$ levels can be quite different from the value expected based on spin-orbit coupling alone.

In NCO, where $\nu_1 \approx 2\nu_2$ in the $\tilde{X}\ ^2\Pi$ and $\tilde{A}\ ^2\Sigma^+$ electronic states, effects from Fermi resonances have been observed in both states.[14,20b] Although ν_2 has not been measured for the $\tilde{B}\ ^2\Pi$ state, a recent theoretical study by Alexander and Werner[39] predicts harmonic frequencies of 1080 cm^{-1} and 522 cm^{-1} for these stretching and bending modes, so one might expect Fermi resonances in this state as well. The observation of at

least six sub-bands in Figure 5.2 is consistent with this expectation, as is the deviation of the splitting between bands (b) and (d) from |A'-A"|.

Figure 5.6, adapted from Johns' paper,[35] shows an energy level diagram consistent with the structure in the (100)←(000) band. The (100) and (020) $\tilde{B}$ state levels with Π vibronic symmetry are shown, from left to right, with no interactions, spin-orbit and Renner-Teller coupling, and finally, the additional effect of Fermi resonances. On the left, the unperturbed (100) level lies above the (020) level in accordance with Alexander's calculation. In the middle set of levels, Dixon's value for the $\tilde{B}$ state spin-orbit coupling constant (A' = -76.6 cm^{-1}) is used for the (100) and (020) manifolds, along with a Renner-Teller parameter sufficiently large so that the (100) levels lie within the (020) levels with Π symmetry (the two (020) levels with Φ symmetry are not shown). The rightmost set of levels shows the qualitative effect of the Fermi resonance between the (100) and (020) manifolds. Since nearby levels with the same Ω value repel one another, the (100) $^2\Pi_{1/2}$ and $^2\Pi_{3/2}$ levels are forced closer together. This results in a splitting between the main $^2\Pi_{1/2}$←$^2\Pi_{1/2}$ and $^2\Pi_{3/2}$←$^2\Pi_{3/2}$ transitions which is larger than |A'-A"|, as is seen in the experimental spectrum. Carrying this one step further, the six sub-bands identified in Figure 5.2 can be assigned to the transitions labeled in Figure 5.6, assuming the energy level spacings indicated in the caption of Figure 5.6.

While Figure 5.6 offers an explanation for the appearance of the (100)←(000) band, other possible sources for the multiple sub-bands must also be considered. For example, perturbations from the lower-lying $\tilde{A}$ $^2\Sigma^+$ state are prominent in the $\tilde{B}$ $^2\Pi$ (000) ←$\tilde{X}$ $^2\Pi$ (000) band,[34] and may well be contributing to the (100)←(000) band. At higher

---

Figure 5.6 Schematic showing our tentative assignment for the sub-bands observed in the (100)←(000) transition region. The observed transitions yield the following spacings for the 100 and 020 energy levels in the $\tilde{B}$ $^2\Pi$ electronic state, relative to the lowest μ-$^2\Pi_{3/2}$ level: μ-$^2\Pi_{1/2}$, 0.6 cm^{-1}, $^2\Pi_{3/2}$, 54 cm^{-1}, $^2\Pi_{1/2}$, 85 cm^{-1}, κ-$^2\Pi_{3/2}$, 130 cm^{-1}, κ-$^2\Pi_{1/2}$, 164.6 cm^{-1}.

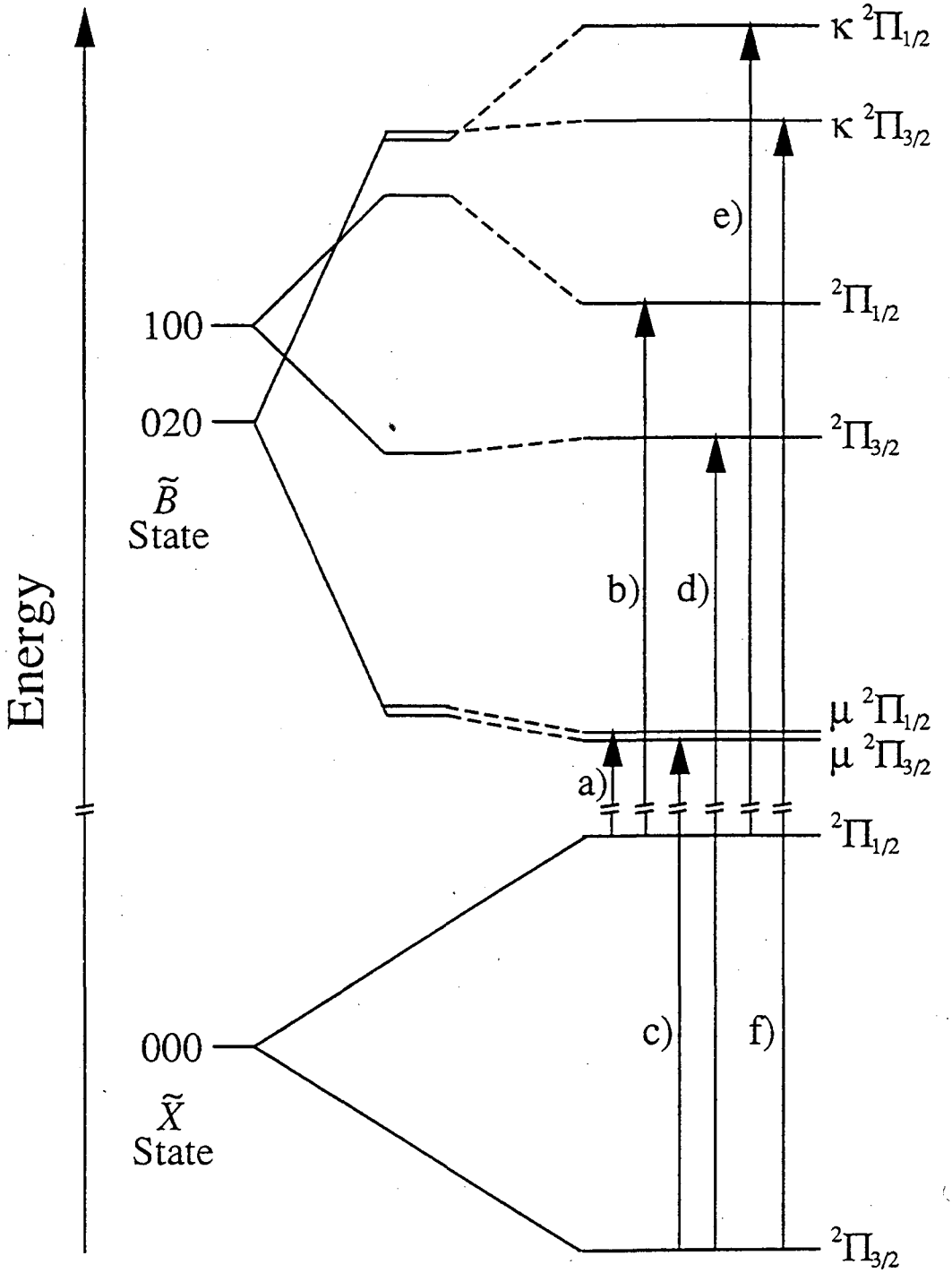

Energy
100
020
$\tilde{B}$
State
000
$\tilde{X}$
State
$\kappa\ ^2\Pi_{1/2}$
$\kappa\ ^2\Pi_{3/2}$
$^2\Pi_{1/2}$
$^2\Pi_{3/2}$
$\mu\ ^2\Pi_{1/2}$
$\mu\ ^2\Pi_{3/2}$
$^2\Pi_{1/2}$
$^2\Pi_{3/2}$
a)
b)
c)
d)
e)
f)

Figure 5.6

resolution sub-band (a), in particular, appears to actually consist of two distinct features, one centered at 32665 cm^{-1} (the value used in the above discussion) and the other at 32689 cm^{-1}, possibly resulting from extensive $\tilde{A}$ state perturbations. Similar perturbations have been identified in a double resonance study of CO_2^+.[40] In addition, the (100)←(000) transition in NCO may be overlapped by sequence bands in the NCO bend (the (110)← (010) band, for example). Any bend excitation in the NCO^- is transferred to NCO when the radical is produced by photodetachment, because bend sequence bands in the NCO^- photoelectron spectrum fall on top of the NCO(000)←NCO^-(000) transition. We note that bend sequence bands were observed in our photodissociation study of N_3.[5]

With higher resolution data over a wider frequency range than in the rotationally-resolved spectrum in Figure 5.3, we might be able to definitively sort out these possibilities. As it is, the qualitative similarities with the CO_2^+ and BO_2 spectra support the notion that most of the detailed structure in the (100)←(000) band in NCO is from Fermi resonances within the $\tilde{B}\ ^2\Pi$ state. These effects will become even more complex for higher energy transitions in the $\tilde{B}\ ^2\Pi \leftarrow \tilde{X}\ ^2\Pi$ system.

5.4.2 Dissociation Dynamics

In this section, the photofragment time-of-flight (TOF) results are analyzed in order to learn about the dissociation dynamics of the NCO $\tilde{B}\ ^2\Pi$ state. This data can be considered at two levels of detail. We obtain the approximate maximum photofragment kinetic energy release at each wavelength from the edges of the TOF distribution. We also obtain more precise and detailed photofragment energy and angular distributions by comparing experimental and simulated TOF spectra at various laser wavelengths and polarizations.

We first consider the five TOF spectra shown in Figure 5.4. In the center-of-mass frame of reference, the fastest N atoms scattered parallel and anti-parallel to the ion beam

Wavelength [nm]	$h\nu$ [eV]	$\Delta\tau$ [ns]	KER [eV]	hν - KER [eV]
305.23	4.06	220	1.74	2.32
260.69	4.76	265	2.52	2.24
255.20	4.86	75	0.20	4.66
247.58	5.01	100	0.36	4.65
240.83	5.15	120	0.52	4.63

Table 5.1 Summary of results obtained directly from TOF spectra shown in Figure 5.4. Kinetic energy release (KER) values are determined using Eqn. (5.1). See text.

direction correspond to the minimum (t_{min}) and maximum (t_{max}) flight times at which photofragment signal is observed. If effects due to the energy spread of the radical beam and the extent of the beam block are ignored (these *are* considered explicitly below), then $\tau = t_{max} - t_{min}$ is related to the maximum center-of-mass kinetic energy release, KER, by

$$KER = \frac{v_0^4}{8L^2} \cdot \left(\frac{m_N \cdot m_{NCO}}{m_{CO}} \right) \cdot \tau^2, \qquad (5.1)$$

where L is the distance the fragments travel to the detector (101.1 cm for these spectra) and v_0 is the beam velocity (1.91×10^7 cm/sec). The results for the spectra in Figure 5.4 are shown in Table 5.1.

Within Table 5.1, the quantity (hν - KER) corresponds to the bond dissociation energy if ground state products are formed. Any excitation present in the products reduces the KER and makes this quantity larger. The more detailed simulations discussed below show that the values for the KER in Table 5.1 are only approximate upper bounds to the maximum photofragment kinetic energy release, and therefore, the values of

(hν – KER) are only approximate lower bounds for the bond dissociation energy plus product internal energy.

Nonetheless, several features in this table are noteworthy. First, consider the value for the KER at λ = 305 nm, which corresponds to excitation of the (100)←(000) transition. If the upper state were dissociating to the lowest energy spin-allowed dissociation channel, N(^2D) + CO, then KER = 1.78 eV yields $\Delta_f H^0_{298\,K}$(N-CO) = 88 kcal/mol, more than double the accepted value. This value for $\Delta_f H^0_{298\,K}$(N-CO) would mean that the CN + O_2 → NCO + O reaction is endothermic by 45 kcal/mol, a physically unreasonable result because this reaction is rapid at room temperature. The 305 nm and 260 nm TOF results therefore indicate that the upper states for both transitions are dissociating to the energetically allowed but spin-forbidden N(^4S) + CO channel, which lies 2.38 eV (54.9 kcal/mol) below the N(^2D) + CO channel. However, at λ = 255 nm, the TOF distribution narrows abruptly. This suggests an energy level diagram for NCO and its photofragments as given in Figure 5.7; at λ = 255 nm, the spin-allowed channel N(^2D) + CO channel is energetically accessible, whereas it is not at λ = 260 nm.

The results in Table 5.1 therefore imply that the N(^2D) + CO channel becomes accessible between 4.76 and 4.86 eV above the NCO ground state and allows us to bracket $\Delta_f H^0_{298\,K}$(NCO) between 31.5 and 29.0 kcal/mol. This bracketing is validated by the additional observation that the KER is small at the longest wavelength at which N(^2D) + CO is formed, implying no substantial barrier to dissociation. Since this range of $\Delta_f H^0_{298\,K}$(NCO) values lies below the accepted literature[26] value of 36.1 kcal/mol, we have performed more detailed analyses of photofragment TOF spectra which will now be discussed.

We wish to determine the N-CO bond dissociation energy and the photofragment

---

Figure 5.7 Energy level diagram showing the position of the NCO electronic energy levels relative to the asymptotic energies of the fragment channels as determined by this work.

Energy [kcal/mol]
120
100
80
60
40
20
0
N(2P) + CO(1Σ+, v=0)
O(3P) + CN(1Σ+, v=0)
v=1
N(2D) + CO(1Σ+, v=0)
v=6
v=4
v=2
N(4S) + CO(1Σ+, v=0)
260.69 nm
255.20 nm
247.58 nm
NCO B̃ 2Π
NCO Ã 2Σ
NCO X̃ 2Π
Energy [eV]
5
4
3
2
1
0

Figure 5.7

energy and angular distributions consistent with the TOF spectra in Figures 5.4 and 5.5. This is done using a Monte Carlo computer simulation program,[41] which generates a TOF spectrum for given dissociation wavelength, laser polarization and radical beam characteristics. The simulation program also accounts for the finite size of the photofragment detector and the beam block. The radical beam characteristics are checked by comparing simulated and experimental TOF spectra of the undissociated radical beam; the agreement, shown in Figure 5.4(b), is excellent. The simulation assumes a spread of 8 eV (0.1 %) in the ion beam energy. This value is consistent with the small energy spread expected from the beam-modulation TOF mass spectrometer.[42]

To carry out a photofragment TOF simulation, we specify a trial photofragment angular distribution and bond dissociation energy. The simulation program accounts for contributions to the TOF spectra by both N and CO fragments, since the experimental spectrum has contributions from both. Figure 5.5(a) shows the individual contributions of the N (dotted lines) and CO (dashed lines) fragments as determined from the simulation program together with their sum that fits the experimental TOF spectrum. The fact that only N atoms contribute to the TOF signal close to t_{min} and t_{max} is illustrated clearly by this figure.

The photofragment center-of-mass kinetic energy and angular distribution $P(E_T,\theta)$ is taken to be

$$P(E_T,\theta) = \sum_{i=S,D} n_i P_i(E_T) \cdot (1+\beta_i P_2(\cos\theta)). \qquad (5.2)$$

The index i specifies the N atom electronic state (^4S or ^2D). Here n_i refers to the electronic state populations, $P_i(E_T)$ is the photofragment kinetic energy distribution in the ith electronic state, θ is the angle between the fragment recoil and laser polarization vector, and β_i is the anisotropy parameter[43] for the ith electronic state. $P_i(E_T)$ is

λ [nm]	^4S:^2D Ratio	β (for ^2D channel)	β (for ^4S channel)	D_0(N-CO) [kcal/mol]
260.69	100:0	N/A	+0.1	54.9
255.20	6:94	+0.2	+0.1	54.9
247.58	2:98	+0.2	+0.1	54.9

Table 5.2 Simulation parameters used to fit all data in Figure 5.5 (both polarizations) and the common wavelengths in Figure 5.4. The value shown for D_0(N-CO) was determined by obtaining the best fit for *all* data in both Figures 5.4 and 5.5.

λ [nm]	v = 0 pop.	v = 1 pop.	v = 2 pop.	v = 3 pop.	v = 4 pop.	v =5 pop.	v =6 pop.	v =7 pop.	v =8 pop.
260.69	1.0	1.0	1.0	1.0	1.5	2.5	3.0	2.0	1.0

Table 5.3 CO fragment vibrational distribution used in the simulations of the three TOF spectra at 260.69 nm in Figures 5.4 and 5.5. The simulations assumed no rotational excitation of the CO.

---

determined by specifying the bond dissociation energy and the CO vibration-rotation distribution associated with the *i*th N atom electronic state.

We first consider the TOF spectra at the three wavelengths for which the photofragment flight length was 67.8 cm (Figure 5.5), as these were taken with both laser polarization directions. The broad TOF spectrum at λ = 260 nm indicates dissociation to N(^4S) + CO, while the two narrower spectra at λ = 255 and 247 nm are from dissociation primarily to N(^2D) + CO. However, the narrow peak in the 255 nm TOF spectrum sits atop a small, broad pedestal due to a small amount of N(^4S) + CO production. At all three wavelengths, the spectra taken with the laser horizontally polarized (parallel to the radical

beam axis) are slightly broader than the spectra taken with vertical laser polarization. This indicates that the anisotropy parameter β is positive for both N atom product electronic states; we find β=+0.1 for the N(^4S) channel and β=+0.2 for the N(^2D) channel. The complete set of parameters used to arrive at the solid line fits for the three sets of TOF spectra are listed in Table 5.2 and Table 5.3.

The narrow peak in the 255 nm TOF spectrum can be fit successfully assuming a single value for the photofragment kinetic energy of 0.10 eV. If this corresponds to the translational energy of N(^2D) + CO (v=0, J=0) fragments, then the N(^2D) + CO dissociation channel lies 4.76 eV (109.8 kcal/mol) above the ground state of NCO, yielding an N-CO bond dissociation energy (to form N(^4S) + CO) of 2.38 eV (54.9 kcal/mol). This value of the bond dissociation energy is the one used to determine the relative energies of NCO and the dissociation product channels in Figure 5.7.

Our interpretation of the 255 nm spectrum is supported by the 247 nm spectrum using the same bond dissociation energy in which the edges of the distribution are fit accurately assuming they are also from N(^2D) + CO (v=0, J=0) fragments. However, closer to the center of the distribution, the simulated TOF spectrum is too broad (Figure 5.5(e), dashed line) if *all* the CO is assumed to be in its (v=0, J=0) state, and the best simulation (solid line) includes rotational excitation in the CO fragment [CO(v=1) is not accessible at 247 nm]. The rotational excitation was incorporated using the following empirical functional form for the translational energy distribution:

$$P(E_T) = 1 - \frac{E_T}{E_{TOT}} \qquad (5.3)$$

Here E_T is translational energy while E_{TOT} is hν - [D_0(N-CO) + ΔE(^2D-^4S)], with ΔE(^2D-^4S) equal to the difference in energy between the ^2D and ^4S N atom electronic states.

λ [nm]	v = 0 pop.	v = 1 pop.	v = 2 pop.	v = 3 pop.	v = 4 pop.	v =5 pop.
305.23	1.0	1.5	2.5	3.0	3.0	1.0
240.83	1.0	1.0	---	---	---	---

Table 5.4 CO fragment vibrational distribution (neglecting rotational excitation) used in the simulations in of the TOF spectra Figure 5.4 with TOF spectra at 305.23 (for N(^4S) + CO products) and 240.83 nm (for N(^2D) + CO products).

In the simulated 260 nm TOF spectrum, the photofragment kinetic energy distribution was determined using the bond dissociation energy, determined above, of 2.38 eV. The CO (v, J=0) distribution used is given in Table 5.3. A more detailed distribution involving CO vibrational and rotational excitation is not justified by the data. However, one can safely say that, unless the CO is extremely rotationally excited, the CO vibrational distribution is highly inverted. More significantly, from the perspective of the NCO energetics, t_{min} and t_{max} in the simulated TOF distribution, which are due to N(^4S) + CO(v=0), match the experimental spectrum very well, further supporting the bond dissociation energy obtained from the 255 nm spectrum. As mentioned above, the 255 nm and, to an even lesser extent, the 247 nm spectra have a small contribution from N(^4S) + CO product. The electronic branching ratios in Table 5.2 were determined assuming the same CO vibrational distribution for the N(^4S) + CO product that was determined for the 260 nm spectrum.

As a further check on the values determined in these simulations, the parameters in Table 5.2 and Table 5.3 were used to fit the three TOF spectra in Figure 5.4 at the same three wavelengths but where the photofragment flight length is 101.1 cm. Excellent agreement is found between the experimental and simulated TOF spectra. Finally, the TOF spectra in Figure 5.4 at both 305 nm and 240 nm were fit assuming the respective β

values obtained above for the two N atom electronic channels. The fragment state populations that gave the best-fitting simulations (solid lines, Figure 5.4) are listed in Table 5.4. The CO vibrational distribution at 305 nm appears to be highly inverted, just as at 260 nm, while the excellent fit at the extrema of the simulation to the data further confirms our choice of D_0(N-CO). At 240 nm, again using D_0(N-CO) = 2.38 eV in our simulation, the N(^2D) + CO(v=1) channel should be energetically accessible. Indeed, the shape of the TOF spectrum is noticeably different from the 247 nm spectrum, with broader wings under a central peak, and we find by varying only the relative vibrational level population of v=0 and v=1 (neglecting rotational excitation) that an adequate fit is obtained if 50% of the CO is in the v=1 level.

The value of 2.38 eV for D_0(N-CO) was obtained from our analysis assuming the NCO is in its rotational ground state. Including the correction for the average rotational energy of the NCO (0.1 kcal/mol at 50 K), we obtain $\Delta_f H^0$=30.4 kcal/mol at 0 K. $\Delta_f H^0$(NCO) at 298 K is only slightly higher, 30.5 kcal/mol.[44] This value is significantly lower than the currently accepted literature value[26] of 36.1 ± 1 kcal/mol (D_0(N-CO) = 2.13 eV). As a measure of the sensitivity of our TOF spectra to the assumed value of D_0(N-CO), Figure 5.4(c) shows the results of a simulation (dotted line) at 255 nm with D_0(N-CO) = 2.13 eV but with all the other parameters the same as in Table 5.2. This is clearly inferior to the simulation with D_0(N-CO) = 2.38 eV. Based on trial simulations at various dissociation wavelengths using different bond dissociation energies, we estimate the error in our determination of $\Delta_f H^0_{298\,K}$(NCO) to be ± 1 kcal/mol.

5.5 Discussion

The results presented here not only settle the question of the onset of predissociation in the NCO $\tilde{B}\ ^2\Pi$ state, but also explain the observations in previous experiments. In Dixon's original study of the NCO $\tilde{B}\ ^2\Pi \leftarrow \tilde{X}\ ^2\Pi$ system, he attributed the

spectral broadening of the (200)←(000) transition to the onset of predissociation forming N(^2D) + CO products, whereas Crosley[23] interpreted the lower fluorescence lifetime of the $\tilde{B}$ $^2\Pi$ state (100) level relative to the (000) level to mean that the (100) level could dissociate to N(^2D) + CO. Our photodissociation cross section measurements clearly show that *all* vibrational levels of the NCO $\tilde{B}$ $^2\Pi$ state predissociate. The explanation of these seemingly discordant results is given by the photofragment TOF spectra. These show that, in Dixon's and Crosley's experiments, predissociation of the $\tilde{B}$ $^2\Pi$ state was indeed occurring, but it was spin-forbidden dissociation to N(^4S) + CO products. Their results can therefore be explained if the dissociation rate to these products increases with the level of $\tilde{B}$ $^2\Pi$ state vibrational excitation, causing a noticeable decrease in the fluorescence lifetime for the (100) level and observable broadening in the absorption spectrum for the (200) and higher levels. As mentioned in the Results section, the intensity of the photofragment signal from the (000) level relative to the signal from the higher levels appears to be smaller than in the $\tilde{B}$ $^2\Pi$←$\tilde{X}$ $^2\Pi$ absorption spectrum. This suggests that the dissociation rate from the (000) level is sufficiently slow that the photofragment signal is depleted by spontaneous emission.

The next important issue is the discrepancy between our value for $\Delta_f H^0_{298\,\mathrm{K}}$(NCO) and the literature value. Our experiment is, in principle, a direct measurement of the N-CO bond dissociation energy, but there are several assumptions in our analysis which must be discussed. The $\tilde{B}$ $^2\Pi$←$\tilde{X}$ $^2\Pi$ transitions in the range of the N(^2D) + CO threshold have not been definitively assigned. In using the photofragment TOF spectra to determine this threshold, we have assumed all the spectroscopic transitions originate from the (000) level of the NCO ground electronic state. This may not be correct, as some of these transitions may be sequence bands originating from vibrationally excited NCO. As discussed in the Analysis section, vibrationally excited NCO can result from photodetachment of vibrationally excited NCO^-. However, in the unlikely event that this

were somehow the case for all the transitions in Figure 5.4, then the true bond dissociation energy would be higher than 2.38 eV and the true $\Delta_f H^0_{298\,K}$(NCO) would be *lower* than 30.5 kcal/mol, a shift in the opposite direction from the literature value. We also assume in our simulations that dissociation occurs in a time which is less than 1% of the total flight time (~3.5 μs). This is strongly supported by the observation that the TOF spectra which were taken at both photofragment flight lengths, l = 67.8 cm and 101.1 cm, could be fit with a single fragment translational energy distribution.

Another source of possible error is that our analysis assumes that t_{min} and t_{max} in the TOF distributions are from N + CO(v=0) fragments, and, in particular, that the TOF spectra at 255 nm and 247 nm are due entirely to CO(v=0). Suppose, however, that no CO(v=0) is produced at those wavelengths, and that the TOF spectra are from N(^2D) + CO(v=1). If this were the case, D_0(N-CO) would have to be lowered by 5 kcal/mol (corresponding to the CO vibrational frequency) from our value and $\Delta_f H^0_{298\,K}$(NCO) would be raised by the same amount, which would bring it in close agreement with the literature value. However, this would imply that the CO distribution at 241 nm [Figure 5.4(e)] is approximately 50% v=2, 50% v=1, and no v=0, a most unusual distribution. In addition, at 260 nm and 305 nm, where only N(^4S) + CO is formed, the edges of the TOF spectra are reproduced in our simulations using D_0(N-CO) = 2.38 eV. In order to fit these spectra with a 5 kcal/mol lower bond dissociation energy, one would have to assume no CO(v=0) was produced at these wavelengths, as well. This also seems unlikely, considering the substantial range of CO vibrational states populated at both of these wavelengths, and the different mechanism producing the spin-forbidden product channel. We therefore believe that the bond dissociation energy obtained from our TOF spectra is correct, and that the literature value of $\Delta_f H^0_{298\,K}$(NCO) is too large by several kcal/mol.

The literature value[26] of $\Delta_f H^0_{298\,K}(NCO)$ is obtained from three quantities which contain possible sources of error: (1) D^A(H-NCO), the threshold for formation of NCO $\tilde{A}\ ^2\Sigma^+$ from the photolysis of HNCO, (2) D_0(HN-CO), the dissociation energy of HNCO to form NH + CO, and (3) $\Delta_f H^0_{298\,K}(NH)$. The H-NCO bond dissociation energy, D_0(H-NCO), is found from D^A(H-NCO) by subtracting 2.82 eV, the electronic energy of the NCO $\tilde{A}\ ^2\Sigma^+$ state. We then have

$$\Delta_f H^0_{0K}(NCO) = D_o(H-NCO) - D_o(HN-CO) + \Delta_f H^0_{0\,K}(NH) + \Delta_f H^0_{0\,K}(CO) - \Delta_f H^0_{0\,K}(H). \qquad (5.4)$$

The standard heats of formation of CO and H are well known.[44] The literature value of $\Delta_f H^0_{298\,K}(NCO)$ is based on Okabe's value[22] of 7.73 eV for D^A(H-NCO), Chandler's value[25] of D(HN-CO) = 82.9 (+2.8,-0.7) kcal/mol, which was obtained from the threshold for NH($a^1\Delta$) production from HNCO photodissociation, and Berkowitz's value[45] of $\Delta_f H^0_{298\,K}(NH)$ = 85.2 kcal/mol, which was obtained from NH_2 ionization and dissociative ionization potentials. Two more recent independent measurements[46] of $\Delta_f H^0_{298\,K}(NH)$ are in excellent agreement with Berkowitz's value. However, a more recent measurement of D^A(H-NCO) by Shobatake[47] yields a slightly lower value, 7.65 eV, lowering $\Delta_f H^0_{298\,K}(NCO)$ by 1.8 kcal/mol. This brings $\Delta_f H^0_{298\,K}(NCO)$ down from 36.1 to 34.3 kcal/mol. In addition, using the upper bound on D_0(HN-CO) of 85.7 kcal/mol from Chandler's study lowers $\Delta_f H^0_{298\,K}(NCO)$ still more, to 31.5 kcal/mol, bringing it within the error bars of our measurement. Our value for $\Delta_f H^0_{298\,K}(NCO)$ therefore is not as inconsistent with previous work as it might first appear, and a further lowering of D^A(H-NCO) would yield even better agreement.

We also note that Coombe[2] obtained a lower bound to $\Delta_f H^0_{298\,K}(NCO)$ of 37 kcal/mol in a photodissociation study of NCO at 193 nm, based on the highest internal energy seen in the CN fragment. However, his value assumes no internal excitation in the

NCO, which is generated by the F + HNCO reaction. The CO bond distances in HNCO and NCO are 1.167 Å[48] and 1.21 Å[49] respectively. Based on a simplistic Franck-Condon view of the F + HNCO reaction, this geometry change would lead one to expect the NCO product to have significant stretching excitation which could, in principle, appear as CN internal excitation when the NCO is photodissociated.

Finally, we consider the mechanism for the various NCO dissociation processes. Until very recently, little was known about the detailed structure and interaction of the excited state potential energy surfaces of NCO. Past calculations have been completed at the INDO-CI[50] and MINDO/3[51] level; predictions of electronic state levels varied widely between these two calculations, with neither being close to the experimentally determined values for the $\tilde{A}\ ^2\Sigma^+$ and $\tilde{B}\ ^2\Pi$ states. However, a theoretical study just completed by Alexander and Werner[39] addresses in detail the nature of the potential energy surfaces and the mechanism of the spin-forbidden decomposition of the NCO $\tilde{B}\ ^2\Pi$ state using *ab initio* methods. They propose that the dissociation pathway for this process first passes through a conical intersection between the $\tilde{B}\ ^2\Pi$ and $\tilde{A}\ ^2\Sigma^+$ states, both of which have components of A' symmetry when the molecule is bent. Dissociation then occurs through the interaction, via spin-orbit coupling, of the $\tilde{A}\ ^2\Sigma^+$ state with the repulsive $^4\Sigma^-$ state which correlates with N(^4S) + CO products. An attractive feature of their calculation is that, for collinear geometries, the minimum in the crossing seam between the $\tilde{B}\ ^2\Pi$ and $\tilde{A}\ ^2\Sigma^+$ states occurs slightly above the $\tilde{B}\ ^2\Pi$ state minimum. This is consistent with the apparent rapid increase in dissociation rate as the vibrational energy of the $\tilde{B}\ ^2\Pi$ state increases.

This calculation also predicts that the lowest energy crossing between the $\tilde{A}$ state and the repulsive $^4\Sigma^-$ state occurs at a bond angle of 145.5° and CO bond length of 1.21 Å, which is considerably longer than the bond length of 1.13 Å in diatomic CO. One therefore expects considerable rotational and vibrational excitation of the CO produced by spin-forbidden dissociation of NCO. While we cannot distinguish rotational from

vibrational excitation in our TOF spectra of the spin-forbidden channel, the spectra are certainly consistent with extensive excitation of the CO product. Planned experiments with a two-particle position and time sensing detector similar to that used by Los and co-workers[52] should yield more complete product energy distributions, enabling a detailed comparison with theory.

Finally, although Alexander and Werner did not explicitly discuss photodissociation to the spin-allowed products, one can speculate on the possible mechanism based on their results. Neither the $\tilde{A}\ ^2\Sigma^+$ nor $\tilde{B}\ ^2\Pi$ states correlate to N(^2D) + CO products for linear NCO geometries, but both states correlate to these products for bent geometries. The excited stretching levels of the $\tilde{B}\ ^2\Pi$ state that are accessed in our experiment should have some bending character due to Fermi resonances, so dissociation to spin-allowed products should be reasonably facile once this channel becomes energetically accessible. This is consistent with the experimental observation that the spin-allowed channel dominates immediately above the N(^2D) + CO threshold.

5.6 Conclusions

We have used photodissociation spectroscopy to study the $\tilde{B}\ ^2\Pi \leftarrow \tilde{X}\ ^2\Pi$ electronic transition in NCO. Photodissociation cross section measurements show that predissociation occurs throughout the $\tilde{B}\ ^2\Pi$ state, including the ground vibrational level. Time-of-flight spectroscopy of the N + CO photofragments reveal that the first several vibrational levels of the NCO $\tilde{B}\ ^2\Pi$ state undergo spin-forbidden dissociation to N(^4S) + CO products, but 20.3 kcal/mol above the $\tilde{B}\ ^2\Pi$ state origin, the spin-allowed N(^2D) + CO channel becomes energetically accessible and immediately dominates the photofragment distribution. From our determination of this threshold, we obtain a $\Delta_f H^0_{298\,\mathrm{K}}$ of 30.5 kcal/mol for the NCO free radical, which is significantly lower than the current literature value of 36.1 kcal/mol. We also obtain approximate CO internal energy

distributions from the time-of-flight spectrum. The dynamics of NCO photodissociation are briefly discussed in terms of recent theoretical results by Alexander and Werner which indicate that spin-forbidden dissociation from the $\tilde{B}\ ^2\Pi$ state occurs via a conical intesection with the $\tilde{A}\ ^2\Sigma^+$ state followed by a non-adiabatic transition to a repulsive quartet state which correlates to $N(^4S)$ + CO products.

5.7 References

[1] *Molecular Photodissociation Dynamics*, edited by M. N. R. Ashfold and J. E. Baggott (Royal Society of Chemistry, London, 1987).

[2] X. Liu and R. D. Coombe, J. Chem. Phys. **91**, 7543 (1989).

[3] J. C. Loison, S. H. Kable, P. L. Houston, and I. Burak, J. Chem. Phys. **94**, 1796 (1991).

[4] E. J. Hintsa, X. S. Zhao, W. M. Jackson, W. B. Miller, A, M. Wodtke, and Y. T. Lee, J. Phys. Chem. **95**, 2799 (1991).

[5] R. E. Continetti, D. R. Cyr, R. B. Metz, and D. M. Neumark, Chem. Phys. Lett., **182**, 406 (1991).

[6] B. S. Haynes, Combust. Flame **28**, 113 (1977); J. A. Miller, M. C. Branch, W. J. McLean, D. W. Chandler, M. D. Smooke, and R. J. Kee, Symp. (Int.) Combust. [Proc]., 12th, 673 (1984).

[7] a) K. N. Wong, W. R. Anderson, A. J. Kotlar, and J. A. Vanderhoff, J. Chem. Phys. **81**, 2970 (1984).
b) W. R. Anderson, J. A. Vanderhoff, A. J. Kotlar, M. A. Dewilde, and R. A. Beyer, J. Chem. Phys. **77**, 1677 (1982).
c) R. A. Copeland and D. R. Crosley, Can J. Phys. **62**, 1488 (1984).

[8] K. Schmatjko and J. Wolfram, Ber. Bunsenges. Physik. Chem. **82**, 419 (1978).

[9] R. Holland, D. W. G. Style, R. N. Dixon, and D. A. Ramsay, Nature **182**, 337 (1958).

[10] R. N. Dixon, Phil. Trans. Roy. Soc. (London), A**252**, 165 (1960).

[11] R. N. Dixon, Can. J. Phys., **38**, 10, (1960).

[12] D. E. Milligan and M. E. Jacox, J. Chem. Phys. **47**, 5157 (1967).

[13] P. S. H. Bolman, J. M. Brown, A. Carrington, I. Kopp, and D. A. Ramsay, Proc. R. Soc. Lond. A. **343**, 17 (1975).

[14] V. E. Bondybey and J. H. English, J. Chem. Phys. **67**, 2868 (1977).

[15] T. R. Charlton, T. Okamura and B. A. Thrush, Chem. Phys. Lett. **89**, 98 (1982).

[16] D. R. Woodward, D. A. Fletcher, and J. M. Brown, Mol. Phys. **62**, 517 (1987).

[17] D. Patel-Misra, D. G. Sauder, and P. J. Dagdigian, J. Chem. Phys. **93**, 5448 (1990).

[18] A. Carrington, A. R. Fabris, B. J. Howard, and N. J. D. Lucas, Mol. Phys., **20**, 961 (1971).

[19] a) K. Kawaguchi, S. Saito, and E. Hirota, Mol. Phys. **49**, 663 (1983).
b) K. Kawaguchi, S. Saito, and E. Hirota, Mol. Phys. **55**, 341 (1985).

[20] a) C. E. Barnes, J. M. Brown, A. D. Fackerell, and T. J. Sears, J. Mol. Spectrosc. **92**,485 (1982).
b) J. Werner, W. Seebass, M. Koch, R. F. Curl, W. Urban, and J. M. Brown, Mol. Phys. **65**, 453 (1985).

c) P. B. Davies and I. H. Davis, Mol. Phys. **69**, 175 (1990).

[21] a) D. M. Sonnenfroh, R. G. MacDonald, and K. Liu, J. Chem. Phys. **93**, 1478 (1990).
b) D. G. Sauder, D. Patel-Misra, and P. J. Dagdigian, J. Chem. Phys. **95**, 1696 (1991).
c) L. F. Phillips, I. A. W. Smith, R. P. Tuckett, and C. J. Whitham, Chem. Phys. Lett. **183**, 254 (1991).

[22] H. Okabe, J. Chem. Phys. **53**, 5307 (1970).

[23] B. J. Sullivan, G. P. Smith and D. R. Crosley, Chem. Phys. Lett. **96**, 307 (1983).

[24] R. J. Balla and K. H. Casleton, J. Phys. Chem. **95**, 2344 (1991), and references therein.

[25] a) T. A. Spiglanin, R. A. Perry, and D. W. Chandler, J. Phys. Chem. **90**, 6184 (1986).
b) T. A. Spiglanin and D. W. Chandler, Chem. Phys. Lett. **141**, 428 (1987).

[26] K. -Y. Du and D. W. Setser, Chem. Phys. Lett. **153**, 393 (1988).

[27] M. A. Johnson, M. L. Alexander, and W. C. Lineberger, Chem. Phys. Lett. **112**, 285 (1984).

[28] R. B. Metz, S. E. Bradforth, and D. M. Neumark, Adv. Chem. Phys. **81**, 1 (1992).

[29] D. Proch and T. Trickl, Rev. Sci. Instrum. **60**, 713 (1989).

[30] E. Illenberger, P. B. Comita, J. I. Brauman, H. -P. Fenzlaff, M. Heni, N. Heinrich, W. Koch, and G. Frenking, Ber. Bunsenges Physik. Chem. **89**, 1026 (1985).

[31] R. E. Continetti, D. R. Cyr, and D. M. Neumark, Rev. Sci. Instrum. **63**, 1840 (1992).

[32] J. M. B. Bakker, J. Phys. E **6**, 785 (1973); **7**, 364 (1974).

[33] S. E. Bradforth, E. H. Kim, D. W. Arnold, and D. M. Neumark, J. Chem. Phys. **98**, 800 (1993).

[34] R. N. Dixon, M. J. Trenouth, and C. M. Western, Mol. Phys. **60**, 779 (1987).

[35] J. W. C. Johns, Can. J. Phys. **39**, 1738 (1961).

[36] a) D. Gauyacq, C. Larcher, and J. Rostas, Can. J. Phys. **57**, 1634 (1979).
b) C. Larcher, D. Gauyacq and J. Rostas, J. de Chim. Phys. **77**, 655 (1980).

[37] J. A. Pople, Mol. Phys. **3**, 16 (1960).

[38] J. T. Hougen, J. Chem. Phys. **36**, 519 (1962); J. Chem. Phys. **37**, 403 (1962).

[39] M. H. Alexander and H.-J. Werner, submitted to J. Chem. Phys.

[40] a) M. A. Johnson, J. Rostas, and R. N. Zare, Chem. Phys. Lett. **92**, 225 (1982).
b) M. A. Johnson, R. N. Zare, J. Rostas and S. Leach, J. Chem. Phys. **80**, 2407 (1984).

[41] R. E. Continetti, Ph. D. Thesis, University of California, Berkeley, 1989.

[42] T. N. Kitsopoulos, Ph. D. Thesis, University of California, Berkeley, 1991.

[43] R. N. Zare, Mol. Photochem. **4**, 1 (1972).

[44] M. W. Chase, Jr., C. A. Davies, J. R. Downey, Jr., D. J. Frurip, R. A. McDonald, and A. N. Syverud, *JANAF Thermochemcial Tables, 3rd Edition* (American Chemical Society and American Institute of Physics, New York, 1986).

[45] S. T. Gibson, J. P. Greene, and J. Berkowitz, J. Chem. Phys. **83**, 4319 (1985).

[46] K. M. Ervin and P. B. Armentrout, J. Chem. Phys. **86**, 2659 (1987); J. W. Sutherland and J. V. Michael, J. Chem. Phys. **88**, 830 (1988).

[47] K. Uno, T. Hikida, A. Hiraya, and K. Shobatake, Chem. Phys. Lett. **166**, 475 (1990).

[48] K. Yamada, J. Mol. Spectrosc. **79**, 323 (1980).

[49] P. Misra, C. W. Mathews, and D. A. Ramsay, J. Mol. Spectrosc. **130**, 419 (1988).

[50] P. Carskey, J. Kuhn, and R. Zahradnik, J. Mol. Spectrosc. **55**, 120 (1975).

[51] B. F. Minaev, N. M. Ivanova, and Z. M. Muldahmetov, Spectrosc. Lett. **22**, 901 (1989).

[52] D. P. de Brujin and J. Los, Rev. Sci. Instrum. **53**, 1020 (1982).

Chapter 6

Photodissociation of the CH_2NO_2 Radical

6.1 Introduction

The study of molecular photodissociation processes has developed into one of the most productive and valuable areas of chemical physics in recent years.[1] This body of work has led to a detailed understanding of both the spectroscopy and dynamics associated with the dissociative electronic states in a wide variety of molecules. Moreover, photodissociation experiments provide one of the most direct means of determining bond dissociation energies. However, the vast body of these investigations have been undertaken on stable molecules; only a handful of photodissociation experiments on reactive free radicals have been performed to date. For most radicals, the excited states in general and the dissociative states in particular are poorly characterized, and the bond dissociation energies are not known very accurately. Hence, these species present an inviting target for photodissociation experiments.

This situation provided the motivation for the development in our laboratory of a photodissociation experiment especially well-suited for studies of free radicals, based on a novel form of photofragment translational spectroscopy. In the standard version of photofragment translational spectroscopy,[2] a molecular beam is photolyzed, and one measures the translational energy and angular distributions of the resulting fragments by a variety of techniques.[3] Thus far, the primary challenge met in extending this method to reactive free radicals[4,5] has been the difficulties involved in generating a clean, well-characterized molecular beam of these species. Our experiment uses a different approach in order to eliminate this problem. We produce mass selected, neutral free radicals from photodetachment of a fast (6-8 keV) beam of the corresponding negative ions. The resulting radicals are then photodissociated, and the photofragments are detected with high efficiency. We can map out the dissociative excited states of a given radical by

determining how the photodissociation cross section varies with dissociation photon energy. Subsequently, we perform dynamics experiments at selected wavelengths, using a coincidence detection scheme to determine the masses, kinetic energy release (KER), and recoil angle with respect to the laser polarization for each pair of photofragments formed. From this, we obtain photofragment translational energy and angular distributions as well as the yield for each fragmentation channel. Previously, we have applied this method to diatomic and triatomic species.[6,7,8,9] In this paper, we report our first results for a polyatomic molecule, the nitromethyl radical CH_2NO_2.

The nitromethyl radical is thought to be a primary intermediate in the combustion of nitromethane, CH_3NO_2, formed by simple hydrogen abstraction.[10,11] Past experimental work on CH_2NO_2 is fairly limited. Chachaty and coworkers[12,13] obtained electron spin resonance (ESR) spectra of the nitromethyl radical following γ irradiation of nitromethane at 77 and 195 K. Matrix isolation work using isotopic substitution performed by Jacox[11,14] resulted in the identification of all but one of the twelve ground state vibrational frequencies by infrared absorption, and the subsequent determination of the planar, C_{2v} geometry of the ground state. Jacox also ascertained that the threshold for photodissociation lies between 280 and 300 nm. The photoproducts resulting from dissociation at 280 nm were determined to be H_2CO and NO. It was noted that these could be either the nascent fragments or the products from caged recombination of $CH_2 + NO_2$ formed by C-N fission. More recently, Metz *et al.*[15] recorded the photoelectron spectrum of the nitromethyl anion ($CH_2NO_2^-$). They determined the electron affinity of CH_2NO_2 to be 2.475 ± 0.010 eV, and located an excited electronic state (the $1\ ^2A_2$ state) 1.591 eV above the $\tilde{X}\ ^2B_1$ ground state. In addition, *ab initio* calculations were performed to assist in identifying the one remaining unknown ground-state vibrational frequency (for the torsional mode) from the experimental data. Other theoretical work on the nitromethyl radical includes two previous *ab initio* calculations

completed at the single and multiconfigurational self-consistent field (MCSCF) levels by McKee,[16,17] which attempted to identify and characterize the lowest electronic states.

In contrast to the relatively few studies of CH_2NO_2, the photodissociation of the analogous closed shell species, nitromethane, has been thoroughly studied. Depending on the method of excitation, three major pathways can contribute to dissociation:

$$CH_3NO_2 \rightarrow CH_3NO + O \qquad (6.1)$$

$$\rightarrow CH_3O + NO \qquad (6.2)$$

$$\rightarrow CH_3 + NO_2. \qquad (6.3)$$

The ultraviolet photodissociation of CH_3NO_2 in the wavelength range from 190 to 220 nm has been investigated in the gas phase[18,19,20,21] and also in solution.[22] This is a $\pi^* \leftarrow \pi$ excitation and leads exclusively to C-N bond fission, Eqn. (6.3). Butler[20] has proposed two pathways to these products. The primary pathway is adiabatic, involving crossing onto a $\sigma^* \leftarrow$n surface of the same symmetry as the initially excited state and leading to the production of NO_2 in the *1* $^2B_2$ electronic state. At the highest energies in this range a diabatic dissociation pathway becomes energetically allowed, producing NO_2 in the *2* $^2B_2$ electronic state. Schoen *et al.*[23] have reported picosecond photodissociation studies of nitromethane at wavelengths between 238 nm and 337 nm, again finding only C-N bond fission in this region. The amount of electronically excited NO_2 was shown to increase with increasing photon energy, as determined by the ratio of fluorescence yield to laser-induced fluorescence yield measurements on the photoproducts. In contrast, infrared multiphoton dissociation (IRMPD) studies by Wodtke et al.[24,25] determined that when dissociation occurs on the ground state surface, there is a competition between the C-N bond fission process and rearrangement to form methyl nitrite, followed by nitric oxide elimination, as illustrated by Eqn. (6.2). Beijersbergen *et al.*[26] have recently published a study in which the dissociative charge transfer neutralization of the nitromethane radical cation is described. This technique results in the formation of neutrals with excitation energies centered at 5.9 eV (Na charge exchange) or 7.2 eV (Cs charge exchange). In

their study, they determined the major channels to be N-O and C-N fission [Eqns. (6.1) and (6.3)], while nitric oxide elimination [Eqn. (6.2)] was observed as a minor channel.

The work on the nitromethyl radical reported here was largely motivated by our interest in learning how its photodissociation dynamics might differ from nitromethane. The matrix results obtained by Jacox[11] are particularly intriguing in this regard. If the H_2CO + NO photoproducts observed by Jacox are indeed nascent fragments, this would indicate that the primary UV photochemistry of CH_2NO_2 involves NO elimination rather than C-N bond fission as in CH_3NO_2. The collision-free environment in our experiment makes it possible to clearly identify the primary photofragments.

In the following section details of the experiment are discussed, while results and analysis of the photodissociation of the nitromethyl radical are presented in § 6.3. We unambiguously identify and characterize the nascent photoproducts of the ultraviolet photodissociation of CH_2NO_2. In doing so, we identify the following two dissociation channels: (I) $CH_2NO_2 \rightarrow CH_2NO + O$ and (II) $CH_2NO_2 \rightarrow H_2CO + NO$. The dissociation process that involves simple C-N bond fission, (III) $CH_2NO_2 \rightarrow CH_2 + NO_2$, is not observed. For the two observed channels, the branching ratio, expressed as the quotient (I)/(II), changes from 1.19 to 1.76 on varying the photodissociation wavelength from 270 nm to 240 nm. In addition, the translational energy distribution, $P(E_T)$, and the recoil anisotropy parameter as a function of translational energy release, $\beta(E_T)$, are determined for each channel. These data imply fundamental differences in the dissociation dynamics for the two observed channels, providing insight into the dissociative mechanism involved in each channel. In § 6.4, we discuss the implications of these observations on the dissociation mechanisms for the various channels. It is noted that a preliminary account of this work has been published elsewhere.[27]

6.2 Experimental Approach

Due to their open shell nature, free radicals typically have positive electron affinities. Our experiment utilizes this property in order to generate a pure, well-characterized packet of free radicals via photodetachment of the corresponding mass-selected negative ion. As an added advantage, the high laboratory frame kinetic energy imparted to the radicals during their acceleration prior to detachment allows the efficient collection and detection of photofragments simply by striking the face of a microchannel plate (MCP) detector.

A schematic of the fast radical beam photodissociation spectrometer used in this work is shown in Figure 2.1. Details concerning the design of the fast radical beam spectrometer and the photofragment detection scheme are given in Chapter 2, but there are several aspects of the experiment particularly germane to the present study which will be described here.

The basic experiment is as follows:

$$CH_2NO_2^- \xrightarrow{h\nu_1} CH_2NO_2 + e^- \xrightarrow{h\nu_2} \text{fragments.} \qquad (6.4)$$

The nitromethyl anion, $CH_2NO_2^-$, is prepared following the procedure of Metz *et al.*,[15] using a pulsed molecular beam ion source similar to that developed by Lineberger and coworkers.[28] We first bubble a mixture of 10% NF_3 in N_2 through neat nitromethane (Eastman) at room temperature and 18 PSIG total pressure. A pulsed molecular beam valve operating at 60 Hz introduces this mixture to the source vacuum chamber. Immediately outside the nozzle orifice, the supersonic expansion is crossed with a continuous 1 keV electron beam. Low energy secondary electrons formed by electron impact ionization result in the production of F^- via dissociative attachment to NF_3. This fluorine anion abstracts a proton from nitromethane, forming $CH_2NO_2^-$, which undergoes cooling during the remainder of the expansion. Ions pass through a 3 mm diameter

skimmer and are accelerated to 6 keV. They are then rereferenced to ground potential by the use of a potential 'switch',[29] and mass selected via a Bakker-type time-of-flight mass spectrometer.[30] After passing through a 1-mm beam-defining aperture, the anions are then photodetached using a tunable, pulsed, excimer-pumped-dye laser system, timed to intersect the mass of interest. Detachment laser energies are 30-50 mJ/pulse and the laser is loosely focused with a 1 m focal length lens, 50 cm prior to intersecting the ion packet. Attenuation of the ion beam by the photodetachment light pulse can be as high as 50%.

The anions are vibrationally cold, and the detachment wavelength is chosen to avoid the production of vibrationally excited radicals, as is discussed in § 1.4.2. This is done with the aid of Figure 6.1, which shows the fixed frequency photoelectron spectrum of $CH_2NO_2^-$ obtained previously using a separate apparatus in our laboratory. This photoelectron spectrum was taken using a photon energy of 3.49 eV (λ_{det} = 355 nm), and revealed the electron affinity of CH_2NO_2 to be 2.475 eV.[15] In this spectrum, the measured kinetic energy of the electrons represents the photon energy minus the difference in energy between the ground vibrational state of the anion and vibrational levels in the neutral. Hence, in Figure 6.1 the peak at highest electron kinetic energy corresponds to the ground vibrational level in the neutral. The spectrum consists of a strong vibrational progression in the ν_3 symmetric stretch mode. We therefore photodetach the anions at a photon energy of 2.59 eV (λ_{det} = 479 nm), just above the electron affinity of CH_2NO_2. As illustrated by the location of the arrow in Figure 6.1, this is not energetic enough to excite the active ν_3 mode of the neutral.

Following detachment, all remaining negative ions are deflected out of the beam. The radicals must pass through a second 1 mm collimating aperture and are then

---

Figure 6.1 Photoelectron spectrum of nitromethyl anion obtained using a photon energy of 3.49 eV (λ_{det} = 355 nm). The prominent ν_3 progression is labeled, and the position of the arrow relative to the origin peak indicates the energy available upon photodetachment using 2.59 eV photons (λ_{det} = 479 nm).

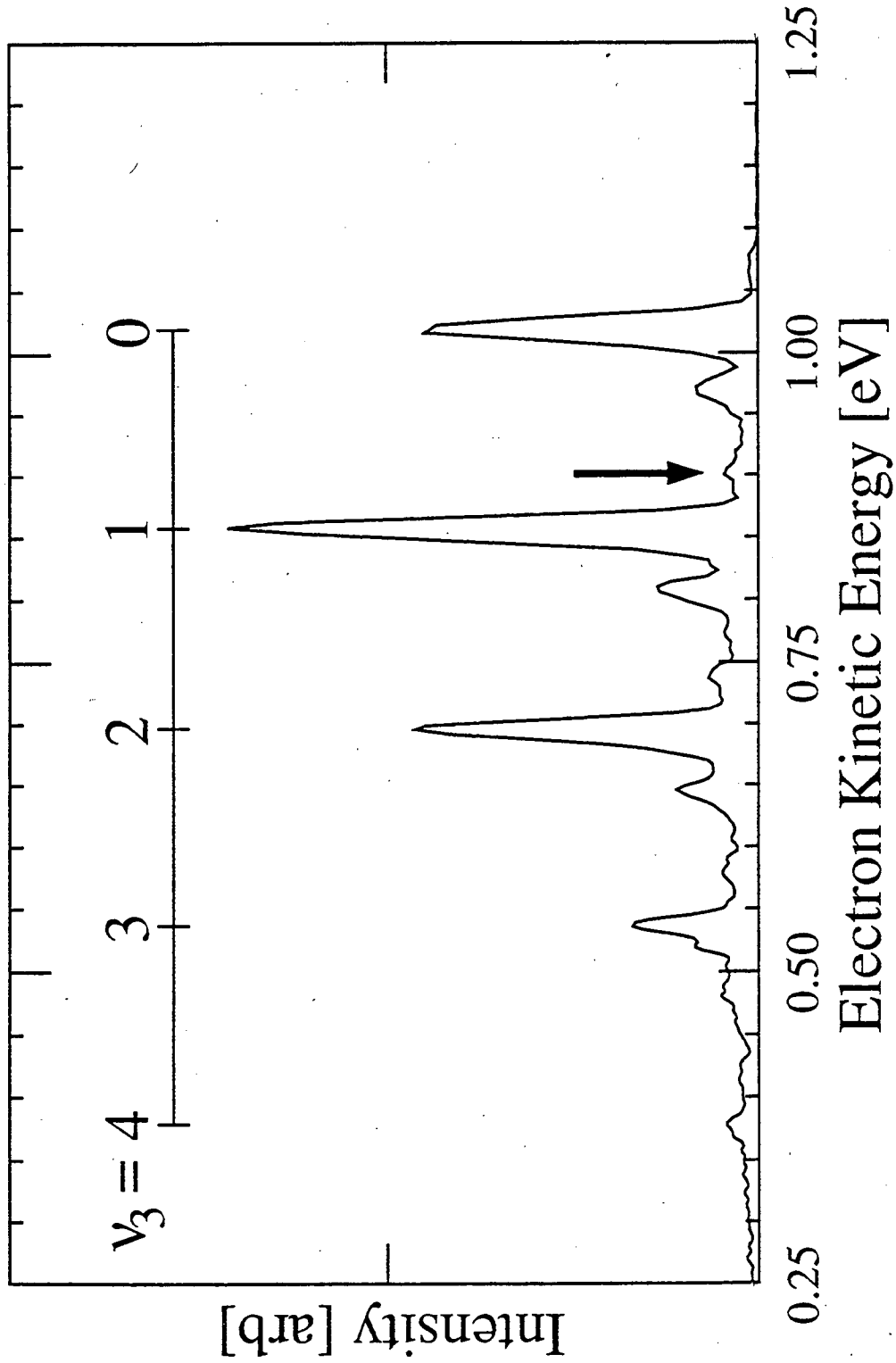

Intensity [arb]
$\nu_3 = 4$
3
2
1
0
0.25
0.50
0.75
1.00
1.25
Electron Kinetic Energy [eV]

Figure 6.1

intersected in the photodissociation region by the horizontally polarized, frequency-doubled output of a second pulsed excimer-pumped-dye laser, loosely focused by a 2 m lens. The photofragments are detected with one of two microchannel plate detectors, lying 0.67 and 1.0 m downstream of the photodissociation region respectively; the closer detector is movable and is removed from the beam path when the more distant detector is in use. Photofragments typically have enough recoil kinetic energy to scatter out from the beam axis, thereby missing the blocking strips placed across the center of the detectors (3 or 8 mm wide, respectively) to prevent undissociated radicals from impinging on the active area. The laboratory kinetic energies of the photofragments are generally > 1 keV, ensuring a high (~50%) detection efficiency for those that strike the active area. The detector at 0.67 m is used for photodissociation cross section measurements; the total photofragment yield is monitored as the dissociation laser wavelength is varied. In general, this yields an approximate mapping of the dissociative electronic states of a given radical.

The photodissociation dynamics experiments are done using a more complex photofragment coincidence detection scheme. The laser is tuned to a wavelength at which photodissociation occurs, and the photofragments are detected with the second detector, 1 m from the photodissociation region. This detector is a two-particle time- and position-sensitive detector, similar in principle to the detector first developed by de Bruijn and Los,[31] although we use a coincidence wedge-and strip anode (C-WSA)[8,32,33] rather than capacitive charge division for the position sensing. As illustrated in Figure 6.2, for each photodissociation event we measure the position of both fragments at the detector, denoted (x_1, y_1) and (x_2, y_2), as well as the time delay between their arrival, τ. This enables us to determine the photofragment masses, kinetic energy release, and scattering

---

Figure 6.2 Schematic of a typical dissociation event, with the measured quantities necessary for calculation of fragment mass ratio, KER and recoil angle with respect to laser polarization labeled.

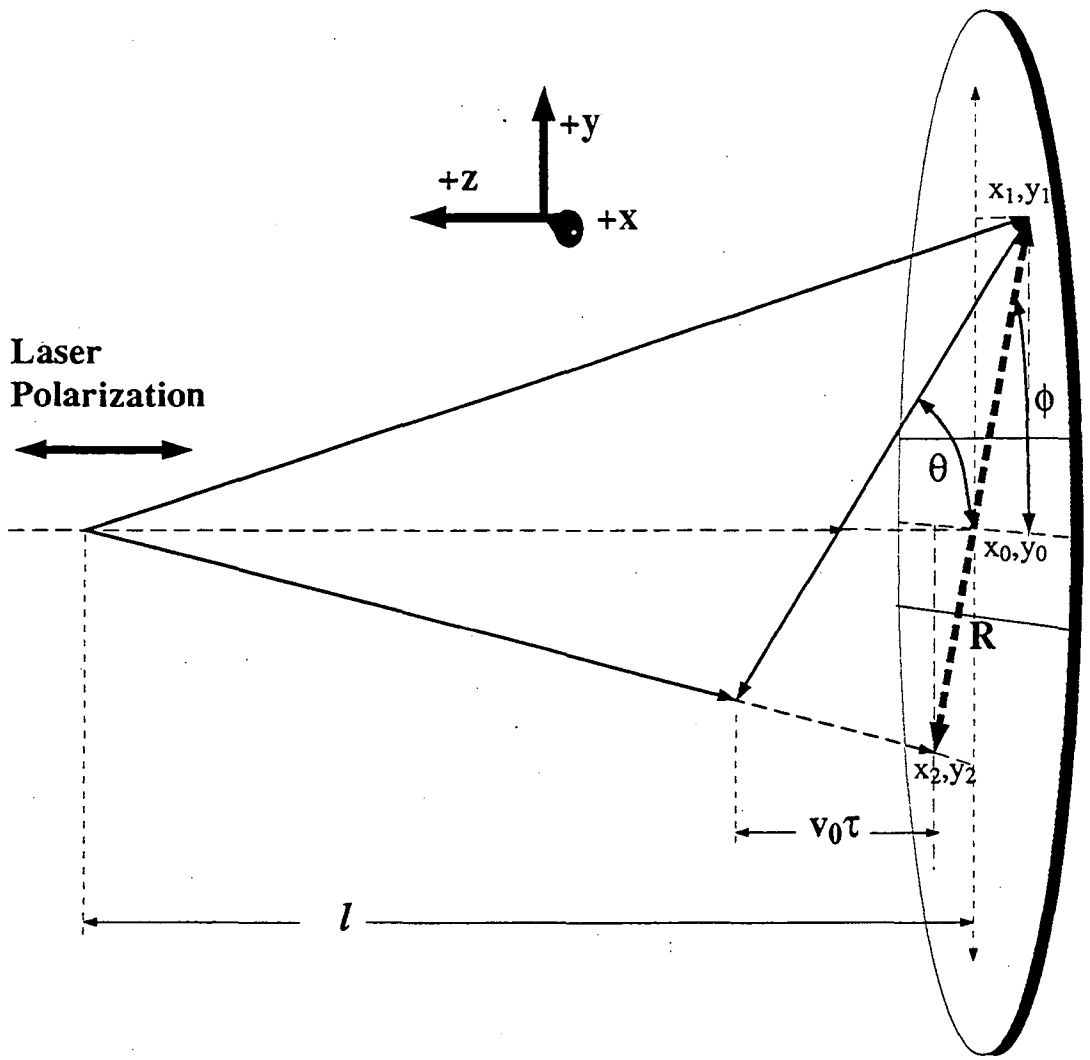


Figure 6.2

angle with respect to the dissociation laser polarization direction for each event, as fully discussed in § 3.2. Because this is a coincidence measurement, we can process at most one event per laser pulse; ion densities in this experiment are sufficiently low so that this was not a problem. By summing over large numbers of dissociation events (2–8 × 10^4), we have obtained branching ratios as well as translational energy and angular distributions for each channel observed in this experiment.

The mass ratio of the two fragments is found by first determining $r_i = \sqrt{(x_i - x_0)^2 + (y_i - y_0)^2}$, the displacement of each fragment from the location of the radical beam center (x_0, y_0) on the detector face. The mass ratio of the two fragments is then given approximately by the inverse of the ratio of these displacements:

$$\frac{m_1}{m_2} = \frac{r_2}{r_1} \cdot \left[1 - \left(\frac{v_0 \tau}{L}\right)\right] \cong \frac{r_2}{r_1} \tag{6.5}$$

Once the fragment masses are determined, Eqns. (6.6) and (6.7), originally derived by De Bruijn and Los,[31] are used to obtain the center-of-mass kinetic energy release (KER) and fragment recoil angle θ, measured with respect to the polarization vector of the laser, for each dissociation event.

$$\theta = \tan^{-1}\left(\frac{R}{v_0 \tau}\right) \tag{6.6}$$

$$KER = E_0 \cdot \left(\frac{m_1 \cdot m_2}{M^2}\right) \cdot \frac{\left[(v_0 \tau)^2 + R^2\right]}{L^2} \cdot \left[1 + 2\left(\frac{m_2 - m_1}{M}\right) \cdot \frac{v_0 \tau}{L}\right] \tag{6.7}$$

Here $R = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$ is the relative separation of the two fragments, M is the parent radical mass, m_1 and m_2 the fragment masses striking the upper and lower half of the detector, respectively, L the flight length, and E_0 and v_0 the primary beam

energy and velocity, respectively. τ is taken to be positive if the upper fragment strikes the detector first.

In Eqn. (6.5) the chief source of uncertainty is the dependence of r_1 and r_2 on an explicit knowledge of (x_0, y_0), the detector coordinates where the center of mass of the dissociating system would be found (i.e. where the parent radical, if it had not dissociated, would strike the detector). The radical beam is about 1 mm FWHM in diameter, resulting in a fragment mass resolution ($m/\Delta m$) of about 15 for the results reported here. In contrast, Eqns. (6.6) and (6.7) do not require an explicit knowledge of (x_0, y_0), and therefore the center-of-mass uncertainty does not contribute to the energy and scattering angle, assuming the product masses can be identified unambiguously. The measured spatial resolution is on the order of 75 μm full-width-at-half-maximum in y and 130 μm in x, while timing resolution is about 0.5 ns. Energy resolution as high as 10 meV has been demonstrated during calibrations using the predissociation of O_2 through the Schumann-Runge band,[9] but is approximately 30 meV (0.69 kcal/mol, 240 cm^{-1}) in these studies.

As discussed in Chapter 3, upon the completion of a photodissociation dynamics experiment, we first separate the data into different mass channels following Eqn. (6.5). For each channel we then construct a two-dimensional array in KER and theta into which each coincident dissociation event is binned. The channel specific energy-angle arrays, denoted here as $\mathcal{A}(E_T,\theta)$, are determined using Eqns. (6.6) and (6.7). From this, we obtain the true joint energy and angular distribution $\mathcal{P}(E_T,\theta)$ for each product channel by dividing by the "detector acceptance function" $\mathcal{B}(E_T,\theta)$:

$$\mathcal{P}(E_T,\theta) = \mathcal{A}(E_T,\theta) / \mathcal{B}(E_T,\theta). \qquad (6.8)$$

The detector acceptance function accounts for the finite acceptance of the detector as a function of E_T and theta. For example, fragments with lower recoil energies and/or values of theta close to 0 or 180 degrees may not strike the detector because of the beam block, while some high energy fragments at values of theta close to 90 degrees miss the detector

because of its finite size. The straightforward numerical determination of $\mathcal{I}(E_T,\theta)$ is discussed in § 3.2.3.

As mentioned above, the dual-wedge-and-strip anode requires an 8 mm wide horizontal blocking strip which shields the microchannel plates over a small region above the interface of the two anodes to prevent the undissociated radical beam from impinging on the detector face. However, it also prevents photofragments with very low center-of-mass kinetic energy from being detected. This effect is magnified in the case of fragments with unequal masses. The heavy fragment scatters out of the beam more slowly than the light fragment and is more likely to be blocked, but both fragments must be detected to be recorded as a coincidence event. In order to improve coincidence detection of low energy, unequal mass fragments, the detector can be positioned slightly off-center vertically with respect to the radical beam axis. In the experiments reported here, such a product channel was identified early on, and the data were then collected with the center of the detector 1.9 mm lower than the radical beam axis. This enables heavy fragments that would have otherwise struck the blocking strip to clear the top edge. Because the correlated light fragments have a greater recoil velocity due to momentum conservation, they are able to travel the correspondingly larger distance necessary to clear the bottom of the blocking strip. The effect of this offset on the detector acceptance function is included in the analysis of the data.

6.3 Results and Analysis

Photodissociation cross section measurements of the CH_2NO_2 radical were taken across the wavelength range from 240 to 270 nm. The photodissociation cross section scans appeared to be structureless, with the cross section peaking near 240 nm and decreasing monotonically by about a factor of five at 270 nm. Most of the photodissociation dynamics data were taken at 240 nm, corresponding to a photon energy of 119.1 kcal/mol, where the dissociation signal was most intense. Supplementary data

were recorded using longer wavelengths of 255 nm and 270 nm, corresponding to photon energies of 112.1 and 105.9 kcal/mol, respectively.

6.3.1 Identification of Photodissociation Channels

Prior to any further analysis, it is necessary to determine which fragmentation channels occur in the photodissociation of CH_2NO_2. Many channels are energetically allowed in the present case of CH_2NO_2, a six atom radical to which ~5 eV excitation energy is imparted. Even if only one dissociation channel is active, the analysis of time and position data to obtain the translational energy distribution for that channel depends on first identifying the masses of the photofragments produced. The presence of two or more active dissociation channels demands the ability to determine which channel *each* dissociation event corresponds to.

The photofragment mass spectrum for CH_2NO_2 at 240 nm is shown in Figure 6.3. Note that this plot is constructed from the data by determining the photofragment mass ratio in each event, and thus the fraction of the total mass M (= 60 amu) of the parent radical for each fragment, resulting in a mirror image about M/2. The mass spectrum indicates the presence of two dissociation channels. Although the mass resolution is limited, it appears that in one channel, the fragment masses are 16 and 44 amu, while in the other, both fragments have mass 30 amu. This would indicate that the two dissociation channels are (I) $CH_2NO + O$ and (II) $H_2CO + NO$.

In order to verify these assignments, the photofragment mass spectrum of CD_2NO_2 was also obtained at 240 nm, and the results are superimposed on the CH_2NO_2 mass

---

Figure 6.3 Photofragment mass spectrum from the 240 nm dissociation of the nitromethyl radical (CH_2NO_2; solid line) and the d_2-nitromethyl radical (CD_2NO_2; dashed line). Areas under the mass peaks do not accurately reflect the mass channel branching ratio because they are uncorrected for detector acceptance (see text).

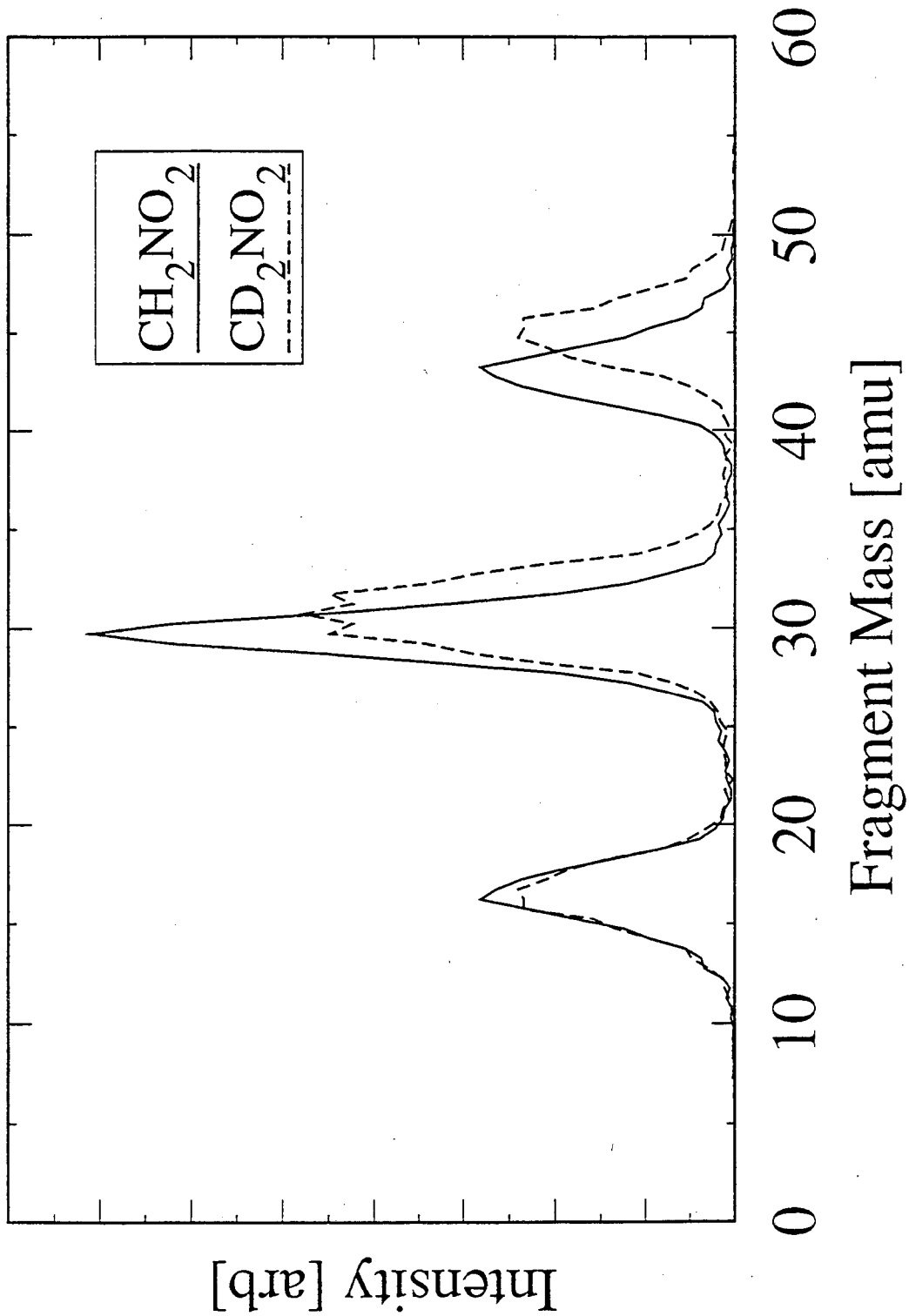

Intensity [arb]
0
10
20
30
40
50
60
Fragment Mass [amu]
$\frac{CH_2NO_2}{CD_2NO_2}$

Figure 6.3

spectrum in Figure 6.3. The differences in the two spectra upon deuteration are the following: the mass peak centered at 16 amu remains fixed while the mass peak at 44 amu shifts to 46 amu, and the mass peak centered at 30 amu broadens towards higher mass and appears to be centered at 31 amu. The first difference shows that the photofragments with mass ratio close to 3:1 indeed corresponds to (I) $CH_2NO_2 \rightarrow CH_2NO + O$, since there can be no H/D atoms in the lighter mass fragment. Although the mass fragments from channel (III), $CH_2NO_2 \rightarrow CH_2 + NO_2$, (14 and 46) are similar to those of channel (I), upon isotopic substitution the lower mass peak would shift from 14 to 16 amu, while the heavier mass fragment would remained fixed at 46 amu. We therefore estimate an upper limit for the presence of channel (III) products of 4 %. The second observed difference confirms the presence of channel (II), $CH_2NO_2 \rightarrow H_2CO + NO$, since the dissociation products from CH_2NO_2 should both have a mass of 30 amu while the deuterated analogue, CD_2NO_2, should have dissociation products with masses of 32 and 30 amu respectively. The deuterated mass spectrum eliminates another energetically feasible channel with comparable masses corresponding to O_2 elimination, $CH_2NO_2 \rightarrow H_2CN + O_2$. The fragment masses of this excluded channel would change from 28:32 to 30:32 upon deuteration resulting in an expected narrowing of the central peak of the mass spectrum, contrary to our observation. This channel is assigned a similar estimated upper limit of 4%.

While the mass spectra we have obtained do not show evidence for the presence of any dissociation channels other that the two identified above, we are unable to comment on channels involving extreme mass ratios because of the unfavorable kinematics involved. When two fragments have very disparate masses, as in the case of hydrogen bond fission, either the light fragment will recoil outside the maximum radius of the detector, or the heavy fragment will fail to clear the blocking strip, or both, depending on the recoil energy and angle of the event. Therefore, a conceivable channel that would go undetected in our experiment involves loss of a hydrogen atom, forming $HCNO_2 + H$. Although there is no

literature value for the C-H bond energy of nitromethyl radical, we expect it to be close to the 100 kcal/mol value found for the C-H bond in CH_3NO_2,[15,34] largely unaffected by the open shell nature of the radical. This is quite high compared to the available energy in these experiments and the energies of the channels which are known to be occurring (see below), so it is doubtful that this channel is active.

6.3.2 Translational Energy and Angular Distributions

Once the photodissociation channels are identified, we can determine the center-of-mass translational energy and angular distributions for each channel. The energy and angular distribution, $\mathcal{P}(E_T,\theta)$, for a one-photon dissociation can be expressed as

$$\mathcal{P}(E_T,\theta) = P(E_T)[1+\beta(E_T)P_2(\cos\theta)], \qquad (6.9)$$

where θ is the angle between the photofragment recoil vector and the electric vector of the dissociation laser, $P_2(\cos\theta)$ is the second Legendre polynomial, $\beta(E_T)$ is the anisotropy parameter as a function of translational (or kinetic) energy, and $P(E_T)$ is the angle-integrated translational energy distribution. The anisotropy parameter β can range from $\beta = +2$ to $\beta = -1$, corresponding to a $\cos^2\theta$ distribution and a $\sin^2\theta$ distribution of recoil vectors, respectively.[35]

$\mathcal{P}(E_T,\theta)$ is derived from the data as described in § 6.2, and, in greater detail, § 3.2.4. The photofragment anisotropy parameter $\beta(E_T)_i$ is found for each kinetic energy interval $(E_i-\Delta E/2, E_i+\Delta E/2)$ by performing a least squares fit to Eqn. (6.9). The ΔE intervals chosen for this procedure were 4.6 kcal/mol for channel (I) and 9.2 kcal/mol for channel (II). Only when the data becomes very sparse (i.e. in the 'tail' of the KER distribution near the energetic limit for a given channel) do we find that this method of analysis breaks down. In these restricted regions, containing very little data, we cannot determine $\beta(E_T)$, and it is assumed to be zero (implying an isotropic recoil angular distribution). The final step in this direct inversion analysis is to find the angle-integrated

translational energy distribution $P(E_T)$ at the highest energy resolution. Using the appropriate $\beta(E_T)$ as determined above, we determine $P(E_T)$ for energy intervals of 0.575 kcal/mol in channel (I) and 1.150 kcal/mol in channel (II). These intervals are comparable to the instrumental resolution.

Table 6.1 shows the energies relative to the nitromethyl radical of the two observed product channels in addition to the absent C-N bond fission channel. The heat of formation of CH_2NO_2 is taken to be 30.4 ± 3 kcal/mol,[15] while the heat of formation for CH_2NO has been calculated by Melius[36] to be 43 ± 6 kcal/mol. All other thermochemical data are relatively well known and are obtained from Reference 34.

Figure 6.4 shows the $P(E_T)$ distribution of channel (I), $CH_2NO_2 \rightarrow CH_2NO + O$, at the three wavelengths used in this study, 240, 255 and 270 nm. All three of these distributions are peaked close to zero KER, with the 240 nm distribution peaking furthest away at 5-8 kcal/mol. The $P(E_T)$ distributions obtained at the other wavelengths peak closer to zero, but larger statistical uncertainties make the exact determination of their maxima difficult. For this channel the lowest observable KER, because of the presence of the blocker, is 2.0 kcal/mol. The observed mean KERs are 11.4, 11.0, and 10.8 kcal/mol for 240, 255 and 270 nm photodissociation, respectively. These values are upper bounds for the true mean KER, because of the unobserved fragments with a KER of less than 2.0 kcal/mol. Barring the presence of an excursion in the $P(E_T)$ distribution below 2.0 kcal/mol, the overestimate in the mean KERs listed above is less than 10%. The maximum kinetic energy release in the 240 nm data, for which the statistical error bars are smallest, is 44±1 kcal/mol. This observation, combined with the well known $\Delta_f H^0_{(0\,K)}$ of an oxygen atom (59.0 kcal/mol)[34] and the $\Delta_f H^0_{(0\,K)}$ for CH_2NO_2 given by Metz *et al.*,[15] places an upper bound on the $\Delta_f H^0_{(0\,K)}$ for CH_2NO of 47±3 kcal/mol. This upper bound is

---

Figure 6.4 $P(E_T)$ distribution plots for the $CH_2NO + O$ photodissociation channel taken at the three labeled wavelengths. Error bars represent 1 σ uncertainty.

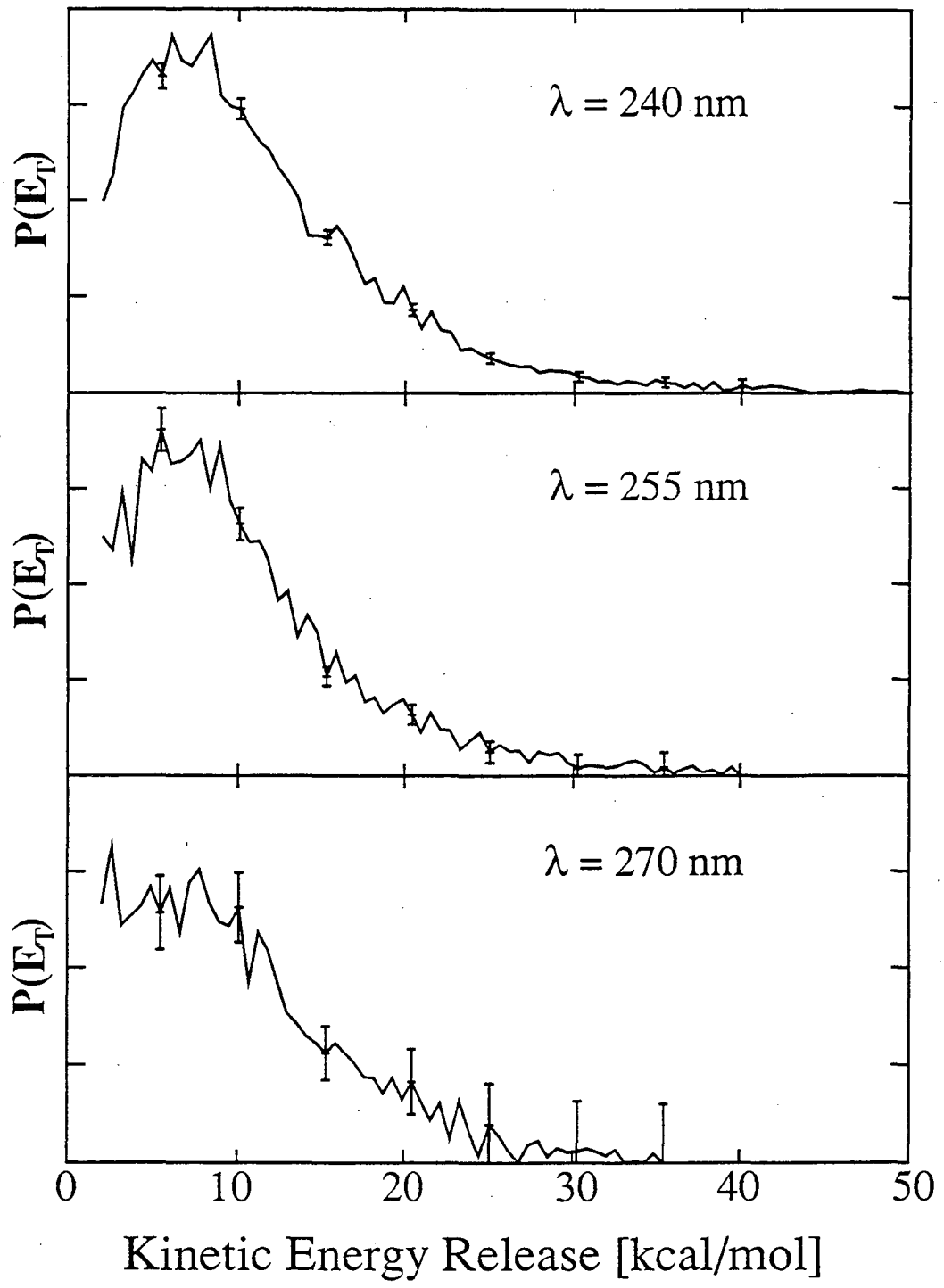

λ = 240 nm
λ = 255 nm
λ = 270 nm
P(E_T)
P(E_T)
P(E_T)
0
10
20
30
40
50
Kinetic Energy Release [kcal/mol]

Figure 6.4

Product Channel	Masses [amu]	D_0 [kcal/mol]	$h\nu - D_0$ [kcal/mol]		
			240 nm	255 nm	270 nm
(I) $CH_2NO + O$	44 + 16	72±6	47	40	34
(II) $H_2CO + NO$	30 + 30	−33.7±3	152.8	145.8	139.6
(III) $CH_2 + NO_2$	14 + 46	71.2±3	47.9	40.9	34.7

Table 6.1 Some possible product channels for the ultraviolet photodissociation of the nitromethyl radical. On the right, the energy available to the products following dissociation at the three wavelengths used in this work are listed. Heats of formation for CH_2NO_2 and CH_2NO are taken from Metz *et al.*[15] and Melius,[36] respectively, while all other thermochemical information is obtained from Ref. 34. Channels (I) and (II) were observed as major channels in these experiments, while channel (III), corresponding to C-N bond fission, was found to be absent.

---

very close to the $\Delta_f H^0_{(0\ K)}$ for CH_2NO of 43±6 kcal/mol given by Melius.[36] It is well below the upper bound of $\Delta_f H^0_{(298\ K)}$ (CH_2NO) = 99 kcal/mol given by Beijersbergen *et al.*;[26] however, in their calculations they use the most probable kinetic energy release rather than the maximum observed kinetic energy release. This, coupled with their assumption that all excess energy is measured as kinetic energy, results in a very conservative upper bound.

Figure 6.5 shows the $P(E_T)$ of channel (II), $CH_2NO_2 \rightarrow H_2CO + NO$, for 240, 255 and 270 nm photolysis. All three energy distributions have a broad peak near 55 kcal/mol, with a mean KER of 58 kcal/mol. High-energy tails in the $P(E_T)$ plots extend up to about 20 kcal/mol below the energetic limit of this channel, which is 153 kcal/mol at 240 nm. Upon changing photon energies, the mean KER changes very little; the major effect

---

Figure 6.5 $P(E_T)$ distribution plots for the $H_2CO + NO$ photodissociation channel taken at the three labeled wavelengths. Error bars represent 1 σ uncertainty.

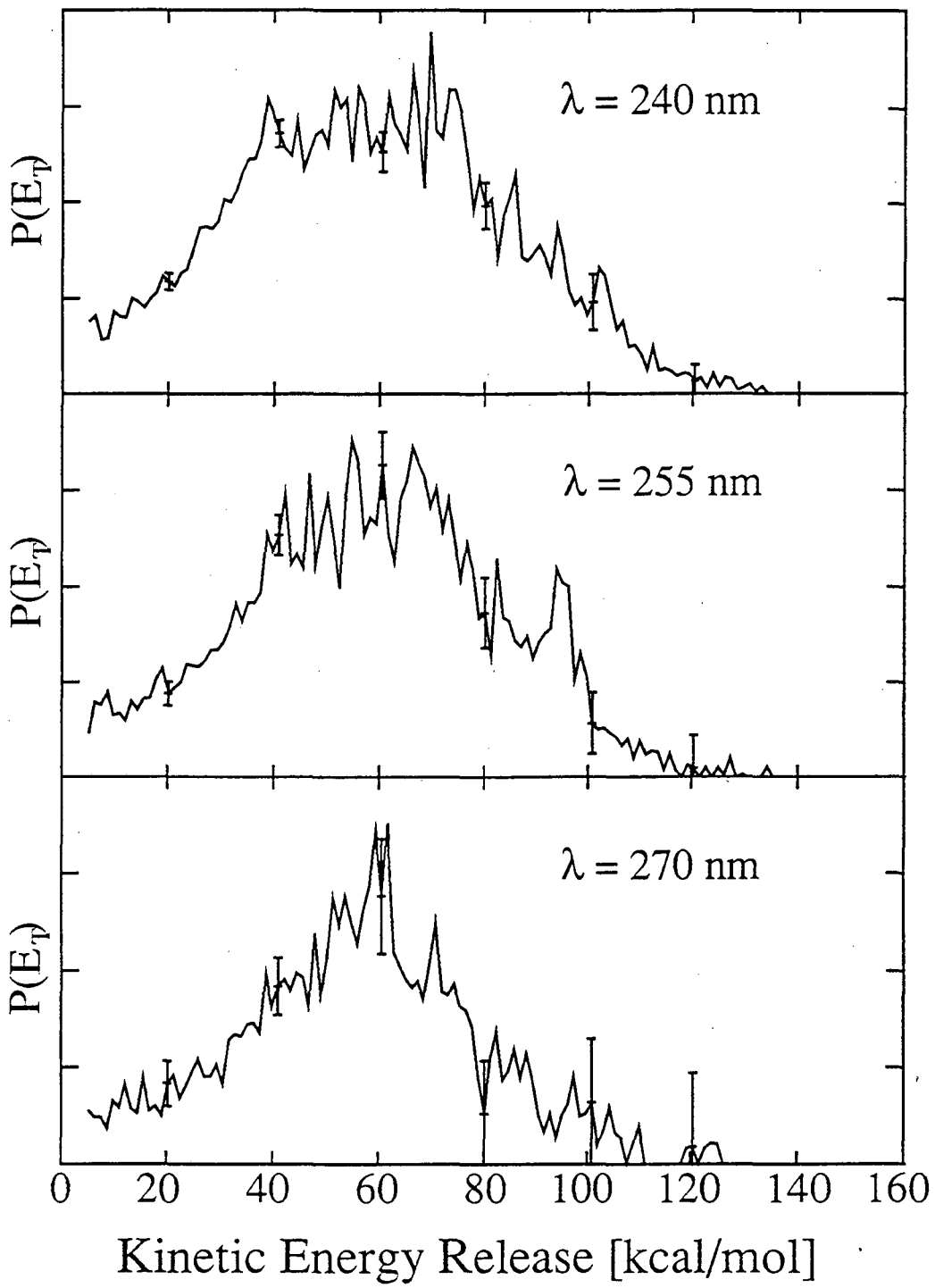

λ = 240 nm
λ = 255 nm
λ = 270 nm
P(E_T)
P(E_T)
P(E_T)
0
20
40
60
80
100
120
140
160
Kinetic Energy Release [kcal/mol]

Figure 6.5

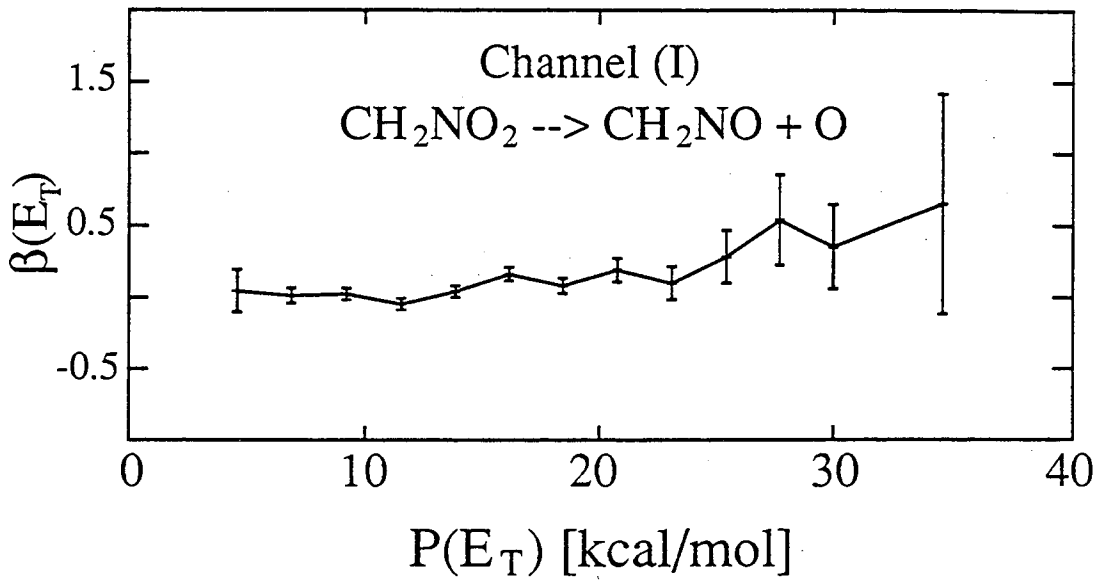
instead is that the higher wavelength $P(E_T)$ distributions are narrower, being more strongly peaked near the mean KER.

Figure 6.6 is a plot of the fragment recoil anisotropy parameter, β, as a function of recoil energy, for channels (I) and (II) as determined from the 240 nm data. β for channel (I) appears to be essentially zero until the upper limit of observed KER's, where the uncertainty associated with the measured values of β includes zero in all but one case. β for channel (II) is also found to be quite close to zero at low values of KER. Both of these plots are representative of the dependence of β on KER at all three wavelengths for the respective channels; within the experimental uncertainty, the $\beta(E)$ plots did not appear to depend on the photon energy. We note that nearly isotropic angular distributions were also seen in the 193 nm photodissociation of nitromethane.[18]

Channel (II) may also involve the secondary dissociation of the most highly internally excited portion of the H_2CO formed. The exothermicity of channel (II) is 34 kcal/mol, so that at 240 nm (119.1 kcal/mol), fully 153 kcal/mol of energy is available for distribution among the products' degrees of freedom. This energy is well above the 80 kcal/mol barrier to dissociation to H_2 + CO. Of course, Figure 6.5 shows that much of the available energy appears as product translation. In addition, the NO bond order changes from 1.5 to 2.5 during elimination, which is likely to result in vibrationally excited NO, further decreasing the energy available for partitioning into H_2CO internal degrees of freedom. Nonetheless, some fraction of H_2CO might be expected to possess sufficient internal energy to undergo secondary dissociation on the nanosecond time scale. A second dissociation pathway for H_2CO, forming H (^2S) + HCO $\tilde{X}$ with very little KER, is expected to dominate the decomposition of H_2CO much above its threshold of 96 kcal/mol.[37] In both cases, however, the kinematics are such that the velocity of the

---

Figure 6.6 Recoil anisotropy factor β, determined from the data as a function of recoil energy E_T, for both channel (I), $CH_2NO_2 \rightarrow CH_2NO + O$, and channel (II), $CH_2NO_2 \rightarrow H_2CO + NO$, at 240 nm.


Channel (I)
CH_2NO_2 --> CH_2NO + O
1.5
0.5
-0.5
$\beta(E_T)$
0
10
20
30
40
$P(E_T)$ [kcal/mol]

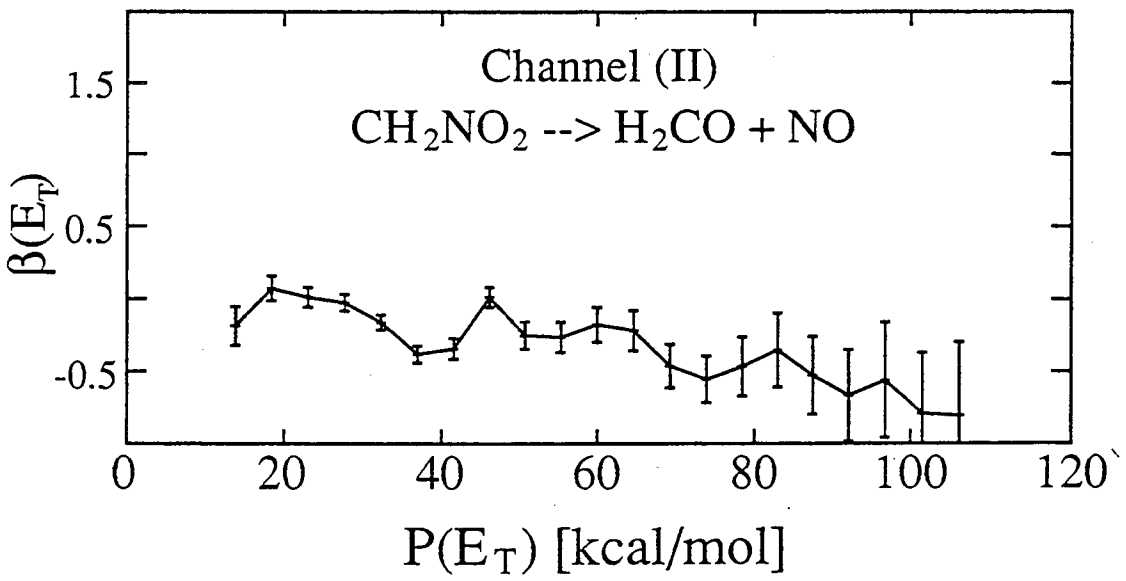

Channel (II)
CH_2NO_2 --> H_2CO + NO
1.5
0.5
-0.5
$\beta(E_T)$
0
20
40
60
80
100
120
$P(E_T)$ [kcal/mol]

Figure 6.6

		Channel (I)	Channel (II)	
Wavelength	Photon Energy	$CH_2NO + O$ <KER>	$H_2CO + NO$ <KER>	Branching Ratio
[nm]	[kcal/mol]	[kcal/mol]	[kcal/mol]	[Channel(I)/Channel(II)]
270	105.9	11.4	59	1.19 (+0.37, -0.21)
255	112.1	11.0	59	1.29 (+0.25, -0.16)
240	119.1	10.8	56	1.76 (+0.27, -0.16)

Table 6.2 This table is a compilation of the observed data for both channels at the three wavelengths (photon energies) used in our experiments. The average kinetic energy release for each channel and the branching ratio measurements [expressed as the quotient channel (I)/channel (II)] are given.

heavy fragment (CO or HCO) will be very close to that of the original H_2CO photofragment, whereas the light fragment will fly out of the beam and miss the detector. The net result, at worst, will be a slight blurring of the translational energy distribution for channel (II). Even in the absence of secondary dissociation, one would not expect to resolve any structure in the translational energy distribution for channel (II) since two molecular fragments are produced. Hence, the effects of secondary dissociation on the overall appearance of the distribution should be very small.

6.3.3 Branching Ratios

Branching ratios are easily determined from the foregoing KER analysis by summing the area under the KER curves to obtain the total population of each channel at each wavelength. As can be seen from Table 6.2, the branching ratio is near unity at the lowest photon energy, but definitely favors channel (I), involving an oxygen atom loss, as

the photon energy is increased. The uncertainties given with these numbers derive chiefly from two sources: statistical error, and uncertainly in $\beta(E_T)$. Statistical error in each channel is well modeled by counting statistics. The second source of uncertainty is due primarily to the absence of knowledge of $\beta(E_T)$ for the small amount of data found at energies where the $P(E_T)$ is very low. By assuming values of $\beta(E_T)$ that represent the possible extremes in these regions, and the limits of uncertainty in the regions where $\beta(E_T)$ is well determined, we estimate the uncertainty in branching ratio from this source.

6.4 Discussion

In the following discussion, we compare our results for the photodissociation of the nitromethyl radical to the known photodissociation behavior of nitromethane. We then identify the initial electronic transition used in the excitation of CH_2NO_2 in the present experiment. Subsequently, we develop the argument that the observed product branching ratios (particularly the absence of C-N bond fission), combined with the translational energy distribution for channel (II), imply that dissociation is not occurring via a statistical process on the ground state CH_2NO_2 surface. Rather, NO elimination most likely results from nuclear dynamics on an excited state surface, aided by the coordinative unsaturation of the carbon atom in the nitromethyl radical. We will also argue that the other observed channel, involving N-O bond fission, must similarly be occurring along an excited pathway, and we identify a plausible mechanism.

6.4.1 Comparison to Nitromethane

The identities of the product channels we observe in the ultraviolet photodissociation of the nitromethyl radical represent a significant departure from what one might expect based on the known UV photofragmentation behavior of the closest closed-shell analogue, nitromethane. Nitromethane exclusively undergoes C-N bond fission following UV absorption,[18,19,20,21,23] whereas the analogous channel [channel (III)]

is absent in CH_2NO_2 photodissociation. Instead, we have identified two major channels: channel (I), N-O bond fission, and channel (II), the elimination of NO. Primary N-O bond fission has not been observed in any collision-free nitromethane photodissociation experiments, although it has been observed in the nitromethane radical cation dissociative charge neutralization experiments of Beijersbergen *et al.*[26] They also reported a minor channel identified as H_2CO + NOH (or HNO). Only in infrared multiphoton dissociation (IRMPD) studies[24,25] has the analogous CH_3O + NO photodissociation product channel been seen for nitromethane.

Some of the differences between CH_2NO_2 and CH_3NO_2 can be understood in terms of the relevant bond dissociation energies. As shown in Table 6.1, the N-O and C-N bond strengths are roughly comparable in CH_2NO_2 ($\approx$ 71 kcal/mol). In contrast, the bond dissociation energies for the N-O and C-N bonds in CH_3NO_2 are 95 and 60.6 kcal/mol, respectively.[34] The slightly stronger C-N bond in CH_2NO_2 presumably arises from the C-N π bonding interaction via the half-filled $2b_1$ molecular orbital. The N-O bond is considerably stronger in CH_3NO_2 than in CH_2NO_2 or other related radicals: e.g., the N-O bond dissociation energies of NO_2 and NO_3 are 72.7[34] and 49.2 kcal/mol,[5] respectively. While the absence of N-O fission in the UV photodissociation of nitromethane is perhaps not so surprising in light of the bond strengths, the preference of N-O over C-N bond fission in CH_2NO_2 photodissociation is an intriguing bond selective effect worthy of further consideration.

The observation of NO elimination in the UV photodissociation of CH_2NO_2 but not CH_3NO_2 is also of interest. As mentioned above, CH_3O + NO products have been seen in the infrared multiphoton dissociation of nitromethane;[25] these are produced in competition with the products of C-N bond fission with a branching ratio of $(CH_3O + NO)/(CH_3 + NO_2) \approx 0.6$. In these studies, the mechanism proposed for CH_3O + NO formation was isomerization to methyl nitrite (CH_3ONO) over a barrier involving a three-center $\overline{C-N-O}$ transition state followed by CH_3O-NO bond fission, all

on the ground state electronic surface of the system. Using a statistical model for the competition between isomerization and C-N bond fission, the isomerization barrier height was estimated to be 55.5 kcal/mol. Saxon and Yoshimine[38] later performed calculations to more fully characterize this nitro-nitrite isomerization transition state and determined that the C-N and C-O bond lengths are 3.4 Å and 3.65 Å respectively. These bond lengths are very large compared with the CH_3NO_2 ground state C-N equilibrium bond distance of 1.489 Å,[39] resulting in a transition state whose energy is calculated to be 56.7 kcal/mol, only 0.4 kcal/mol less than their calculated C-N bond dissociation energy. The very unfavorable geometry of the rearrangement transition state and accompanying high barrier is a consequence of the inability of the fully saturated carbon atom in nitromethane to form a new bond without first breaking an existing bond. In the case of NO elimination from CH_2NO_2, one also expects to pass through some type of $\overline{C-N-O}$ transition state, but the formation of such a structure should be facilitated because the carbon atom in CH_2NO_2 is coordinatively unsaturated. This effect should not be limited to the ground state surface.

The dynamics involved in the NO elimination channel that we observe are also quite different from the above-mentioned IRMPD study of CH_3NO_2. In the IRMPD experiment, the translational energy distribution for NO elimination peaked at zero kinetic energy. This is the characteristic distribution for a statistical bond fission in the absence of a barrier in the reverse direction, and indicates that the CH_3ONO well is sufficiently deep for "re-randomization" of the available energy to occur after crossing the isomerization transition state. Indeed, the heat of formation of methylnitrite is virtually identical to that of nitromethane (only ~2 kcal/mol higher), resulting in methylnitrite being ~40 kcal/mol stable with respect to CH_3O + NO products.[34] The situation in our experiment is quite different; the translational energy distribution for NO elimination peaks at high kinetic energy (around 60 kcal/mol). This implies that, regardless of the details, little or no re-randomization occurs prior to dissociation to H_2CO + NO products.

6.4.2 Nature of the Initially Excited Electronic State

In order to have any detailed understanding of the dissociation mechanisms operative in CH_2NO_2, we need to know which electronic transition is responsible for the absorption band probed in our experiment. CH_2NO_2 has a $^2B_1$ ground state with molecular orbital configuration [...$(8a_1)^2$ $(5b_2)^2$ $(1a_2)^2$ $(2b_1)$]. Figure 6.7 gives a qualitative picture of the $1a_2$ and $2b_1$ molecular orbitals as well as the unoccupied $3b_1$ molecular orbital, obtained by undertaking a minimal basis set [Slater-type orbital (STO-3G)] geometry optimization calculation using the GAUSSIAN 88 package[40] on $CH_2NO_2^-$. A low-lying excited state of CH_2NO_2, the *1* $^2A_2$ state, results from the $2b_1 \leftarrow 1a_2$ transition. This was the excited state observed in the $CH_2NO_2^-$ photoelectron spectrum,[15] formed by the removal of an electron from the $1a_2$ molecular orbital of the anion, and found to lie only 36.7 kcal/mol above the ground state. Figure 6.7 shows that the $2b_1 \leftarrow 1a_2$ transition involves excitation from a non-bonding N-O π orbital to a π orbital which is N-O antibonding and C-N bonding. However, because the *1* $^2A_2$ state lies so far below the photon energies used in our experiment (>105 kcal/mol) it is an unlikely candidate for the initial excited state. The $2b_1 \leftarrow 5b_2$ transition yielding a $^2B_2$ excited state is not optically allowed in C_{2v} symmetry. Two other excited states resulting from $\pi^* \leftarrow \pi$ transitions are more attractive options: the *1* $^2B_1$ state, from the $3b_1 \leftarrow 2b_1$ transition, and the *2* $^2A_2$ state from the $3b_1 \leftarrow 1a_2$ transition. The $3b_1 \leftarrow 2b_1$ transition is analogous to the transitions seen around 230 nm for several alkanenitronate ($R'RC=NO_2^-$) anions in solution.[41] The $3b_1 \leftarrow 1a_2$ transition is more localized on the NO_2 group than the other two transitions, and strongly resembles the $\pi^* \leftarrow \pi$ transition in nitromethane between 190-220 nm. This is the transition accessed in most nitromethane UV photodissociation experiments.[18,19,20,21]

The lowest energy transitions in solvated alkanenitronate anion spectra result in

---

Figure 6.7 Form of the $1a_2$, $2b_1$, and $3b_1$ molecular orbitals determined from *ab initio* calculations performed on $CH_2NO_2^-$.

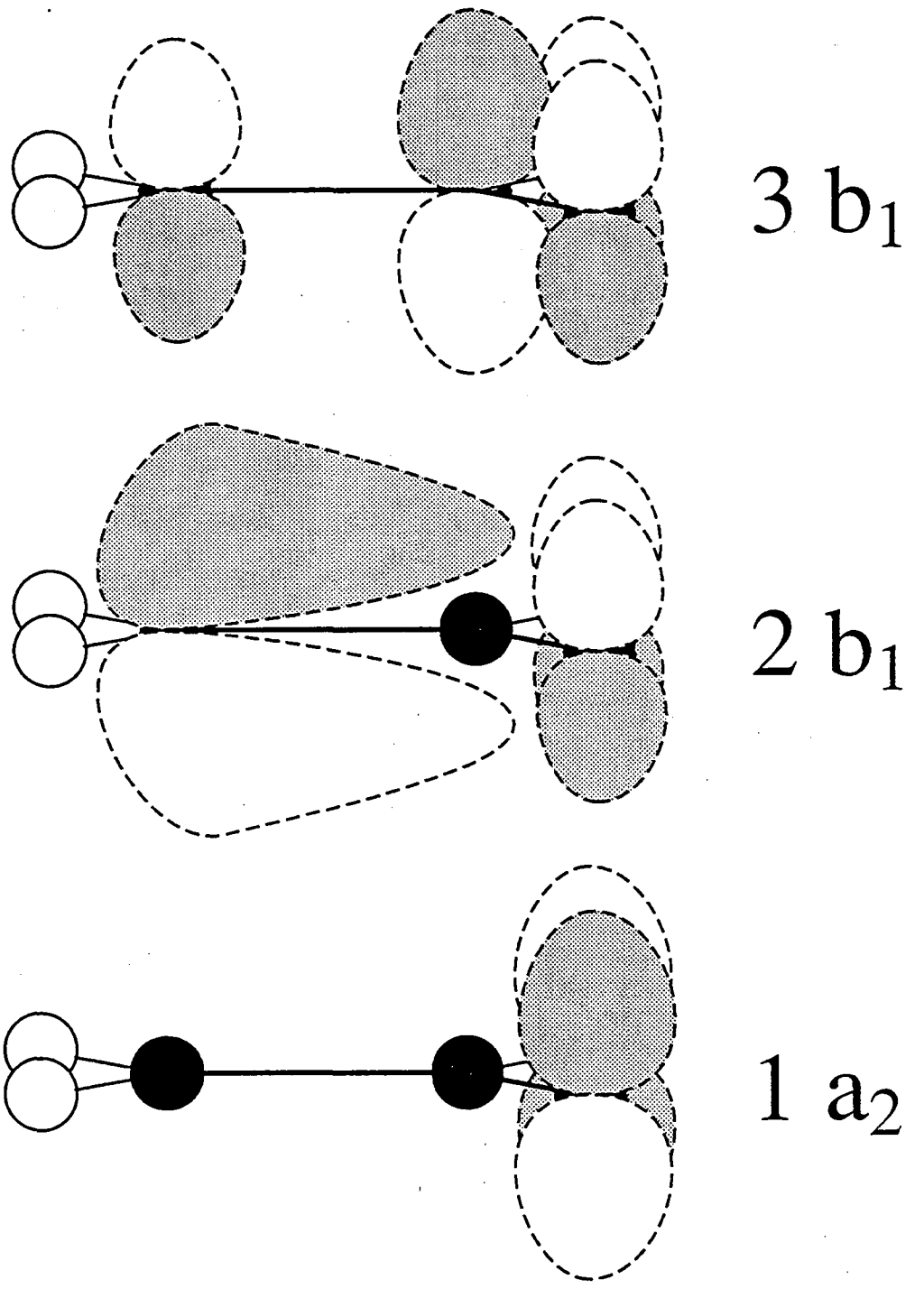


Figure 6.7

bands that peak near 230 nm and extend up to 250-260 nm.[41] This corresponds approximately with our photodissociation cross section measurements, although the band in the radical appears somewhat red-shifted. The C-N bond distance in the anion ground state is considerably shorter than in the neutral (1.347 Å vs. 1.430 Å, according to our previous *ab initio* calculations[15]), and the π^* orbital populated by this transition is C-N antibonding. One would expect the vertical transition energy to be lower in the radical than in the corresponding anion, so the observed red shift appears reasonable. Conversely, one might expect the $3b_1 \leftarrow 1a_2$ transition to occur at a higher energy in CH_2NO_2 than the $\pi^* \leftarrow \pi$ transition in nitromethane because the C-N π^* antibonding interaction in CH_2NO_2 is absent in nitromethane. Any blue shifting of this transition would place it well beyond the range of photodissociation wavelengths used in our experiments. Based on these considerations, we assign the *1* $^2B_1$ state resulting from the $3b_1 \leftarrow 2b_1$ transition as the initially prepared excited electronic state.

Figure 6.8 shows the energetics of the CH_2NO_2 electronic states relative to the dissociation channels (I), (II) and (III). The location of the *1* $^2B_1$ state on this diagram is intended to indicate the lowest energy at which transitions from the ground state to the *1* $^2B_1$ state occur, and is estimated according to the lowest energy at which electronic excitation of CH_2NO_2 leads to dissociation. According to the matrix isolation study of Jacox,[11] the dissociation threshold is known to be between the wavelengths of 300 nm (4.13 eV) and 280 nm (4.42 eV). We therefore place the *1* $^2B_1$ state (somewhat arbitrarily) at 4.25 eV. The energetics of low lying electronic states of channels (I) and (III) are also shown; the excited states of CH_2 and NO_2 are reasonably well-characterized

---

Figure 6.8 Energetics of the nitromethyl radical and selected energetically allowed product channels (not all products shown are observed). States whose energies are known relatively well are represented by solid lines located at their origin, while a dashed line is used to denote the approximate location of the *1* $^2B_1$ state (see text).

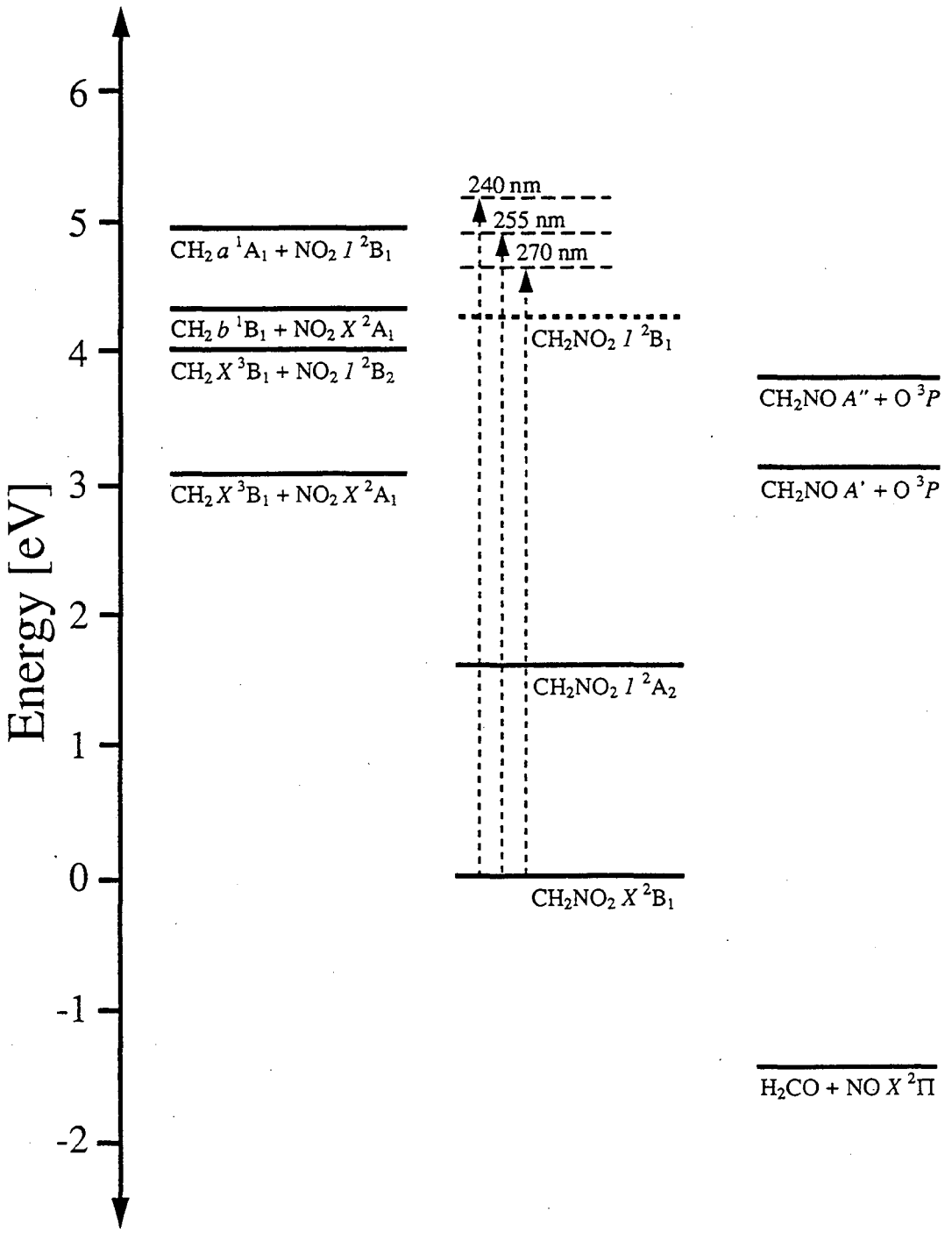

Energy [eV]
6
5
4
3
2
1
0
-1
-2
CH2 a 1A1 + NO2 I 2B1
CH2 b 1B1 + NO2 X 2A1
CH2 X 3B1 + NO2 I 2B2
CH2 X 3B1 + NO2 X 2A1
240 nm
255 nm
270 nm
CH2NO2 I 2B1
CH2NO2 I 2A2
CH2NO2 X 2B1
CH2NO A'' + O 3P
CH2NO A' + O 3P
H2CO + NO X 2Π

Figure 6.8

experimentally, while the locations of the ground and first excited states of CH_2NO are from an MP4-BAC *ab initio* calculation by Melius.[36]

6.4.3 Dissociation Mechanisms

Our assignment of the electronic transition accessed in these experiments offers an immediate if somewhat unspectacular reason for the different product channels seen in the CH_2NO_2 and CH_3NO_2 photodissociation experiments. In nitromethane dissociation, the π^* state accessed near 200 nm, which correlates diabatically to $CH_3 + NO_2$ ($2\ ^2B_2$) products, is believed to be predissociated by a σ^* repulsive state of the same symmetry that correlates to lower energy $CH_3 + NO_2$ ($1\ ^2B_2$) products. Since we believe we are exciting a qualitatively different transition in CH_2NO_2, involving a different excited state surface, this mechanism would not be operative. It remains to be understood how CH_2NO_2 does dissociate, and this will now be considered in detail.

6.4.3.1 Possible Role of Ground State Statistical Dissociation

To explain product branching ratios and translational energy distributions in a given photodissociation experiment, a simple model to consider is rapid internal conversion to the ground state potential energy surface. One should then be able to predict the product branching ratios by a statistical model such as Rice-Ramsperger-Kassel-Marcus (RRKM) theory.[42,43] Such a statistical treatment was successfully used by Wodtke *et al.* to interpret the IRMPD of nitromethane.[25] Figure 6.9 shows a schematic of the potential energy along the reaction coordinates for channels (I)-(III) on the ground state surface of CH_2NO_2. We would expect "loose" transition states for the simple bond fission channels (I) and (III), and a "tight" transition state, presumably involving a three-

---

Figure 6.9 Schematic of the adiabatic minimum energy pathways to all three product channels on the ground state surface of CH_2NO_2.

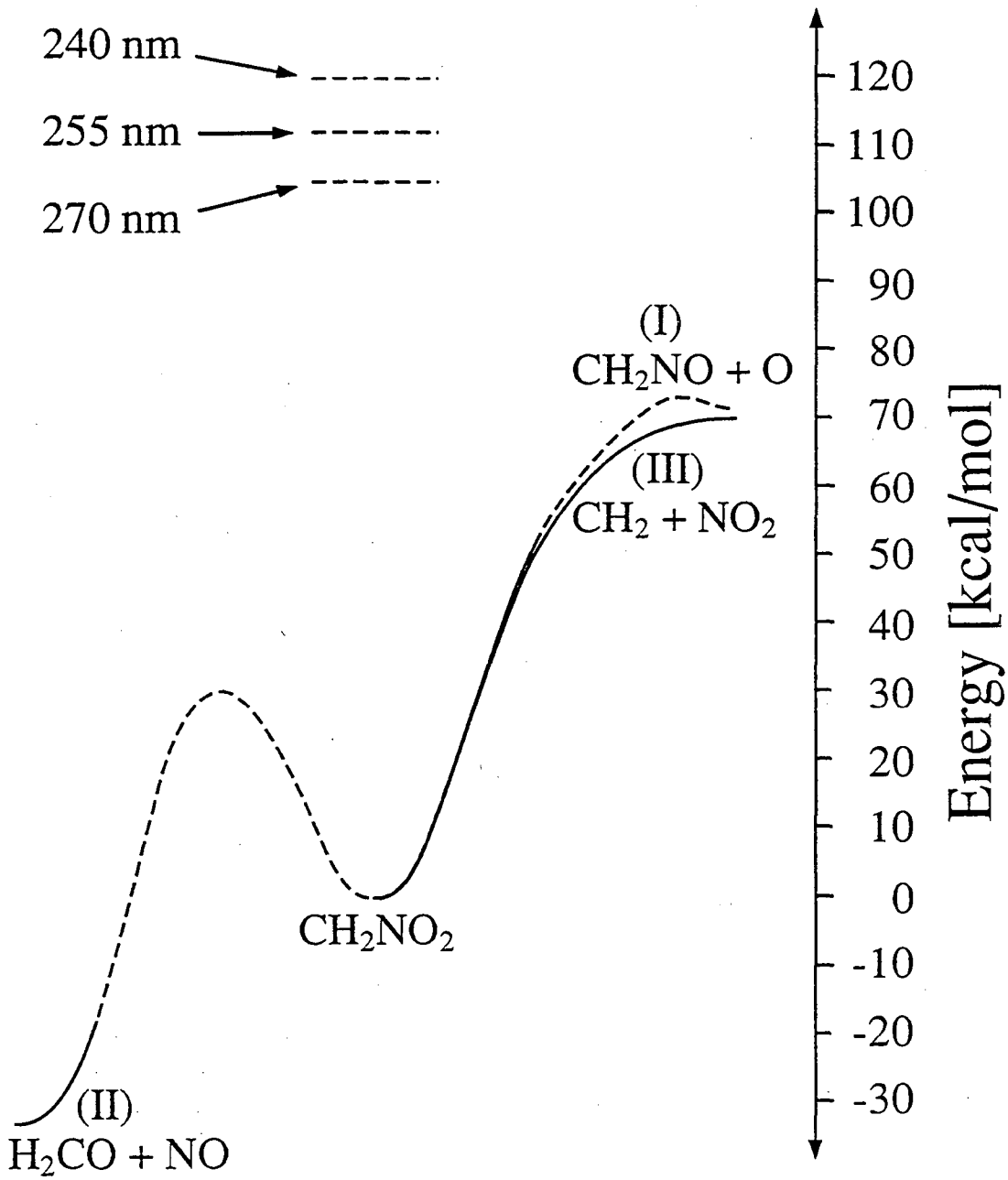

240 nm
255 nm
270 nm
(I)
$CH_2NO + O$
(III)
$CH_2 + NO_2$
CH_2NO_2
(II)
$H_2CO + NO$
120
110
100
90
80
70
60
50
40
30
20
10
0
-10
-20
-30
Energy [kcal/mol]

Figure 6.9

membered $\overline{C-N-O}$ ring, for NO elimination channel (II).[44] The high kinetic energy release associated with channel (II) is consistent with passage over a barrier associated with the tight transition state, while the translational energy distribution for channel (I) peaks near zero, the expected "statistical distribution" for simple bond fission (perhaps over a small barrier in the reverse direction, consistent with the $P(E_T)$ distribution maximum at several kilocalories per mole.) However, ground state CH_2 + NO_2 correlate to the ground state of CH_2NO_2 both diabatically and adiabatically. This means there should be no barrier to the CH_2 + NO_2 recombination reaction, so the loose transition state associated with C-N bond fission will lie about 71 kcal/mol (i.e. the bond dissociation energy) above the CH_2NO_2 well. Since the N-O and C-N bond dissociation energies are very close, the transition state for channel (I) must also lie at least as high; it will be higher if there is a barrier along the reaction coordinate for channel (I). In addition, the preexponential A factor in the Arrhenius rate equation for the C-N bond fission channel will be larger since channel (III) yields two polyatomics, as opposed to a polyatomic plus an atom, and will thus have a greater gain in entropy on going from the reactant to the activated complex.[43] Thus, even with no barrier to N-O bond fission above the bond dissociation energy, we expect a statistical model to yield a greater rate of C-N bond fission than N-O bond fission. This is incompatible with the absence of C-N bond fission products observed in the present experiment and immediately indicates that the N-O bond fission channel cannot be explained by statistical dissociation on the ground state surface. We shall return to this channel subsequently.

While it appears certain that a statistical dissociation on the ground state surface cannot explain *all* of the observed dissociation dynamics, it may be possible that N-O bond fission [channel (I)] occurs on an excited state surface, while rearrangement and elimination of NO [channel (II)] is statistically favored on the ground state surface and dominates C-N bond fission [channel (III)] once on that surface. This requires that the RRKM rate constant for NO elimination at a reasonable barrier height be substantially

larger than that for C-N bond fission. We have performed RRKM calculations[45] for the two channels to test if this is plausible.

RRKM theory is the most widely accepted and generally employed method of describing and modeling unimolecular decomposition processes. It is based on two assumptions: that vibrational energy is randomized rapidly on the time scale of a reaction throughout all vibrational degrees of freedom; and that all trajectories of a system which cross a judiciously chosen surface separating reactants from products are considered to have irreversibly reacted. RRKM theory therefore allows the rate of reaction at a given energy to be expressed as:

$$k(E) = \frac{W^{\dagger}(E)}{h\rho(E)} \qquad (6.10)$$

Here, $W^{\dagger}(E)$ is the sum of internal states of the dissociating complex at the transition state, and $\rho(E)$ is the density of states of the reactant. Because any recrossing of the dividing surface is neglected by the second assumption given above, a slight overestimation of reaction rate can result. Therefore, this expression is strictly speaking an upper bound.

The parameters required to undertake the sum- and density-of-states calculations which are necessary to evaluate Eqn (6.10) are the vibrational frequencies and parameters that describe any internal rotational degrees of freedom present in either the reactant or the transition state. While these quantities are usually well known for the reactant, transition states are often not as well characterized. Fortunately, it is generally acknowledged that the calculated RRKM unimolecular decomposition rate is insensitive to the exact values given the vibrational frequencies in the critical configuration, so long as they are consistent with the correct high-pressure preexponential A-factor.

The high-pressure preexponential A-factor, denoted A_{∞} in Eqn (6.11),[46]

$$k_{uni}^{\infty} = A_{\infty} \cdot \exp\left\{-\frac{E_{\infty}}{k_B T}\right\} \qquad (6.11)$$

can be recognized as the preexponential term in the familiar Arrhenius rate equation. It is at high pressures that collisionally activated unimolecular decomposition truly follows first-order kinetics, since the rate limiting step will not be the second order (bimolecular) activation process. Values of A_∞ and E_∞ are determined experimentally by examining the temperature dependence of reaction rates under high pressure conditions.

The A_∞ factor is related to the change in entropy on going from the reactant to the activated complex in the following way:

$$A_\infty = \frac{ek_BT}{h} \cdot \exp\left\{\frac{\Delta S}{k_BT}\right\} \qquad (6.12)$$

For this reason 'loose' unimolecular decomposition transition states involving a long separation of incipient fragments, conversion of vibrational modes to hindered rotors and/or a net lowering of remaining vibrational mode frequencies relative to the reactant molecule have large A_∞ factors.

Since we know all the reactant vibrational frequencies from the works of Jacox and Metz *et al.*, we then calculated vibrational frequencies for the critical configurations that are consistent with the expected A-factors for the individual processes under consideration. Vibrational frequencies for the tight three-center transition state in NO elimination were chosen to yield a preexponential $\log_{10}$ A-factor in the Arrhenius rate expression of 13, based on comparisons to previously determined values for comparable types of unimolecular decompositions.[43,47] The vibrational frequencies in the transition state for C-N bond fission were chosen to be consistent with a $\log_{10}$ A-factor of 15.6, determined by Benson and O'Neal from the analysis of kinetics data.[48] The program AFAC[49] is used, which calculates the A-factor associated with changing from the known reactant frequencies to a proposed set of frequencies for the critical configuration. By iteratively varying the proposed critical configuration vibrational frequencies, one can find

a set of frequencies that is realistic and which results in the desired A-factor. These vibrational frequencies were then used in RRKM rate calculations.

In our RRKM rate calculations, all vibrations were assumed to be separable harmonic oscillators. The rotational temperature was estimated to be 50 K, which is consistent with previous experiments using this apparatus.[6,7,8,9] The overall rotations were treated adiabatically, so that the change in moments of inertial from reactant to critical configuration served only to modify slightly the energy available above the barrier to a particular decomposition pathway.

The moments of inertia for the reactant, CH_2NO_2, were obtained from the geometry calculated by Metz *et al.*[15] For the C-N bond fission critical configuration, moments of inertia were calculated assuming a C-N bond distance of 2.3 Å,[50] with both of the nascent CH_2 and NO_2 fragments having the same bond lengths and angles as the respective free molecules. The critical configuration for the isomerization-elimination channel was taken to be an equilateral $\overline{C\text{-}N\text{-}O}$ ring, with the second oxygen extending out from the nitrogen, bent out of the plane of the ring by 60°. The change in available energy due to change in moments of inertia, and thus changes in rotational energy, on transforming from reactant to transition state geometries was very small compared to the overall excess energy in all cases.

The most informative results from our RRKM calculations are shown in Table 6.3. First, the statistical rates of C-N bond fission at the three different wavelengths used in our study are reported. Then, the rate of the rearrangement-elimination channel is examined by assuming several values for the height of the barrier to a possible ground state rearrangement. We find that even for a barrier as low as 10 kcal/mol with respect to ground state CH_2NO_2, NO elimination is only four times as fast as C-N bond fission at λ_{diss} = 240 nm. This is the dissociation wavelength where our signal-to-noise ratio is best, and we would definitely observe C-N fission were it occurring to this extent. Moreover, a 10 kcal/mol barrier is not consistent with the translational energy distribution for the NO

Channel	$\log_{10}$ A-factor	E^* [kcal/mol]	E_0 [kcal/mol]	$E^\dagger$ [kcal/mol]	Harmonic k(E) [10^{12} s^{-1}]
$CH_2 + NO_2$	15.6	105.9	71.1	34.8	0.223
	15.6	112.5	71.1	41.4	0.487
	15.6	119.1	71.1	48.0	0.931
$H_2CO + NO$	13	105.9	10	95.9	3.62
	13	112.5	10	102.5	3.81
	13	119.1	10	109.1	3.99
$H_2CO + NO$	13	105.9	30	75.9	0.444
	13	112.5	30	82.5	0.531
	13	119.1	30	89.1	0.621

Table 6.3 The results of several RRKM calculations which were performed to investigate the possibility that the experimentally observed products could be a result of statistical unimolecular decomposition on the ground electronic state of CH_2NO_2. Specifically, the statistical rate for channel (III), C-N bond fission, is calculated and compared to that of channel (II), the rearrangement-elimination channel, at two different values of the main unknown quantity, the barrier to channel (II). The labels E^*, E_0, and $E^\dagger$ refer to the photon energy, the energy below which classical reaction cannot occur, and the difference between the two, respectively. See text.

elimination channel. This distribution peaks around 60 kcal/mol (see Figure 6.4), indicating that the barrier with respect to the products is at least this high. Since NO elimination from CH_2NO_2 is exothermic by 33 kcal/mol, the barrier height with respect to CH_2NO_2 must be at least 30 kcal/mol. For this barrier height, statistical C-N bond fission on the ground state surface is faster than NO elimination at 240 nm. Judging by the

observed $P(E_T)$ distribution for this channel, the true barrier height is probably larger than this. A barrier of 30 kcal/mol with respect to CH_2NO_2 implies that the potential energy at the barrier must be very efficiently channeled into product kinetic energy, and this is unlikely considering how strained the transition state must be with respect to the products. Our calculations therefore imply that in the statistical limit, elimination of NO cannot be the dominant dissociation channel on the ground state. While there is considerable leeway in some of the assumptions made in our analysis, this conclusion seems quite secure.

One final scenario that deserves attention is the possibility that statistical C-N bond fission does occur from the ground state but that the statistical kinetic energy release distribution peaks so sharply near zero that detection of the products is overwhelmingly diminished by the blocking strip at the center of the detector.

For molecules the size of CH_2NO_2, a 'prior distribution' model[51] can be used to give a reasonable approximation of the statistical product translational energy distribution. By comparing this distribution to our known low kinetic energy cut-off, we can determine if the products from statistical C-N fission will be undetected.

As given by Levine,[51] the prior distribution has the form:

$$P(E|f_T) \propto f_T^{1/2} \cdot (1-f_T)^{\alpha}, \qquad (6.13)$$

where for a given total energy E, f_T is the fraction of energy found in translation (i.e., $f_T = E_T/E$). The exponent α is found using the formula $\alpha = s - 1 + r/2$, s being the number of vibrational modes and r the number of rotational degrees of freedom in the products. For channel (III), $CH_2NO_2 \rightarrow CH_2 + NO_2$, $s = 6$ and $r = 6$ so α has the value of 8. At a dissociation wavelength of 240 nm this results in the predicted translational energy distribution shown in Figure 6.10. Since the mass ratio is very similar between channel (I)

Figure 6.10 The translational energy distribution predicted by a prior distribution calculation for the channel $CH_2NO_2 \rightarrow CH_2 + NO_2$ at 240 nm.

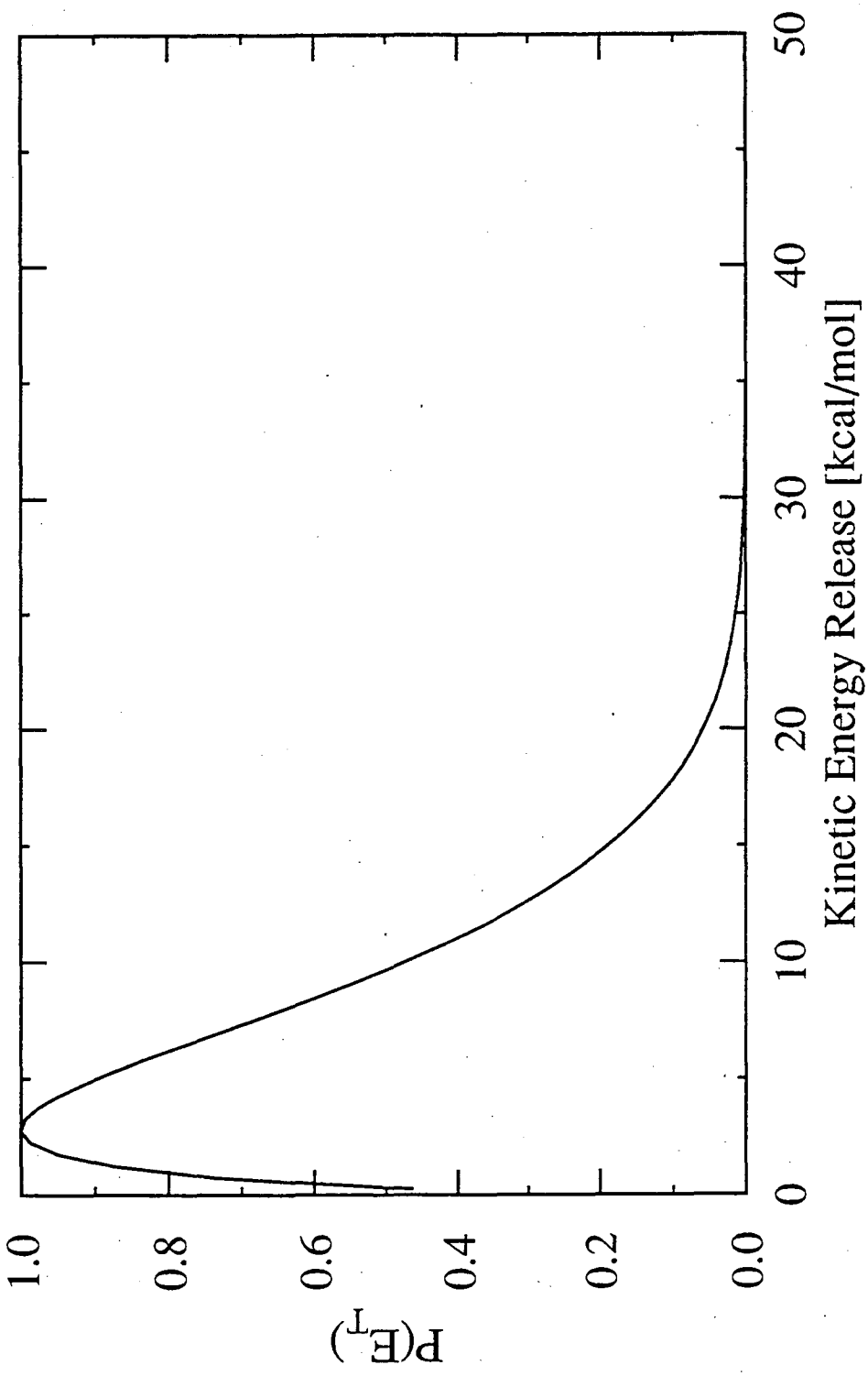

P(E_T)
1.0
0.8
0.6
0.4
0.2
0.0
0
10
20
30
40
50
Kinetic Energy Release [kcal/mol]

Figure 6.10

and channel (III) products, the blocking-strip-induced low kinetic energy release cut-off for the $CH_2 + NO_2$ products will be virtually the same as in channel (I), shown to be ~2 kcal/mol in Figure 6.4. Figure 6.10 clearly shows that a large fraction of the translation energy distribution extends out well past 2 kcal/mol. We are thus very confident that C-N bond fission is not occurring to any significant extent.

6.4.3.2 $CH_2NO_2 \rightarrow H_2CO + NO$ Dissociation Mechanism

We are left with two options regarding channel (II). Either it proceeds on ground state surface in a highly non-statistical (i.e., mode-specific) fashion, or it results from the nuclear dynamics on one or more of the excited state surfaces. Given the considerable molecular rearrangement necessary to produce the observed channel (II) products, in contrast to the very direct C-N simple bond fission channel, the first option seems quite unlikely. This suggests the existence of a mechanism on excited state surfaces that would facilitate dissociation to channel (II) products. After presenting and discussing such a mechanism, we then consider the mechanism for dissociation channel (I), which, based on the above discussion, also most likely occurs on an excited state surface.

Figure 6.11 depicts the likely nuclear dynamics for the rearrangement-elimination process of channel (II), irrespective of the details of the potential energy surfaces involved. This channel presumably involves (a) rotation of the CH_2 plane perpendicular to the NO_2 plane, (b) oxygen atom transfer though a $\overline{C-N-O}$ three-membered-ring, and (c) elimination of H_2CO from NO. We note that steps (a) and (b) are the first steps in the mechanism proposed by Yamada and co-workers[52,53] for the photochemical rearrangement of $R'RC{=}NO_2^-$ anions in solution as a synthetic route to the corresponding hydroxamic acids [i.e. RC(O)-N(OH)R]. In their mechanism, as applied to $CH_2NO_2^-$, the anion first

---

Figure 6.11 Nuclear dynamics presumably involved in channel (II) rearrangement-elimination mechanism resulting in the reaction $CH_2NO_2 \rightarrow H_2CO + NO$.

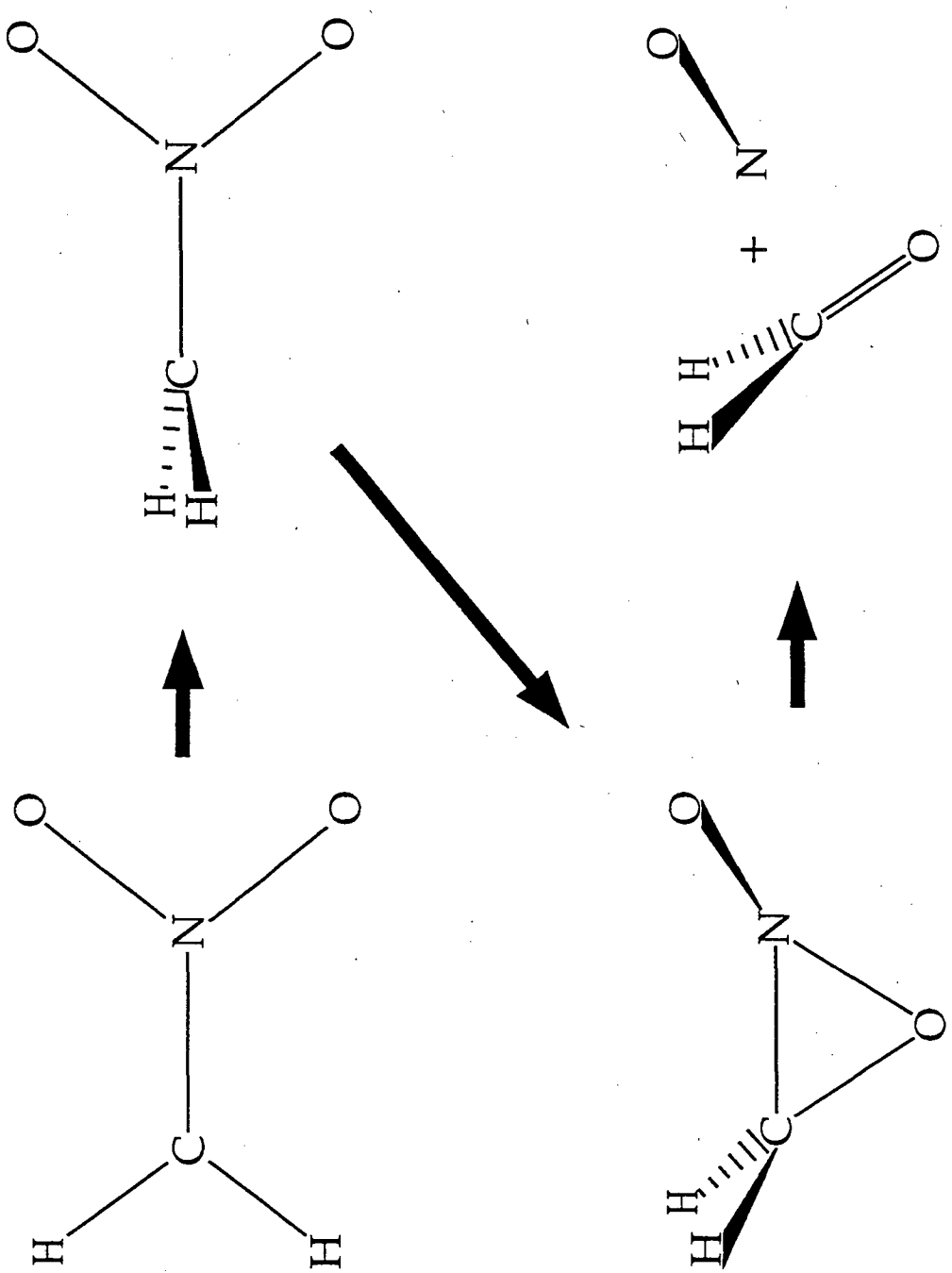


Figure 6.11

undergoes $\pi^* \leftarrow \pi$ excitation corresponding to one electron being transferred from the $2b_1$ HOMO to the $3b_1$ molecular orbital (see Figure 6.7). The subsequent dynamics involve CH_2 moiety rotation, which minimizes the C-N antibonding interaction, formation of a three-membered oxaziridine ring as an oxygen atom is transferred from nitrogen to carbon atoms, and finally hydrogen atom migration resulting in the observed net isomerization from $CH_2NO_2^-$ to $HC(O)N(O)H^-$. The similarities between the observed rearrangement-elimination products in the neutral CH_2NO_2 radical and the postulated $CH_2NO_2^-$ anion rearrangement dynamics are compelling. Not only are the nuclear motions in the initial two stages of the anion rearrangement mechanism very similar to what must occur in the rearrangement-elimination channel observed with nitromethyl radical, but the initial electronic transition is proposed to be the same in each case (as discussed in § 6.4.2).

Although the anion has an additional $2b_1$ electron in both the ground and excited states, one expects the $1\,^2B_1$ state in the radical to be significantly stabilized by a 90° internal rotation about the C-N bond for the same reason as in the anion: reduction of the C-N antibonding interaction as a result of the half-filled $3b_1$ orbital. The resulting stabilization may be greater in the radical because there is no half-filled C-N π-bonding orbital as there is in the anion excited state, or it might be smaller due to the larger C-N bond distance in the radical. In any case, let us consider what effect the 90° internal rotation might have on the relevant molecular orbitals in the excited state. Figure 6.12 (a) shows the $3b_1$ molecular orbital after the internal rotation has occurred. While this orbital has some amplitude on the carbon atom in the planar geometry, it becomes an orbital localized on the NO_2 group in the 90° geometry, leaving the p-orbital on the carbon atom perpendicular to the CH_2 plane empty. This geometry places an excess positive charge on

---

Figure 6.12 (a) The form of the $3b_1$ molecular orbital following a 90° internal rotation.
(b) Top view of CH_2NO_2 subsequent to internal rotation, showing how a C-O bond may form from the interaction of the $5b_2$ oxygen atom lone pair molecular orbital with the empty p orbital on the carbon atom.

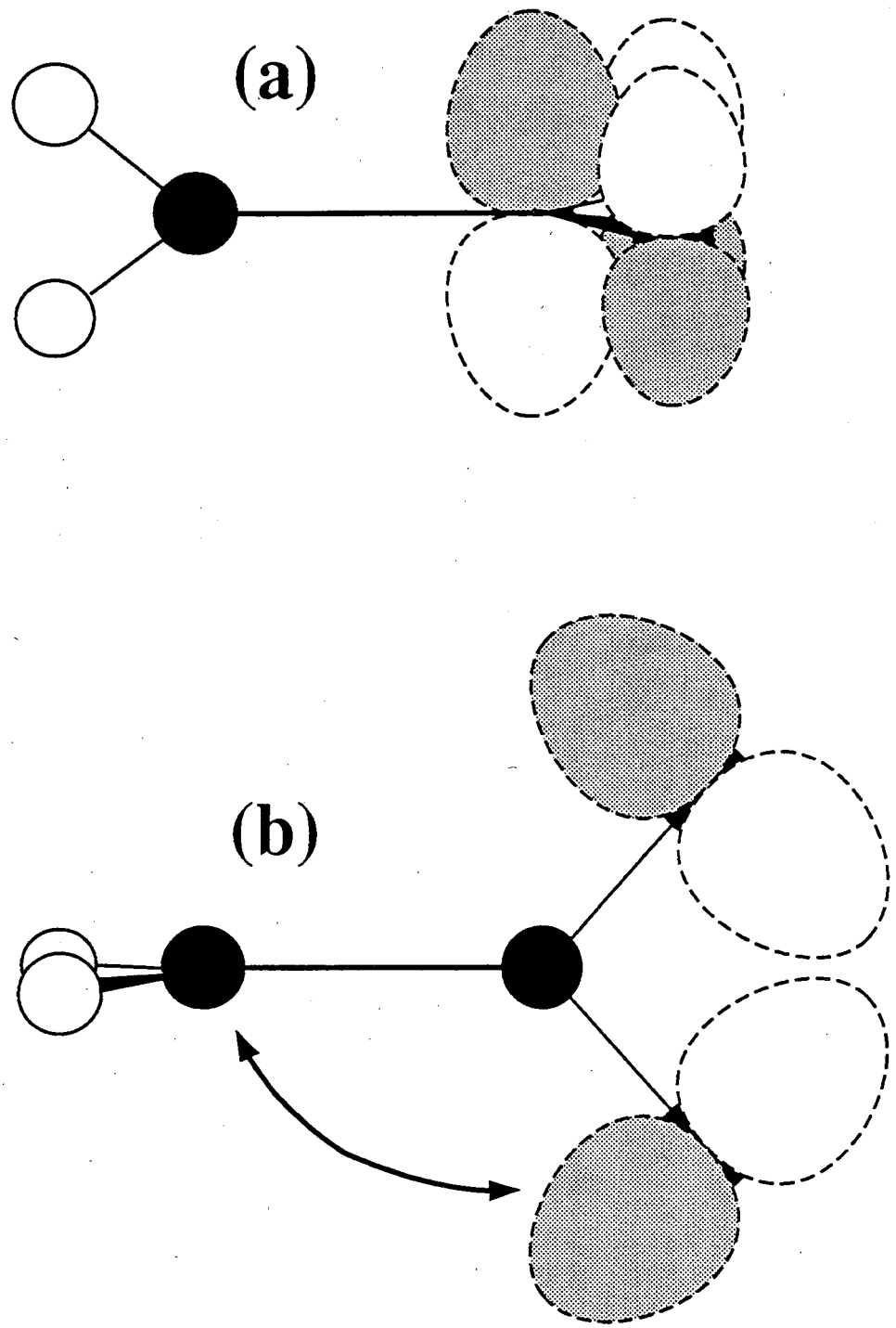


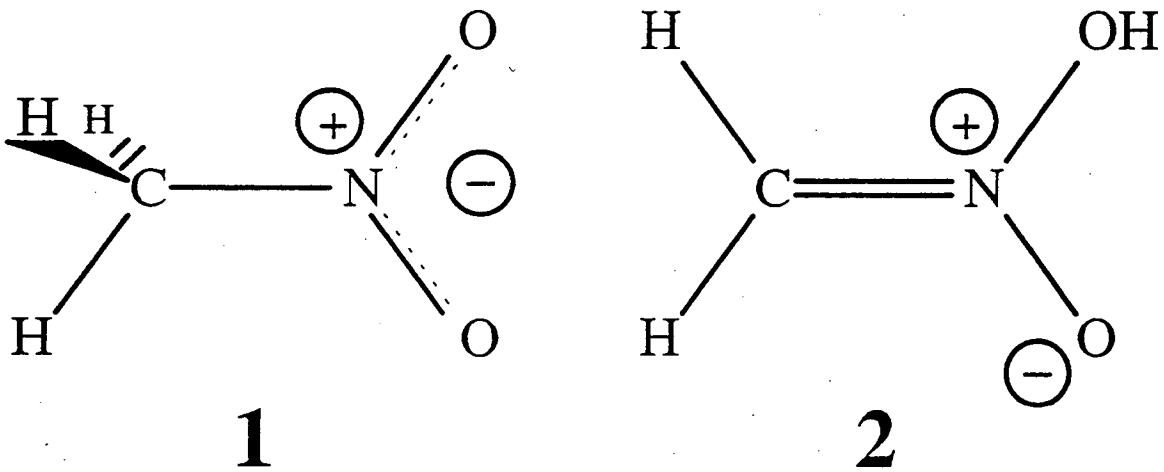
Figure 6.12

the carbon atom and corresponding excess negative charge on the NO_2 moiety. This should facilitate formation of the three-member $\overline{C-N-O}$ ring intermediate, since the lone pair on either of the oxygen atoms in the NO_2 plane [see Figure 6.12 (b)] will be attracted to the empty orbital on the electron-poor carbon atom, forming an O-C bond. Figure 6.12 may be somewhat of a simplification in that rehybridization of the nitrogen atom may occur subsequent to internal rotation, resulting in an sp^3 hybridization with a lone pair of electrons localized on the nitrogen, but this does not affect the overall argument.

As far as the energetics along the rather complex reaction coordinate associated with the nuclear dynamics depicted in Figure 6.11 are concerned, we expect the highest energy geometry to occur either en route to formation of the three-member ring, or at the three-member ring itself. While one might wonder if the three-member oxaziridine ring represents a local minimum, prior theoretical studies[53] suggest that the structurally identical intermediate postulated for $CH_2NO_2^-$ rearrangement is unstable. In addition, only oxaziridines having aryl-, alkyl-, H-, or acyl-substituted nitrogen atoms have been isolated; no stable oxaziridines with O-substituted nitrogen atoms (i.e. the current case) are known.[54] In any case, regardless of the exact location of the barrier along the reaction coordinate, we expect it to lie well above ground state H_2CO + NO products, consistent with the high translational energy seen for this channel. On the other hand, the geometry found at the barrier is only beginning to resemble the products, and as a result there should be extensive internal excitation found in the products. This implies that the actual barrier height with respect to the products is significantly higher than the minimum value of ~60 kcal/mol implied by the translational energy distribution.

The central element in the above mechanism is that the coordinatively unsaturated carbon atom provides a relatively facile pathway for NO elimination. Such a pathway does not exist in CH_3NO_2, consistent with the absence of NO production as a result of electronic excitation. Moreover, other examples of dissociation channels analogous to

channel (II) in our study have been observed in which the carbon atom in the parent molecule is unsaturated. For example, in the experiments of Beijersbergen *et al.*,[26] three structural isomers of nitromethane radical cation and their deuterated analogs were neutralized by resonant charge exchange. This process forms the neutral with a significant amount of excitation (6.9 eV for Na and 7.2 eV for Cs charge exchange). Below are the structures of two of the isomers studied, nitromethane **1** and its tautomer, *aci*-nitromethane **2**:



Rearrangement followed by elimination of NOH (or HNO) was found to be the major pathway only upon charge exchange neutralization yielding the *aci* form of nitromethane, which contains a CH_2 group rather than a CH_3 group bonded to the nitrogen atom. Also, in the recent study by Galloway *et al.*[55] of the UV photodissociation of nitrobenzene, NO elimination was observed, along with both C-N and N-O bond fission. While nitrobenzene is a closed shell molecule, the carbon atom bound to the NO_2 group is coordinatively unsaturated, and this may enhance the NO elimination channel by the mechanism discussed above.

6.4.3.3 $CH_2NO_2 \rightarrow CH_2NO + O$ Dissociation Mechanism

Having proposed a possible mechanism by which channel (II) proceeds to products, the next step is to determine whether the absence of C-N bond fission and the presence of N-O fission can also be explained in terms excited state dynamics. We first consider why C-N fission might not readily occur from the initially accessed *1* $^2B_1$ state. As shown in Figure 6.8, the three lowest lying C-N bond fission channels which could correlate to this state in C_{2v} symmetry are (IIIa) CH_2 ($\tilde{X}$ $^3B_1$) + NO_2 ($\tilde{X}$ $^2A_1$), (IIIb) CH_2($\tilde{b}$ $^1B_1$) + NO_2 ($\tilde{X}$ $^2A_1$), and (IIIc) CH_2($\tilde{a}$ $^1A_1$) + NO_2 (*1* $^2B_1$). The observed absence of C-N bond fission in our results requires that there is no low energy pathway for dissociation of this state to the ground state fragments [channel (IIIa)]. If we examine how the molecular orbitals of the parent molecule correlate with those of the fragments, we indeed find that the ground state fragments correlate diabatically (and adiabatically) with the $\tilde{X}$ $^2B_1$ state of CH_2NO_2 and not with the *1* $^2B_1$ state. Hence, channel (IIIc) is the lowest C-N bond fission channel to which the *1* $^2B_1$ state can correlate diabatically. This channel lies 45 kcal/mol above the lowest CH_2NO + O channel. C-N bond fission is therefore less favorable than might be expected at first glance, and any additional barrier along the reaction coordinate further reduces the expected contribution from this channel.

To complete the overall picture, we need to identify a facile dissociation pathway to N-O bond fission consistent with the observed translational energy distribution. This distribution, which at 240 nm peaks at less than 15% of the available kinetic energy, indicates that the surface on which dissociation occurs is not strongly repulsive along the N-O coordinate. We first consider which electronic states of CH_2NO_2 might lead to CH_2NO + O production. The geometry of the ground state of CH_2NO calculated by Balakina *et al.*[56] is shown in Figure 6.13. This radical has C=N double bond character,

---

Figure 6.13 Geometry of ground state CH_2NO, as determined in Ref. 56. Bond lengths are in Ångstroms, bond angles are in degrees.

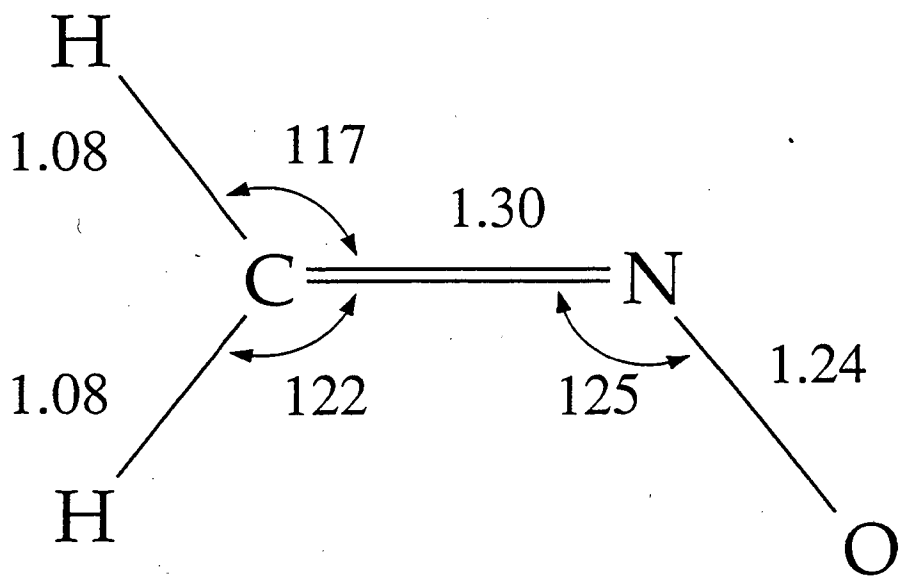


Figure 6.13

while the HOMO of the *1* $^2B_1$ state of CH_2NO_2 (the $3b_1$ molecular orbital) is C-N π antibonding. Hence, the *1* $^2B_1$ state should not diabatically correlate to the ground state of CH_2NO. On the other hand, the *1* $^2A_2$ state has a C-N double bond and lies well below the CH_2NO + O asymptote. Moreover, the C-N double bond makes dissociation to CH_2 + NO_2 unlikely. Dissociation to CH_2NO + O from the *1* $^2A_2$ state therefore presents an attractive possibility.

The next issue to address is the mechanism which allows the radical to transfer onto the *1* $^2A_2$ surface after initial excitation to the *1* $^2B_1$ state. The molecular orbital occupancies for these states in the planar geometry, $...(1a_2)(2b_1)^2$ for the former and $...(1a_2)^2(3b_1)$ for the latter, differ by two electrons, and are therefore coupled only by the configuration interaction term in the electronic Hamiltonian. This coupling should be strong only in the vicinity of an intersection between the two surfaces; such an intersection can be brought about by internal rotation about the C-N bond. As discussed above, this torsional motion is expected to stabilize the initially excited *1* $^2B_1$ state. In contrast, it should strongly destabilize the *1* $^2A_2$ state as it disrupts the C-N double bond in the planar geometry. One can therefore imagine a crossing between the two surfaces at an internal rotation angle between 0° and 90°, as shown in Figure 6.14. The radical has no symmetry at the intermediate angles, so these would actually be avoided crossings, with the adiabatic surfaces repelling each other near where the intersections would occur in the absence of configuration interaction.

However, it is perhaps easier to think about the overall dynamics in terms of diabatic surfaces. If, at the crossing points, the radical remains on the initial diabatic surface, it will continue to undergo internal rotation until it reaches the 90° geometry, at which point NO elimination is most favorable. On the other hand, it can undergo a

---

Figure 6.14 Schematic of the potentials for the *1* $^2B_1$ state and the *1* $^2A_2$ state along the torsion coordinate.

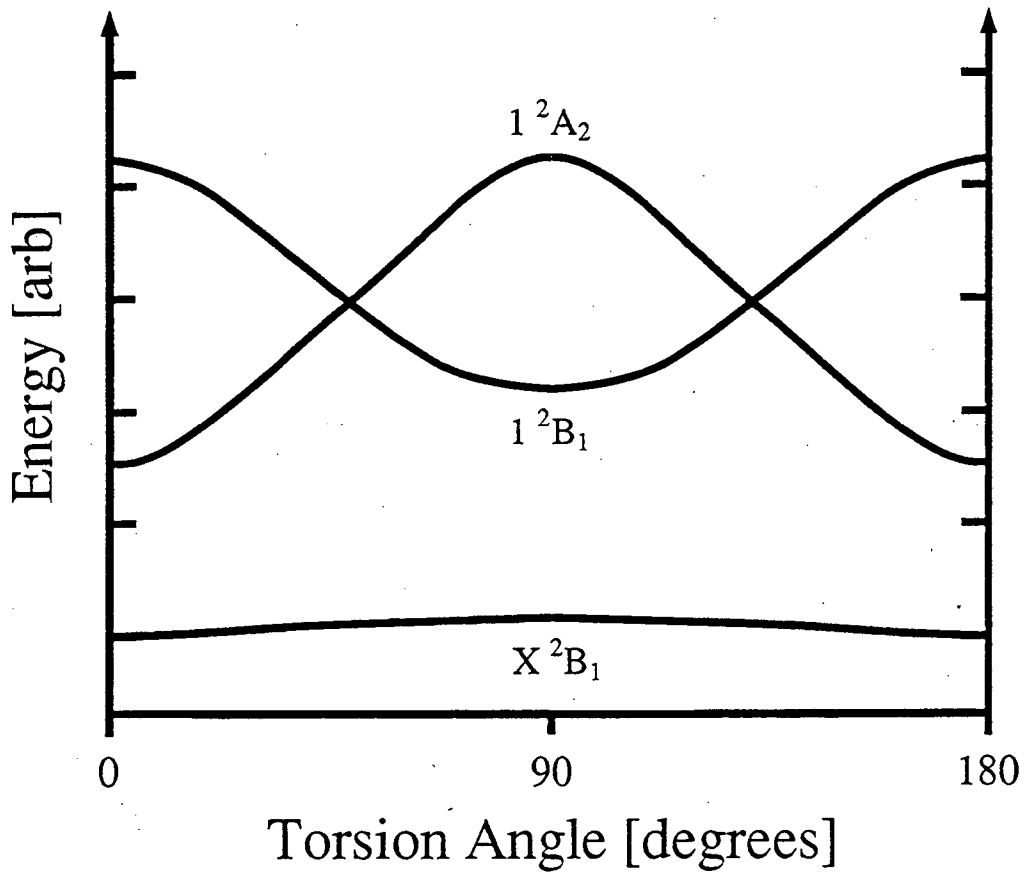


Figure 6.14

transition to the diabatic surface which correlates to the $1\ ^2A_2$ state in the planar geometry. Such a diabatic transition would most likely result in a highly vibrationally excited molecule. If we apply the concepts developed for statistical dissociation on ground state surfaces to this situation, we would expect that dissociation to fragments with high kinetic energy is unlikely because the $1\ ^2A_2$ state is strongly bound with respect to $CH_2NO + O$. This is consistent with our observed distribution. However, a translational energy distribution described by phase space theory would peak very close to zero, while ours actually peaks at 5-8 kcal/mole. This may indicate a small barrier to N-O bond fission. Alternatively, peaking away from zero may indicate that the dissociation rate is sufficiently rapid as to render the statistical perspective somewhat inappropriate.

Overall, the picture we are proposing to explain the photodissociation of CH_2NO_2 is that the important nuclear dynamics are occurring on excited state potential energy surfaces. In particular, torsional motion on the initially excited surface plays a major role in the two observed dissociation channels, and the outcome depends on whether a diabatic transition occurs while the radical is undergoing internal rotation. We acknowledge that aspects of these mechanisms are somewhat speculative. Nonetheless, as the overall scheme is consistent with the experimental results, it should at least provide a starting point for more detailed experimental and theoretical studies of this species. We believe that *ab initio* investigations of the excited states of CH_2NO_2 would be particularly helpful in assessing some of the ideas put forward here.

6.5 Conclusions

We have studied the photodissociation dynamics of the CH_2NO_2 radical at three wavelengths: 240, 255, and 270 nm. We have measured branching ratios for the various dissociation channels, as well as translational energy and angular distributions for each major channel. We observe only two channels and find these have comparable yields: (I) $CH_2NO_2 \rightarrow CH_2NO_2 + O$ and (II) $CH_2NO_2 \rightarrow H_2CO + NO$. No C-N bond fission is

observed. These results are in sharp contrast to the ultraviolet photodissociation of nitromethane, which undergoes exclusive C-N bond fission near 200 nm. We have assigned the electronic transition in these experiments to the $1\ ^2B_1 \leftarrow \tilde{X}\ ^2B_1$ transition, in which the $2b_1$ C-N π-bonding electron is promoted to the $3b_1$ π^* molecular orbital. An analysis of the observed branching ratios and translational energy distributions indicates that statistical dissociation on the ground state surface is most likely not occurring, and that both of the observed channels are occurring on excited state surfaces. Mechanisms for both channels are proposed in which the initial nuclear dynamics involves internal rotation about the C-N bond in the initially excited state. A 90° internal rotation facilitates formation of the three-center $\overline{C-N-O}$ transition state needed for NO elimination, while internal rotation also provides a reasonable coupling mechanism between the $1\ ^2B_1$ and lower-lying $1\ ^2A_2$ surface, with N-O bond fission occurring on the latter surface.

6.6 References

[1] *Molecular Photodissociation Dynamics*, edited by M. N. R. Ashfold and J. E. Baggott (Royal Society of Chemistry, London, 1987).

[2] G. E. Busch, J. F. Cornelius, R. T. Mahoney, R. I. Morse, D. W. Schlosser, and K. R. Wilson, Rev. Sci. Instrum. **41**, 1066 (1970); G. E. Busch and K. R. Wilson, J. Chem Phys. **56**, 3626 (1972).

[3] For a recent review, see M. N. R. Ashfold, I. R. Lambert, D. H. Mordaunt, G. P. Morley and C. M. Western, J. Phys. Chem. **96**, 2938 (1992).

[4] E. J. Hintsa, X. Zhao, W. M. Jackson, W. B. Miller, A. M. Wodtke, and Y. T. Lee, J. Phys. Chem. **95**, 2799 (1991).

[5] H. F. Davis, B. Kim, H. S. Johnston, and Y. T. Lee, J. Phys. Chem. **97**, 2172 (1993).

[6] R. E. Continetti, D. R. Cyr, R. B. Metz, and D. M. Neumark, Chem. Phys. Lett. **182**, 406 (1991).

[7] D. R. Cyr, R. E. Continetti, R. B. Metz, D. L. Osborn, and D. M. Neumark, J. Chem. Phys. **97**, 4937 (1992).

[8] R. E. Continetti, D. R. Cyr, D. L. Osborn, D. J. Leahy and D. M. Neumark, J. Chem. Phys. **99**, 2616 (1993).

[9] D. J. Leahy, D. R. Cyr, D. L. Osborn, and D. M.Neumark, Chem. Phys. Lett. (in press).

[10] L. F. Salter and B. A. Thrush, J. Chem. Soc., Faraday Trans. I **37**, 2025 (1977).

[11] M. E. Jacox, J. Phys. Chem. **87**, 3126 (1983).

[12] C. Chachaty, J. Chim. Phys. **62**, 728 (1965).

[13] C. Chachaty, C. Rosilio, J. Chim. Phys. **64**, 777 (1967).

[14] M. E. Jacox, J. Phys. Chem. **91**, 5038 (1987).

[15] R. B. Metz, D. R. Cyr, and D. M. Neumark, J. Phys. Chem. **95**, 2900 (1991).

[16] M. L. McKee, J. Am. Chem. Soc. **107**, 1900 (1985).

[17] M. L. McKee, J. Chem. Phys. **81**, 3580 (1984).

[18] L. J. Butler, D. Krajnovich, Y. T. Lee, G. Ondrey, and R. Bersohn, J. Chem. Phys. **79**, 1708 (1983).

[19] N. C. Blais, J. Chem. Phys. **79**, 1723 (1983).

[20] K. Q. Lao, E. Jensen, P. W. Kash, and L. J. Butler, J. Chem. Phys. **93**, 3958 (1990).

[21] D. B. Moss, K. A. Trentelman, and P. L. Houston, J. Chem. Phys. **96**, 237 (1991).

[22] D. L. Phillips and A. B. Myers, J. Phys. Chem. **95**, 7164 (1991).

[23] P. E. Schoen, M. J. Marrone, J. M. Schnur, and L. S. Goldberg, Chem. Phys. Lett. **90**, 272 (1982).

[24] A. M. Wodtke, E. J. Hintsa, and Y. T. Lee, J. Chem. Phys. **84**, 1044 (1986).

[25] A. M. Wodtke, E. J. Hinsta, and Y. T. Lee, J. Phys. Chem. **90**, 3549 (1986).

[26] J. H. M. Beijersbergen, W. J. van der Zande, P. G. Kistemaker, J. Los, T. Drewello, and N. M. M. Nibbering, J. Phys. Chem. **96**, 9288 (1992).

[27] D. J. Leahy, D. R. Cyr, D. L. Osborn, and D. M. Neumark, SPIE Proceedings **1858**, 49 (1993).

[28] M. A. Johnson, M. L. Alexander, and W. C. Lineberger, Chem. Phys. Lett. **112**, 285 (1984).

[29] R. E. Continetti, D. R. Cyr, and D. M. Neumark, Rev. Sci. Instrum. **63**, 1840 (1992).

[30] J. M. B. Bakker, J. Phys. E **6**, 785 (1973); **7**, 364 (1974).

[31] D. P. de Bruijn and J. Los, Rev. Sci. Instrum. **53**, 1020 (1982).

[32] C. Martin, P. Jelinsky, M. Lampton, R. F. Malina, and H. O. Anger, Rev. Sci. Instrum. **52**, 1067 (1981).

[33] D. F. Ogletree, G. S. Blackman, R. Q. Hwang, U. Stark, G. A. Somorjai and J. E. Katz, Rev. Sci. Instrum. **63**, 104 (1992).

[34] S. G. Lias, J. E. Bartmess, J. F. Liebman, J. L. Holmes, R. D. Levin, and W. G. Mallard, J. Phys. Chem. Ref. Data, **17**, *Supplement 1* (1988).

[35] R. N. Zare, Mol. Photochem. **4**, 1 (1972).

[36] C. F. Melius, private communication.

[37] P. Ho, D. J. Bamford, R. J. Buss, Y. T. Lee, and C. B. Moore, J. Chem. Phys. **76**, 3630 (1982).

[38] R. P. Saxon and M. Yoshimine, Can. J. Chem. **70**, 572 (1992).

[39] A. P. Cox and S. Waring, J. Chem. Soc. Faraday Trans. **2**, 1060 (1972).

[40] M. J. Frisch, M. Head-Gordon, H. B. Schlegel, K. Raghavachari, J. S. Binkley, C. Gonzalez, D. J. DeFrees, D. J. Fox, R. A. Whiteside, R. Seeger, C. F. Melius, J. Baker, R. L. Martin, L. R. Kahn, J. J. P. Stewart, E. M. Fluder, S. Topiol, and J. A. Pople, Gaussian Inc., Pittsburgh, PA.

[41] F. T. Williams, Jr., P. W. K. Flanagan, W. J. Taylor, and H. Schechter, J. Org. Chem. **30**, 2674 (1965).

[42] For general reviews of RRKM theory, see W. Forst, J. Chim. Phys. **87**, 715 (1990); *Theory of Unimolecular Reactions*, (Academic, New York, 1973); P. J. Robinson and K. A. Holbrook, *Unimolecular Reactions*, (Wiley-Interscience, New York, 1972).

[43] R. G. Gilbert and S. C. Smith, *Theory of Unimolecular and Recombination Reactions*, (Blackwell, Oxford, 1990).

[44] 'Loose' and 'tight' relate to the relative gain in entropy on going from the equilibrium geometry to the activated complex, with 'loose' implying a larger gain and vise-versa.

[45] D. L. Bunker and W. L. Hase, Quantum Chemistry Program Exchange catalog number QCPE-234, Department of Chemistry, Indiana University.

[46] D. J. Krajnovich, Ph. D. Thesis, UC Berkeley, 1983, p. 85.

[47] S. W. Benson, *Thermochemical Kinetics*, 2nd ed., (Wiley, New York, 1976).

[48] S. W. Benson and H. E. O'Neal, Natl. Stand. Ref. Data Ser., Natl. Bur. of Stand. **21**, 473 (1970).

[49] AFAC was obtained from the group of Prof. Y. T. Lee. It calculates the expected A_∞ factor based on Eqn. 6.12, by first calculating ΔS given vibrational frequencies for both the reactant and transition state.

[50] Dr. Sean Smith determined this C-N bond distance value, using a variational calculation, to have the minimum sum of states in the dissociating complex at the energy available when a 240 nm photon is used in our experiment.

[51] R. D. Levine in *Theory of Chemical Reaction Dynamics*, Vol. IV, M. Baer, Ed., (CRC Press, Boca Raton, FL, 1984), p. 31.

[52] K. Yamada, T. Kanekiyo, S. Tanaka, K. Naruchi, and M. Yamamoto, J. Am. Chem. Soc. **103**, 7003 (1981).

[53] O. Kikuchi, T. Kanekiyo, S. Tanaka, K. Naruchi, and K. Yamada, Bull. Chem. Soc. Jpn. **55**, 1509 (1982).

[54] E. Schmitz in *Comprehensive Heterocyclic Chemistry*, W. Lwowski, Ed., (Pergamon Press, Oxford, 1984).

[55] D. B. Galloway, J. A. Bartz, L. G. Huey, and F. F. Crim, J. Chem. Phys. **98**, 2107 (1993).

[56] M. Y. Balakina, M. B. Zuev, and I. D. Morozova, J. Mol. Struct. (Theochem) **183**, 291 (1989).

Appendix A

FRBM — The Ion Mass Spectrum and Total Photodissociation Cross Section Experiment Control and Data Acquisition Program

This Appendix presents FRBM the data acquisition program used both to collect ion mass spectra (see § 2.6.4.1) as well as radical total photodissociation cross section spectra (see § 2.7.5). Both options are invoked as selections from the main menu. Other main menu items include provision for 'manual' control of the lasers, setting experimental parameters, plotting and file manipulation. Sub menus are found within these options. In addition to the source files listed below, several units supplied by Turbo Pascal are compiled into the program, such as crt, dos and graph. A third-party unit, graphadd, allows extra printing capabilities. A file containing routines for interface to the CAMAC crate, namely camturbo.v4, must be compiled as an include file. Far calls must be disabled via compiler directive in the sections where this file is included. See the Turbo Pascal manuals for details and ramifications.

1 Description of Files

frbm.pas	Contains the list of units to be built into the program; calls initialization routines, followed by the main menu.
init_ovr.pas	Initializes the Turbo Pascal overlay manager to automatically overlay all units in frbm.pas marked with the compiler directive $O+.
initial.pas	Initializes all parameters necessary for the program to run properly, checks with the user that the system knows the proper date and time for the .log files that will be made, insures subdirectories for data exist (or

creates them), installs external graphing units and initializes GPIB controller and the dye lasers.

mainmenu.pas Presents the user with an array of menu items, calling the routine corresponding to the selection made by the user.

globals.pas Declares all variables, constants, data types, strings, etc. that need to be shared among units.

utility.pas This is a catch-all unit which includes somewhat common functions and procedures that can be called from other units. The general headings that are included here are: math, user input, saving and reading data files, and 'miscellaneous'

insgrdrv.pas Installs the third-party graphics driver that allows us to print to the laserwriter from within the program

grafutil.pas A pared-down version of graphlib.tgt (written by William Polik) modified to include routines from the Turbo Pascal graph unit where possible.

txsplot.pas Handles plotting for total photodissociation cross section data, including raw photodissociation signal, detached electron signal, laser power signal and normalized signal.

tofplot.pas Provides plots of ion signal vs. time-of-flight.

plotutil.pas Contains routines allowing plots to be sent to the laserwriter from within the program.

parameters.pas Searches for and reads in a setup file containing initial parameters for experiments. If this file is not found it prompts the user for this information and allows such a setup file to be written.

dyelaser.pas Allows full control of the detachment (grating) and dissociation (grating and etalon) dye lasers via GPIB interface.

twkplot.pas Provides graphical representation of the status of the various measured quantities (photodetached electrons, laser power, dissociated radical

signal) measured via the charge sensitive ADC's. Useful for efficient 'tweaking' of various parameters to improve output of these signals.

txs_exp.pas Controls the total cross section scanning and data acquisition and related routines such as the signal tweaking mode.

tof_exp.pas Controls the data acquisition during ion time-of-flight experiments.

gpibutil.pas Contains utilities for use with the National Instruments GPIB controller, which is used primarily to interface with the dye lasers.

tpdecl.pas Contains declarations necessary for GPIB operation.

2 Source Code Listing

2.1 frbm.pas

```
{ frbm.pas                                                    }

{  This program is a result of major modifications, additions, deletions and revisions to
   the ACROPOLIS program originally written by J. G. Loeser to control an experiment
   with similar components.  It has been upgraded to version 5.5 of Turbo Pascal, and
   the include files converted to separate units.  Overlays are used to enable the program
   to be quite large.  Graphics routines from TP are now used, with the small number of
   useful routines from the old graphlib unit now found in grafutil.  Graphics drivers
   from Graf/Drive Plus have been utilized to allow graphing directly to our PostScript
   LaserWriter from within the program.
   Questions that may arise concerning various aspects of this code can be addressed by
   the appropriate section of either the Turbo Pascal 5.5 Reference Manual or Users
   Guide supplied by Borland, which are altogether quite helpful.

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

{ main program FRBM                                           }

program FRBM (input, output);

{$F+}

{$M 16384,200000,300000}

uses overlay,init_ovr,initial,mainmenu;

{$O initial}
{$O globals}
```

```
{$O utility}
{$O insgrdrv}
{$O grafutil}
{$O txsplot}
{$O tofplot}
{$O plotutil}
{$O parameters}
{$O dyelaser}
{$O twkplot}
{$O txs_exp}
{$O tofms}
{$O mainmenu}

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

begin { main program FRBM }

 initialize_ovr;
 initialize;
 main_menu;

end. { main program FRBM }
```

2.2 init_ovr.pas

```
{ init_ovr.pas                                    }

{ contents :                                      }
{ procedure initialize_overlay                        }
{     initializes overlay                           }

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

unit init_ovr;

{$F+}

interface

uses overlay;

procedure initialize_overlay;

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

implementation

procedure initialize_overlay;
const
 OvrMaxSize = 40000;
var
```

```
  y : integer;
  OvrName : String[79];
  Size : LongInt;
begin
  OvrName := 'frbm.ovr';
  repeat
    OvrInit(OvrName);
    if OvrResult = ovrNotFound then
      begin
        writeln ('Overlay file not found: ', Ovrname);
        write ('Enter correct overlay file name: ');
        readln (OvrName);
      end;
  until OvrResult <> ovrNotFound;
  if OvrResult <> ovrOK then
    begin
      writeln ('Overlay Manager Error!');
      if OvrResult = ovrNoMemory then
        begin
          writeln ('Not enought memory for overlay buffer');
        end;
      Halt(1);
    end;
  OvrSetBuf(OvrMaxSize);
  if OvrResult <> ovrOk then
    begin
      case OvrResult of
        ovrError:    writeln ('Overlay manager error');
        ovrNoMemory: writeln ('Not enough memory for extra overlay buffer size');
      end;
      write ('OvrResult = ', OvrResult, '. Press <enter> to continue ...');
      readln;
    end;
end; { procedure initialize_overlay }

end. { unit init_ovr }
```

2.3 initial.pas

```
{ initial.pas                                                    }

{ contents :                                                     }
{ procedure initialize                                             }
{     initializes everything (except for overlay manager)          }

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

unit initial;

{$O+}
{$F+}
```

```
interface

uses overlay,globals,parameters,dyelaser,utility,gpibutil,txsplot,tofplot,
   plotutil,insgrdrv,graph,graphadd,grafutil,CRT,DOS;

procedure check_time;
procedure get_data_path (experiment_type : integer);
procedure initialize;

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

implementation

procedure check_time;
var
 date, time : string;
begin
 clrscr;
 form_date_and_time_strings (date, time);
 delete (time, 6, 3);
 GoToXY (1,6);
 TextColor (14);
 writeln ('It is important for the log file to have the ',
       'correct time and date.');
 writeln;
 writeln ('If it is (approximately) ', time, ' on (exactly) ', date,
       ' then continue. ');
 writeln;
 writeln ('If not, quit to DOS and set correct time and/or date.');
 if not ask_continue then
  begin
   clrscr;
   TextColor(15);
   halt(1);
  end;
end;

procedure get_data_path (experiment_type : integer);
var
 date, time, user_data_path, active_directory : string;
begin
 case experiment_type of

   txs :
    begin
     clrscr;
     GoToXY (1,10);
     getdir (0, active_directory);
     form_date_and_time_strings (date, time);
     txs_data_path := 'c:\turbo\frbm\nm';
     writeln ('Please enter the full path to the directory where',
           ' total photodissociation ');
     writeln;
```

```
    writeln ('cross section data should be stored.');
    writeln;
    writeln ('All parts of the path with the exception of the last',
          ' subdirectory should');
    writeln;
    write ('already exist [c:\turbo\frbm\nm] : ');
    readln (user_data_path);
    if length (user_data_path) > 0 then
     begin
      { make sure sub directory exists or is created }
      {$I-}
      chdir (user_data_path);
      if IOResult <> 0 then
       begin
        mkdir (user_data_path);
        if IOResult <> 0 then
         begin
          writeln ('Path to data directory is invalid.  You must try again.');
          writeln;
          get_data_path (txs);
         end;
       end;
      txs_data_path := user_data_path;
     end;
    txs_data_path := txs_data_path + '\' + date;
    chdir (txs_data_path);
    if IOResult <> 0 then
     begin
      mkdir (txs_data_path);
     end;
    {$I+}
    chdir (active_directory);
   end;

tof :
  begin
   clrscr;
   GoToXY (1,10);
   getdir (0, active_directory);
   tof_data_path := 'c:\turbo\frbm\nm\tof';
   writeln ('Please enter the full path to the directory where',
         ' time of flight mass ');
   writeln;
   writeln ('spectra data should be stored.');
   writeln;
   writeln ('All parts of the path with the exception of the last',
         ' subdirectory should');
   writeln;
   write ('already exist [c:\turbo\frbm\nm\tof] : ');
   readln (user_data_path);
   if length (user_data_path) > 0 then
    begin
     { make sure sub directory exists or is created }
     {$I-}
```

```
          chdir (user_data_path);
          if IOResult <> 0 then
            begin
              mkdir (user_data_path);
              if IOResult <> 0 then
                begin
                  writeln ('Path to data directory is invalid.  You must try again.');
                  writeln;
                  get_data_path (tof);
                end;
            end;
          tof_data_path := user_data_path;
        end;
      {SI+}
      chdir (active_directory);
    end;
  end;
end;  { procedure get_data_paths }


procedure initialize;
var
  y : integer;
begin
  setcbreak (TRUE);

  check_time;

  get_data_path (txs);
  get_data_path (tof);

  new (data);

  count := 0;

  x_axis := laser_wavelength;
  y_axis := normalized_signal;
  full_parameter_listing := FALSE;
  ready_for_etalon_scan := FALSE;

  SymbolInit;

  InstallScreenGraphicsDriver;

  InstallGrafPlusDriver (PostScriptLaserWriterFile, PortFILE, '$PS');

  InstallGrafPlusDriver (HP7470Plotter, PortCOM1, '$HP7470');

  GraphicsOn := FALSE;

  Init_GPIB;
  set_gpib_remote (TRUE);
  clrscr;
  GoToXY (1,11);
```

```
get_grating_cal (detachment_dye_laser);
writeln;
get_grating_cal (dissociation_dye_laser);
set_gpib_remote (FALSE);

with stp.etalon_parameters do
        begin
          etalon_normal_position := 0;
          lambda_0 := 275.0;
          upper_lambda := 275.0;
          initial_grating_position :=
              wave_to_grating_drive (dissociation_dye_laser, 275.0);
          max_etalon_scan_range := 2.0;
          min_etalon_delta_lambda := 0.0;
        end;

initial_read := TRUE;
if not read_set_up then
  begin
    with stp do
      begin
        detach_lambda := 380;
        cal_offset := 0.0;
        start_lambda := 270;
        end_lambda := 275;
        delta_lambda := 0.05;
        shots_per_point := 200;
        doubling := TRUE;
        use_etalon := FALSE;
        set_grating_order (dissociation_dye_laser);
        ADC_channel [laser_power] := 0;
        ADC_channel [dissociated_radical_signal] := 2;
        ADC_channel [detached_electron_signal] := 4;
        ADC_channel [iodine_reference] := 6;
        zero [detached_electron_signal] := 0.0;
        zero [laser_power] := 0.0;
        zero [dissociated_radical_signal] := 0.0;
        zero [iodine_reference] := 0.0;
        zero_average_shot_number := 200;
      end;
      show_parameters;
      y := wherey;
      window (1, y, 80, 24);
      clrscr;
      get_detachment_wavelength;
      get_cal_offset;
      get_initial_wavelength;
      get_final_wavelength;
      get_lambda_step;
      get_shots_per_point;
      get_doubling;
      get_use_etalon;
      set_grating_order (dissociation_dye_laser);
      get_ADC_channels;
```

```
      get_zero_average_shot_number;
      write_set_up;
    end;
  initial_read := FALSE;

  set_grating_order (detachment_dye_laser);
  set_gpib_remote (TRUE);
  move_dye_laser (detachment_dye_laser, stp.detach_lambda);
  set_gpib_remote (FALSE);

  delay (1000);

  axis_label [laser_wavelength]          := 'Dissociation Laser Wavelength [nm]';
  axis_label [detached_electron_signal]  := 'Electron Signal';
  axis_label [laser_power]               := 'Laser Power';
  axis_label [dissociated_radical_signal] := 'Fragment Signal';
  axis_label [normalized_signal]         := 'Normalized Signal';
  axis_label [laser_wavenumber]          := 'Dissociation Laser Energy [cm-1]';
  axis_label [iodine_reference]          := 'Iodine Reference Cell';

  tof_axis_label [time]                  := 'Time of Flight [5 ns / channel]';
  tof_axis_label [tof_signal]            := 'TOF Signal';
  tof_axis_label [mass]                  := 'Mass [amu]';

end; { procedure initialize }

end. { unit initial }
```

2.4 mainmenu.pas

```
{ mainmenu.pas                                                    }

{ contents :                                                      }
{ procedure main_menu                                             }
{     menu of things to do before and after experiment            }

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

unit mainmenu;

{$O+}
{$F+}

interface

uses globals,initial,parameters,utility,txs_exp,tofms,txsplot,CRT;

procedure main_menu;

{ ********************************************************************** }
```

```
implementation

procedure main_menu;
var
 ch : char;
 y : integer;
begin { procedure main_menu }
 show_parameters;
 repeat
  window (1, 1, 36, 24);
  clrscr; textcolor (15);
  writeln ('E : Set Experimental Parameters');
  writeln ('Z : Obtain Zeros');
  writeln ('C : Control of Laser & ADC');
  writeln ('M : Mass Spectrometer Experiment');
  writeln ('T : Photodissn Total XS Experiment');
  writeln ('D : Photodissn Dynamics Experiment');
  writeln ('P : Plot Menu');
  writeln ('S : Save Data File');
  writeln ('R : Read Data File');
  writeln ('Q : Quit Program');
  writeln ('-----------------------------------');
  writeln; y := wherey; textcolor (14);
  repeat
   window (1, y, 80, 24); clrscr;
   write ('? > '); ch := Readkey; ch := upcase(ch); writeln (ch);
   writeln;
  until ch in ['E', 'Z', 'C', 'M', 'T', 'D', 'P', 'S', 'R', 'Q'];
  case ch of
   'E' : experimental_parameters;
   'Z' : obtain_zeros;
   'C' : manual_experiment;
   'M' : begin
         switch_to_mass_spec;
         show_parameters;
         window (1, y, 80, 24);
        end;
   'T' : run_total_photodissociation_cross_section_experiment;
   'D' : reserved;
   'P' : begin
         txs_plotting_parameters;
         show_parameters;
        end;
   'S' : begin
         save_data_file (txs);
         show_parameters;
        end;
   'R' : begin
         read_data_file (txs);
         show_parameters;
        end;
  end; { case ch }
 until (ch = 'Q');
 if not ask_for_boolean ('Are you sure you want to quit ? [Y/N] : ') then
```

```
  begin
    main_menu;
  end;
end; { procedure main_menu }

end. { unit mainmenu }
```

2.5 globals.pas

```
{ globals.pas
 this unit contains all global variables neccessary for frbm.pas        }

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

unit globals;

{$O+}
{$F+}

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

interface

uses graphadd;

const
 max_data = 1024;
 max_column = 6;

 laser_wavelength = 0;
 detached_electron_signal = 1;
 laser_power = 2;
 dissociated_radical_signal = 3;
 normalized_signal = 4;
 laser_wavenumber = 5;
 iodine_reference = 6;

 set_up_file_name = 'setup.dat';

 time = 0;
 tof_signal = 1;
 mass = 2;
 memory : integer = 22; { CAMAC crate slot for 4101 }
 digitizer : integer = 19; { CAMAC crate slot for 2001AS }

 txs = 0;
 tof = 1;

 { descriptive constants for CAMAC function calls }
 ADC      : integer = 16; { CAMAC crate station of LeCroy 2249SG ADC }
 test_LAM  : integer =  8; { CAMAC F8 }
```

```
  read_channel : integer =  0; { CAMAC F0 }
  dataway_z : integer =  1; { CAMAC Initialize }
  dataway_c : integer =  2; { CAMAC Clear }
  dataway_i : integer =  4; { CAMAC Inhibit }
  CRIACL    : integer = 64; { Clear Request Inhibit & ACL detection registers }

  max_tweak_limit = 200;

  lines : boolean = TRUE;
  plot_lines : boolean = TRUE;
  plot_from_zero : boolean = TRUE;

  info : DrvINFO = (DrvMem:36000; DrvWrkdrive:'e';
          DrvOutfile : 'e:\plot.out';
       escchk : TRUE; nohead : FALSE );
  PathToGraphics : string = 'c:\turbo\graphics';
  GraphComm : word = Baud9600 or ParNone or Data8 or Stop1;
  XON : boolean = FALSE;

type
  name = string[80];

  data_pointer = ^data_array;
  data_array = array [1..max_data, 0..max_column] of real;

  tweak_data_pointer = ^tweak_data_array;
  tweak_data_array = array [1..max_tweak_limit, 0..max_column] of real;

  axis_label_array = array [0..max_column] of name;
  real_column_array = array [0..max_column] of real;
  int_column_array = array [0..max_column] of integer;

  tof_real_column_array = array [0..1] of real;
  tof_data_pointer = ^tof_data_array;
  tof_data_array = array [0..4095] of real;

  etalon_parameters_type = record
     etalon_normal_position : integer;
     lambda_0 : real;
     upper_lambda : real;
     initial_grating_position : real;
     max_etalon_scan_range : real;
     min_etalon_delta_lambda : real;
   end; { record etalon_parameters_type }

  set_up = record
     detach_lambda : real;
     cal_offset : real;
     start_lambda : real;
     end_lambda : real;
     delta_lambda : real;
     shots_per_point : integer;
     doubling : boolean;
     use_etalon : boolean;
```

```
    etalon_parameters : etalon_parameters_type;
    grating_order_detachment_dye_laser : integer;
    grating_order_dissociation_dye_laser : integer;
    ADC_channel : int_column_array;
    zero : real_column_array;
    zero_average_shot_number : integer;
   end; { record set_up }

var
 Year, Month, Day, DayofWeek, Hour, Minute, Second, Sec100 : word;

 txs_data_path : string;
 tof_data_path : string;
 dyn_data_path : string;

 ch : char;

 data : data_pointer;

 stp : set_up;
 count : integer;

 min, max : real_column_array; { minimum and maximum data values }
 plot_min, plot_max : real_column_array; { min and max for plot axes }
 thresh_min, thresh_max : real_column_array;
                { threshold values for changing plotting scale }
 x_axis : integer;
 y_axis : integer;
 axis_label : axis_label_array;
 tof_axis_label : axis_label_array;
 plot_little_crosses : boolean;
 file_name : name;
 data_file_name : name;
 log_file_name : name;
 day_of_experiment : name;
 time_of_experiment : name;
 data_file : text;
 log_file : text;

 tweak_limit : integer;
 tweaking : boolean;
 tweak_count : integer;
 tweak_shots_per_point : integer;
 tweak_data : tweak_data_pointer;
 tweak_min, tweak_max : real_column_array;
 tweak_thresh_min, tweak_thresh_max : real_column_array;
 tweak_plot_min, tweak_plot_max : real_column_array;

 detachment_dye_laser : integer;
 dissociation_dye_laser : integer;
 sb1, sb2 : integer;

 initial_read : boolean;
 upper_left_corner_y : integer;
```

```
  full_parameter_listing : boolean;

  ready_for_etalon_scan : boolean;
  new_zeros_obtained : boolean;

  sweeps_to_average : integer;
  digitizer_control : integer;
  memory_control : integer;
  data_count : integer;
  tof_data : tof_data_pointer;
  tof_data_max, tof_data_min : real;
  mass_spec_delay : real;
  tof_plot_min, tof_plot_max : tof_real_column_array;
  df_data_count : integer;
  df_digitizer_control : integer;
  df_memory_control : integer;
  df_sweeps_to_average : integer;
  df_mass_spec_delay : real;

  GraphDriver : integer;  { The Graphics device driver }
  GraphMode   : integer;  { The Graphics mode value }
  ErrorCode   : integer;  { Reports any graphics errors }
  MaxColor    : word;     { The maximum color value available }
  OldExitProc : Pointer;  { Saves exit procedure address }
  GraphicsOn  : Boolean;  { Saves status of CRT mode (graphics or text) }
  HP7470Plotter : integer;
  PostScriptLaserWriterFile : integer;
  status : word;
  magic : ^byte;
  wordptr : ^word;
  hold : byte;

implementation
end. { unit globals }
```

2.6 utility.pas

```
{ utility.pas                                          }

{ contents :                                           }

{ i. math                                              }
{ function int_power (base : real;                             }
{                power : integer) : real                }
{    raises any real 'base' to positive integer 'power'        }

{ ii. user input                                       }

{ ii.a. yes/no                                         }
{ function ask_for_boolean (prompt : name) : boolean           }
{    user must answer yes/no question, 'prompt'                 }
```

```
{ ii.b. continue/quit                                    }
{ function ask_continue : boolean                        }
{     user must hit <enter> to continue or Q to quit     }

{ ii.c. numbers                                          }
{ these functions convert strings to numbers and ignore inappropriate    }
{ characters -- beware of misinterpretation of invalid input             }
{ true if user enters 'response' between 'min' and 'max' or              }
{               '<' which sets 'response' to 'min' or                    }
{               '>' which sets 'response' to 'max'                       }
{ false if user just presses 'enter'; 'response' is unchanged            }
{ 'prompt' is repeated if user enters something else                     }

{ function ask_for_integer (var response : integer;                      }
{                   prompt : name;                                       }
{                   min, max : integer) : boolean                        }

{ function ask_for_real (var response : real;                            }
{                 prompt : name;                                         }
{                 min, max : integer) : boolean                          }

{ ii.d. wavelength to wavenumber conversion and vise versa               }

{ function wl2wn (wl : real) : real;                                     }

{ function wn2wl (wn : real) : real;                                     }

{ ii.e. file names                                       }

{ function exist (file_name : name) : boolean                            }
{     true if file called 'file_name' is present on disk                 }

{ function legal_name (file_name : name) : boolean                       }
{     true if file name 'file_name' is a legal one                       }

{ procedure get_out_file_name (var file_name : name;                     }
{                   var overwrite : boolean                              }
{                   prompt : name                                        }
{                   var got_good_out_file_name : boolean                 }
{     true if user enters a new 'file_name' or wants to overwrite an     }
{     existing file; if overwrite, sets overwrite to TRUE                }

{ function get_in_file_name (var file_name : name;                       }
{                   prompt : name) : boolean                             }
{     true if user entered 'file_name' of an existing file               }


{ iii. save and read data                                }

{ procedure save_data (experiment_type : integer);                       }
{     save data to individual'.dat' file, append scan parameter info     }
{     to one 'date.log' file per day                                     }
```

```
{ iv. miscellaneous                                    }

{ procedure reserved;                                  }
{     displays message : 'reserved for future development'          }

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

unit utility;

{$O+}
{$F+}

interface

uses globals,DOS,CRT;

function int_power (base : real; power : integer) : real;
function ask_for_boolean (prompt : name) : boolean;
function ask_continue : boolean;
function ask_for_integer (var response : integer; prompt : name;
                  min, max : integer) : boolean;
function ask_for_real (var response : real; prompt : name;
                min, max : real) : boolean;
function wl2wn (wl : real) : real;
function wn2wl (wn : real) : real;
function I2S (Val, Digit : Integer) : String;
function R2S (Val : real; Digit, Decimal : Integer) : String;
function exist (file_name : name) : boolean;
function legal_name (file_name : name) : boolean;
procedure get_out_file_name (var file_name : name; var overwrite : boolean;
                   experiment_type : integer; prompt : name;
                   var got_good_out_file_name : boolean);
function get_in_file_name (var file_name : name; experiment_type : integer;
                  prompt : name) : boolean;
procedure form_date_and_time_strings (var date_string : name;
                        var time_string : name);
procedure save_data_file (experiment_type : integer);
procedure read_data_file (experiment_type : integer);
procedure reserved;

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

implementation

function int_power (base : real;
            power : integer) : real;
begin { function int_power }
 if (power = 0) then
   int_power := 1.0
 else
   int_power := base * int_power (base, (power - 1));
end;  { function int_power }
```

```
{ ************************************************************************ }

function ask_for_boolean (prompt : name) : boolean;
var
 ch : char;
begin { function ask_for_boolean }
 ch := ' ';
 repeat
  write (prompt); ch := ReadKey; ch := upcase (ch); writeln (ch);
  if ((ch <> 'Y') and (ch <> 'N')) then
   writeln ('Please type "Y" or "N"');
 until ((ch = 'Y') or (ch = 'N'));
 ask_for_boolean := (ch = 'Y');
end; { function ask_for_boolean }

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

function ask_continue : boolean;
var
 ch : char;
begin { function ask_continue }
 repeat
  writeln; writeln;
  write ('Hit <enter> to continue or Q to quit : ');
  ch := readkey; ch := upcase (ch); write (ch);
  if ch = 'Q' then ask_continue := false;
  if ch = #13 then ask_continue := true;
 until ((ch = 'Q') or (ch = #13));
end; { function ask_continue }

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

function ask_for_integer (var response : integer;
                          prompt : name;
                          min, max : integer) : boolean;
var
 answer, code, i : integer;
 input_string : string[8];
begin { function ask_for_integer }
 answer := 0;
 code := 1;
 repeat
  write (prompt); readln (input_string);
  if (input_string = '') then
   begin
    ask_for_integer := FALSE;
    exit;
   end;
  if (input_string = '<') then
   begin
    ask_for_integer := TRUE;
    response := min;
    exit;
   end;
```

```
    if (input_string = '>') then
     begin
      ask_for_integer := TRUE;
      response := max;
      exit;
     end;
    i := 1;
    repeat
     if not (input_string [i] in ['0'..'9']) then
      delete (input_string, i, 1)
     else
      i := i + 1;
    until (i = (length (input_string) + 1));
    val (input_string, answer, code);
  until ((input_string <> '') and (answer >= min) and (answer <= max) and
        (code = 0));
  ask_for_integer := TRUE;
  response := answer;
end;  { function ask_for_integer }

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

function ask_for_real (var response : real;
                       prompt : name;
                       min, max : real) : boolean;
var
  answer : real;
  code, i : integer;
  input_string : string[10];
begin { function ask_for_real }
  answer := 0.0;
  code := 1;
  repeat
    write (prompt); readln (input_string);
    if (input_string = '') then
     begin
      ask_for_real := FALSE;
      exit;
     end;
    if (input_string = '<') then
     begin
      ask_for_real := TRUE;
      response := min;
      exit;
     end;
    if (input_string = '>') then
     begin
      ask_for_real := TRUE;
      response := max;
      exit;
     end;
    i := 1;
    repeat
     if not (input_string [i] in ['0'..'9', '.', 'e', 'E', '-', '+']) then
```

```
      delete (input_string, i, 1)
     else
      i := i + 1;
   until (i = (length (input_string) + 1));
   val (input_string, answer, code);
  until ((input_string <> '') and (answer >= min) and (answer <= max) and
        (code = 0));
  ask_for_real := TRUE;
  response := answer;
end;  { function ask_for_real }

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

function wl2wn (wl : real) : real;
begin
wl2wn := (10000000 / wl);
end;

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

function wn2wl (wn : real) : real;
begin
wn2wl := (10000000 / wn);
end;

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

function I2S (Val, Digit : Integer) : String;
var
  buffer : string;
begin
  str(Val:Digit, Buffer);
  I2S := Buffer;
end;

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

function R2S (Val : real; Digit, Decimal : Integer) : String;
var
  buffer : string;
begin
  str(Val:Digit:Decimal, Buffer);
  R2S := Buffer;
end;

{ ************************************************************************ }
function exist (file_name : name) : boolean;
var
  test_file : file;
begin { function exist }
  assign (test_file, file_name);
  {SI-}
  reset (test_file);
  close (test_file);
```

```
 {$I+}
 exist := (IOresult = 0);
end;  { function exist }

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

function legal_name (file_name : name) : boolean;
var
 test_file : file;
begin { function legal_name }
 assign (test_file, file_name);
 {$I-}
 rewrite (test_file);
 {$I+}
 legal_name := (IOresult = 0);
 close (test_file);
end; { function legal_name }

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

procedure get_out_file_name (var file_name : name; var overwrite : boolean;
                    experiment_type : integer; prompt : name;
                    var got_good_out_file_name : boolean);
var
 extension : string;
 i : integer;
begin { function get_out_file_name }
 case experiment_type of
 txs :
  begin
   extension := '.txs';
  end;
 tof :
  begin
   extension := '.tof';
  end;
 dyn :
  begin
   extension := '.dyn';
  end;
 end;
 file_name := '';
 overwrite := FALSE;
 write (prompt); readln (file_name);
 if pos ('.', file_name) = 0 then
  begin
   file_name := file_name + extension;
  end;
 if (file_name = extension) then
  begin
   writeln;
   if ask_for_boolean
        ('Do you really want to NOT save this file? [Y/N] : ') then
    begin
```

```
      got_good_out_file_name := FALSE;
      exit;
     end
    else
     begin
      get_out_file_name (file_name, overwrite, experiment_type, prompt,
                got_good_out_file_name);
     end;
  end
 else if exist (file_name) then
     begin
      writeln; writeln; write ('          ');
      if ask_for_boolean ('Overwrite '+ file_name + '? [Y/N] : ') then
       begin
        overwrite := TRUE;
       end
      else
       begin
        writeln;
        get_out_file_name (file_name, overwrite, experiment_type, prompt,
                got_good_out_file_name);
       end;
     end
 else if not legal_name (file_name) then
  begin
   writeln ('Not a legal name for a file!');
   get_out_file_name (file_name, overwrite, experiment_type, prompt,
              got_good_out_file_name);
  end
 else
  got_good_out_file_name := TRUE;
end;  { function get_out_file_name }

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

function get_in_file_name (var file_name : name; experiment_type : integer;
                prompt : name) : boolean;
var
 extension : string;
 i : integer;
begin { function get_in_file_name }
 case experiment_type of
 txs :
  begin
   extension := '.txs';
   chdir (txs_data_path);
  end;

 tof :
  begin
   extension := '.tof';
   chdir (tof_data_path);
  end;
```

```
  dyn :
    begin
      extension := '.dyn';
      chdir (dyn_data_path);
    end;
  end;
  file_name := '';
  write (prompt); readln (file_name);
  if pos ('.', file_name) = 0 then
    begin
      file_name := file_name + extension;
    end;
  if (file_name = extension) then
    begin
      get_in_file_name := FALSE;
      exit;
    end
  else if not exist (file_name) then
    begin
      writeln (file_name + ' does not exist.');
      get_in_file_name := get_in_file_name (file_name, experiment_type, prompt);
    end
  else
    get_in_file_name := TRUE;
end;  { function get_in_file_name }

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

procedure form_date_and_time_strings (var date_string : name;
                                      var time_string : name);
var
  Yearstr, Monthstr, Daystr, Hourstr, Minstr, Secstr : string;
begin
  GetTime (Hour, Minute, Second, Sec100);
  time_string := '';
  Hourstr := I2S (Hour,0);
  if length (Hourstr) = 1 then
    begin
      Hourstr := '0' + Hourstr;
    end;
  Minstr := I2S (Minute,0);
  if length (Minstr) = 1 then
    begin
      Minstr := '0' + Minstr;
    end;
  Secstr := I2S (Second,0);
  if length (Secstr) = 1 then
    begin
      Secstr := '0' + Secstr;
    end;
  time_string := Hourstr + ':' + Minstr + ':' + Secstr;

  GetDate (Year, Month, Day, DayofWeek);
  date_string := '';
```

```
  Yearstr := I2S (Year,0);
  delete (Yearstr,1,2);
  Monthstr := I2S (Month,0);
  if length (Monthstr) = 1 then
    begin
      Monthstr := '0' + Monthstr;
    end;
  Daystr := I2S (Day,0);
  if length (Daystr) = 1 then
    begin
      Daystr := '0' + Daystr;
    end;
  date_string := Yearstr + Monthstr + Daystr;
end;

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

procedure save_data_file (experiment_type : integer);
var
  i, IOErrorCode : integer;
  CDriveFreeMemory : longint;
  overwrite, got_good_out_file_name : boolean;
  data_point : real_column_array;
  active_directory : string;
  Yearstr, Monthstr, Daystr, Hourstr, Minstr, Secstr : string;
  ms_time : real;
begin { procedure save_data }

  form_date_and_time_strings (day_of_experiment, time_of_experiment);

  window (1, 1, 80, 24);
  clrscr;
  GoToXY (1, 10);

  CDriveFreeMemory := DiskFree(0) div 1024;
  if CDriveFreeMemory < 400 then
    begin
      TextColor (4 + BLINK);
      writeln ('Free Memory on C Drive less than 400 kb!');
      writeln;
      TextColor (14);
    end;
  getdir (0, active_directory);
  case experiment_type of

    txs :
      begin
        chdir (txs_data_path);
        get_out_file_name (data_file_name, overwrite, txs,
           'Output data file name? [No extension necessary if .txs] : ',
            got_good_out_file_name);
        if not got_good_out_file_name then
          begin
            chdir (active_directory);
```

```
    exit;
  end;

{ write out data in text format }
{$I-}
assign (data_file, data_file_name);
rewrite (data_file);
for i := 1 to count do
  begin
    write (data_file, data^ [i, laser_wavelength]:10:5, ' ');
    write (data_file, data^ [i, detached_electron_signal]:10:5, ' ');
    write (data_file, data^ [i, laser_power]:10:5, ' ');
    write (data_file, data^ [i, dissociated_radical_signal]:10:5, ' ');
    writeln (data_file, data^ [i, iodine_reference]:10:5);
  end;
close (data_file);
IOErrorCode := IOResult;
if IOErrorCode <> 0 then
  begin
    if IOErrorCode = 101 then
      begin
        writeln('C Disk Full!.  Prepare to save to A: drive.');
        if ask_continue then
          begin
            assign (data_file, ('a:\save.txs'));
            rewrite (data_file);
            for i := 1 to count do
              begin
                write (data_file, data^ [i, laser_wavelength]:10:5, ' ');
                write (data_file, data^ [i, detached_electron_signal]:10:5, ' ');
                write (data_file, data^ [i, laser_power]:10:5, ' ');
                write (data_file, data^ [i, dissociated_radical_signal]:10:5, ' ');
                writeln (data_file, data^ [i, iodine_reference]:10:5);
              end;
            close (data_file);
          end;
      end
    else
      begin
        writeln ('IO ERROR! Recommend exiting program',
                 ' until you figure it out.');
      end;
  end;
{$I+}

form_date_and_time_strings (day_of_experiment, time_of_experiment);

log_file_name := day_of_experiment + '.log';
if exist (log_file_name) then
  begin
    assign (log_file, log_file_name);
    append (log_file);
  end
else
```

```
 begin
  assign (log_file, log_file_name);
  rewrite (log_file);
 end;
with stp do
 begin
  writeln (log_file);  writeln (log_file);
  if overwrite then
    begin
     writeln;
     writeln (log_file, '*******  THE FOLLOWING LOG RECORD OVERWRITES ',
          'THE PRIOR LOG RECORD *******');
     writeln (log_file, '********************** WITH THE SAME DATA ',
          'FILE NAME **********************');
     writeln;
    end;
  writeln (log_file, 'Total Photodissociation Cross Section Log Record ',
        'for Data File : ', data_file_name);
  writeln (log_file, 'Recorded in D-21 Latimer at ',
        time_of_experiment, ' on ', day_of_experiment);
  writeln (log_file);
  with stp do
  begin
   writeln (log_file, '     Detachment Wavelength : ', detach_lambda:12:7);
   writeln (log_file, '  Laser Calibration Offset : ', cal_offset:12:7);
   writeln (log_file, '       Initial Wavelength : ', start_lambda:12:7);
   writeln (log_file, '         Final Wavelength : ', end_lambda:12:7);
   writeln (log_file, '              Lambda Step : ', delta_lambda:12:7);
   writeln (log_file, '          Shots per Point : ', shots_per_point:4);
   write (log_file,'   Autotracker (Doubling?) : ');
   if stp.doubling then
     writeln (log_file, ' YES')
   else
     writeln (log_file, '  NO');
   write (log_file,'              Use Etalon : ');
   if stp.use_etalon then
     writeln (log_file, ' YES')
   else
     writeln (log_file, '  NO');
   write (log_file,'Dissociation Grating Order : ');
   writeln (log_file, grating_order_dissociation_dye_laser:4);
   write (log_file,'  Detached Electron Signal : ');
   writeln (log_file, ADC_channel [detached_electron_signal]:4);
   write (log_file,'             Laser Power : ');
   writeln (log_file, ADC_channel [laser_power]:4);
   write (log_file, 'Dissociated Radical Signal : ');
   writeln (log_file, ADC_channel [dissociated_radical_signal]:4);
   write (log_file,'     Iodine Reference Cell : ');
   writeln (log_file, ADC_channel [iodine_reference]:4);
   write (log_file,'  Zero Average Shot Number : ');
   writeln (log_file, zero_average_shot_number:4);
   write (log_file,'    Detached Electron Zero : ');
   writeln (log_file, zero [detached_electron_signal]:9:4);
   write (log_file,'         Laser Power Zero : ');
```

```
          writeln (log_file, zero [laser_power]:9:4);
          write (log_file,' Dissociated Radical Zero : ');
          writeln (log_file, zero [dissociated_radical_signal]:9:4);
          write (log_file, 'Iodine Reference Cell Zero : ');
          writeln (log_file, zero [iodine_reference]:9:4);
        end;
        if overwrite then
          begin
            writeln (log_file);
            writeln (log_file, '******  THE PRECEEDING LOG RECORD OVERWRITES ',
                  'A PREVIOUS LOG RECORD  ******');
            writeln (log_file, '********************** WITH THE SAME DATA ',
                  'FILE NAME **********************');
          end;
      end;
    close (log_file);
  end;

 tof :
  begin
    chdir (tof_data_path);
    get_out_file_name (data_file_name, overwrite, tof,
       'Output data file name? [No extension necessary if .tof] : ',
        got_good_out_file_name);
    if not got_good_out_file_name then
      begin
        chdir (active_directory);
        exit;
      end;

    { write tof data file }
    assign (data_file, data_file_name);
    rewrite (data_file);
    writeln(data_file, data_count);
    writeln(data_file, digitizer_control);
    writeln(data_file, memory_control);
    writeln(data_file, sweeps_to_average);
    writeln(data_file, mass_spec_delay);
    for i := 0 to data_count do
      begin
        writeln (data_file, tof_data^ [i]:10:0);
      end;
    close (data_file);
  end;  { write tof data file }
 end;
chdir (active_directory);
end;  { procedure save_data }

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

procedure read_data_file (experiment_type : integer);
var
 i, j : integer;
 normalized_detached_electron_signal : real;
```

```
 normalized_laser_power : real;
 active_directory : string;
begin { procedure read_data_file }
 getdir (0, active_directory);
 window (1, 1, 80, 24);
 clrscr;
 GoToXY (1, 8);
 writeln ('WARNING : This action will overwrite current data in RAM.');
 writeln;
 writeln ('If you have not done so already, save the ',
       'current data after exiting now.');
 writeln;
 if not ask_continue then
  begin
   exit;
  end;
 writeln;  writeln;
 case experiment_type of

  txs :
   begin
    if not get_in_file_name (data_file_name, txs,
      'Enter file name [No extension necessary if .txs] : ') then
     begin
      chdir (active_directory);
      exit;
     end;

    {initialize}
    for i := 1 to max_data do
     for j := 0 to max_column do
      data^ [i, j] := 0.0;

  {   plot_little_crosses := (abs ((stp.end_lambda - stp.start_lambda) /
              stp.delta_lambda) <= 200.0);}

    x_axis := laser_wavelength;

    max [detached_electron_signal]  := 1.0;
    min [detached_electron_signal]  := 0.0;

    max [laser_power]          := 1.0;
    min [laser_power]          := 0.0;

    max [dissociated_radical_signal] := 1.0;
    min [dissociated_radical_signal] := 0.0;

    max [normalized_signal]       := 1.0;
    min [normalized_signal]       := 0.0;

    max [iodine_reference]    := 1.0;
    min [iodine_reference]    := 0.0;

    { read out data in text format }
```

```
{$I-}
assign (data_file, data_file_name);
reset (data_file);
count := 0;
while not Eof (data_file) do
 begin
  readln (data_file);
  count := count + 1;
 end;
reset (data_file);
for i := 1 to count do
 begin
  read (data_file, data^ [i, laser_wavelength]);
  read (data_file, data^ [i, detached_electron_signal]);
  read (data_file, data^ [i, laser_power]);
  read (data_file, data^ [i, dissociated_radical_signal]);
  readln (data_file, data^ [i, iodine_reference]);
 end;
close (data_file);

for i := 1 to count do
 begin
  normalized_detached_electron_signal :=
       data^ [i, detached_electron_signal];
  if normalized_detached_electron_signal < 1.0 then
       normalized_detached_electron_signal := 1.0;
  normalized_laser_power :=
       data^ [i, laser_power];
  if normalized_laser_power < 1.0 then
       normalized_laser_power := 1.0;
  data^ [i, normalized_signal] :=
       10000 * data^ [i, dissociated_radical_signal]
       / normalized_detached_electron_signal
       / normalized_laser_power;
 end;
if data^ [1, laser_wavelength] < data^ [count, laser_wavelength] then
 begin
  max [laser_wavelength] := data^ [count, laser_wavelength];
  min [laser_wavelength] := data^ [1, laser_wavelength];
  min [laser_wavenumber] := wl2wn (data^ [count, laser_wavelength]);
  max [laser_wavenumber] := wl2wn (data^ [1, laser_wavelength]);
 end
else
 begin
  max [laser_wavelength] := data^ [1, laser_wavelength];
  min [laser_wavelength] := data^ [count, laser_wavelength];
  min [laser_wavenumber] := wl2wn (data^ [1, laser_wavelength]);
  max [laser_wavenumber] := wl2wn (data^ [count, laser_wavelength]);
 end;
for i := 1 to count do
 begin
  for j := 1 to max_column do
   begin
    if data^ [i, j] > max [j] then
```

```
          begin
            max [j] := data^ [i, j];
          end
        else if data^ [i, j] < min [j] then
          begin
            min [j] := data^ [i, j];
          end;
      end;
  end;

  for i := 0 to max_column do
    begin
      plot_max [i]   := max [i] +  0.1 * (max [i] - min [i]);
      plot_min [i]   := min [i] -  0.1 * (max [i] - min [i]);
      thresh_max [i] := max [i] + 0.05 * (max [i] - min [i]);
      thresh_min [i] := min [i] - 0.05 * (max [i] - min [i]);
    end;
  plot_min [laser_power] := 0.0;
  plot_min [detached_electron_signal] := 0.0;
end;

tof :
 begin
  if not get_in_file_name (data_file_name, tof,
     'Enter file name [No extension necessary if .tof] : ') then
    begin
     chdir (active_directory);
     exit;
    end;
  assign (data_file, data_file_name);
  reset (data_file);
  readln (data_file, df_data_count);
  readln (data_file, df_digitizer_control);
  readln (data_file, df_memory_control);
  readln (data_file, df_sweeps_to_average);
  readln (data_file, df_mass_spec_delay);
  for i := 0 to df_data_count do
    begin
     readln (data_file, tof_data^[i]);
    end;
  close (data_file);
  writeln (data_count + 1,' points read');
  tof_data_max := tof_data^ [0];
  tof_data_min := tof_data^ [0];
  for i := 1 to df_data_count do
    begin
     if (tof_data^ [i] > tof_data_max) then
       tof_data_max := tof_data^ [i]
     else if (tof_data^ [i] < tof_data_min) then
       tof_data_min := tof_data^ [i];
    end;
  writeln ('maximum data value = ', tof_data_max :8:0);
  writeln ('minimum data value = ', tof_data_min :8:0);
  writeln ('difference = ',tof_data_max - tof_data_min :8:0);
```

```
      readln;
    end;
  end;
  chdir (active_directory);
end;  { procedure read_data_file }

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


procedure reserved;
begin { procedure reserved }
  writeln ('This key is reserved');
  writeln ('for future development.');
  delay (2000);
end;  { procedure reserved }

end. { of unit utility }
```

2.7 insgrdvr.pas

```
{ insgrdvr.pas                                    }

{ procedure                                       }

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

unit insgrdrv;

{$O+}
{$F+}

interface

uses globals,graph,graphadd,grafutil,CRT,DOS;

procedure GetMagicAddress;
procedure GraphExitProc;
procedure AbortM ( msg:string );
procedure InstallGrafPlusDriver (var driverID : integer;
                                 GraphPort : integer;
                                 GraphDevice : string);
procedure InstallScreenGraphicsDriver;

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

implementation


procedure GetMagicAddress;
{ Find the address needed to use to keep graphics on the screen as they are
```

```
  plotted out on the HP plotter or the PostScript printer              }
begin
   wordptr := ptr(seg(closegraph),8+ofs(closegraph));
   magic := ptr(DSeg,2+wordptr^);
end; { procedure GetMagicAddress }


procedure GraphExitProc;
begin
  ExitProc := OldExitProc; { Restore exit procedure address }
  bCloseGraph;           { Shut down the graphics system }
end; { GraphExitProc }

procedure AbortM ( msg:string );
begin
  WriteLn(msg);
  Halt(1);
end;

procedure InstallGrafPlusDriver (var driverID : integer; GraphPort : integer;
                        GraphDevice : string);

begin { procedure InstallGrafPlusDriver }
   { Install the graphics driver specified by GraphDevice }
   driverID := InstallUserDriver(GraphDevice,nil) + 5;
   ErrorCode := GraphResult;
   if ErrorCode<>0 then
     begin
      AbortM (GraphErrorMsg(ErrorCode));
     end;
end; { procedure InstallGraphPlusDriver }


procedure InstallScreenGraphicsDriver;

begin { procedure InstallScreenGraphicsDriver }
  { Initialize graphics and report any errors that may occur }
  { when using Crt and graphics, turn off Crt's memory-mapped writes }
  {DirectVideo := False;}
  OldExitProc := ExitProc;              { save previous exit proc }
  ExitProc := @GraphExitProc;              { insert custom exit proc in chain }
  GraphDriver := Detect;
  repeat
    bInitGraph(GraphDriver, GraphMode, PathToGraphics, nil);
    ErrorCode := GraphResult;           { preserve error return }
    if ErrorCode <> grOK then           { error? }
    begin
      Writeln('Graphics error: ', GraphErrorMsg(ErrorCode));
      if ErrorCode = grFileNotFound then  { Can't find driver file }
      begin
        Write ('Enter full path to BGI driver or type <Ctrl-Break> to quit : ');
        Readln(PathToGraphics);
        Writeln;
      end
```

```
    else
      Halt(1);                    { Some other error: terminate }
    end;
  until ErrorCode = grOK;

  GetMagicAddress;

  PlotInit;
  RestoreCRTMode;

end; { procedure InstallScreenGraphicsDriver }
end. { unit insgrdvr }
```

2.8 grafutil.pas

```
{ This unit uses a few routines from Turbo Graphics Tools 1.1 in order to
  ensure compatibility between our current programs and the new Turbo 5.5
  Graph unit. Where ever possible, however, the TP unit has been used to
  minimize duplication. }

unit grafutil;

{$O+}
{$F+}

interface

uses graph,graphadd,DOS,CRT;

const
  MaxSymbol   = 9;      {0..any positive integer allowable}
  ULXScreen   = 0;      {screen coordinate system defined here;}
  ULYScreen   = 0;      {may be changed by the user}
  LRXScreen   = 1000;
  LRYScreen   = 1000;

type
  SymbolMask    = Array[0..6] of Byte;

var
  xdpi, ydpi      : word;          {Dots per inch of current graphics output}
  DrawColor       : Integer;
  Symbol          : Array[0..MaxSymbol] of SymbolMask; {Symbol patterns}
  ErrorCode       : Integer;
  XabsGbl,YabsGbl : integer;       {Current graphics cursor location}
  ULXABS,ULYABS,
  LRXABS,LRYABS   : integer;       {Screen corners in pixels}
  MaxX, MaxY      : integer;       {Saves the maximum graphics coordinates}
  ULXWIN,ULYWIN,
  LRXWIN,LRYWIN   : integer;       {Current window corners in pixels}
  ULXSCR,ULYSCR,
```

```
 LRXSCR,LRYSCR   : Real;          {Screen corners in screen coords}
 ULXUSER,ULYUSER,
 LRXUSER,LRYUSER : Real;          {Window corners in user coords}
 XminABS,YminABS,
 XmaxABS,YmaxABS : integer;       {Screen limits in pixels}
 XminWIN,YminWIN,
 XmaxWIN,YmaxWIN : integer;       {Window limits in pixels}

procedure DefineUserCoordinates(ULX,ULY,LRX,LRY: Real);
procedure DefineWindow(ULX,ULY,LRX,LRY: Real);
procedure SCREENtoABS(Xscr,Yscr: Real; var Xabs, Yabs: integer);
procedure USERtoABS(Xuser,Yuser: Real; var Xabs, Yabs: integer);
procedure MoveUser(Xuser,Yuser: Real);
procedure MoveScreen(Xscreen,Yscreen: Real);
procedure DrawLineUser(Xuser,Yuser: Real);
procedure DrawPoint;
procedure DrawBarUser(Xuser,Yuser: Real);
procedure DrawCircleUser(Xuser,Yuser,RadiusUser: Real);
procedure OutTextXYUser(Xuser,Yuser : Real; Text : String);
procedure DefineSymbol(i: Integer; a,b,c,d,e,f,g: Byte);
procedure DrawSymbol(i: Integer);
procedure SymbolInit;
procedure PlotInit;

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

implementation


procedure DefineUserCoordinates(ULX,ULY,LRX,LRY: Real);

{Defines User Coordinates for use in the window}

begin {of procedure DefineUserCoordinates}
ULXUSER := ULX;
ULYUSER := ULY;
LRXUSER := LRX;
LRYUSER := LRY;
end; {of procedure DefineUserCoordinates}


procedure DefineWindow(ULX,ULY,LRX,LRY: Real);

{Input parameters are given in screen coordinates; absolute coordinates
 of new window are calculated and global variables for clipping are set}

var
 XScaleFactor, YScaleFactor : Real;

begin {of procedure DefineWindow}
XScaleFactor := (LRXABS-ULXABS)/(LRXSCR-ULXSCR);   {abs coords of window}
YScaleFactor := (LRYABS-ULYABS)/(LRYSCR-ULYSCR);
ULXWIN := Round((ULX - ULXSCR)*XScaleFactor + ULXABS);
ULYWIN := Round((ULY - ULYSCR)*YScaleFactor + ULYABS);
```

```
LRXWIN := Round((LRX - LRXSCR)*XScaleFactor + LRXABS);
LRYWIN := Round((LRY - LRYSCR)*YScaleFactor + LRYABS);
if ULXWIN < LRXWIN                {window X clipping values}
then begin XminWIN := ULXWIN; XmaxWIN := LRXWIN; end
else begin XminWIN := LRXWIN; XmaxWIN := ULXWIN; end;
if ULYWIN < LRYWIN                {window Y clipping values}
then begin YminWIN := ULYWIN; YmaxWIN := LRYWIN; end
else begin YminWIN := LRYWIN; YmaxWIN := ULYWIN; end;
if XminWIN < XminABS then XminWIN := XminABS;   {if ClippingOn = true,   }
if YminWIN < YminABS then YminWIN := YminABS;   {absolute clipping values }
if XmaxWIN > XmaxABS then XmaxWIN := XmaxABS;   {are not checked; only    }
if YmaxWIN > YmaxABS then YmaxWIN := YmaxABS;   {window values are checked}
end; {of procedure DefineWindow}


procedure SCREENtoABS(Xscr,Yscr: Real; var Xabs, Yabs: integer);

{Converts screen coordinates into absolute coordinates}

var
 XScaleFactor,YScaleFactor,X,Y : Real;

begin {of procedure SCRtoABS}
XScaleFactor := (LRXABS-ULXABS)/(LRXSCR-ULXSCR);
YScaleFactor := (LRYABS-ULYABS)/(LRYSCR-ULYSCR);
X := (Xscr - ULXSCR)*XScaleFactor + ULXABS;
Y := (Yscr - ULYSCR)*YScaleFactor + ULYABS;
if (Abs(X) > 32767.0) or (Abs(Y) > 32767.0) then
 begin
  Errorcode := -11;
  writeln (GraphErrorMsg(ErrorCode));
 end;
Xabs := Round(X);
Yabs := Round(Y);
MoveTo (Xabs,Yabs);
end; {of procedure SCRtoABS}


procedure USERtoABS(Xuser,Yuser: Real; var Xabs, Yabs: integer);

{Converts user coordinates (of window) into absolute coordinates}

var
 XScaleFactor,YScaleFactor,X,Y : Real;

begin {of procedure USERtoABS}
XScaleFactor :=  (LRXWIN-ULXWIN)/(LRXUSER-ULXUSER);
YScaleFactor :=  (LRYWIN-ULYWIN)/(LRYUSER-ULYUSER);
X := (Xuser - ULXUSER)*XScaleFactor + ULXWIN;
Y := (Yuser - ULYUSER)*YScaleFactor + ULYWIN;
if (Abs(X) > 32767.0) or (Abs(Y) > 32767.0) then
begin
  Errorcode := -11;
  writeln (GraphErrorMsg(ErrorCode));
```

```
 end;
Xabs := Round(X);
Yabs := Round(Y);
MoveTo (Xabs,Yabs);
end; {of procedure USERtoABS}


procedure MoveUser(Xuser,Yuser: Real);

{Moves invisible graphics cursor to (Xuser,Yuser) without drawing line}

begin {of procedure Move}
USERtoABS(Xuser,Yuser,XabsGbl,YabsGbl);
end; {of procedure Move}


procedure MoveScreen(Xscreen,Yscreen: Real);

{Moves invisible graphics cursor to (Xscreen,Yscreen) without drawing line}

begin {of procedure MoveScr}
SCREENtoABS(Xscreen,Yscreen,XabsGbl,YabsGbl);
end; {of procedure MoveScr}


procedure DrawLineUser(Xuser,Yuser: Real);

{Draws a line from current invisible graphics cursor to (Xuser,Yuser);
 leaves invisible graphics cursor at (Xuser,Yuser) after drawing line}

var
 X2abs,Y2abs : integer;

begin {of procedure DrawLineUser}
USERtoABS(Xuser,Yuser,X2abs,Y2abs);
Line(XabsGbl,YabsGbl,X2abs,Y2abs);
XabsGbl := X2abs;
YabsGbl := Y2abs;
end; {of procedure DrawLineUser}


procedure DrawPoint;

{Draws a point at current invisible graphics cursor position}

begin {of procedure DrawPoint}
PutPixel(XabsGbl,YabsGbl,DrawColor);
end; {of procedure DrawPoint}


procedure DrawBarUser(Xuser,Yuser: Real);

{Draws a bar with current invisible graphics cursor as one corner and
 the corner diagonal to it given by (Xuser,Yuser); leaves invisible
```

```
 graphics cursor at (Xuser,Yuser) after drawing line}

var
  X2abs,Y2abs : integer;
begin {of procedure DrawBarUser}
USERtoABS(Xuser,Yuser,X2abs,Y2abs);
Bar(XabsGbl,YabsGbl,X2abs,Y2abs);
XabsGbl := X2abs;
YabsGbl := Y2abs;
end;  { procedure DrawBarUser }


procedure DrawCircleUser(Xuser,Yuser,RadiusUser: Real);

{Draws a circle with center given by first two parameters and radius
 given by the third                                          }
var
  Xabs,Yabs,origin,intersection,RadiusAbs : integer;
  RadiusXuser : real;

  pixels : string;

begin {of procedure DrawCircleUser}
USERtoABS(Xuser,Yuser,Xabs,Yabs);
origin := Xabs;
RadiusXuser := Xuser + RadiusUser;
USERtoABS(RadiusXuser,Yuser,Xabs,Yabs);
intersection := Xabs;
RadiusAbs := abs(intersection - origin);

{str (RadiusAbs:1, pixels);
OutTextXY (MaxX div 2, (MaxY div 2 + 50), pixels);

MoveUser (8000,8000);
Bar (XabsGbl,YabsGbl,XabsGbl + 3,YabsGbl + 3);    }

USERtoABS(Xuser,Yuser,Xabs,Yabs);
Circle(Xabs,Yabs,RadiusAbs);
end;  { procedure DrawCircleUser }


procedure OutTextXYUser(Xuser,Yuser : Real; Text : String);
var
  Xabs,Yabs : integer;
begin
  USERtoABS(Xuser,Yuser,Xabs,Yabs);
  OutTextXY(Xabs,Yabs,Text);
end;


procedure DrawSymbol(i: Integer);

{Draws 7x7 symbol[i] centered at invisible graphics cursor position}
```

```
var
 x,y,
 row,
 column  : Integer;
 pattern : Byte;

begin {of procedure DrawSymbol}
if (i<0) or (i>MaxSymbol) then    {check range of Symbol}
begin
   Errorcode := -11;
   writeln (GraphErrorMsg(ErrorCode));
 end;
y := YabsGbl - 3;              {start at top,left of character block}
x := XabsGbl - 3;
for row := 0 to 6 do
 begin
 pattern := Symbol[i,row];
 for column := 0 to 6 do
   if (pattern and (S80 shr column))<>0
   then PutPixel(x+column,y+row,DrawColor);
 end;
end; {of procedure DrawSymbol}


procedure DefineSymbol(i: Integer; a,b,c,d,e,f,g: Byte);

{Defines symbol[i] such that Byte a = top row, Byte b = second row, etc.}

begin {of procedure DefineSymbol}
if (i<0) or (i>MaxSymbol) then          {check range of Symbol}
 begin
   Errorcode := -11;
   writeln (GraphErrorMsg(ErrorCode));
 end;
Symbol[i,0] := a;
Symbol[i,1] := b;
Symbol[i,2] := c;
Symbol[i,3] := d;
Symbol[i,4] := e;
Symbol[i,5] := f;
Symbol[i,6] := g;
end; {of procedure DefineSymbol}


procedure SymbolInit;
begin
DefineSymbol(0,$10,$7C,$7C,$FE,$7C,$7C,$10);    {large point}
DefineSymbol(1,$00,$38,$44,$54,$44,$38,$00);    {dot with circle}
DefineSymbol(2,$00,$00,$10,$38,$10,$00,$00);    {cross}
end;


procedure PlotInit;
var
```

```
  xinch, yinch : word;
  xinchdim, yinchdim : real;
begin { procedure PlotInit }
ULXABS := 0; ULYABS := 0;
if GetDeviceType = ScreenDev then
  begin
    LRXABS := GetMaxX; LRYABS := GetMaxY;
    MaxX := GetMaxX; MaxY := GetMaxY;
  end
else if (GetDeviceType = PrintDev) or (GetDeviceType = PlotDev) then
  begin
    GetDPI (xdpi, ydpi);
    GetPageSize (xinch, yinch);
    xinchdim := xinch / 1000;
    yinchdim := yinch / 1000;
    LRXABS := round(xinchdim * xdpi) - 1;
    LRYABS := round(yinchdim * ydpi) - 1;
    MaxX := LRXABS;
    MaxY := LRYABS;
  end;
XabsGbl := ULXABS;              {Invisible graphics cursor starts}
YabsGbl := ULYABS;              {in upper left corner}
ULXSCR := ULXScreen;            {Screen coordinate system defined here;}
ULYSCR := ULYScreen;            {these may be changed by the user in}
LRXSCR := LRXScreen;            {the const declaration section above}
LRYSCR := LRYScreen;
DefineWindow(ULXSCR,ULYSCR,LRXSCR,LRYSCR);   {Default window is entire screen}
end; {of procedure PlotInit}

end. {of unit grafutil}
```

2.9 txsplot.pas

```
{ txsplot.pas                                    }

{ contents :                                     }

{ i. useful (repetitive)                           }

{ procedure txs_draw_ticks (tick_axis : integer;                  }
{                 constant_limit : real;                 }
{                 code : integer;                      }
{                 label_ticks : boolean)                 }

{ ii. while experiment is running                          }

{ procedure txs_label_axes (axis : integer)                    }

{ procedure txs_draw_axes                                  }

{ procedure txs_x_y_ticks (y_data : integer)                       }
```

```
{ procedure txs_replot (y_data, point_count : integer; visible : boolean)   }

{ procedure txs_plot_point (count, y_data : integer)                        }

{ procedure set_done_flag                                                   }

{ iii. after experiment                                                     }

{ procedure p_txs_draw_axes (x_axis, y_axis : integer)                      }

{ procedure p_txs_plot_points (x_axis, y_axis : integer)                    }

{ procedure j_hard_copy (x_axis, y_axis : integer)                          }

{ function set_txs_axis (var axis : integer) : boolean                      }

{ procedure dump_experiment_screen                                          }

{ procedure show_txs_plot_parameters                                        }

{ procedure plot_menu                                                       }

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

unit txsplot;

{$O+}
{$F+}

interface

uses globals,utility,insgrdrv,plotutil,graph,graphadd,grafutil,CRT,DOS;

const
  Font : Integer = DefaultFont;

procedure txs_draw_ticks (tick_axis : integer; constant_limit : real;
                          code : integer; label_ticks : boolean);
procedure txs_label_axes (axis : integer);
procedure txs_draw_axes;
procedure txs_x_y_ticks (y_data : integer);
procedure txs_replot (y_data, point_count : integer; visible : boolean);
procedure txs_plot_point (count, y_data : integer);
procedure set_done_flag;
procedure p_txs_draw_axes (x_axis, y_axis : integer);
procedure p_txs_plot_points (x_axis, y_axis : integer);
function set_txs_axis (var axis : integer) : boolean;
procedure dump_txs_experiment_screen;
procedure show_txs_plot_parameters;
procedure txs_plotting_parameters;


{ ************************************************************************** }
```

```
implementation

procedure txs_draw_ticks (tick_axis : integer;
                          constant_limit : real;
                          code : integer;
                          label_ticks : boolean);
var
 DeviceType, xdotnum, ydotnum : integer;
 tick_mark, tick_step : real;
 tick_label : string[8];
 decimal_places : integer;
begin { procedure txs_draw_ticks }

 { find tick length for graphics in absolute number of dots }
 DeviceType := GetDeviceType;
 if DeviceType = ScreenDev then
  begin
   xdotnum := 4;
   ydotnum := 4;
  end
 else if DeviceType > 0 then
  begin
   xdotnum := xdpi div 12;
   ydotnum := ydpi div 12;
  end;

 { find tick step size }
 if (abs (plot_max [tick_axis] - plot_min [tick_axis]) >= 1.0) then
  tick_step := exp (ln (10.0) * int
   (ln (abs (plot_max [tick_axis] - plot_min [tick_axis])) / ln (10.0)))
 else
  tick_step := exp (ln (10.0) * int
   (-1.0 + ln (abs (plot_max [tick_axis] - plot_min [tick_axis]))
    / ln (10.0)));
 tick_mark := int (plot_min [tick_axis] / tick_step) * tick_step;
 if (tick_mark < plot_min [tick_axis]) then
  tick_mark := tick_mark + tick_step;
 decimal_places := trunc (-ln (tick_step) / ln (10.0));
 if ((tick_mark + (2.0 * tick_step)) >= plot_max [tick_axis]) then
  begin
   tick_step := tick_step / 2.0;
   decimal_places := decimal_places + 1;
   tick_mark := int (plot_min [tick_axis] / tick_step) * tick_step;
   if (tick_mark < plot_min [tick_axis]) then
    tick_mark := tick_mark + tick_step;
  end
 else if ((tick_mark + (5.0 * tick_step)) < plot_max [tick_axis]) then
  begin
   tick_step := tick_step * 2.0;
   tick_mark := int (plot_min [tick_axis] / tick_step) * tick_step;
   if (tick_mark < plot_min [tick_axis]) then
    tick_mark := tick_mark + tick_step;
  end;
```

```
if (decimal_places < 0) then
 decimal_places := 0;

{ draw tick marks }
while (tick_mark < plot_max [tick_axis]) do
 begin
  case code of
   0 : begin { bottom }
       MoveUser (tick_mark, constant_limit);
       Line (Xabsgbl, (Yabsgbl - 1), Xabsgbl, (Yabsgbl - ydotnum));
       if label_ticks then
         begin
          str (tick_mark : (decimal_places + 1) : (decimal_places + 1),
              tick_label);
          MoveUser (tick_mark, constant_limit);
          MoveRel (0, (TextHeight(tick_label) div 2));
          bSetTextStyle (Font, HorizDir, 1);
          SetTextJustify (CenterText, TopText);
          OutText (tick_label);
         end;
      end;
   1 : begin { left }
       MoveUser (constant_limit, tick_mark);
       Line ((Xabsgbl + 1), Yabsgbl, (Xabsgbl + xdotnum), Yabsgbl);
       if label_ticks then
         begin
          str (tick_mark : (decimal_places + 1) : decimal_places,
              tick_label);
          MoveUser (constant_limit, tick_mark);
          bSetTextStyle (Font, HorizDir, 1);
          SetTextJustify (RightText, CenterText);
          OutText (tick_label);
         end;
      end;

   2 : begin { top }
       MoveUser (tick_mark, constant_limit);
       Line (Xabsgbl, (Yabsgbl + 1), Xabsgbl, (Yabsgbl + ydotnum));
       if label_ticks then
         begin
          str (tick_mark : (decimal_places + 1) : (decimal_places + 1),
              tick_label);
          MoveUser (tick_mark, constant_limit);
          bSetTextStyle (Font, HorizDir, 1);
          SetTextJustify (CenterText, BottomText);
          OutText (tick_label);
         end;
      end;

   3 : begin { right }
       MoveUser (constant_limit, tick_mark);
       Line ((Xabsgbl - 1), Yabsgbl, (Xabsgbl - xdotnum), Yabsgbl);
       if label_ticks then
         begin
```

```
            str (tick_mark : (decimal_places + 1) : decimal_places,
                tick_label);
            MoveUser (constant_limit, tick_mark);
            MoveRel (2, 0);
            bSetTextStyle (Font, HorizDir, 1);
            SetTextJustify (LeftText, CenterText);
            OutText (tick_label);
           end;
         end;
     end; { case code }
     tick_mark := tick_mark + tick_step;
   end;
end;  { procedure draw_ticks }

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

procedure txs_label_axes (axis : integer);
begin
 case axis of

   laser_wavelength :
      begin { laser_wavelength, x axis }
       SetColor (Cyan);
       bSetTextStyle (Font, HorizDir, 1);
       SetTextJustify (CenterText, BottomText);
       OutTextXY ((GetMaxX div 2), GetMaxY,
             axis_label [laser_wavelength]);
      end;

   detached_electron_signal :
      begin
       SetColor (Green);
       bSetTextStyle (Font, VertDir, 1);
       SetTextJustify (LeftText,CenterText);
       OutTextXY (TextHeight(axis_label [detached_electron_signal]),
             (GetMaxY * 3 div 4),
             axis_label [detached_electron_signal]);
      end;

   laser_power :
      begin
       SetColor (Red);
       bSetTextStyle (Font, VertDir, 1);
       SetTextJustify (RightText,CenterText);
       OutTextXY (GetMaxX,(GetMaxY * 3 div 4),
             axis_label [laser_power]);
      end;

   dissociated_radical_signal :
      begin
       SetColor (Blue);
       bSetTextStyle (Font, VertDir, 1);
       SetTextJustify (LeftText,CenterText);
       OutTextXY (TextHeight(axis_label [dissociated_radical_signal]),
```

```
            (GetMaxY div 4),
            axis_label [dissociated_radical_signal]);
      end;

    normalized_signal :
      begin
       SetColor (LightGray);
       bSetTextStyle (Font,  VertDir, 1);
       SetTextJustify (RightText,CenterText);
       OutTextXY (GetMaxX,(GetMaxY div 4),
            axis_label [normalized_signal]);
      end;

    end;  { case axis }
end;  { procedure txs_label_axes }

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

procedure txs_draw_axes;
begin { procedure txs_draw_axes }
 { draw box }
 DefineWindow (80, 60, 920, 940);
 DefineUserCoordinates (plot_min [laser_wavelength], 1.0,
              plot_max [laser_wavelength], -1.0);
 MoveUser (plot_min [laser_wavelength], -1.0);
 SetColor (Green);
 DrawLineUser (plot_min [laser_wavelength], 0.0);
 SetColor (Blue);
 DrawLineUser (plot_min [laser_wavelength], 1.0);
 SetColor (Cyan);
 DrawLineUser (plot_max [laser_wavelength], 1.0);
 SetColor (LightGray);
 DrawLineUser (plot_max [laser_wavelength], 0.0);
 SetColor (Red);
 DrawLineUser (plot_max [laser_wavelength], -1.0);
 SetColor (Cyan);
 DrawLineUser (plot_min [laser_wavelength],-1.0);
 MoveUser (plot_min [laser_wavelength], 0.0);
 DrawLineUser (plot_max [laser_wavelength], 0.0);
 txs_label_axes (laser_wavelength);
 txs_label_axes (detached_electron_signal);
 txs_label_axes (laser_power);
 txs_label_axes (dissociated_radical_signal);
 txs_label_axes (normalized_signal);

 SetColor (Cyan);
 txs_draw_ticks (laser_wavelength, 1.0, 2, TRUE);
 txs_draw_ticks (laser_wavelength, 0.0, 0, FALSE);
 txs_draw_ticks (laser_wavelength, 0.0, 2, FALSE);
 txs_draw_ticks (laser_wavelength, -1.0, 0, TRUE);

 SetColor (White);
 bSetTextStyle (Font, HorizDir, 1);
 SetTextJustify (CenterText, TopText);
```

```
 OutTextXY (GetMaxX div 2, 0,
       'Total Photodissociation Cross Section Experiment');

end;  { procedure txs_draw_axes }

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

procedure txs_x_y_ticks (y_data : integer);
begin { procedure txs_x_y_ticks }
 { y ticks }
 if (y_data = dissociated_radical_signal) then
     txs_draw_ticks (dissociated_radical_signal,
          plot_min [laser_wavelength], 1, TRUE)
 else if (y_data = normalized_signal) then
     txs_draw_ticks (normalized_signal,
          plot_max [laser_wavelength], 3, TRUE)
 else if (y_data = detached_electron_signal) then
     txs_draw_ticks (detached_electron_signal,
          plot_min [laser_wavelength], 1, TRUE)
 else if (y_data = laser_power) then
     txs_draw_ticks (laser_power,
          plot_max [laser_wavelength], 3, TRUE);
end;  { procedure txs_x_y_ticks }

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

procedure txs_replot (y_data, point_count : integer; visible : boolean);
var
 i : integer;
begin { procedure txs_replot }
 if (y_data = dissociated_radical_signal) or (y_data = normalized_signal) then
  DefineWindow (80, 60, 920, 500)
 else
  DefineWindow (80, 500, 920, 940);
 DefineUserCoordinates (plot_min [laser_wavelength],
                plot_max [y_data],
                plot_max [laser_wavelength],
                plot_min [y_data]);
 if visible then
  begin
   txs_label_axes (y_data);
   if (y_data = detached_electron_signal) then DrawColor := Green
   else if (y_data = laser_power) then DrawColor := Red
   else if (y_data = dissociated_radical_signal) then DrawColor := Blue
   else if (y_data = normalized_signal) then
    begin
     DrawColor := LightGray;
     SetColor (LightGray);
    end;
  end
 else
  begin
   DrawColor := Black;
   SetColor (Black);
```

```
    end;
  txs_x_y_ticks (y_data);
  if GetDeviceType = 0 then
    begin
     for i := 1 to point_count do
      begin
       MoveUser (data^ [i, laser_wavelength], data^ [i, y_data]);
       if plot_little_crosses then
        DrawSymbol (2)
       else
        DrawPoint;
       end;
    end;
  if ((y_data = normalized_signal) and (lines = TRUE)) or (GetDeviceType > 0)
     then
      begin
       MoveUser (data^ [1, laser_wavelength], data^ [1, y_data]);
       for i := 2 to point_count do
        DrawLineUser (data^ [i, laser_wavelength], data^ [i, y_data]);
      end;
end;  { procedure txs_replot }

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

procedure txs_plot_point (count, y_data : integer);
begin
  if ((count = 1) and (y_data = dissociated_radical_signal) or
     (count = 1) and (y_data = normalized_signal)) then
    begin
     min [y_data] :=
        data^ [count, y_data] - 1.0;
     max [y_data] :=
        data^ [count, y_data] + 1.0;

     plot_max [y_data] :=
           max [y_data] +
           0.1 * (max [y_data] -
           min [y_data]);

     plot_min [y_data] :=
           min [y_data] -
           0.1 * (max [y_data] -
           min [y_data]);

     thresh_max [y_data] :=
            max [y_data] +
            0.05 * (max [y_data] -
            min [y_data]);

     thresh_min [y_data] :=
            min [y_data] -
            0.05 * (max [y_data] -
            min [y_data]);
```

```
    DefineWindow (80, 60, 920, 500);
    DefineUserCoordinates (plot_min [laser_wavelength],
                    plot_max [y_data],
                    plot_max [laser_wavelength],
                    plot_min [y_data]);
    if y_data = dissociated_radical_signal then
     begin
      SetColor (Blue);
      DrawColor := Blue;
     end
    else
     begin
      SetColor (LightGray);
      DrawColor := LightGray;
     end;
    txs_x_y_ticks (y_data);
    MoveUser (data^ [count, laser_wavelength],
          data^ [count, y_data]);
    if plot_little_crosses then
     DrawSymbol (2)
    else
     DrawPoint;
  end;
if (data^ [count, y_data] > thresh_max [y_data]) then
  begin
   txs_replot (y_data, (count - 1), FALSE);
   max [y_data] := data^ [count, y_data];
   plot_max [y_data] := max [y_data] +
                (0.1 * (max [y_data] - min [y_data]));
   thresh_max [y_data] := max [y_data] +
                (0.05 * (max [y_data] - min [y_data]));
   txs_replot (y_data, count, TRUE);
  end
else if (data^ [count, y_data] < thresh_min [y_data]) then
  begin
   txs_replot (y_data, (count - 1), FALSE);
   min [y_data] := data^ [count, y_data];
   plot_min [y_data] := min [y_data] -
                (0.1 * (max [y_data] - min [y_data]));
   thresh_min [y_data] := min [y_data] -
                (0.05 * (max [y_data] - min [y_data]));
   txs_replot (y_data, count, TRUE);
  end
else
  begin
   if (y_data = dissociated_radical_signal)
         or (y_data = normalized_signal) then
    DefineWindow (80, 60, 920, 500)
   else
    DefineWindow (80, 500, 920, 940);
   DefineUserCoordinates (plot_min [laser_wavelength],
                    plot_max [y_data],
                    plot_max [laser_wavelength],
                    plot_min [y_data]);
```

```
    if (y_data = detached_electron_signal) then DrawColor := Green
    else if (y_data = laser_power) then DrawColor := Red
    else if (y_data = dissociated_radical_signal) then DrawColor := Blue
    else if (y_data = normalized_signal) then DrawColor := LightGray;
    MoveUser (data^ [count, laser_wavelength], data^ [count, y_data]);
    if plot_little_crosses then
     DrawSymbol (2)
    else
     DrawPoint;
    if (y_data = normalized_signal) and (count > 1) and (lines = TRUE) then
     begin
      SetColor (LightGray);
      MoveUser (data^ [count - 1, x_axis], data^ [count - 1, y_data]);
      DrawLineUser (data^ [count, x_axis], data^ [count, y_data]);
     end;
   end;
end;  { procedure txs_plot_point }

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

procedure set_done_flag;
var
 i : integer;
begin { procedure set_done_flag }
 bSetTextStyle (Font, HorizDir, 1);
 SetTextJustify (CenterText,TopText);
 SetColor (Yellow);
 OutTextXY ((GetMaxX div 2), (GetMaxY * 4 div 5),
        'DONE!  Hit any key to continue ...');
 For i := 1 to 8 do
  begin
   Sound (1000);
   Delay (100);
   NoSound;
   Delay (100);
  end;
 Repeat
  SetFillStyle (EmptyFill, EmptyFill);
  Bar ((GetMaxX div 2 - (TextWidth ('DONE!  Hit any key to continue ...') div 2)),
    (GetMaxY * 4 div 5),
    (GetMaxX div 2 + (TextWidth ('DONE!  Hit any key to continue ...') div 2)),
    (GetMaxY * 4 div 5 + TextHeight('D')));
  Delay (200);
  SetColor (Yellow);
  OutTextXY ((GetMaxX div 2), (GetMaxY * 4 div 5),
        'DONE!  Hit any key to continue ...');
  Delay (200);
 until KeyPressed;
end;  { procedure set_done_flag }

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

procedure p_txs_draw_axes (x_axis, y_axis : integer);
```

```
begin
  DefineWindow (100, 100, 1000, 900);
  DefineUserCoordinates (plot_min [x_axis], plot_max [y_axis],
                plot_max [x_axis], plot_min [y_axis]);

  SetColor (White);

  { draw box }
  MoveUser (plot_min [x_axis], plot_min [y_axis]);
  DrawLineUser (plot_min [x_axis], plot_max [y_axis]);
  DrawLineUser (plot_max [x_axis], plot_max [y_axis]);
  DrawLineUser (plot_max [x_axis], plot_min [y_axis]);
  DrawLineUser (plot_min [x_axis], plot_min [y_axis]);

  { label y axis }
  bSetTextStyle (Font, VertDir, 1);
  SetTextJustify (LeftText, CenterText);
  OutTextXY (TextHeight(axis_label [y_axis]), (GetMaxY div 2),
        axis_label [y_axis]);

  { label x axis }
  bSetTextStyle (Font, HorizDir, 1);
  SetTextJustify (CenterText, BottomText);
  OutTextXY ((GetMaxX div 2 + GetMaxX div 20), GetMaxY,
        axis_label [x_axis]);

  SetColor (White);
  txs_draw_ticks (x_axis, plot_min [y_axis], 0, TRUE);
  txs_draw_ticks (x_axis, plot_max [y_axis], 2, FALSE);
  txs_draw_ticks (y_axis, plot_min [x_axis], 1, TRUE);
  txs_draw_ticks (y_axis, plot_max [x_axis], 3, FALSE);

  { label title }
  bSetTextStyle (Font, HorizDir, 1);
  SetTextJustify (CenterText, TopText);
  OutTextXY (GetMaxX div 2 + GetMaxX div 20, 0,
        'Total Photodissociation Cross Section Experiment');
  OutTextXY (GetMaxX div 2 + GetMaxX div 20, (TextHeight ('T') * 3),
        (axis_label [y_axis] + ' vs. ' + axis_label [x_axis]));

  { label file name }
  SetColor (Red);
  bSetTextStyle (Font, HorizDir, 1);
  SetTextJustify (LeftText, BottomText);
  OutTextXY (0, GetMaxY, data_file_name);
end;  { procedure p_txs_draw_axes }

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

procedure p_txs_plot_points (x_axis, y_axis : integer);
var
  plot_little_crosses : boolean;
  i : integer;
begin { procedure p_txs_plot_points }
```

```
  plot_little_crosses := (count < 500);
  DrawColor := Red;
  if plot_lines then
    begin
      SetColor (Red);
      MoveUser (data^ [1, x_axis], data^ [1, y_axis]);
      for i := 2 to count do
        DrawLineUser (data^ [i, x_axis], data^ [i, y_axis]);
    end
  else
    begin
      for i := 1 to count do
        begin
          MoveUser (data^ [i, x_axis], data^ [i, y_axis]);
          if plot_little_crosses then
            DrawSymbol (2)
          else
            DrawPoint;
        end;
    end;
end;  { procedure p_txs_plot_points }

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

function set_txs_axis (var axis : integer) : boolean;
var
  i, old_axis : integer;
  ch : char;
begin { function set_txs_axis }
  old_axis := axis;
  if not GraphicsOn then
    begin
      for i := 0 to max_column do
        writeln (i:1, ' : ', axis_label [i]);
      writeln; write ('? > ');
    end;
  ch := Readkey;  ch := upcase(ch);
  if not GraphicsOn then
    writeln (ch);
  if ((ord(ch) - ord('0')) in [0..max_column]) then
    begin
      axis := (ord(ch) - ord('0'));
      if (axis = old_axis) then
        set_txs_axis := FALSE
      else
        set_txs_axis := TRUE;
    end
  else
    set_txs_axis := FALSE;
end;  { function set_txs_axis }

procedure draw_txs_dump_screen;
begin
  { draw axes }
```

```
  txs_draw_axes;

  { plot data }
  SetLineStyle (3,0,1);
  txs_replot (laser_power, count, TRUE);
  SetLineStyle (2,0,1);
  txs_replot (dissociated_radical_signal, count, TRUE);
  SetLineStyle (1,0,1);
  txs_replot (detached_electron_signal, count, TRUE);
  SetLineStyle (0,0,1);
  txs_replot (normalized_signal, count, TRUE);

  { label file name }
  {SetColor (Red);}
  bSetTextStyle (Font, HorizDir, 1);
  SetTextJustify (LeftText, BottomText);
  OutTextXY (0, GetMaxY, data_file_name);
end;

procedure dump_txs_experiment_screen;
begin
  {show_screen_dump_parameters;
  screen_dump_menu;}

  { Put on screen what will go to output device }
  SetGraphMode(GraphMode);
  GraphicsOn := TRUE;
  draw_txs_dump_screen;

  { Now output to PS LaserWriter }

  prevent_clear_screen;

  erase_old_portfile;

  enter_laserwriter_graphics_mode;

  { Draw Plot of Data }
  draw_txs_dump_screen;

  plot_graph_and_exit;

  lpr_plot_to_laserwriter;

  InstallScreenGraphicsDriver;
  TextColor (14);
end;

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

procedure put_plot_to_screen;

{ called just before putting plot to graphics device }
```

```
begin
 if not GraphicsOn then
  begin
   SetGraphMode(GraphMode);
   GraphicsOn := TRUE;
   p_txs_draw_axes (x_axis, y_axis);
   p_txs_plot_points (x_axis, y_axis);
  end;
end;

procedure show_txs_plot_parameters;
var
 x, y : integer;
begin { procedure show_txs_plot_parameters }
 x := wherex; y := wherey;
 window (37, 1, 80, 24);
 clrscr; textcolor (9);
 GoToXY (1,5);
 writeln ('Plot Parameters Status');
 writeln ('----------------------');
 writeln;
 writeln ('X axis : ', axis_label [x_axis]);
 writeln ('Y axis : ', axis_label [y_axis]);
 if plot_lines then
  writeln ('Plot with Lines')
 else
  writeln ('Plot with Dots');
 writeln;
 if plot_from_zero then
  writeln ('Scale Plot from Zero')
 else
  writeln ('Scale Plot to Fill Screen');
end;  { procedure show_txs_plot_parameters }

procedure txs_plotting_parameters;
var
 ch : char;
 axis, upper_y : integer;
 procedure show_txs_plot_menu;
 begin
  if not GraphicsOn then
   begin
    window (1, 1, 36, 24);
    clrscr; textcolor (2);
    writeln ('   Plot Menu');
    writeln ('----------------');
    writeln;
    writeln ('S : Plot on Screen');
    writeln ('P : Plot on LaserWriter');
    writeln ('H : Plot on HP Plotter');
    writeln;
    writeln ('X : X axis');
    writeln ('Y : Y axis');
    writeln ('E : Experiment Screen');
```

```
      writeln ('D : Dots');
      writeln ('L : Lines');
      writeln ('Z : Zero (toggle)'); writeln;
      writeln ('Q : Quit Plot Menu'); writeln;
      upper_y := wherey;
    end;
  end;

begin { procedure txs_plotting_parameters }
  repeat
    if not GraphicsOn then
      begin
        show_txs_plot_menu;
        show_txs_plot_parameters;
        window (1, upper_y, 80, 24);  clrscr;
        textcolor (14);
        write ('? > ');
      end;
    ch := Readkey;  ch := upcase (ch);  writeln (ch);  writeln;
    if not GraphicsOn then
      begin
        writeln (ch);
      end;
    case ch of

    'S' : begin
            put_plot_to_screen;
          end;

    'H' : begin
            { Plot it to screen first }
            put_plot_to_screen;
            prevent_clear_screen;
            enter_hp7470plotter_graphics_mode;

            { Draw Plot of Data }
            p_txs_draw_axes (x_axis, y_axis);
            p_txs_plot_points (x_axis, y_axis);

            plot_graph_and_exit;

            { The screen is still in graphics mode.  This puts it back in
            text mode.  It is then neccessary to reload the Screen
            Graphics driver and replace the menu on the screen.          }

            textmode(LastMode);
            magic^ := hold;

            InstallScreenGraphicsDriver;
            TextColor (14 + Blink);
            GoToXY (1,12);
            write ('Graphics plotting completed.  ',
                   'Press <enter> to continue ...');
            readln;
```

```
        end;

'P' : begin
      put_plot_to_screen;
      prevent_clear_screen;
      erase_old_portfile;

      enter_laserwriter_graphics_mode;

      { Draw Plot of Data }
      p_txs_draw_axes (x_axis, y_axis);
      p_txs_plot_points (x_axis, y_axis);

      plot_graph_and_exit;

      lpr_plot_to_laserwriter;

      InstallScreenGraphicsDriver;
        end;

'X' : begin
      if set_txs_axis (x_axis) then
       if GraphicsOn then
        begin
         ClearDevice;
         p_txs_draw_axes (x_axis, y_axis);
         p_txs_plot_points (x_axis, y_axis);
        end
       else
        show_txs_plot_parameters;
    end;

'Y' : begin
      if set_txs_axis (y_axis) then
       if GraphicsOn then
        begin
         ClearDevice;
         p_txs_draw_axes (x_axis, y_axis);
         p_txs_plot_points (x_axis, y_axis);
        end
       else
        show_txs_plot_parameters;
    end;
'E' : begin
      if not GraphicsOn then
       begin
        SetGraphMode(GraphMode);
        GraphicsOn := TRUE;
       end;
      draw_txs_dump_screen;
      {if ask_for_boolean ('Output screen display to LaserWriter? ')
       then
       begin
        dump_txs_experiment_screen;
```

```
      end;}
    end;
'D' : begin
      if (GraphicsOn and plot_lines) then
       begin
        plot_lines := FALSE;
        ClearDevice;
        p_txs_draw_axes (x_axis, y_axis);
        p_txs_plot_points (x_axis, y_axis);
       end
      else
       begin
        plot_lines := FALSE;
        show_txs_plot_parameters;
       end;
    end;

'L' : begin
      if (GraphicsOn and not plot_lines) then
       begin
        plot_lines := TRUE;
        p_txs_plot_points (x_axis, y_axis);
       end
      else
       begin
        plot_lines := TRUE;
        show_txs_plot_parameters;
       end;
    end;

'Z' : begin
      if plot_from_zero then
       for axis := 1 to max_column do
        begin
         plot_min [axis] := min [axis] - (0.1 *
                        (max [axis] - min [axis]));
         plot_max [axis] := max [axis] + (0.1 *
                        (max [axis] - min [axis]));
        end
      else
       for axis := 1 to max_column do
        begin
         plot_min [axis] := -(0.1 * max [axis]);
         plot_max [axis] := max [axis] + (0.1 * max [axis]);
        end;
      plot_from_zero := not plot_from_zero;
      if GraphicsOn then
       begin
        ClearDevice;
        p_txs_draw_axes (x_axis, y_axis);
        p_txs_plot_points (x_axis, y_axis);
       end
      else
       show_txs_plot_parameters;
```

```
        end;

    #13 : begin
          if GraphicsOn then
            begin
              RestoreCRTMode;
              GraphicsOn := FALSE;
            end;
          show_txs_plot_parameters;
        end;

    'Q' : begin
          if GraphicsOn then
            begin
              RestoreCRTMode;
              GraphicsOn := FALSE;
            end;
        end;
  end; { case ch }
 until (ch = 'Q');
end;  { procedure txs_plotting_parameters }


end. { unit txsplot }
```

2.10 tofplot.pas

```
{ tofplot.pas                                                    }

{ contents :                                                   }

{ i. useful (repetitive)                                         }

{ procedure draw_ticks (tick_axis : integer;                          }
{                       constant_limit : real;                     }
{                       code : integer;                          }
{                       label_ticks : boolean)                     }

{ ii. while experiment is running                                  }

{ procedure label_axes (axis : integer)                            }

{ procedure x_draw_axes                                          }

{ procedure x_y_ticks (y_data : integer)                           }

{ procedure x_replot (y_data, point_count : integer; visible : boolean)   }

{ procedure x_plot_point (count, y_data : integer)                      }

{ procedure set_done_flag                                        }
```

```
{ iii. after experiment                                    }

{ procedure p_draw_axes (x_axis, y_axis : integer)          }

{ procedure p_plot_points (x_axis, y_axis : integer)        }

{ procedure j_hard_copy (x_axis, y_axis : integer)          }

{ function set_axis (var axis : integer) : boolean          }

{ procedure dump_experiment_screen                          }

{ procedure show_plot_parameters                            }

{ procedure plot_menu                                       }

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

unit tofplot;

{$O+}
{$F+}

interface

uses globals,utility,insgrdrv,graph,plotutil,graphadd,grafutil,CRT,DOS;

const
  Font : Integer = DefaultFont;

procedure tof_draw_ticks (tick_axis : integer;
                          constant_limit : real;
                          code : integer;
                          label_ticks : boolean);
procedure tof_draw_axes (x_axis, y_axis : integer);
procedure dump_tof_experiment_screen (x_axis, y_axis : integer;
                                      data_count : integer);

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

implementation


procedure tof_draw_ticks (tick_axis : integer;
                          constant_limit : real;
                          code : integer;
                          label_ticks : boolean);
var
  xdotnum, ydotnum : integer;
  tick_mark, tick_step : real;
  tick_label : string[8];
  decimal_places : integer;
begin { procedure tof_draw_ticks }
```

```
xdotnum := 4;
ydotnum := 4;

{ find tick step size }
if (abs (tof_plot_max [tick_axis] - tof_plot_min [tick_axis]) >= 1.0)
 then
  tick_step := exp (ln (10.0) *
           int (ln (abs (tof_plot_max [tick_axis]
           - tof_plot_min [tick_axis])) / ln (10.0)))
else
 tick_step := exp (ln (10.0) *
          int (-1.0 + ln (abs (tof_plot_max [tick_axis]
          - tof_plot_min [tick_axis])) / ln (10.0)));
tick_mark := int (tof_plot_min [tick_axis] / tick_step) * tick_step;
if (tick_mark < tof_plot_min [tick_axis]) then
 tick_mark := tick_mark + tick_step;
decimal_places := trunc (-ln (tick_step) / ln (10.0));
if ((tick_mark + (2.0 * tick_step)) >= tof_plot_max [tick_axis]) then
 begin
  tick_step := tick_step / 2.0;
  decimal_places := decimal_places + 1;
  tick_mark := int (tof_plot_min [tick_axis] / tick_step) * tick_step;
  if (tick_mark < tof_plot_min [tick_axis]) then
   tick_mark := tick_mark + tick_step;
 end
else if ((tick_mark + (5.0 * tick_step)) < tof_plot_max [tick_axis]) then
 begin
  tick_step := tick_step * 2.0;
  tick_mark := int (tof_plot_min [tick_axis] / tick_step) * tick_step;
  if (tick_mark < tof_plot_min [tick_axis]) then
   tick_mark := tick_mark + tick_step;
 end;
if (decimal_places < 0) then
 decimal_places := 0;

{ draw tick marks }
while (tick_mark < tof_plot_max [tick_axis]) do
 begin
  case code of
   0 : begin { bottom }
       MoveUser (tick_mark, constant_limit);
       Line (Xabsgbl, (Yabsgbl - 1), Xabsgbl, (Yabsgbl - ydotnum));
       if label_ticks then
         begin
          str (tick_mark : (decimal_places + 1) : (decimal_places + 1),
              tick_label);
          MoveUser (tick_mark, constant_limit);
          MoveRel (0, (TextHeight(tick_label) div 2));
          bSetTextStyle (Font, HorizDir, 1);
          SetTextJustify (CenterText, TopText);
          OutText (tick_label);
         end;
      end;
```

```
      1 : begin { left }
          MoveUser (constant_limit, tick_mark);
          Line ((Xabsgbl + 1), Yabsgbl, (Xabsgbl + xdotnum), Yabsgbl);
          if label_ticks then
            begin
             str (tick_mark : (decimal_places + 1) : decimal_places,
                tick_label);
             MoveUser (constant_limit, tick_mark);
             bSetTextStyle (Font, HorizDir, 1);
             SetTextJustify (RightText, CenterText);
             OutText (tick_label);
            end;
        end;

      2 : begin { top }
          MoveUser (tick_mark, constant_limit);
          Line (Xabsgbl, (Yabsgbl + 1), Xabsgbl, (Yabsgbl + ydotnum));
          if label_ticks then
            begin
             str (tick_mark : (decimal_places + 1) : (decimal_places + 1),
                tick_label);
             MoveUser (tick_mark, constant_limit);
             bSetTextStyle (Font, HorizDir, 1);
             SetTextJustify (CenterText, BottomText);
             OutText (tick_label);
            end;
        end;

      3 : begin { right }
          MoveUser (constant_limit, tick_mark);
          Line ((Xabsgbl - 1), Yabsgbl, (Xabsgbl - xdotnum), Yabsgbl);
          if label_ticks then
            begin
             str (tick_mark : (decimal_places + 1) : decimal_places,
                tick_label);
             MoveUser (constant_limit, tick_mark);
             MoveRel (2, 0);
             bSetTextStyle (Font, HorizDir, 1);
             SetTextJustify (LeftText, CenterText);
             OutText (tick_label);
            end;
        end;
    end; { case code }
    tick_mark := tick_mark + tick_step;
  end;

end;  { procedure tof_draw_ticks }


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


procedure tof_draw_axes (x_axis, y_axis : integer);
begin
```

```
  DefineWindow (100, 100, 1000, 900);
  DefineUserCoordinates (tof_plot_min [time], tof_plot_max [tof_signal],
                tof_plot_max [time], tof_plot_min [tof_signal]);

  SetColor (LightGray);

  MoveUser (tof_plot_min [time], tof_plot_min [tof_signal]);
  DrawLineUser (tof_plot_min [time], tof_plot_max [tof_signal]);
  DrawLineUser (tof_plot_max [time], tof_plot_max [tof_signal]);
  DrawLineUser (tof_plot_max [time], tof_plot_min [tof_signal]);
  DrawLineUser (tof_plot_min [time], tof_plot_min [tof_signal]);

  { label y axis }
  bSetTextStyle (Font, VertDir, 1);
  SetTextJustify (LeftText, CenterText);
  OutTextXY (TextHeight(tof_axis_label [y_axis]), (GetMaxY div 2),
          tof_axis_label [y_axis]);

  { label x axis }
  bSetTextStyle (Font, HorizDir, 1);
  SetTextJustify (CenterText, BottomText);
  OutTextXY ((GetMaxX div 2 + GetMaxX div 20), GetMaxY,
          tof_axis_label [x_axis]);

  tof_draw_ticks (time, tof_plot_min [tof_signal], 0, TRUE);
  tof_draw_ticks (time, tof_plot_max [tof_signal], 2, FALSE);
  tof_draw_ticks (tof_signal, tof_plot_max [time], 3, FALSE);
  tof_draw_ticks (tof_signal, tof_plot_min [time], 1, TRUE);

  { label title }
  bSetTextStyle (Font, HorizDir, 1);
  SetTextJustify (CenterText, TopText);
  OutTextXY (GetMaxX div 2 + GetMaxX div 20, 0,
          'Time of Flight Mass Spectrum');
  OutTextXY (GetMaxX div 2 + GetMaxX div 20, (TextHeight ('T') * 3),
          (tof_axis_label [y_axis] + ' vs. ' + tof_axis_label [x_axis]));

  { label file name }
  SetColor (Red);
  bSetTextStyle (Font, HorizDir, 1);
  SetTextJustify (LeftText, BottomText);
  OutTextXY (0, GetMaxY, data_file_name);
end;  { procedure tof_draw_axes }


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


procedure draw_tof_dump_screen (x_axis, y_axis : integer;
                        data_count : integer);
var
  i : integer;
begin
  tof_draw_axes (x_axis, y_axis);
```

```
  SetColor (LightGray);
  MoveUser(0, tof_data^ [0]);
  for i := 1 to (data_count - 1) do
    begin
      DrawLineUser (i, tof_data^ [i]);
    end;
end; { procedure draw_tof_dump_screen }


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


procedure dump_tof_experiment_screen (x_axis, y_axis : integer;
                                      data_count : integer);
begin
  {show_screen_dump_parameters;
  screen_dump_menu;}

  { Put on screen what will go to output device }
  SetGraphMode(GraphMode);
  GraphicsOn := TRUE;
  draw_tof_dump_screen (x_axis, y_axis, data_count);

  { Now output to PS LaserWriter }

  prevent_clear_screen;

  erase_old_portfile;

  enter_laserwriter_graphics_mode;

  { Draw Plot of Data }
  draw_tof_dump_screen (x_axis, y_axis, data_count);

  plot_graph_and_exit;

  lpr_plot_to_laserwriter;

  InstallScreenGraphicsDriver;
  TextColor (14);
end;

end. { unit tofplot }
```

2.11 plotutil.pas

```
{ plotutil.pas                                    }

unit plotutil;

{$O+}
```

```
{$F+}

interface

uses globals,utility,insgrdrv,graph,graphadd,grafutil,CRT,DOS;

const
 Font : Integer = DefaultFont;

procedure set_done_flag;
procedure prevent_clear_screen;
procedure plot_graph_and_exit;
procedure lpr_plot_to_laserwriter;
procedure erase_old_portfile;
procedure enter_laserwriter_graphics_mode;
procedure enter_hp7470plotter_graphics_mode;

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

implementation

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

procedure set_done_flag;
var
 i : integer;
begin { procedure set_done_flag }
 bSetTextStyle (Font, HorizDir, 1);
 SetTextJustify (CenterText,TopText);
 SetColor (Yellow);
 OutTextXY ((MaxX div 2), (MaxY * 4 div 5),
        'DONE!  Hit any key to continue ...');
 For i := 1 to 8 do
  begin
   Sound (1000);
   Delay (100);
   NoSound;
   Delay (100);
  end;
 Repeat
  SetFillStyle (EmptyFill, EmptyFill);
  Bar ((MaxX div 2 - (TextWidth ('DONE!  Hit any key to continue ...') div 2)),
     (MaxY * 4 div 5),
     (MaxX div 2 + (TextWidth ('DONE!  Hit any key to continue ...') div 2)),
     (MaxY * 4 div 5 + TextHeight('D')));
  Delay (200);
  SetColor (Yellow);
  OutTextXY ((MaxX div 2), (MaxY * 4 div 5),
        'DONE!  Hit any key to continue ...');
  Delay (200);
 until KeyPressed;
end;  { procedure set_done_flag }

{ ************************************************************************** }
```

```
procedure prevent_clear_screen;
begin

 { After putting a graph on the screen, save the magic address
 contents, then set it to $a5 before calling closegraph to
 prevent clearing the screen.                              }

 hold := magic^;  magic^ := $a5;
 bclosegraph;
end; { procedure prevent_clear_screen }

procedure plot_graph_and_exit;
begin

 { Plot the graph }
 ClearDevice;
 status := GraphStatus;
 if (status and $8000) <> 0 then begin
    bCloseGraph;
    AbortM (GraphStatusMsg(status));
 end;

 { Exit Graphics }
 bCloseGraph;
end; { procedure plot_graph_and_exit }


procedure erase_old_portfile;
var
 f : file;
begin

 { Erase any existing old PortFILE plots so we don't print them as well }
 if exist('e:\plot.out') then
  begin
   assign (f, 'e:\plot.out');
   erase (f);
  end;
end; { procedure erase_old_portfile }

procedure lpr_plot_to_laserwriter;
begin
 SwapVectors;
 Exec('c:\command.com','/c lpr e:\plot.out');
 SwapVectors;
 if DosError <> 0 then
  begin
   TextMode(LastMode);
   magic^ := hold;
   TextColor (14 + Blink);
   GraphicsOn := FALSE;
   GoToXY (10,12);
   writeln ('Dos Error # ', DosError);
```

```
    write ('Hit <enter> to continue ...');
    readln;
   end
  else
   begin
    TextMode(LastMode);
    magic^ := hold;
    TextColor (14 + Blink);
    GraphicsOn := FALSE;
    GoToXY (10,10);
    writeln ('Transfer of plot to LaserWriter successful.');
    writeln;
    writeln ('              LPR exit code = ', DosExitCode);
    writeln;
    write ('            Hit <enter> to continue ...');
    readln;
   end;
end;  { procedure lpr_plot_to_laserwriter }

procedure enter_laserwriter_graphics_mode;
begin
 { Enter LaserWriter Graphics Mode }
 Graphmode := Land or PortFile;
 bInitGraph (PostScriptLaserWriterFile, Graphmode, PathToGraphics, @info);
 ErrorCode := GraphResult;
 if ErrorCode<>0 then AbortM (GraphErrorMsg(ErrorCode));
 status := GraphStatus;
 if (status and $8000) <> 0 then begin
    bCloseGraph;
    AbortM (GraphStatusMsg(status));
 end;
 PlotInit;
 Plot_Lines := TRUE;
end; { procedure enter_laserwriter_graphics_mode }

procedure enter_hp7470plotter_graphics_mode;
begin
 Graphmode := DraftPL or PortCOM1;
 bInitGraph (HP7470Plotter, Graphmode, PathToGraphics, @info);
 ErrorCode := GraphResult;
 if ErrorCode<>0 then AbortM (GraphErrorMsg(ErrorCode));
 status := GraphStatus;
 if (status and $8000) <> 0 then begin
  bCloseGraph;
  AbortM (GraphStatusMsg(status));
 end;
 PlotInit;
 Plot_Lines := TRUE;
end; { procedure  }

end. { unit plotutil }
```

2.12 parameter.pas

```
{ parameters.pas                                    }

{ contents :                                        }

{ procedure experimental_parameters                 }
{     menu for changing experimental set_up parameters }

{ procedure get_detachment_wavelength               }

{ procedure get_cal_offset                          }

{ procedure check_range_vs_step                     }

{ procedure get_initial_wavelength                  }

{ procedure get_final_wavelength                    }

{ procedure get_lambda_step                         }

{ procedure get_shots_per_point                     }

{ procedure get_doubling                            }

{ procedure get_use_etalon                          }

{ procedure get_grating_orders                      }

{ procedure get_ADC_channels                        }

{ procedure get_zero_average_shot_number            }

{ function read_set_up : boolean                    }
{     initialize experimental set-up parameters from a file }

{ procedure write_set_up                            }
{     write experimental set_up parameters to a file }

{ procedure show_parameters                         }
{     display experimental set_up parameters on right hand side of screen }

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

unit parameters;

{$O+}
{$F+}

interface

uses globals,utility,gpibutil,dyelaser,DOS,CRT;
```

```
procedure show_parameters;
procedure get_detachment_wavelength;
procedure get_cal_offset;
procedure check_range_vs_step;
procedure check_max_data;
procedure get_initial_wavelength;
procedure get_final_wavelength;
procedure get_lambda_step;
procedure get_shots_per_point;
procedure get_doubling;
procedure get_use_etalon;
procedure get_grating_orders;
procedure get_ADC_channels;
procedure get_zero_average_shot_number;
function read_set_up : boolean;
procedure write_set_up;
procedure experimental_parameters;

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

implementation

procedure show_parameters;
var
 x, y : integer;
begin { procedure show_parameters }
 x := wherex; y := wherey;
 window (37, 1, 80, 24);
 clrscr; textcolor (11);
 with stp do
  begin
   write ('    Detachment Wavelength : ');
   writeln (detach_lambda:12:7);
   if full_parameter_listing then
     begin
      write ('  Detachment Grating Order : ');
      writeln (grating_order_detachment_dye_laser:4);
      write ('  Laser Calibration Offset : ');
      writeln (cal_offset:12:7);
     end;
   write ('       Initial Wavelength : ');
   writeln (start_lambda:12:7);
   write ('         Final Wavelength : ');
   writeln (end_lambda:12:7);
   write ('              Lambda Step : ');
   writeln (delta_lambda:12:7);
   write ('          Shots per Point : ');
   writeln (shots_per_point:4);
   write ('  Autotracker (Doubling?) : ');
   if stp.doubling then
     writeln (' YES')
   else
     writeln ('  NO');
```

```
    write ('              Use Etalon : ');
    if stp.use_etalon then
      writeln (' YES')
    else
      writeln ('  NO');
    if full_parameter_listing then
      begin
       write ('Dissociation Grating Order : ');
       writeln (grating_order_dissociation_dye_laser:4);
       write ('  Detached Electron Signal : ');
       writeln (ADC_channel [detached_electron_signal]:4);
       write ('              Laser Power : ');
       writeln (ADC_channel [laser_power]:4);
       write ('Dissociated Radical Signal : ');
       writeln (ADC_channel [dissociated_radical_signal]:4);
       write ('    Iodine Reference Cell : ');
       writeln (ADC_channel [iodine_reference]:4);
       write ('  Zero Average Shot Number : ');
       writeln (zero_average_shot_number:4);
      end;
    write ('   Detached Electron Zero : ');
    writeln (zero [detached_electron_signal]:9:4);
    write ('         Laser Power Zero : ');
    writeln (zero [laser_power]:9:4);
    write ('  Dissociated Radical Zero : ');
    writeln (zero [dissociated_radical_signal]:9:4);
    write ('Iodine Reference Cell Zero : ');
    writeln (zero [iodine_reference]:9:4);
  end;
 window (1, upper_left_corner_y, 80, 24); textcolor (14);
 gotoxy (x, y);
end; { procedure show_parameters }

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

procedure get_detachment_wavelength;
begin { procedure get_detachment_wavelength }
 if ask_for_real (stp.detach_lambda, 'Detachment wavelength : ',
                                        200, 900) then
   begin
    set_grating_order (detachment_dye_laser);
    set_gpib_remote (TRUE);
    move_dye_laser (detachment_dye_laser, stp.detach_lambda);
    set_gpib_remote (FALSE);
    show_parameters;
   end;
end; {procedure get_detachment_wavelength }

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

procedure get_cal_offset;
begin { procedure get_cal_offset }
 if ask_for_real (stp.cal_offset, 'Calibration offset [nm] : ', -10, 10) then
   show_parameters;
```

```
end; { procedure cal_offset }

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

procedure check_range_vs_step;
begin { procedure check_range_vs_step }
 while abs (stp.end_lambda - stp.start_lambda) < abs (stp.delta_lambda) do
  begin
   get_lambda_step;
  end;
end; { check_range_vs_step }

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

procedure check_max_data;
begin {procedure check_max_data }
 if abs (stp.end_lambda - stp.start_lambda) / stp.delta_lambda > 1024 then
   begin
     writeln;
     write ('          *** WARNING : # OF DATA POINTS EXCEEDS ');
     writeln ('LIMIT ***');
     write ('    CHANGE INITIAL, FINAL WAVELENGTHS, STEP SIZE OR ');
     writeln ('SOME COMBINATION');
     writeln;
     write ('                          Hit <enter> to continue ....');
     readln;
   end;
end; {procedure check_max_data }

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

procedure get_initial_wavelength;
var
 old_start : real;
begin { procedure get_initial_wavelength }
 if ((stp.use_etalon) and (stp.end_lambda < stp.start_lambda)) then
  begin
   writeln ('Changing this parameter will require new etalon set up.');
   if not ask_continue then exit;
   stp.use_etalon := FALSE;
  end;
 old_start := stp.start_lambda;
 if ask_for_real (stp.start_lambda, 'Initial wavelength : ', 200, 900) then
   begin
      set_grating_order (dissociation_dye_laser);
      if stp.start_lambda = stp.end_lambda then
         stp.end_lambda := old_start;
      if (stp.start_lambda > stp.end_lambda) then
        stp.delta_lambda := -abs (stp.delta_lambda)
      else
         stp.delta_lambda := abs (stp.delta_lambda);
      check_range_vs_step;
      check_max_data;
      show_parameters;
```

```
    end;
end; {procedure get_initial_wavelength}

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

procedure get_final_wavelength;
var
 old_end : real;
begin { procedure get_final_wavelength }
 if ((stp.use_etalon) and (stp.end_lambda > stp.start_lambda)) then
   begin
    writeln ('Changing this parameter will require new etalon set up.');
    if not ask_continue then exit;
    stp.use_etalon := FALSE;
   end;
 if ask_for_real (stp.end_lambda, 'Final wavelength : ', 200, 900) then
    begin
       set_grating_order (dissociation_dye_laser);
       if stp.end_lambda = stp.start_lambda then
          stp.start_lambda := old_end;
       if (stp.start_lambda > stp.end_lambda) then
         stp.delta_lambda := -abs (stp.delta_lambda)
       else
          stp.delta_lambda := abs (stp.delta_lambda);
       check_range_vs_step;
       check_max_data;
       show_parameters;
    end;
end; { procedure get_final_wavelength }

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

procedure get_lambda_step;
var lower_limit : real;
begin { procedure get_lambda_step }
 if not stp.use_etalon then
    begin
      lower_limit := (0.00315 / stp.grating_order_dissociation_dye_laser);
    end
 else
    begin
      lower_limit := stp.etalon_parameters.min_etalon_delta_lambda;
 end;
 if ask_for_real (stp.delta_lambda, 'Lambda step : ',
    lower_limit, abs (stp.end_lambda - stp.start_lambda))
    then
    begin
       if (stp.start_lambda > stp.end_lambda) then
         stp.delta_lambda := -abs (stp.delta_lambda)
       else
          stp.delta_lambda := abs (stp.delta_lambda);
       check_range_vs_step;
       check_max_data;
       show_parameters;
```

```
    end;
end { procedure get_lambda_step };

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

procedure get_shots_per_point;
begin { procedure shots_per_point }
 if ask_for_integer (stp.shots_per_point, 'Shots per Point : ',
    1, 32767) then
    show_parameters;
end { procedure get_shots_per_point };

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

procedure get_doubling;
begin { procedure get_doubling }
 stp.doubling := ask_for_boolean ('Use Autotracker? (Double?)');
 set_grating_order (dissociation_dye_laser);
 show_parameters;
end {procedure get_doubling };

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

procedure get_use_etalon;
var
 lambda, old_lambda : real;
 normal_position_set : boolean;
 end_step : integer;
begin { procedure get_use_etalon }
 stp.use_etalon := ask_for_boolean ('Use Etalon? : ');
 if stp.use_etalon then
   begin
    with stp do
     begin
       clrscr;  get_cal_offset;
       use_etalon := FALSE;
       get_initial_wavelength;
       get_final_wavelength;
       use_etalon := TRUE;
       get_lambda_step;  clrscr;  writeln;
       writeln ('Be patient, etalon is proceeding to set up position!');
       if start_lambda < end_lambda then
         begin
           lambda := end_lambda;
         end
       else
         begin
           lambda := start_lambda;
         end;
       etalon_parameters.upper_lambda := lambda;
       etalon_parameters.lambda_0 :=
                          lambda / (1 - sqr(800 * 3.125e-6) / 2);
       use_etalon := FALSE;
       set_gpib_remote (TRUE);
```

```
if doubling = TRUE then
  begin
   old_lambda :=
          get_laser_wavelength (dissociation_dye_laser);
   move_dye_laser_slowly_since_doubling (dissociation_dye_laser,
                  old_lambda, etalon_parameters.lambda_0);
  end;
move_dye_laser (dissociation_dye_laser,
                           etalon_parameters.lambda_0);
use_etalon := TRUE;
if set_etalon_normal_position then
 begin
  set_gpib_remote (FALSE);
  stp.etalon_parameters.max_etalon_scan_range :=
       lambda * 0.5 * sqr((24000 -
       (etalon_parameters.etalon_normal_position + 800)) *
       3.125e-6);
  if abs(start_lambda - end_lambda) >
    etalon_parameters.max_etalon_scan_range then
    begin
      writeln ('Sorry, the maximum etalon scan range is : ',
           etalon_parameters.max_etalon_scan_range:0:6,
           ' nm.');
      writeln;
      write ('Re-enter etalon mode and adjust initial ');
      writeln ('wavelength, final wavelength, or both.');
      writeln ('Hit <enter> to continue ....');
      readln;
      use_etalon := FALSE;
      exit;
    end
  else
     begin
      writeln;  writeln;  write ('              ');
      writeln ('Etalon scan range is satisfactory.');  writeln;
      writeln ('          The maximum etalon scan range is : ',
           etalon_parameters.max_etalon_scan_range:0:6,' nm.');
      writeln;  writeln;
      write ('              Hit <enter> to continue ....');
      readln;
     end;
  end_step := etalon_parameters.etalon_normal_position + 800
         + round (sqrt (2 * abs(start_lambda - end_lambda)
         * 1.024e11 / lambda));
  etalon_parameters.min_etalon_delta_lambda :=
               lambda * 9.76563e-12 * (end_step -
         (etalon_parameters.etalon_normal_position + 800));
  while abs (delta_lambda) <
      etalon_parameters.min_etalon_delta_lambda do
      begin
       writeln ('Sorry, minimum step size is : ',
          etalon_parameters.min_etalon_delta_lambda:0:8);
       get_lambda_step;
      end;
```

```
          writeln;  writeln;  write ('                ');
          writeln ('Step size is also satisfactory.');  writeln;
          write ('    ');
          write ('The minimum etalon step size for this scan range is : ');
          writeln (etalon_parameters.min_etalon_delta_lambda:0:8, ' nm.');
          writeln;  writeln;
          write ('                Hit <enter> to continue ....');
          readln;  clrscr;  writeln;  writeln;
          writeln ('Proceeding to initial dissociation wavelength ....');
          if (end_lambda > start_lambda) then
            begin
              set_gpib_remote (TRUE);
              move_dye_laser (dissociation_dye_laser, start_lambda);
              set_gpib_remote (FALSE);
            end;
          end
        else
          begin
            set_gpib_remote (FALSE);
            use_etalon := FALSE;
          end; { if normal position set }
      end; {with stp do }
    end; { if use_etalon is TRUE }
    show_parameters;
end; { procedure get_use_etalon }

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

procedure get_grating_orders;
begin {procedure get_grating_orders}
  if ask_for_integer (stp.grating_order_detachment_dye_laser,
              'Detachment dye laser grating order : ', 3, 8) then
     show_parameters;
  if ask_for_integer (stp.grating_order_dissociation_dye_laser,
              'Dissociation dye laser grating order : ', 3, 8) then
     show_parameters;
end { procedure get_grating_orders };

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

procedure get_ADC_channels;
begin {procedure get_ADC_channels}
  writeln ('Detached Electron Signal in ADC Channel ',
               stp.ADC_channel [detached_electron_signal]:1);
  if ask_for_integer (stp.ADC_channel [detached_electron_signal],
              'New Detached Electron Signal Channel : ', 0, 11) then
                 show_parameters;
  writeln ('Laser Power in ADC Channel ',
             stp.ADC_channel [laser_power]:1);
  if ask_for_integer (stp.ADC_channel [laser_power],
              'New Laser Power Channel : ', 0, 11) then
                 show_parameters;
  writeln ('Dissociated Radical Signal in ADC Channel ',
              stp.ADC_channel [dissociated_radical_signal]:1);
```

```
 if ask_for_integer (stp.ADC_channel [dissociated_radical_signal],
              'New Dissociated Radical Signal Channel : ', 0, 11) then
                 show_parameters;
 writeln ('Iodine Reference Cell Signal in ADC Channel ',
              stp.ADC_channel [iodine_reference]:1);
 if ask_for_integer (stp.ADC_channel [iodine_reference],
              'New Iodine Reference Cell Channel : ', 0, 11) then
                 show_parameters;
end { procedure get_ADC_channels };

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

procedure get_zero_average_shot_number;
begin { procedure get_zero_average_shot_number }
 writeln ('Currently zero-averaging for ', stp.zero_average_shot_number,
       ' shots.');
 if ask_for_integer (stp.zero_average_shot_number,
   'New number of shots to average for zeros : ', 1, 32767) then
   show_parameters;
end {procedure get_zero_average_shot_number };

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

function read_set_up : boolean;
var
 i : integer;
 shg, high_res, ch : char;
begin { function read_set_up }
 if not initial_read then
 begin
  writeln ('Are you sure you want to read old set up file?');
  if not ask_continue then exit;
 end;
 if not exist (set_up_file_name) then
  begin
   read_set_up := FALSE;
   exit;
  end;
 assign (data_file, set_up_file_name);
 reset (data_file);
 with stp do
  begin
   readln (data_file, detach_lambda);
   readln (data_file, cal_offset);
   readln (data_file, start_lambda);
   readln (data_file, end_lambda);
   readln (data_file, delta_lambda);
   readln (data_file, shots_per_point);
   readln (data_file, shg);
    if shg = 'T' then
     begin
      doubling := TRUE;
     end
    else
```

```
        begin
         doubling := FALSE;
        end;
      readln (data_file, high_res);
       if high_res = 'T' then
        begin
         use_etalon := TRUE;
        end
       else
        begin
         use_etalon := FALSE;
        end;
      with etalon_parameters do
         begin
          read (data_file, etalon_normal_position);
          read (data_file, lambda_0);
          read (data_file, upper_lambda);
          read (data_file, initial_grating_position);
          read (data_file, max_etalon_scan_range);
          read (data_file, min_etalon_delta_lambda);
         end;
      readln (data_file, grating_order_detachment_dye_laser);
      readln (data_file, grating_order_dissociation_dye_laser);
      readln (data_file, ADC_channel [detached_electron_signal]);
      readln (data_file, ADC_channel [laser_power]);
      readln (data_file, ADC_channel [dissociated_radical_signal]);
      readln (data_file, ADC_channel [iodine_reference]);
      readln (data_file, zero [detached_electron_signal]);
      readln (data_file, zero [laser_power]);
      readln (data_file, zero [dissociated_radical_signal]);
      readln (data_file, zero [iodine_reference]);
      readln (data_file, zero_average_shot_number);
    end;
  close (data_file);
  read_set_up := TRUE;
  set_grating_order (dissociation_dye_laser);
end;  { function read_set_up }

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

procedure write_set_up;
var
  i : integer;
  shg, high_res : char;
begin { procedure write_set_up }
  stp.use_etalon := FALSE;
  assign  (data_file, set_up_file_name);
  rewrite (data_file);
  with stp do
    begin
      writeln (data_file, detach_lambda:10:5);
      writeln (data_file, cal_offset:10:5);
      writeln (data_file, start_lambda:10:5);
      writeln (data_file, end_lambda:10:5);
```

```
    writeln (data_file, delta_lambda:10:5);
    writeln (data_file, shots_per_point:1);
    if doubling = TRUE then
     begin
      shg := 'T';
     end
    else
     begin
      shg := 'F';
     end;
    writeln (data_file, shg);
    if use_etalon = TRUE then
     begin
      high_res := 'T';
     end
    else
     begin
      high_res := 'F';
     end;
    writeln (data_file, high_res);
    with etalon_parameters do
       begin
        writeln (data_file, etalon_normal_position:1);
        writeln (data_file, lambda_0:10:5);
        writeln (data_file, upper_lambda:10:5);
        writeln (data_file, initial_grating_position:1);
        writeln (data_file, max_etalon_scan_range:10:8);
        writeln (data_file, min_etalon_delta_lambda:10:8);
       end;
    writeln (data_file, grating_order_detachment_dye_laser:1);
    writeln (data_file, grating_order_dissociation_dye_laser:1);
    writeln (data_file, ADC_channel [detached_electron_signal] :1);
    writeln (data_file, ADC_channel [laser_power] :1);
    writeln (data_file, ADC_channel [dissociated_radical_signal] :1);
    writeln (data_file, ADC_channel [iodine_reference] :1);
    writeln (data_file, zero [detached_electron_signal]:5:3);
    writeln (data_file, zero [laser_power]:5:3);
    writeln (data_file, zero [dissociated_radical_signal]:5:3);
    writeln (data_file, zero [iodine_reference]:5:3);
    writeln (data_file, zero_average_shot_number:1);
   end;
  close (data_file);
end; { procedure write_set_up }

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

procedure experimental_parameters;
var
 ch : char;
 y : integer;
 min_lambda, max_lambda : real;
  procedure show_menu;
   begin { procedure show_menu }
    window (1, 1, 36, 24);
```

```
      clrscr; textcolor (15);
      writeln ('D : Detachment Wavelength');
      writeln ('G : Force Dye Laser Grating Orders');
      writeln ('O : Laser Calibration Offset');
      writeln ('I : Initial Wavelength');
      writeln ('F : Final Wavelength');
      writeln ('L : Lambda Step');
      writeln ('S : Shots per Point');
      writeln ('A : Autotracker (Doubling?)');
      writeln ('E : Use Etalon');
      writeln ('G : Force Dye Laser Grating Orders'); writeln;
      writeln ('C : Change ADC Channels'); writeln; writeln;
      writeln ('Z : Zero Average Shot Number');
      writeln ('R : Read Set-up from File');
      writeln ('W : Write Current Set-up to File');
      writeln;
      writeln ('Q : Quit Set-up Menu');
      writeln;
      textcolor (14);
      upper_left_corner_y := wherey;
    end;  { procedure show_menu }

begin { procedure experimental_parameters }
 show_menu;
 full_parameter_listing := TRUE;
 show_parameters;
 repeat
  window (1, upper_left_corner_y, 80, 24); clrscr;
  write ('? > ');
  ch := Readkey; ch := upcase (ch); writeln (ch); writeln;
  case ch of
    'D' : get_detachment_wavelength;
    'O' : get_cal_offset;
    'I' : get_initial_wavelength;
    'F' : get_final_wavelength;
    'L' : get_lambda_step;
    'S' : get_shots_per_point;
    'A' : get_doubling;
    'E' : get_use_etalon;
    'G' : get_grating_orders;
    'C' : get_ADC_channels;
    'Z' : get_zero_average_shot_number;
    'R' : begin
          if read_set_up then
            show_parameters;
        end;
    'W' : write_set_up;
  end; { case ch }
 until (ch = 'Q');
 full_parameter_listing := FALSE;
 show_parameters;
end;  { procedure experimental_parameters }

{ ********************************************************************** }
```

```
end. { unit parameters }
```

2.13 dyelaser.pas

```
{ dyelaser.pas
 procedures used for GPIB and dye laser control

********************************************************************************* }

unit dyelaser;

{$O+}
{$F+}

interface

uses globals,graph,utility,gpibutil,tpdecl,CRT;

type
 string_80 = string[80];
 string_8 = string[8];
 string_2 = string[2];

var
 Bd, dye_laser : integer;
 laser_buffer : string_80;
 IBBuf : array[1..$FF] of char;
 grating_cal_detachment_dye_laser : real;
 grating_cal_dissociation_dye_laser : real;

procedure read_dye_laser (dye_laser : integer);
procedure write_dye_laser (dye_laser : integer; w_laser_buffer : string_80);
function real_laser_position (start_byte, end_byte : byte) : real;
function ASCII_convert (value : integer) : string_2;
procedure ASCII_laser_position (grating : real;
                    etalon, crystal, reserve : integer);
procedure get_grating_cal (dye_laser : integer);
function grating_cal_mismatch : boolean;
procedure set_grating_order (dye_laser : integer);
function drive_to_wave (dye_laser : integer; grating : real;
                      etalon : integer) : real;
function wave_to_grating_drive (dye_laser : integer; lambda : real) : real;
procedure wave_to_etalon_drive (lambda : real;
                    var grating : real; var etalon : integer);
function get_laser_wavelength (dye_laser : integer) : real;
procedure wait_for_laser_ready (dye_laser : integer);
procedure move_dye_laser (dye_laser : integer; lambda : real);
procedure move_dye_laser_slowly_since_doubling (dye_laser : integer;
                    old_lambda, lambda : real);
function set_etalon_normal_position : boolean;
```

```
{ ************************************************************************ }

implementation

{ ************************************************************************ }
{ read a message from the dye laser into the string laser_buffer     }

procedure read_dye_laser (dye_laser : integer);
var
 i : integer;

begin { procedure read_dye_laser }
 IBRd(dye_laser, IBBuf, $FF);
 if (IBSta < 0) then GPIB_error('IBRd');
 for i := 1 to (IBCnt - 1) do
  laser_buffer [i] := IBBuf [i];
 laser_buffer [0] := chr(IBCnt - 1);
end; { procedure read_dye_laser }

{ ************************************************************************ }
{ write a message from the string w_laser_buffer to a dye laser      }

procedure write_dye_laser (dye_laser : integer; w_laser_buffer : string_80);
var
 i : integer;
 buffer_length : integer;
begin { procedure write_dye_laser }
 w_laser_buffer := w_laser_buffer + chr($0D);
 buffer_length := length (w_laser_buffer);
 for i := 1 to buffer_length do
  IBBuf [i] := w_laser_buffer [i];

 IBWrt(dye_laser, IBBuf, buffer_length);
 if (IBSta < 0) then GPIB_error('IBWrt');
end;  { procedure write_dye_laser }

{ ************************************************************************ }
{ convert a substring of laser_buffer (stacked ASCII coded) within }
{ the string positions start_byte and end_byte to a real value     }

function real_laser_position (start_byte, end_byte : byte) : real;
var
 i : byte;
 real_value : real;
begin { function real_laser_position }
 real_value := 0;
 for i := start_byte downto end_byte do
  real_value := real_value * 16 + (ord(laser_buffer[i]) - $41);
 real_laser_position := real_value;
end;  { function real_laser_position }

{ ************************************************************************ }
```

```
function ASCII_convert (value : integer) : string_2;
var
 first_letter, second_letter : char;
begin { function ASCII_convert }
 first_letter := chr ((value mod 16) + $41);
 second_letter := chr ((value div 16) + $41);
 ASCII_convert := concat (first_letter, second_letter);
end; { function ASCII_convert }

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

procedure ASCII_laser_position (grating : real;
                        etalon, crystal, reserve : integer);
const
 HI_GRATE = 65536.00;
 LO_GRATE = 256.00;
var
 high_grating, mid_grating, low_grating : real;
begin { procedure ASCII_laser_position }
 laser_buffer := 'SA';
 high_grating := grating / HI_GRATE;
 mid_grating := frac (high_grating) * HI_GRATE / LO_GRATE;
 low_grating := frac (mid_grating) * LO_GRATE;
 laser_buffer := laser_buffer + ASCII_convert (trunc (low_grating))
                        + ASCII_convert (trunc (mid_grating))
                        + ASCII_convert (trunc (high_grating))
                        +.ASCII_convert (lo (etalon))
                        + ASCII_convert (hi (etalon))
                        + ASCII_convert (lo (crystal))
                        + ASCII_convert (hi (crystal))
                        + ASCII_convert (lo (reserve))
                        + ASCII_convert (hi (reserve));
end; { procedure ASCII_laser_position }

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

procedure get_grating_cal (dye_laser : integer);
begin { procedure get_grating_cal }
 write_dye_laser (dye_laser,'?G');
 read_dye_laser (dye_laser);
 if dye_laser = dissociation_dye_laser then
   begin
     grating_cal_dissociation_dye_laser := real_laser_position (6, 1)
                                  / 1000.0;
     writeln ('Dissociation Dye Laser Grating Calibration : ',
        grating_cal_dissociation_dye_laser:0:7);
   end
 else
   begin
     grating_cal_detachment_dye_laser := real_laser_position (6, 1)
                                 / 1000.0;
     writeln ('  Detachment Dye Laser Grating Calibration : ',
                 grating_cal_detachment_dye_laser:0:7);
 end;
```

```
end;  { procedure get_grating_cal }

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

function grating_cal_mismatch : boolean;
var
 grating_cal : real;
 dye_laser : integer;
begin
 dye_laser := detachment_dye_laser;
 write_dye_laser (dye_laser,'?G');
 read_dye_laser (dye_laser);
 grating_cal := real_laser_position (6, 1) / 1000.0;
 if not (grating_cal = grating_cal_detachment_dye_laser) then
  begin
   grating_cal_mismatch := TRUE;
   exit;
  end;
 dye_laser := dissociation_dye_laser;
 write_dye_laser (dye_laser,'?G');
 read_dye_laser (dye_laser);
 grating_cal := real_laser_position (6, 1) / 1000.0;
 if not (grating_cal = grating_cal_dissociation_dye_laser) then
  begin
   grating_cal_mismatch := TRUE;
   exit;
  end;
 grating_cal_mismatch := FALSE;
end; { function grating_cal_mismatch }

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

procedure set_grating_order (dye_laser : integer);
var grating_cal : real;  grating_order : integer;
begin { procedure set_grating_order }
 with stp do
  begin
   if (dye_laser = detachment_dye_laser) then
     begin
      grating_cal := grating_cal_detachment_dye_laser;
      grating_order := trunc ((900.00225 + grating_cal) / detach_lambda);
      if (grating_order > 8) then grating_order := 8;
      if (grating_order < 3) then grating_order := 3;
      stp.grating_order_detachment_dye_laser := grating_order;
     end
   else
     begin
      grating_cal := grating_cal_dissociation_dye_laser;
      if doubling then
       begin
        if (end_lambda > start_lambda) then
         begin
          grating_order := trunc ((900.00225 + grating_cal)
                    / (2.0 * end_lambda));
```

```
            if ((trunc (grating_cal / (2.0 * start_lambda)) + 1) >
                              grating_order) then
              grating_order := grating_order + 1;
            end
           else
            begin
             grating_order := trunc ((900.00225 + grating_cal)
                          / (2.0 * start_lambda));
             if ((trunc (grating_cal / (2.0 * end_lambda)) + 1) >
                              grating_order) then
               grating_order := grating_order + 1;
            end;
          end
         else
          begin
           if (end_lambda > start_lambda) then
            begin
             grating_order := trunc ((900.00225 + grating_cal)
                                  / end_lambda);
             if ((trunc (grating_cal / start_lambda) + 1)
                                       > grating_order) then
               grating_order := grating_order + 1;
            end
           else
            begin
             grating_order := trunc ((900.00225 + grating_cal)
                          / start_lambda);
             if ((trunc (grating_cal / end_lambda) + 1)
                                       > grating_order) then
               grating_order := grating_order + 1;
            end;
          end;
       if (grating_order > 8) then grating_order := 8;
       if (grating_order < 3) then grating_order := 3;
       stp.grating_order_dissociation_dye_laser := grating_order;
       end;
  end;
end;  { procedure set_grating_order }

{ ************************************************************** }
{ convert grating position (in motor steps) to wavelength (in nm) }

function drive_to_wave (dye_laser : integer; grating : real;
                        etalon : integer) : real;
var grating_cal, result : real; grating_order : integer;
begin { function drive_to_wave }
 with stp do
  begin
   if dye_laser = dissociation_dye_laser then
     begin
       grating_cal := grating_cal_dissociation_dye_laser;
       grating_order := grating_order_dissociation_dye_laser;
     end
   else
```

```
        begin
          grating_cal := grating_cal_detachment_dye_laser;
          grating_order := grating_order_detachment_dye_laser;
        end;
      if (use_etalon) and (dye_laser = dissociation_dye_laser) then
        begin
         drive_to_wave := stp.etalon_parameters.lambda_0
                    * (1 - 0.5 * sqr(3.125e-6
              * (etalon - stp.etalon_parameters.etalon_normal_position)));
        end
      else
        begin
         result := ((grating * 0.00315)
                  + grating_cal) / grating_order;
         if doubling and (dye_laser = dissociation_dye_laser) then
          begin
           result := result * 0.5;
          end;
         drive_to_wave := result;
        end;
    end;
end;  { function drive_to_wave }

{ ********************************************************************** }
{ convert wavelength (in nm) to grating position (in motor steps) }

function wave_to_grating_drive (dye_laser : integer; lambda : real) : real;
var
 grating_cal, result : real; grating_order : integer;
begin { function wave_to_grating_drive }
 if (stp.doubling) and (dye_laser = dissociation_dye_laser) then
  lambda := lambda * 2.0;
 if dye_laser = detachment_dye_laser then
   begin
    grating_cal := grating_cal_detachment_dye_laser;
    grating_order := stp.grating_order_detachment_dye_laser;
   end
 else
   begin
    grating_cal := grating_cal_dissociation_dye_laser;
    grating_order := stp.grating_order_dissociation_dye_laser;
   end;
 result := ((lambda * grating_order) - grating_cal) / 0.00315;
 if ((result > 285715.0) and (grating_order > 3)) then
  begin
   grating_order := grating_order - 1;
   result := ((lambda * grating_order) - grating_cal) / 0.00315;
  end
 else if ((result < 0.0) and (grating_order < 8)) then
  begin
   grating_order := grating_order + 1;
   result := ((lambda * grating_order) - grating_cal) / 0.00315;
  end;
 if dye_laser = dissociation_dye_laser then
```

```
    stp.grating_order_dissociation_dye_laser := grating_order
  else
    stp.grating_order_detachment_dye_laser := grating_order;
  wave_to_grating_drive := result;
end;  { function wave_to_drive }

{ ********************************************************************** }
{ convert wavelength (in nm) to etalon and grating position (in motor steps) }

procedure wave_to_etalon_drive (lambda : real;
                        var grating : real; var etalon : integer);
var
  doubling_lambda_0, doubling_upper_lambda : real;
begin { procedure wave_to_etalon_drive }
  with stp.etalon_parameters do
    begin
      if (stp.doubling) then
        begin
          lambda := lambda * 2;
          doubling_lambda_0 := lambda_0 * 2;
          doubling_upper_lambda := upper_lambda * 2;

          grating := initial_grating_position -
                  (doubling_upper_lambda - lambda)
                  * 3.1746e2 * stp.grating_order_dissociation_dye_laser;

          etalon := round (etalon_normal_position +
                  sqrt (2 * abs(doubling_lambda_0 - lambda)
                  / doubling_lambda_0) * 3.2e5);
        end
      else
        begin
          grating := initial_grating_position - (upper_lambda - lambda)
                  * 3.1746e2 * stp.grating_order_dissociation_dye_laser;

          etalon := round (etalon_normal_position +
                  sqrt (2 * abs(lambda_0 - lambda) / lambda_0) * 3.2e5);
        end;
    end;
end; { procedure wave_to_etalon_drive }

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

function get_laser_wavelength (dye_laser : integer) : real;
begin { function get_laser_wavelength }
  write_dye_laser (dye_laser,'?A');
  read_dye_laser (dye_laser);
  get_laser_wavelength := drive_to_wave (dye_laser,
     (real_laser_position (6, 1)), (round (real_laser_position (10,7))));
end;  { function get_laser_wavelength }

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

procedure wait_for_laser_ready (dye_laser : integer);
```

```
begin { procedure wait_for_laser_ready }
 write_dye_laser (dye_laser,'?S');
 read_dye_laser (dye_laser);
 if (laser_buffer [1] <> 'R') then
  repeat
   delay (1);
   write_dye_laser (dye_laser,'?S');
   read_dye_laser (dye_laser);
  until (laser_buffer [1] = 'R');
end;  { procedure wait_for_laser_ready }

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

procedure move_dye_laser (dye_laser : integer; lambda : real);
var
 grating : real;
 etalon : integer;
begin { procedure move_dye_laser }
 if (stp.use_etalon) and (dye_laser = dissociation_dye_laser) then
   begin
     wave_to_etalon_drive (lambda, grating, etalon);
     ASCII_laser_position (grating + 5, etalon + 5, 0, 0);
     write_dye_laser (dye_laser, laser_buffer);
     wait_for_laser_ready (dye_laser);
     ASCII_laser_position (grating, etalon, 0, 0);
     write_dye_laser (dye_laser, laser_buffer);
     wait_for_laser_ready (dye_laser);
   end
 else
   begin
     grating := wave_to_grating_drive (dye_laser, lambda);
     ASCII_laser_position (grating + 5, 0, 0, 0);
     write_dye_laser (dye_laser, laser_buffer);
     wait_for_laser_ready (dye_laser);
     ASCII_laser_position (grating, 0, 0, 0);
     write_dye_laser (dye_laser, laser_buffer);
     wait_for_laser_ready (dye_laser);
   end;
end;  { procedure move_dye_laser }

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

procedure move_dye_laser_slowly_since_doubling (dye_laser : integer;
                        old_lambda, lambda : real);
var
 i : integer;
begin { procedure move_dye_laser_slowly_since_doubling }
 if abs (lambda - old_lambda) > 0.1 then
   begin
    for i := 1 to trunc (abs (lambda - old_lambda) / 0.1) do
     begin
      old_lambda := old_lambda + 0.1 * (lambda - old_lambda)
                        / (abs (lambda - old_lambda));
      move_dye_laser (dissociation_dye_laser, old_lambda);
```

```
        wait_for_laser_ready (dissociation_dye_laser);
       end;
  end;
end; { procedure move_dye_laser_slowly_since_doubling }

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


function set_etalon_normal_position : boolean;
var
  ch : char;
  grating : real;
  etalon : integer;
begin { procedure set_etalon_normal_position }
  write_dye_laser (dissociation_dye_laser,'?A');
  read_dye_laser (dissociation_dye_laser);
  grating := real_laser_position (6,1);
  wait_for_laser_ready (dissociation_dye_laser);
  set_gpib_remote (FALSE);
  writeln;
  writeln ('Please set etalon normal position.');
  if not ask_continue then
    begin
      set_etalon_normal_position := false;
      exit;
    end;
  set_gpib_remote (TRUE);
  write_dye_laser (dissociation_dye_laser,'?A');
  read_dye_laser (dissociation_dye_laser);
  stp.etalon_parameters.etalon_normal_position :=
                    round (real_laser_position (10,7));
  etalon := stp.etalon_parameters.etalon_normal_position + 800;
  ASCII_laser_position (grating, etalon, 0, 0);
  write_dye_laser (dissociation_dye_laser, laser_buffer);
  wait_for_laser_ready (dissociation_dye_laser);
  set_gpib_remote (FALSE);
  writeln;
  writeln ('Now tune grating slightly to establish tracking.');
  if not ask_continue then
    begin
      set_etalon_normal_position := false;
      exit;
    end;
  set_gpib_remote (TRUE);
  write_dye_laser (dissociation_dye_laser,'?A');
  read_dye_laser (dissociation_dye_laser);
  stp.etalon_parameters.initial_grating_position :=
                              (real_laser_position (6,1));
  set_etalon_normal_position := true;
end; { procedure set_etalon_normal_position }

end. { unit dyelaser }
```

2.14 twkplot.pas

```
{ twkplot.pas                                                    }

{ contents :                                                     }

{ procedure tweak_draw_axes                                      }
{ procedure tweak_y_ticks (y_data : integer)                     }
{ procedure tweak_replot (y_data, tweak_count : integer; visible : boolean)}
{ procedure tweak_plot_point (tweak_count, y_data : integer)     }

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

unit twkplot;

{$O+}
{$F+}

interface

uses globals,txsplot,plotutil,graph,graphadd,grafutil,CRT;

procedure tweak_draw_ticks (tick_axis : integer;
                            constant_limit : real;
                            code : integer;
                            label_ticks : boolean);
procedure tweak_draw_axes;
procedure tweak_y_ticks (y_data : integer);
procedure tweak_replot (y_data, tweak_count : integer; visible : boolean);
procedure tweak_replot_all (tweak_count : integer);
procedure tweak_plot_point (tweak_count, y_data : integer);

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

implementation

procedure tweak_draw_ticks (tick_axis : integer;
                            constant_limit : real;
                            code : integer;
                            label_ticks : boolean);
var
 xdotnum, ydotnum : integer;
 tick_mark, tick_step : real;
 tick_label : string[8];
 decimal_places : integer;
begin { procedure tweak_draw_ticks }

 xdotnum := 4;
 ydotnum := 4;

 { find tick step size }
 if (abs (tweak_plot_max [tick_axis] - tweak_plot_min [tick_axis]) >= 1.0)
```

```
 then
  tick_step := exp (ln (10.0) *
           int (ln (abs (tweak_plot_max [tick_axis]
           - tweak_plot_min [tick_axis])) / ln (10.0)))
else
 tick_step := exp (ln (10.0) *
          int (-1.0 + ln (abs (tweak_plot_max [tick_axis]
          - tweak_plot_min [tick_axis])) / ln (10.0)));
tick_mark := int (tweak_plot_min [tick_axis] / tick_step) * tick_step;
if (tick_mark < tweak_plot_min [tick_axis]) then
 tick_mark := tick_mark + tick_step;
decimal_places := trunc (-ln (tick_step) / ln (10.0));
if ((tick_mark + (2.0 * tick_step)) >= tweak_plot_max [tick_axis]) then
 begin
  tick_step := tick_step / 2.0;
  decimal_places := decimal_places + 1;
  tick_mark := int (tweak_plot_min [tick_axis] / tick_step) * tick_step;
  if (tick_mark < tweak_plot_min [tick_axis]) then
   tick_mark := tick_mark + tick_step;
 end
else if ((tick_mark + (5.0 * tick_step)) < tweak_plot_max [tick_axis]) then
 begin
  tick_step := tick_step * 2.0;
  tick_mark := int (tweak_plot_min [tick_axis] / tick_step) * tick_step;
  if (tick_mark < tweak_plot_min [tick_axis]) then
   tick_mark := tick_mark + tick_step;
 end;
if (decimal_places < 0) then
 decimal_places := 0;

{ draw tick marks }
while (tick_mark < tweak_plot_max [tick_axis]) do
 begin
  case code of
   0 : begin { bottom }
       MoveUser (tick_mark, constant_limit);
       Line (Xabsgbl, (Yabsgbl - 1), Xabsgbl, (Yabsgbl - ydotnum));
       if label_ticks then
         begin
          str (tick_mark : (decimal_places + 1) : (decimal_places + 1),
              tick_label);
          MoveUser (tick_mark, constant_limit);
          MoveRel (0, (TextHeight(tick_label) div 2));
          bSetTextStyle (Font, HorizDir, 1);
          SetTextJustify (CenterText, TopText);
          OutText (tick_label);
         end;
      end;
   1 : begin { left }
       MoveUser (constant_limit, tick_mark);
       Line ((Xabsgbl + 1), Yabsgbl, (Xabsgbl + xdotnum), Yabsgbl);
       if label_ticks then
         begin
          str (tick_mark : (decimal_places + 1) : decimal_places,
```

```
                tick_label);
              MoveUser (constant_limit, tick_mark);
              bSetTextStyle (Font, HorizDir, 1);
              SetTextJustify (RightText, CenterText);
              OutText (tick_label);
            end;
          end;

        2 : begin { top }
              MoveUser (tick_mark, constant_limit);
              Line (Xabsgbl, (Yabsgbl + 1), Xabsgbl, (Yabsgbl + ydotnum));
              if label_ticks then
                begin
                  str (tick_mark : (decimal_places + 1) : (decimal_places + 1),
                    tick_label);
                  MoveUser (tick_mark, constant_limit);
                  bSetTextStyle (Font, HorizDir, 1);
                  SetTextJustify (CenterText, BottomText);
                  OutText (tick_label);
                end;
            end;

        3 : begin { right }
              MoveUser (constant_limit, tick_mark);
              Line ((Xabsgbl - 1), Yabsgbl, (Xabsgbl - xdotnum), Yabsgbl);
              if label_ticks then
                begin
                  str (tick_mark : (decimal_places + 1) : decimal_places,
                    tick_label);
                  MoveUser (constant_limit, tick_mark);
                  MoveRel (2, 0);
                  bSetTextStyle (Font, HorizDir, 1);
                  SetTextJustify (LeftText, CenterText);
                  OutText (tick_label);
                end;
            end;
      end; { case code }
      tick_mark := tick_mark + tick_step;
    end;
end;  { procedure tweak_draw_ticks }


procedure tweak_draw_axes;
begin { procedure tweak_draw_axes }
  { draw box }
  DefineWindow (80, 10, 920, 990);
  DefineUserCoordinates (-1.0, 1.0, 1.0, -1.0);
  MoveUser (0, -1.0);
  SetColor (Green);
  DrawLineUser (-1.0, -1.0);
  DrawLineUser (-1.0, 0.0);
  SetColor (Blue);
  DrawLineUser (-1.0, 1.0);
  DrawLineUser (0, 1.0);
```

```
  SetColor (LightGray);
  DrawLineUser (1.0, 1.0);
  DrawLineUser (1.0, 0.0);
  SetColor (Red);
  DrawLineUser (1.0, -1.0);
  DrawLineUser (0, -1.0);
  SetColor (Cyan);
  DrawLineUser (0, 1.0);
  MoveUser (-1.0, 0.0);
  DrawLineUser (1.0, 0.0);
  txs_label_axes (detached_electron_signal);
  txs_label_axes (laser_power);
  txs_label_axes (dissociated_radical_signal);
  txs_label_axes (normalized_signal);

end;  { procedure tweak_draw_axes }

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

procedure tweak_y_ticks (y_data : integer);
begin { procedure tweak_y_ticks }
  { y ticks }
  if (y_data = dissociated_radical_signal) then
          tweak_draw_ticks (dissociated_radical_signal,
                  -1, 1, TRUE)
  else if (y_data = normalized_signal) then
              tweak_draw_ticks (normalized_signal,
              (tweak_limit + 1), 3, TRUE)
  else if (y_data = detached_electron_signal) then
              tweak_draw_ticks (detached_electron_signal,
              -1, 1, TRUE)
  else if (y_data = laser_power) then
              tweak_draw_ticks (laser_power,
              (tweak_limit + 1), 3, TRUE);
end;  { procedure tweak_y_ticks }

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

procedure tweak_replot (y_data, tweak_count : integer; visible : boolean);
var
  i : integer;
begin { procedure tweak_replot }
  if y_data = dissociated_radical_signal then
    begin
      DefineWindow (80, 10, 500, 500);
      SetColor (Blue);
      SetFillStyle (SolidFill, Blue);
    end
  else if y_data = normalized_signal then
    begin
      DefineWindow (500, 10, 920, 500);
      SetColor (LightGray);
      SetFillStyle (SolidFill, LightGray);
    end
```

```
  else if y_data = detached_electron_signal then
   begin
    DefineWindow (80, 500, 500, 990);
    SetColor (Green);
    SetFillStyle (SolidFill, Green);
   end
  else if y_data = laser_power then
   begin
    DefineWindow (500, 500, 920, 990);
    SetColor (Red);
    SetFillStyle (SolidFill, Red);
   end;
  DefineUserCoordinates (-1, tweak_plot_max [y_data],
                (tweak_limit + 1), tweak_plot_min [y_data]);
  if visible then
   begin
    txs_label_axes (y_data);
    MoveUser (-1, tweak_plot_min [y_data]);
    DrawLineUser (-1, twcak_plot_max [y_data]);
    DrawLineUser ((tweak_limit + 1), tweak_plot_max [y_data]);
   end
  else
   begin
    SetColor (Black);
    SetFillStyle (SolidFill, Black);
   end;
  tweak_y_ticks (y_data);

  MoveUser (0, tweak_plot_min [y_data]);
  for i := 1 to tweak_count do
   begin
    DrawBarUser (i, tweak_data^ [i, y_data]);
    MoveUser (i, tweak_plot_min [y_data]);
   end;

end;  { procedure tweak_replot }

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

procedure tweak_replot_all (tweak_count : integer);
begin
  tweak_replot (detached_electron_signal, tweak_count, TRUE);
  tweak_replot (dissociated_radical_signal, tweak_count, TRUE);
  tweak_replot (normalized_signal, tweak_count, TRUE);
  tweak_replot (laser_power, tweak_count, TRUE);
end;
{ ********************************************************************** }

procedure tweak_plot_point (tweak_count, y_data : integer);
begin
  if (tweak_count = 1) then
   begin
    tweak_min [y_data] :=
                tweak_data^ [tweak_count, y_data] - 1.0;
```

```
    tweak_max [y_data] :=
                tweak_data^ [tweak_count, y_data] + 1.0;

    tweak_plot_max [y_data] :=
          tweak_max [y_data] +
          0.1 * (tweak_max [y_data] -
          tweak_min [y_data]);

    tweak_plot_min [y_data] :=
          tweak_min [y_data] -
          0.1 * (tweak_max [y_data] -
          tweak_min [y_data]);

    tweak_thresh_max [y_data] :=
           tweak_max [y_data] +
           0.05 * (tweak_max [y_data] -
           tweak_min [y_data]);

    tweak_thresh_min [y_data] :=
           tweak_min [y_data] -
           0.05 * (tweak_max [y_data] -
           tweak_min [y_data]);
  end;

if (tweak_data^ [tweak_count, y_data] > tweak_thresh_max [y_data]) then
  begin
    tweak_replot (y_data, (tweak_count - 1), FALSE);
    tweak_max [y_data] := tweak_data^ [tweak_count, y_data];
    tweak_plot_max [y_data] := tweak_max [y_data] +
                (0.1 * (tweak_max [y_data] - tweak_min [y_data]));
    tweak_thresh_max [y_data] := tweak_max [y_data] +
                 (0.05 * (tweak_max [y_data] - tweak_min [y_data]));
    tweak_replot (y_data, tweak_count, TRUE);
  end
else if (tweak_data^ [tweak_count, y_data] < tweak_thresh_min [y_data]) then
  begin
    tweak_replot (y_data, (tweak_count - 1), FALSE);
    tweak_min [y_data] := tweak_data^ [tweak_count, y_data];
    tweak_plot_min [y_data] := tweak_min [y_data] -
                (0.1 * (tweak_max [y_data] - tweak_min [y_data]));
    tweak_thresh_min [y_data] := tweak_min [y_data] -
                 (0.05 * (tweak_max [y_data] - tweak_min [y_data]));
    tweak_replot (y_data, tweak_count, TRUE);
  end
else
  begin
    if y_data = dissociated_radical_signal then
      begin
        DefineWindow (80, 10, 500, 500);
        SetColor (Blue);
        SetFillStyle (SolidFill, Blue);
      end
    else if y_data = normalized_signal then
      begin
```

```
        DefineWindow (500, 10, 920, 500);
        SetColor (LightGray);
        SetFillStyle (SolidFill, LightGray);
       end
     else if y_data = detached_electron_signal then
       begin
        DefineWindow (80, 500, 500, 990);
        SetColor (Green);
        SetFillStyle (SolidFill, Green);
       end
     else if y_data = laser_power then
       begin
        DefineWindow (500, 500, 920, 990);
        SetColor (Red);
        SetFillStyle (SolidFill, Red);
       end;
     DefineUserCoordinates (-1, tweak_plot_max [y_data],
                    (tweak_limit + 1), tweak_plot_min [y_data]);
     tweak_y_ticks (y_data);
     MoveUser (tweak_count - 1, tweak_plot_min [y_data]);
     DrawBarUser (tweak_count, tweak_data^ [tweak_count, y_data]);

   end;
end;  { procedure tweak_plot_point }

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

end. { unit twkplot }
```

2.15 txs_exp.pas

```
{ txs_exp.pas                                          }

{ contents :                                           }

{ procedure read_LeCroy_ADC (var data_point : real_column_array)        }
{     read and average data from LeCroy ADC over 'shots_per_point' counts }
{     without controlling laser                                  }

{ procedure manual_experiment                                    }
{     move laser from computer keyboard, also read LeCroy ADC          }

{ procedure obtain_zeros                                         }
{     obtain and store pedestal values of LeCroy ADC channels          }

{ procedure save_data                                            }
{     save data to individual '.***' file, (extension depends on type of  }
{     experiment) append scan parameter info to one 'date.log' file      }
{     per day                                                    }

{ function pause_menu : boolean                                  }
```

```
{     menu of things to do when experiment is paused                    }

{ function pause_experiment (var txs_count : integer) : boolean         }
{     pauses, cleans up, restarts experiment                            }

{ procedure initialize_for_txs_experiment                               }
{     initializes all variables that must be fresh each time            }

{ procedure run_txs_experiment                                          }
{     data collection and display with computer controlled              }
{     LeCroy ADC and laser                                              }

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

unit txs_exp;

{$O+}
{$F+}

interface

uses globals,txsparam,plotutil,txsplot,twkplot,dyelaser,
   gpibutil,grafutil,graph,utility,tpdecl,codes,CRT,DOS;

procedure read_LeCroy_ADC (var data_point : txs_real_column_array);
procedure manual_experiment;
procedure obtain_zeros;
function pause_menu : boolean;
function pause_experiment (var txs_count : integer) : boolean;
procedure initialize_for_txs_experiment;
procedure run_txs_experiment;

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

implementation

{$F-}

{$I camturbo.v4}

{$F+}

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

procedure read_LeCroy_ADC (var data_point : txs_real_column_array);
var
 D, Q, X, txs_count : integer;
 normalized_laser_power, normalized_detached_electron_signal : real;
begin { procedure read_LeCroy_ADC }
 data_point [detached_electron_signal] := 0.0;
 data_point [laser_power] := 0.0;
 data_point [dissociated_radical_signal] := 0.0;
 data_point [normalized_signal] := 0.0;
```

```
CAMCL (CRIACL);
CAMCL (dataway_z);

for txs_count := 1 to txs_stp.shots_per_point do
 begin
  repeat
   CAMI (LeCroy_ADC, test_LAM,
       txs_stp.LeCroy_ADC_channel [laser_power], D, Q, X);
   delay (1);
  until (Q = 1);
  CAMI (LeCroy_ADC, read_channel,
     txs_stp.LeCroy_ADC_channel [detached_electron_signal], D, Q, X);
  data_point [detached_electron_signal] :=
        data_point [detached_electron_signal] + D;
  CAMI (LeCroy_ADC, read_channel,
     txs_stp.LeCroy_ADC_channel [laser_power], D, Q, X);
  data_point [laser_power] :=
        data_point [laser_power] + D;
  CAMI (LeCroy_ADC, read_channel,
     txs_stp.LeCroy_ADC_channel [dissociated_radical_signal], D, Q, X);
  data_point [dissociated_radical_signal] :=
        data_point [dissociated_radical_signal] + D;

  CAMCL (dataway_c);
 end;

CAMCL (dataway_z);
CAMCL (dataway_i);

data_point [detached_electron_signal] :=
      data_point [detached_electron_signal] / txs_stp.shots_per_point;
data_point [laser_power] :=
      data_point [laser_power] / txs_stp.shots_per_point;
data_point [dissociated_radical_signal] :=
      data_point [dissociated_radical_signal] / txs_stp.shots_per_point;

normalized_detached_electron_signal := data_point [detached_electron_signal]
                        - txs_stp.zero [detached_electron_signal];
if normalized_detached_electron_signal < 1.0 then
                    normalized_detached_electron_signal := 1.0;

normalized_laser_power := data_point [laser_power] - txs_stp.zero [laser_power];
if normalized_laser_power < 1.0 then normalized_laser_power := 1.0;

data_point [normalized_signal] :=        1000000 *
                    (data_point [dissociated_radical_signal]
                    - txs_stp.zero [dissociated_radical_signal])
                     / normalized_detached_electron_signal
                         / normalized_laser_power;
end; {procedure read_LeCroy_ADC }

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

procedure initialize_for_tweaking;
```

```
var
 i, j : integer;
begin { procedure initialize_for_tweaking }

 new (tweak_data);

 tweak_limit := 100;
 if ask_for_integer (tweak_limit,
              'How many cycles to tweak for? [Min 5, Max 200] : ',
               5, max_tweak_limit) then ;

 for i := 1 to tweak_limit do
  for j := 1 to max_txs_column do
   tweak_data^ [i, j] := 0.0;

 tweak_max [detached_electron_signal]  := 1.0;
 tweak_min [detached_electron_signal]  := 0.0;

 tweak_max [laser_power]            := 1.0;
 tweak_min [laser_power]            := 0.0;

 tweak_max [dissociated_radical_signal] := 1.0;
 tweak_min [dissociated_radical_signal] := 0.0;

 tweak_max [normalized_signal]      := 1.0;
 tweak_min [normalized_signal]      := 0.0;

 for i := 0 to max_txs_column do
  begin
   tweak_plot_max [i]  := tweak_max [i] +
                0.1 * (tweak_max [i] - tweak_min [i]);
   tweak_plot_min [i]  := tweak_min [i] -
                0.1 * (tweak_max [i] - tweak_min [i]);
   tweak_thresh_max [i] := tweak_max [i] +
                0.05 * (tweak_max [i] - tweak_min [i]);
   tweak_thresh_min [i] := tweak_min [i] -
                0.05 * (tweak_max [i] - tweak_min [i]);
  end;
  writeln;  write ('Hit <enter> when ready to tweak ....');
  readln;
end;  { procedure initialize_for_tweaking }

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

procedure tweak;
var
 ch : char;
 y, shot_count : integer;
 D, Q, X : integer;
 normalized_detached_electron_signal, normalized_laser_power : real;
begin { procedure run_experiment }

 initialize_for_tweaking;
```

```
SetGraphMode (GraphMode);
tweak_draw_axes;

tweak_count := 0;

CAMCL (CRIACL);
CAMCL (dataway_i);

repeat { until user quits or tweak_data array is full }

 tweak_count := tweak_count + 1;

 if tweak_count > 1 then
  begin
   tweak_plot_point ((tweak_count - 1), detached_electron_signal);
   tweak_plot_point ((tweak_count - 1), laser_power);
   tweak_plot_point ((tweak_count - 1), dissociated_radical_signal);
   tweak_plot_point ((tweak_count - 1), normalized_signal);
  end;

 CAMCL (dataway_z);

 shot_count := 0;
 while (shot_count < txs_stp.shots_per_point) do
  begin
   { wait for LAM from LeCroy ADC }
   repeat
    CAMI (LeCroy_ADC, test_LAM, txs_stp.LeCroy_ADC_channel [laser_power], D, Q, X);
    delay (1); { give LeCroy ADC more time to set a LAM }

    { check keyboard for Q (for quit) }
    if keypressed then
     begin
      ch := Readkey;
      if (ch = ESC) or (upcase (ch) = 'Q') then
       begin
        CAMCL (dataway_c);
        CAMCL (dataway_z);
        RestoreCRTMode;
        GraphicsOn := FALSE;
        exit;
       end;
     end;
   until (Q = 1); { LAM from LeCroy ADC }

   { read data from ADCs }
   CAMI (LeCroy_ADC, read_channel,
      txs_stp.LeCroy_ADC_channel [detached_electron_signal], D, Q, X);
   tweak_data^ [tweak_count, detached_electron_signal] :=
      tweak_data^ [tweak_count, detached_electron_signal] + D;
   CAMI (LeCroy_ADC, read_channel,
      txs_stp.LeCroy_ADC_channel [laser_power], D, Q, X);
   tweak_data^ [tweak_count, laser_power] :=
      tweak_data^ [tweak_count, laser_power] + D;
```

```
      CAMI (LeCroy_ADC, read_channel,
          txs_stp.LeCroy_ADC_channel [dissociated_radical_signal], D, Q, X);
      tweak_data^ [tweak_count, dissociated_radical_signal] :=
          tweak_data^ [tweak_count, dissociated_radical_signal] + D;

      { clear all ADCs and LAM }
      CAMCL (dataway_c);
      shot_count := shot_count + 1;
    end;

    tweak_data^ [tweak_count, detached_electron_signal] :=
        tweak_data^ [tweak_count, detached_electron_signal]
            / txs_stp.shots_per_point
            - txs_stp.zero [detached_electron_signal];
    tweak_data^ [tweak_count, laser_power] :=
        tweak_data^ [tweak_count, laser_power] / txs_stp.shots_per_point
            - txs_stp.zero [laser_power];
    tweak_data^ [tweak_count, dissociated_radical_signal] :=
            tweak_data^ [tweak_count, dissociated_radical_signal]
            / txs_stp.shots_per_point
            - txs_stp.zero [dissociated_radical_signal];
    normalized_detached_electron_signal :=
            tweak_data^ [tweak_count, detached_electron_signal];
    if normalized_detached_electron_signal < 1.0 then
            normalized_detached_electron_signal := 1.0;
    normalized_laser_power :=
            tweak_data^ [tweak_count, laser_power];
    if normalized_laser_power < 1.0 then
            normalized_laser_power := 1.0;
    tweak_data^ [tweak_count, normalized_signal] :=
            1000000 * tweak_data^ [tweak_count, dissociated_radical_signal]
                / normalized_detached_electron_signal
                    / normalized_laser_power;

  until tweak_count = tweak_limit;

  tweak_plot_point (tweak_count, detached_electron_signal);
  tweak_plot_point (tweak_count, laser_power);
  tweak_plot_point (tweak_count, dissociated_radical_signal);
  tweak_plot_point (tweak_count, normalized_signal);

  CAMCL (dataway_z);
  CAMCL (dataway_i);

  set_done_flag;
  ch := Readkey;

  dispose (tweak_data);

  RestoreCRTMode;

end; { procedure tweak }

{ ************************************************************************** }
```

```
procedure manual_experiment;
var
 ch : char;
 lambda, old_lambda : real;
 y, i : integer;
 data_point : txs_real_column_array;
begin { procedure manual_experiment }
 set_gpib_remote (TRUE);
 lambda := get_laser_wavelength (dissociation_dye_laser);
 set_gpib_remote (FALSE);
 window (1, 1, 36, 24); clrscr; textcolor (15);
 writeln ('C : Change wavelength to ...');
 writeln ('+ : Wavelength + step');
 writeln ('- : Wavelength - step');
 writeln ('R : Read ADC'); writeln;
 writeln ('T : Tweak');
 writeln ('Z : Obtain Zeros');
 writeln;
 writeln ('Q : Quit');
 writeln;
 writeln ('------------------------------'); writeln;
 textcolor(14); y := wherey;
 repeat
  window (1, y, 80, 24); clrscr;
  set_gpib_remote (TRUE); writeln;
  lambda := get_laser_wavelength (dissociation_dye_laser);
  writeln ('Dissociation dye laser wavelength is ', lambda:5:3);
  writeln; set_gpib_remote (FALSE);
  write ('? > '); ch := Readkey; ch := upcase (ch); writeln (ch); writeln;
  case ch of
    'C' : begin
          old_lambda := lambda;
          if ask_for_real (lambda, 'New Wavelength? > ', 200, 900) then
            begin
             set_gpib_remote (TRUE);
             if txs_stp.doubling then
               begin
                writeln;
                writeln ('Moving wavelength slowly since Autotracker',
                      ' is in use!');
                writeln;
                writeln ('      [ESC to escape, Alt/O to override]');
                move_dye_laser_slowly_since_doubling
                  (dissociation_dye_laser, old_lambda, lambda);
               end;
             wait_for_laser_ready (dissociation_dye_laser);
             set_gpib_remote (FALSE);
            end;
         end;
    '+' : begin
          lambda := lambda + txs_stp.delta_lambda;
          set_gpib_remote (TRUE);
          move_dye_laser (dissociation_dye_laser, lambda);
```

```
            wait_for_laser_ready (dissociation_dye_laser);
            set_gpib_remote (FALSE);
           end;
        '-' : begin
            lambda := lambda - txs_stp.delta_lambda;
            set_gpib_remote (TRUE);
            move_dye_laser (dissociation_dye_laser, lambda);
            wait_for_laser_ready (dissociation_dye_laser);
            set_gpib_remote (FALSE);
           end;
        'R' : begin
            read_LeCroy_ADC (data_point);
            write ('                RAW  -   ');
            writeln ('ZERO =   ACTUAL');
            writeln ('  ', txs_axis_label [detached_electron_signal],' : ',
                  data_point [detached_electron_signal]:9:3,' - ',
                  txs_stp.zero [detached_electron_signal]:9:3,' = ',
                  (data_point [detached_electron_signal] -
                  txs_stp.zero [detached_electron_signal]):9:3);
            writeln ('     ',txs_axis_label [laser_power],' : ',
                  data_point [laser_power]:9:3,' - ',
                  txs_stp.zero [laser_power]:9:3,' = ',
                  (data_point [laser_power] -
                  txs_stp.zero [laser_power]):9:3);
            writeln ('   ',txs_axis_label [dissociated_radical_signal], ' : ',
                  data_point [dissociated_radical_signal]:9:3,' - ',
                  txs_stp.zero [dissociated_radical_signal]:9:3,' = ',
                  (data_point [dissociated_radical_signal] -
                  txs_stp.zero [dissociated_radical_signal]):9:3);
            write ('Normalized Signal : ');
            writeln ('                  ',
                        data_point [normalized_signal]:9:3);
            writeln;
            write  ('                                ');
            write ('press < enter > to continue ...');
            readln;
           end;
           'T' : begin
            tweak;
            ch := 'Q';
           end;
           'Z' : obtain_zeros;
      end; { case ch }
   until (ch = 'Q');
   show_txs_parameters;
  end;  { procedure manual_experiment }

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

  procedure obtain_zeros;
  var
   D, Q, X, txs_count : integer;
   temp_zero : txs_real_column_array;
  begin { procedure obtain_zeros }
```

```
if ask_for_boolean ('Obtain new zeros? : ') then
 begin
  repeat { until zeros are satisfactory to user }
   temp_zero [detached_electron_signal] := 0.0;
   temp_zero [laser_power] := 0.0;
   temp_zero [dissociated_radical_signal] := 0.0;

   CAMCL (CRIACL);
   CAMCL (dataway_z);

   for txs_count := 1 to txs_stp.zero_average_shot_number do
    begin
     repeat
      CAMI (LeCroy_ADC, test_LAM,
          txs_stp.LeCroy_ADC_channel [detached_electron_signal], D, Q, X);
      delay (1);
     until (Q = 1);
     CAMI (LeCroy_ADC, read_channel,
        txs_stp.LeCroy_ADC_channel [detached_electron_signal], D, Q, X);
     temp_zero [detached_electron_signal] :=
           temp_zero [detached_electron_signal] + D;
     CAMI (LeCroy_ADC, read_channel,
        txs_stp.LeCroy_ADC_channel [laser_power], D, Q, X);
     temp_zero [laser_power] :=
           temp_zero [laser_power] + D;
     CAMI (LeCroy_ADC, read_channel,
        txs_stp.LeCroy_ADC_channel [dissociated_radical_signal], D, Q, X);
     temp_zero [dissociated_radical_signal] :=
           temp_zero [dissociated_radical_signal] + D;

     CAMCL (dataway_c);
    end;

   CAMCL (dataway_z);
   CAMCL (dataway_i);

   temp_zero [detached_electron_signal] :=
                 temp_zero [detached_electron_signal]
                 / txs_stp.zero_average_shot_number;
   temp_zero [laser_power] :=
                 temp_zero [laser_power]
                 / txs_stp.zero_average_shot_number;
   temp_zero [dissociated_radical_signal] :=
                 temp_zero [dissociated_radical_signal]
                 / txs_stp.zero_average_shot_number;

   writeln;  writeln ('New zeros:');  writeln;
   writeln (txs_axis_label [detached_electron_signal], ' : ',
   temp_zero [detached_electron_signal]:5:3);
   writeln (txs_axis_label [laser_power], ' : ',
   temp_zero [laser_power]:5:3);
   writeln (txs_axis_label [dissociated_radical_signal], ' : ',
   temp_zero [dissociated_radical_signal]:5:3);  writeln;
```

```
    until ask_for_boolean ('Accept new zeros? : ');

    new_zeros_obtained := true;

    txs_stp.zero [detached_electron_signal] :=
                temp_zero [detached_electron_signal];
    txs_stp.zero [laser_power] :=
                temp_zero [laser_power];
    txs_stp.zero [dissociated_radical_signal] :=
                temp_zero [dissociated_radical_signal];
    show_txs_parameters;
   end; { if obtain new zeros}
end; {procedure obtain_zeros}

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

function pause_menu : boolean;
{ returns TRUE if experiment is to be continued; }
{ returns FALSE if experiment is to be stopped.  }
var
 ch : char;
begin { function pause_menu }
 show_txs_parameters;
 repeat
  window (1, 1, 36, 24);
  clrscr;
  writeln ('M : Move laser & read ADC'); writeln;
  writeln ('C : Continue experiment'); writeln;
  writeln ('S : Stop experiment'); writeln;
  write ('? > '); ch := Readkey; ch := upcase (ch); writeln (ch);
  case ch of
   'M' : manual_experiment;
   'C' : pause_menu := TRUE;
   'S' : pause_menu := FALSE;
  end; { case ch }
 until ((ch = 'C') or (ch = 'S'));
end; { function pause_menu }

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

function pause_experiment (var txs_count : integer) : boolean;
{ returns TRUE if experiment is to be continued; }
{ returns FALSE if experiment is to be stopped.  }
var
 end_of_scan : real;
begin { function pause_experiment }
 txs_count := txs_count - 1;
 set_gpib_remote (FALSE);
 RestoreCRTMode;
 if pause_menu then
  begin
   pause_experiment := TRUE;
  end
 else
```

```
  begin
   pause_experiment := FALSE;
   end_of_scan := txs_stp.start_lambda + (txs_count - 1) * txs_stp.delta_lambda;
   if (txs_stp.start_lambda < txs_stp.end_lambda) then
    begin
     max [laser_wavelength] := end_of_scan;
     min [laser_wavenumber] := wl2wn (end_of_scan);
    end
   else
    begin
     min [laser_wavelength] := end_of_scan;
     max [laser_wavenumber] := wl2wn (end_of_scan);
    end;
   plot_max [laser_wavelength]  := max [laser_wavelength]
                          + 0.1 * (max [laser_wavelength]
                          - min [laser_wavelength]);

   plot_min [laser_wavelength]  := min [laser_wavelength]
                          - 0.1 * (max [laser_wavelength]
                          - min [laser_wavelength]);

   plot_max [laser_wavenumber]  := max [laser_wavenumber]
                          + 0.1 * (max [laser_wavenumber]
                          - min [laser_wavenumber]);

   plot_min [laser_wavenumber]  := min [laser_wavenumber]
                          - 0.1 * (max [laser_wavenumber]
                          - min [laser_wavenumber]);
   save_data_file;
   writeln;
   if ask_for_boolean ('Plot Screen Display ? ') then
    begin
     dump_txs_experiment_screen;
    end;
   exit;
  end;
 set_gpib_remote (TRUE);
 move_dye_laser (dissociation_dye_laser, txs_data^ [txs_count, laser_wavelength]);
 SetGraphMode (GraphMode);
 txs_draw_axes;
 txs_replot (laser_power, txs_count, TRUE);
 txs_replot (dissociated_radical_signal, txs_count, TRUE);
 txs_replot (detached_electron_signal, txs_count, TRUE);
 txs_replot (normalized_signal, txs_count, TRUE);
 wait_for_laser_ready (dissociation_dye_laser);
end;  { function pause_experiment }

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

procedure initialize_for_txs_experiment;
var
 i, j : integer;
begin { procedure initialize_for_txs_experiment }
 for i := 1 to max_txs_data do
```

```
  for j := 1 to max_txs_column do
    txs_data^ [i, j] := 0.0;

plot_little_crosses := (abs ((txs_stp.end_lambda - txs_stp.start_lambda) /
                txs_stp.delta_lambda) <= 200.0);

ready_for_etalon_scan := FALSE;

x_axis := laser_wavelength;

set_gpib_remote (TRUE);
if grating_cal_mismatch then
  begin
    write ('Problem with dye laser ');
    write ('communication, grating ');
    writeln ('calibration mismatch!');  writeln;
    writeln ('Exiting program now.');
    set_gpib_remote (FALSE);
    halt;
  end;
if txs_stp.doubling then
  begin
    writeln ('Have patience, moving to initial wavelength');
    writeln ('slowly since autotracker is in operation.');
    move_dye_laser_slowly_since_doubling
     (dissociation_dye_laser, get_laser_wavelength (dissociation_dye_laser),
      txs_stp.start_lambda);
  end;
set_gpib_remote (FALSE);

if (txs_stp.start_lambda < txs_stp.end_lambda) then
  begin
    max [laser_wavelength] := txs_stp.end_lambda;
    min [laser_wavelength] := txs_stp.start_lambda;
    min [laser_wavenumber] := wl2wn (txs_stp.end_lambda);
    max [laser_wavenumber] := wl2wn (txs_stp.start_lambda);
  end
else
  begin
    max [laser_wavelength] := txs_stp.start_lambda;
    min [laser_wavelength] := txs_stp.end_lambda;
    min [laser_wavenumber] := wl2wn (txs_stp.start_lambda);
    max [laser_wavenumber] := wl2wn (txs_stp.end_lambda);
  end;

max [detached_electron_signal]  := 1.0;
min [detached_electron_signal]  := 0.0;

max [laser_power]           := 1.0;
min [laser_power]           := 0.0;

max [dissociated_radical_signal] := 1.0;
min [dissociated_radical_signal] := 0.0;
```

```
  max [normalized_signal]        := 1.0;
  min [normalized_signal]        := 0.0;

  max [iodine_reference]     := 1.0;
  min [iodine_reference]     := 0.0;

  for i := 0 to max_txs_column do
   begin
    plot_max [i]   := max [i] +  0.1 * (max [i] - min [i]);
    plot_min [i]   := min [i] -  0.1 * (max [i] - min [i]);
    thresh_max [i] := max [i] + 0.05 * (max [i] - min [i]);
    thresh_min [i] := min [i] - 0.05 * (max [i] - min [i]);
   end;

  new_zeros_obtained := false;
  obtain_zeros;

  if new_zeros_obtained = true then
   begin
    writeln;  writeln;  writeln;  writeln;  writeln;
    write ('Hit <enter> when ready to scan ....');
    readln;
   end;
end;  { procedure initialize for experiment }

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

procedure run_txs_experiment;
var
 ch : char;
 y, shot_count : integer;
 D, Q, X : integer;
 normalized_detached_electron_signal, normalized_laser_power : real;
begin { procedure run_experiment }
 if ready_for_etalon_scan then
  begin
   writeln;  writeln ('Are you sure you are ready to run ????');
   if not ask_continue then
    begin
     ready_for_etalon_scan := FALSE;
     exit;
    end;
  end;

 initialize_for_txs_experiment;

 SetGraphMode (GraphMode);
 txs_draw_axes;

 txs_count := 0;
 txs_data^ [1, laser_wavelength] := txs_stp.start_lambda;
 txs_data^ [1, laser_wavenumber] := wl2wn (txs_stp.start_lambda);

 CAMCL (CRIACL);
```

```
CAMCL (dataway_i);

set_gpib_remote (TRUE);

repeat { until scan is done or data array is full }

 txs_count := txs_count + 1;
 move_dye_laser (dissociation_dye_laser, txs_data^ [txs_count, laser_wavelength]);

 { while laser is changing wavelength, plot last point }
 if (txs_count > 1) then
  begin
   txs_plot_point ((txs_count - 1), detached_electron_signal);
   txs_plot_point ((txs_count - 1), laser_power);
   txs_plot_point ((txs_count - 1), dissociated_radical_signal);
   txs_plot_point ((txs_count - 1), normalized_signal);
  end;

 wait_for_laser_ready (dissociation_dye_laser);

 { ask dissociation dye laser what its wavelength really is }
 txs_data^ [txs_count, laser_wavelength] :=
          get_laser_wavelength (dissociation_dye_laser);
 txs_data^ [txs_count, laser_wavenumber] :=
          wl2wn (txs_data^ [txs_count, laser_wavelength]);

 CAMCL (dataway_z);

 shot_count := 0;
 while (shot_count < txs_stp.shots_per_point) do
  begin
   { wait for LAM from LeCroy ADC }
   repeat
    CAMI (LeCroy_ADC, test_LAM, txs_stp.LeCroy_ADC_channel [laser_power], D, Q, X);
    delay (1); { give LeCroy ADC more time to set a LAM }

    { check keyboard for P (for pause) }
    if keypressed then
     begin
      ch := Readkey;
      if (upcase (ch) = 'P') then
       begin
        CAMCL (dataway_z); CAMCL (dataway_i); shot_count := 0;
        txs_data^ [txs_count, detached_electron_signal] := 0.0;
        txs_data^ [txs_count, laser_power] := 0.0;
        txs_data^ [txs_count, dissociated_radical_signal] := 0.0;
        if not pause_experiment (txs_count) then
         begin
          show_txs_parameters;
          exit;
         end;
        CAMCL (dataway_c);
       end;
     end;
```

```
    until (Q = 1); { LAM from LeCroy ADC }

    { read data from ADCs }
    CAMI (LeCroy_ADC, read_channel,
        txs_stp.LeCroy_ADC_channel [detached_electron_signal], D, Q, X);
    txs_data^ [txs_count, detached_electron_signal] :=
        txs_data^ [txs_count, detached_electron_signal] + D;
    CAMI (LeCroy_ADC, read_channel,
        txs_stp.LeCroy_ADC_channel [laser_power], D, Q, X);
    txs_data^ [txs_count, laser_power] :=
        txs_data^ [txs_count, laser_power] + D;
    CAMI (LeCroy_ADC, read_channel,
        txs_stp.LeCroy_ADC_channel [dissociated_radical_signal], D, Q, X);
    txs_data^ [txs_count, dissociated_radical_signal] :=
        txs_data^ [txs_count, dissociated_radical_signal] + D;

    { clear all ADCs and LAM }
    CAMCL (dataway_c);
    shot_count := shot_count + 1;
  end;

  txs_data^ [txs_count, detached_electron_signal] :=
      txs_data^ [txs_count, detached_electron_signal] / txs_stp.shots_per_point
          - txs_stp.zero [detached_electron_signal];
  txs_data^ [txs_count, laser_power] :=
      txs_data^ [txs_count, laser_power] / txs_stp.shots_per_point
          - txs_stp.zero [laser_power];
  txs_data^ [txs_count, dissociated_radical_signal] :=
      txs_data^ [txs_count, dissociated_radical_signal] / txs_stp.shots_per_point
          - txs_stp.zero [dissociated_radical_signal];
  normalized_detached_electron_signal :=
          txs_data^ [txs_count, detached_electron_signal];
  if normalized_detached_electron_signal < 1.0 then
          normalized_detached_electron_signal := 1.0;
  normalized_laser_power :=
          txs_data^ [txs_count, laser_power];
  if normalized_laser_power < 1.0 then
          normalized_laser_power := 1.0;
  txs_data^ [txs_count, normalized_signal] :=
          1000000 * txs_data^ [txs_count, dissociated_radical_signal]
              / normalized_detached_electron_signal
                  / normalized_laser_power;

  txs_data^ [(txs_count + 1), laser_wavelength] :=
          txs_stp.start_lambda + (txs_count * txs_stp.delta_lambda);
  txs_data^ [(txs_count + 1), laser_wavenumber] :=
          wl2wn (txs_data^ [(txs_count + 1), laser_wavelength]);

until (((txs_stp.start_lambda < txs_stp.end_lambda) and
     (txs_data^ [(txs_count+1), laser_wavelength] >
     (txs_stp.end_lambda + (txs_stp.delta_lambda / 2.0)))) or
    ((txs_stp.start_lambda > txs_stp.end_lambda) and
     (txs_data^ [(txs_count+1), laser_wavelength] <
     (txs_stp.end_lambda + (txs_stp.delta_lambda / 2.0)))) or
```

```
      (txs_count = max_txs_data));

  txs_plot_point (txs_count, detached_electron_signal);
  txs_plot_point (txs_count, laser_power);
  txs_plot_point (txs_count, dissociated_radical_signal);
  txs_plot_point (txs_count, normalized_signal);

  set_gpib_remote (FALSE);

  CAMCL (dataway_z);
  CAMCL (dataway_i);

  set_done_flag;
  ch := Readkey;

  RestoreCRTMode;
  save_data_file;
  writeln;  writeln;
  if ask_for_boolean ('Output screen display to LaserWriter? ') then
   begin
    dump_txs_experiment_screen;
   end;
  show_txs_parameters;

end;  { procedure run_txs_experiment }

end. { unit txs_exp }
```

2.16 tofms.pas

```
{  tofms.pas

   This unit allows the main FRBM program to take mass spectra as an
   option and stores the mass spectra data files as .tof files with
   associated log files to be stored as .tof files                    }

unit tofms;

{$O+}
{$F+}

interface

uses
  graph,graphadd,grafutil,tofplot,plotutil,globals,utility,DOS,CRT;

type
  integer_24 = array [0..2] of byte;

var
  f, a, d, x, q,
```

```
 record_length : integer;
 done : boolean;
 firsttime : boolean;

procedure initialize_m_s;
procedure setup;
function real_convert (var a : integer_24) : real;
procedure display_data (current_data : boolean);
procedure read_data;
procedure loop_read_data;
procedure checkLAM;
procedure m_s_menu;
procedure dokey(ch : char);
procedure switch_back_to_main_expt;
procedure switch_to_mass_spec;

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

implementation

{$F-}

{$I camturbo.v4}

{$F+}

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

procedure initialize_m_s;
var
 f, a, x, q : integer;
 d : integer;
 ErrorCode : integer;
begin { procedure initialize_m_s }
 new (tof_data);
 data_count := -1;
 mass_spec_delay := 0.0;
 firsttime := TRUE;
 window (1, 1, 80, 24);
 clrscr; writeln; writeln;
 writeln ('   Change over to mass spectra data collection configuration.');
 writeln; writeln;
 writeln ('1. Ensure detector is unblocked and not amplified (turn off');
 writeln ('   detector power supply before removing amplifier !!!).');
 writeln;
 writeln ('2. Take radical dissociation signal from LeCroy channel 2 and plug');
 writeln ('   it in to the DSP 2001AS transient recorder signal-in connector.');
 writeln;
 writeln ('3. Take the trigger from section 2 of the gate generator and');
 writeln ('   plug it in to the DSP 4101 averaging memory external trigger.');
 writeln;
 writeln ('4. After hitting <enter> to continue, select Read from mass');
 writeln ('   spec main menu to initialize mass spec operation.');
 writeln;
```

```
  readln;

  { disable averaging }
  f := 24; a := 1; d := 0;
  CAMO (memory, f, a, d, x, q);

  { enable LAM on CAMAC dataway }
  f := 26; a := 0; d := 0;
  CAMO (memory, f, a, d, x, q);

  { initialize digitizer control register for : }
  {   0 pretrigger samples, 1k record length, 10ns sampling interval }
  digitizer_control := $0018;
  f := 16; a := 0;
  CAMO (digitizer, f, a, digitizer_control, x, q);

  { initialize memory control register for : }
  {   0 TRAQ channels, sweep terminated after each trigger, subtraction mode, }
  {   2's complement arithmetic, stop averaging on overflow, }
  {   external trigger disabled }
  memory_control := $0060;
  f := 16; a := 2;
  CAMO (memory, f, a, memory_control, x, q);

  { initialize memory record length to 1k }
  record_length := $0400;
  f := 16; a := 3;
  CAMO (memory, f, a, record_length, x, q);

  { initialize memory sweeps register to 200 sweeps }
  sweeps_to_average := $00C8;
  f := 17; a := 0; d := $FFFF - sweeps_to_average + 1;
  CAMO (memory, f, a, d, x, q);

  { reset internal LAM }
  f := 10; a := 0; d := 0;
  CAMO (memory, f, a, d, x, q);

  { reset memory and digitizer -- dataway Z }
  f := 1;
  CAMCL (f);
  delay ((record_length div 4000) + 1);
end;  { procedure initialize_m_s }

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

procedure setup;
var
  f, a, q, x : integer;
  d : integer;
  dummy : integer;
  ch : char;
begin { procedure setup }
  clrscr;
```

```
writeln;
writeln ('0 : quit');
writeln;
writeln ('1 : set number of pretrigger samples');
writeln ('   (0/8, 1/8, 2/8, 3/8, 4/8, 5/8, 6/8, 7/8)');
writeln;
writeln ('2 : set record length');
writeln ('   (256, 512, 1024, 2048, 4096)');
writeln;
writeln ('3 : set sampling time interval');
writeln ('   (10ns, 20ns, 50ns, 100ns, 200ns, 500ns, 1us, EXTCLK)');
writeln;
writeln ('4 : set number of sweeps to average');
writeln ('   (0 - 32k)');
writeln;
writeln ('5 : add or subtract data');
writeln;
writeln ('6 : offset binary or twos complement arithmetic');
writeln;
writeln ('7 : stop or continue averaging on arithmetic overflow');

gotoxy (60, 4); write ((digitizer_control and S7):1, '/8');
gotoxy (60, 7);
    write ((S8000 SHR ((((digitizer_control SHR 3) and 7) + 2) mod 8)):1);
gotoxy (60, 10); case ((digitizer_control SHR 6) and 7) of
              0 : write ('10ns');
              1 : write ('20ns');
              2 : write ('50ns');
              3 : write ('100ns');
              4 : write ('200ns');
              5 : write ('500ns');
              6 : write ('1us');
              7 : write ('EXTCLK');
            end; { case sampling interval }
gotoxy (60, 13); write (sweeps_to_average);
gotoxy (60, 16); if ((memory_control and S20) = S20) then
              write ('subtract data')
            else write ('add data');
gotoxy (60, 18); if ((memory_control and S40) = S40) then
              write ('twos complement')
            else write ('offset binary');
gotoxy (60, 20); if ((memory_control and S80) = S80) then
              write ('continue')
            else write ('stop on overflow');

repeat
 gotoxy (1, 22); clreol; gotoxy (1, 23); clreol; gotoxy (1, 24); clreol;
 gotoxy (1, 22); write ('? '); ch := Readkey; ch := upcase (ch);
 writeln (ch);
 case ch of
   '1' : begin
         writeln ('0 : 0/8, 1 : 1/8, 2 : 2/8, 3 : 3/8, ' +
               '4 : 4/8, 5 : 5/8, 6 : 6/8, 7 : 7/8');
         write ('pretrigger samples? '); readln (dummy);
```

```
      if ((dummy >= 0) and (dummy <= 7)) then
       begin
        gotoxy (60, 4); clreol; write (dummy:1, '/8');
        digitizer_control := (digitizer_control and $FFF8) +
                             dummy;
        f := 16; a := 0;
        CAMO (digitizer, f, a, digitizer_control, x, q);
       end;
     end;
'2' : begin
      writeln ('0 = 256, 1 = 512, 2 = 1k, 3 = 2k, 4 = 4k');
      write ('record length? '); readln (dummy);
      if ((dummy >= 0) and (dummy <= 4)) then
       begin
        record_length := ($100 SHL dummy) - 1;
        gotoxy (60, 7); clreol; write ((record_length + 1) :1);
        digitizer_control := (digitizer_control and $FFC7) +
                             (((13 - dummy) mod 8) SHL 3);
        f := 16; a := 0;
        CAMO (digitizer, f, a, digitizer_control, x, q);
        f := 16; a := 3; d := record_length;
        CAMO (memory, f, a, d, x, q);
       end;
     end;
'3' : begin
      writeln ('0 = 10ns, 1 = 20ns, 2 = 50ns, 3 = 100ns, ' +
            '4 = 200ns, 5 = 500ns, 6 = 1us, 7 = EXT');
      write ('sampling interval? '); readln (dummy);
      if ((dummy >= 0) and (dummy <= 7)) then
       begin
        gotoxy (60, 10); clreol;
        case dummy of
           0 : write ('10ns');
           1 : write ('20ns');
           2 : write ('50ns');
           3 : write ('100ns');
           4 : write ('200ns');
           5 : write ('500ns');
           6 : write ('1us');
           7 : write ('EXTCLK');
         end; { case sampling interval }
        digitizer_control := (digitizer_control and $FE3F) +
                             (dummy SHL 6);
        f := 16; a := 0;
        CAMO (digitizer, f, a, digitizer_control, x, q);
       end;
     end;
'4' : begin { set sweeps register }
      write ('average for how many sweeps? ');
      readln (dummy);
      if ((dummy > 0) and (dummy <= $7FFF)) then
       begin
        sweeps_to_average := dummy;
        gotoxy (60, 13); clreol; write (sweeps_to_average :1);
```

```
        f := 17; a := 0; d := ($FFFF - sweeps_to_average + 1);
        CAMO (memory, f, a, d, x, q);
       end;
     end;
'5' : begin
      if ((memory_control and $20) = $20) then
       begin { switch to addition mode }
        memory_control := memory_control and $FFDF;
        gotoxy (60, 16); clreol; write ('add data');
       end
      else
       begin { switch to subtraction mode }
        memory_control := memory_control or $20;
        gotoxy (60, 16); clreol; write ('subtract data');
        if ((memory_control and $40) = 0) then
         begin { also switch to two's complement arithmetic }
          memory_control := memory_control or $40;
          gotoxy (60, 18); clreol; write ('twos complement');
         end;
       end;
      f := 16; a := 2;
      CAMO (memory, f, a, memory_control, x, q);
     end;
'6' : begin
      if ((memory_control and $40) = $40) then
       begin { switch to offset binary arithmetic }
        memory_control := memory_control and $FFBF;
        gotoxy (60, 18); clreol; write ('offset binary');
        if ((memory_control and $20) = $20) then
         begin { also switch to addition mode }
          memory_control := memory_control and $FFDF;
          gotoxy (60, 16); clreol; write ('add data');
         end;
       end
      else
       begin { switch to two's complement arithmetic }
        memory_control := memory_control or $40;
        gotoxy (60, 18); clreol; write ('twos complement');
       end;
      f := 16; a := 2;
      CAMO (memory, f, a, memory_control, x, q);
     end;
'7' : begin
      if ((memory_control and $80) = $80) then
       begin { switch to stop averaging on arithmetic overflow }
        memory_control := memory_control and $FF7F;
        gotoxy (60, 20); clreol; write ('stop on overflow');
       end
      else
       begin { switch to continue averaging on overflow }
        memory_control := memory_control or $80;
        gotoxy (60, 20); clreol; write ('continue');
       end;
      f := 16; a := 2;
```

```
          CAMO (memory, f, a, memory_control, x, q);
         end;
    end; { case ch }
 until (ch = '0');
end;  { procedure setup }

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

function real_convert (var a : integer_24) : real;
begin { function real_convert }
 real_convert := (65536.0 * a[2]) + (256.0 * a[1]) + a[0];
end;  { function real_convert }

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

procedure display_data (current_data : boolean);
var
 i : integer;
 display_data_count : integer;

begin { procedure display_data }
 if current_data then
   begin
    display_data_count := data_count;
   end
 else
   begin
    display_data_count := df_data_count;
   end;
 if (display_data_count = -1) then exit; { no data has been read yet }

 SetGraphMode (GraphMode);
 GraphicsOn := TRUE;
 if (tof_data_max = tof_data_min) then
  begin
   tof_data_max := tof_data_max + 0.5;
   tof_data_min := tof_data_min - 0.5;
  end;
 tof_plot_min [tof_signal] := tof_data_min - 0.05 * (tof_data_max - tof_data_min);
 tof_plot_max [tof_signal] := tof_data_max + 0.05 * (tof_data_max - tof_data_min);;
 tof_plot_min [time] := -0.05 * display_data_count;
 tof_plot_max [time] := 1.05 * display_data_count;

 tof_draw_axes (time, tof_signal);
 SetColor (LightGray);

 MoveUser (0, tof_data^[0]);
 for i := 1 to (display_data_count - 1) do
  begin
   DrawLineUser (i, tof_data^[i]);
  end;

 readln;
 RestoreCRTMode;
```

```
 GraphicsOn := FALSE;
 clrscr;
 if current_data then
  begin
   GoToXY (5, 12);
   save_data_file (tof);
  end;
 writeln;  writeln;
 if ask_for_boolean ('Output mass spectrum to plotter? [Y/N] : ') then
  begin
   dump_tof_experiment_screen (time, tof_signal, display_data_count);
  end;
 TextColor (14);
end;  { procedure display_data }

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

procedure read_data;
type
 ms_data_array = array [0..4095] of integer_24;
var
 f, a, x, q, e : integer;
 d : integer;
 starting_address : integer;
 sweeps_done : integer;
 i, j : integer;
 ms_data : ms_data_array;
 dummy : integer absolute ms_data;
 crate, nob, qbl, ntr : integer;
 maxi : integer;
 left, right, half, frac, sum : real;
 tot, cutoff : real;

begin { procedure read_data }
 starting_address := 0;

 if (record_length > 4095) then
  begin
   writeln ('array space for 4k record length max');
   write ('enter starting address in 4101 memory :');
   readln (starting_address);
  end;

 for i := 0 to record_length do
  for j := 0 to 2 do
   ms_data [i,j] := 0;

 { disable averaging }
 f := 24; a := 1; d := 0;
 CAMO (memory, f, a, d, q, x);

 { read sweep counter }
 f := 1; a := 0; d := 0;
 CAMI (memory, f, a, d, q, x);
```

```
sweeps_done := d - ($FFFF - sweeps_to_average + 1);

{ set address of first data word to starting_address }
f := 16; a := 1; d := starting_address;
CAMO (memory, f, a, d, q, x);

crate := 1; nob := 3; qbl := 1;
if (record_length < 4095) then
 ntr := record_length + 1
else
 ntr := 4096;
DMASET (crate, nob, qbl, ntr);
f := 2; a := 0;
DMAI (memory, f, a, dummy, e);
if (e <> 0) then
 begin writeln ('e = ', e:1); readln; end
else
 begin
  camcyc (ntr); data_count := (ntr - 1);
  writeln ('completed ', ntr:1, ' DMA transfers');
 end;

tof_data^ [0] := -1 * real_convert (ms_data [0]);
tof_data_max := tof_data^ [0];
tof_data_min := tof_data^ [0];
for i := 1 to (data_count - 1) do
 begin
  tof_data^ [i] := -1 * real_convert (ms_data [i]);
  if (tof_data^ [i] > tof_data_max) then
   begin
    tof_data_max := tof_data^ [i];
    maxi := i;
   end
  else if (tof_data^ [i] < tof_data_min) then
   begin
    tof_data_min := tof_data^ [i];
   end;
 end;
for i := 0 to (data_count - 1) do
 begin
  tof_data^ [i] := tof_data^ [i] - tof_data_min;
 end;
tof_data_max := tof_data_max - tof_data_min;
tof_data_min := 0.0;
writeln ('maximum data value = ', tof_data_max :8:0,' at pt # ',maxi:0);
writeln ('minimum data value = ', tof_data_min :8:0);
writeln ('difference = ',tof_data_max - tof_data_min :0:0);

if (tof_data_max = tof_data_min) then
 begin tof_data_max := tof_data_max + 0.5; tof_data_min := tof_data_min - 0.5; end
else { find fwhm of largest peak and area under it }
 begin
  sum := 0.0;
  for i := (data_count - 50) to (data_count - 1) do
```

```
    sum := sum  + tof_data^[i];
  sum := sum/50;  { this is 'baseline' }
  writeln('Largest peak is ',tof_data_max-sum:0:0,' above baseline');
  half := (tof_data_max + sum)/2.0;
  i := maxi;
  repeat
    i := i + 1;
  until ((tof_data^[i] < half) or (i >= data_count));
  if (i < data_count) then
    frac :=  (half - tof_data^[i-1])/(tof_data^[i]-tof_data^[i-1])
  else
    frac := 0.0;
  right := i - 1 + frac;
  i := maxi;
  repeat
    i := i - 1;
  until ((tof_data^[i] < half) or (i <= 1));
  if (i > 1) then
    frac := (half - tof_data^[i+1])/(tof_data^[i]-tof_data^[i+1])
  else
    frac := 0.0;
  left := i + 1 - frac;
  writeln('FWHM = ',right-left:0:2,' channels');
  cutoff := sum + (tof_data_max - sum)*0.1;
  tot := 0.0;
  i := maxi;
  repeat
    i := i - 1;
    tot := tot + tof_data^[i] - sum;
  until ((tof_data^[i] < cutoff) or (i <= 1));
  tot := tot - (tof_data^[i] - sum)/2;
  i := maxi - 1;
  repeat
    i := i + 1;
    tot := tot + tof_data^[i] - sum;
  until ((tof_data^[i] < cutoff) or (i >= data_count));
  tot := tot - (tof_data^[i] - sum)/2;
  writeln('Area to 10% is ',tot:0:0);
  tot := 0.0;
  for i := maxi - 2 to maxi + 2 do
    tot := tot + tof_data^[i] - sum;
  writeln('Area in 5 points is ',tot:0:0);
  tot := tof_data^[maxi] + tof_data^[maxi+1] + tof_data^[maxi-1]
       + (tof_data^[maxi+2] + tof_data^[maxi-2])/2 - 4*sum;
  writeln('Area in 4 points is ',tot:0:0);
end;
readln;

{ reset LAM }
f := 10; a := 0; d := 0;
CAMO (memory, f, a, d, q, x);

{ enable averaging }
{ f := 26; a := 1; d := 0;
```

```
 CAMO (memory, f, a, d, q, x); }
end;  { procedure read_data }

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

procedure loop_read_data;
begin { procedure loop_read_data }
end;  { procedure loop_read_data }

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

procedure checkLAM;
var oldX, oldY : integer;
begin
   { test for internal LAM on }
   f := 8; a := 0; d := 0;
   CAMO (memory, f, a, d, q, x);
   if ((q = 0) and firsttime) then
    begin
     oldX := WhereX; oldY := WhereY;
     gotoxy(1,20); writeln ('internal LAM not on');
     gotoxy(oldX,oldY);
     firsttime := FALSE;
    end;
   if ((q = 1) and (NOT firsttime)) then
    begin
     oldX := WhereX; oldY := WhereY;
     gotoxy(1,20); writeln ('internal LAM is on  ');
     firsttime := TRUE;

     { read LAM register }
     f := 0; a := 0; d := 0;
     CAMI (memory, f, a, d, q, x);
     if ((d and $0400) = $0400) then
      write ('DONE ');
     if ((d and $0800) = $0800) then
      write ('OVERFLOW ');
     if ((d and $0E00) = $0200) then
      write ('F24A1 ');
     writeln;

     { read sweep counter }
     f := 1; a := 0; d := 0;
     CAMI (memory, f, a, d, q, x);
     writeln (d + sweeps_to_average:1, ' sweeps completed');
     gotoxy(oldX,oldY);
    end;
end; { procedure checkLAM }

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

procedure m_s_menu;
begin
   clrscr;
```

```
    writeln ('D : disable external trigger');
    writeln ('E : enable external trigger');
    writeln ('Z : reset and enable external trigger');
    writeln ('R : read data and display on screen');
    writeln ('C : reload sweeps counter');
    writeln ('S : set up control registers');
    writeln ('W : write data to file');
    writeln ('G : get data from file');
    writeln ('Q : quit');
    writeln; write ('? ');
end; { procedure m_s_menu }

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

procedure dokey(ch : char);
begin
   case ch of
      'D' : begin
            { disable external trigger }
            memory_control := memory_control or $0100;
            f := 16; a := 2;
            CAMO (memory, f, a, memory_control, x, q);
           end;
      'E' : begin
            { enable external trigger }
            memory_control := memory_control and $FEFF;
            f := 16; a := 2;
            CAMO (memory, f, a, memory_control, x, q);
           end;
      'Z' : begin
            { disable external trigger }
            memory_control := memory_control or $0100;
            f := 16; a := 2;
            CAMO (memory, f, a, memory_control, x, q);
            { disable averaging }
            f := 24; a := 1; d := 0;
            CAMO (memory, f, a, d, x, q);
            { reset }
            f := 9; a := 0; d := 0;
            CAMO (memory, f, a, d, x, q);
            { wait for averaging memory to clear }
            delay ((record_length div 4000) + 1);
            { enable external trigger }
            memory_control := memory_control and $FEFF;
            f := 16; a := 2;
            CAMO (memory, f, a, memory_control, x, q);
           end;
      'R' : begin
            { disable averaging }
            f := 24; a := 1; d := 0;
            CAMO (memory, f, a, d, x, q);
            read_data;
            { reset internal LAM }
            f := 10; a := 0; d := 0;
```

```
            CAMO (memory, f, a, d, x, q);
            { enable averaging }
            f := 26; a := 1; d := 0;
            CAMO (memory, f, a, d, x, q);
            display_data (TRUE);
          end;
      'C' : begin
            { disable averaging }
            f := 24; a := 1; d := 0;
            CAMO (memory, f, a, d, x, q);
            { reload sweep counter }
            f := 25; a := 1; d := 0;
            CAMO (memory, f, a, d, x, q);
            { reset internal LAM }
            f := 10; a := 0; d := 0;
            CAMO (memory, f, a, d, x, q);
            { enable averaging }
            f := 26; a := 1; d := 0;
            CAMO (memory, f, a, d, x, q);
          end;
      'S' : begin
            { disable averaging }
            f := 24; a := 1; d := 0;
            CAMO (memory, f, a, d, x, q);
            setup;
            { reset internal LAM }
            f := 10; a := 0; d := 0;
            CAMO (memory, f, a, d, x, q);
            { enable averaging }
            f := 26; a := 1; d := 0;
            CAMO (memory, f, a, d, x, q);
          end;
      'W' : save_data_file (tof);
      'G' : begin
            read_data_file (tof);
            delay(500);
            display_data (FALSE);
          end;
      'Q' : done := TRUE;
    end; { case ch }
end; { procedure dokey }

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

procedure switch_back_to_main_expt;
begin { procedure switch_back_to_main_expt }
 clrscr; writeln; writeln;
 writeln ('   Change over to total photodissociation cross section');
 writeln ('   data collection configuration.');
 writeln; writeln;
 writeln ('1. Take radical dissociation signal from the DSP 2001AS');
 writeln ('   transient recorder signal-in connector and plug');
 writeln ('   it in to channel 2 of the LeCroy 2249SG analog to digital');
 writeln ('   converter.');
```

```
  writeln;
  writeln ('2.  Take the trigger from the DSP 4101 averaging memory external');
  writeln ('    trigger and plug it in to section 2 of the gate generator.');
  writeln;
  write ('    Hit <enter> when these steps are complete....');
  readln;
end; { procedure switch_back_to_main_expt }

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

procedure switch_to_mass_spec;
begin { procedure switch_to_mass_spec }
  initialize_m_s;
  done := FALSE;
  m_s_menu;

  repeat
    checkLAM;
    if keypressed then
    begin
      ch := readkey;
      if (ch = #0) then ch := readkey; { function key pressed }
      ch := upcase (ch); writeln (ch);
      dokey(ch);
      m_s_menu;
    end;
  until done;
  dispose (tof_data);
  switch_back_to_main_expt;
end; { procedure switch_to_mass_spec }

end.  { unit tofms }
```

2.17 gpibutil.pas

```
{ gpibutil.pas
  procedures used for GPIB control

************************************************************************** }

unit gpibutil;

{SO+}
{SF+}

interface

uses globals,graph,utility,tpdecl,CRT;

type
  string_80 = string[80];
```

```
 string_8 = string[8];
 string_2 = string[2];

var
 Bd, dye_laser, stanford_box : integer;
 laser_buffer : string_80;
 IBBuf : array[1..$FF] of char;

procedure GPIB_error (call : string_8);
procedure init_GPIB;
procedure set_gpib_remote (remote_on : boolean);

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

implementation

procedure GPIB_error (call : string_8);
begin { procedure GPIB_error }
 if GraphicsOn then
  RestoreCRTMode;
 writeln ('GPIB error on call ', call);
 halt;  { stop the entire program }
end;  { procedure GPIB_error }

{ ************************************************************************** }
{ initialize the GPIB card to talk to dye lasers and stanford boxes  }

procedure init_GPIB;
begin { procedure init_GPIB }

 detachment_dye_laser := -1;
 BdName := 'LASER1 ';
 while (detachment_dye_laser < 0) do
  begin
   detachment_dye_laser := IBFind(BdName);
   if (detachment_dye_laser < 0) then
    if not ask_for_boolean
     ('Detachment Dye Laser IBFind error.  Try again? ') then
     halt;
  end;

 dissociation_dye_laser := -1;
 BdName := 'LASER2 ';
 while (dissociation_dye_laser < 0) do
  begin
   dissociation_dye_laser := IBFind(BdName);
   if (dissociation_dye_laser < 0) then
    if not ask_for_boolean
     ('Dissociation Dye Laser IBFind error.  Try again? ') then
     halt;
  end;

 sb1 := -1;
 BdName := 'SB1   ';
```

```
  while (sb1 < 0) do
   begin
    sb1 := IBFind(BdName);
    if (sb1 < 0) then
     if not ask_for_boolean
      ('Stanford Box #1 IBFind Error.  Try again? ') then
      halt;
   end;

  sb2 := -1;
  BdName := 'SB2    ';
  while (sb2 < 0) do
   begin
    sb2 := IBFind(BdName);
    if (sb2 < 0) then
     if not ask_for_boolean
      ('Stanford Box #2 IBFind error.  Try again? ') then
      halt;
   end;

  Bd := -1;
  BdName := 'GPIB0  ';
  while (Bd < 0) do
   begin
    Bd := IBFind(BdName);
    if (Bd < 0) then
     if not ask_for_boolean ('GPIB0 Board IBFind error.  Try again? ') then
      halt;
   end;

  IBSIC(Bd); { Send Interface Clear }
  if (IBSta < 0) then GPIB_error ('IBSIC');
end;  { procedure init_GPIB }

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

procedure set_gpib_remote (remote_on : boolean);
begin { procedure set_gpib_remote }
  if remote_on then
   IBSRe (Bd, 1)
  else
   IBSRe (Bd, 0);
  if (IBSta < 0) then GPIB_error ('IBSRe');
end;  { procedure set_gpib_remote }

end. { unit gpibutil }
```

2.18 tpdecl.pas

```
unit tpdecl;
```

```
{$L tpib}

interface

Const

 (* GPIB Commands:                                    *)

 UNL = $3f;          (* GPIB unlisten command        *)
 UNT = $5f;          (* GPIB untalk command          *)
 GTL = $01;          (* GPIB go to local             *)
 SDC = $04;          (* GPIB selected device clear   *)
 PPC = $05;          (* GPIB parallel poll configure *)
 GGET = $08;         (* GPIB group execute trigger   *)
 TCT = $09;          (* GPIB take control            *)
 LLO = $11;          (* GPIB local lock out          *)
 DCL = $14;          (* GPIB device clear            *)
 PPU = $15;          (* GPIB parallel poll unconfigure *)
 SPE = $18;          (* GPIB serial poll enable      *)
 SPD = $19;          (* GPIB serial poll disable     *)
 PPE = $60;          (* GPIB parallel poll enable    *)
 PPD = $70;          (* GPIB parallel poll disable   *)

 (* GPIB status bit vector:                           *)

 ERR  = $8000;       (* Error detected               *)
 TIMO = $4000;       (* Timeout                      *)
 UEND = $2000;       (* EOI or EOS detected          *)
 SRQI = $1000;       (* SRQ detected by CIC          *)
 RQS  = $800;        (* Device needs service         *)
 CMPL = $100;        (* I/O completed                *)
 LOK  = $80;         (* Local lockout state          *)
 REM  = $40;         (* Remote state                 *)
 CIC  = $20;         (* Controller-in-Charge         *)
 ATN  = $10;         (* Attention asserted           *)
 TACS = $8;          (* Talker active                *)
 LACS = $4;          (* Listener active              *)
 DTAS = $2;          (* Device trigger state         *)
 DCAS = $1;          (* Device clear state           *)

 (* Error messages returned in global variable IBERR:  *)

 EDVR = 0;           (* DOS error                    *)
 ECIC = 1;           (* Function requires GPIB board to be CIC *)
 ENOL = 2;           (* Write function detected no Listeners   *)
 EADR = 3;           (* Interface board not addressed correctly*)
 EARG = 4;           (* Invalid argument to function call      *)
 ESAC = 5;           (* Function requires GPIB board to be SAC *)
 EABO = 6;           (* I/O operation aborted        *)
 ENEB = 7;           (* Non-existent interface board *)
 EOIP = 10;          (* I/O operation started before previous *)
                     (* operation completed          *)
 ECAP = 11;          (* No capability for intended operation   *)
 EFSO = 12;          (* File system operation error  *)
```

```
  EBUS  = 14;                    (* Command error during device call     *)
  ESTB  = 15;                    (* Serial poll status byte lost         *)
  ESRQ  = 16;              (* SRQ remains asserted                       *)

(* EOS mode bits:                                                        *)

  BIN  = $1000;              (* Eight bit compare                        *)
  XEOS = $800;               (* Send EOI with EOS byte                   *)
  REOS = $400;               (* Terminate read on EOS                    *)


(* Timeout values and meanings:                                         *)

  TNONE   = 0;                     (* Infinite timeout (disabled)      *)
  T10us   = 1;               (* Timeout of 10 us (ideal)               *)
  T30us   = 2;               (* Timeout of 30 us (ideal)               *)
  T100us  = 3;               (* Timeout of 100 us (ideal)              *)
  T300us  = 4;               (* Timeout of 300 us (ideal)              *)
  T1ms    = 5;               (* Timeout of 1 ms  (ideal)               *)
  T3ms    = 6;               (* Timeout of 3 ms  (ideal)               *)
  T10ms   = 7;               (* Timeout of 10 ms  (ideal)              *)
  T30ms   = 8;               (* Timeout of 30 ms  (ideal)              *)
  T100ms  = 9;               (* Timeout of 100 ms (ideal)              *)
  T300ms  = 10;                    (* Timeout of 300 ms (ideal)        *)
  T1s     = 11;              (* Timeout of 1 s   (ideal)               *)
  T3s     = 12;              (* Timeout of 3 s   (ideal)               *)
  T10s    = 13;              (* Timeout of 10 s  (ideal)               *)
  T30s    = 14;              (* Timeout of 30 s  (ideal)               *)
  T100s   = 15;              (* Timeout of 100 s (ideal)               *)
  T300s   = 16;              (* Timeout of 300 s (ideal)               *)
  T1000s  = 17;              (* Timeout of 1000 s (maximum)            *)

(* Miscellaneous:                                                       *)

  S  = S08;                  (* Parallel poll sense bit                *)
  LF = S0A;                  (* ASCII linefeed character               *)

(************************************************************************)

        nbufsize = 7;              (* Length of board/device names -- hard-coded
                                      in TPIB *)
        flbufsize = 50;            (* A generous length for filenames -- the
                                   minimum allowed by the handler is 32.
                                      50 is hard-coded in TPIB *)

(************************************************************************)

Type    nbuf  = array[1..nbufsize]  of char; (*  device/board names   *)
        flbuf = array[1..flbufsize] of char; (*  filenames           *)

(************************************************************************)

(* These three variables are to be accessed directly in application program. *)
```

```
var ibsta : word;                (* status word                  *)
var iberr : word;                (* GPIB error code              *)
var ibcnt : word;                (* number of bytes sent or DOS error *)

(* The following variables may be used directly in your application program. *)
Var
  bname  : nbuf;                 (* board name buffer            *)
  bdname : nbuf;                 (* board or device name buffer  *)
  flname : flbuf;                (* filename buffer              *)

  procedure ibbna (bd:integer;var bname:nbuf);
  procedure ibcac (bd:integer;v:integer);
  procedure ibclr (bd:integer);
  procedure ibcmd (bd:integer; var cmd;cnt:word);
  procedure ibcmda (bd:integer; var cmd;cnt:word);
  procedure ibdiag (bd:integer;var rd;cnt:word);
  procedure ibdma (bd:integer;v:integer);
  procedure ibeos (bd:integer;v:integer);
  procedure ibeot (bd:integer;v:integer);
  function  ibfind (var bdname:nbuf):integer;
  procedure ibgts (bd:integer;v:integer);
  procedure ibist (bd:integer;v:integer);
  procedure ibloc (bd:integer);
  procedure ibonl (bd:integer;v:integer);
  procedure ibpad (bd:integer;v:integer);
  procedure ibpct (bd:integer);
  procedure ibppc (bd:integer;v:integer);
  procedure ibrd  (bd:integer;var rd;cnt:word);
  procedure ibrda (bd:integer;var rd;cnt:word);
  procedure ibrdf (bd:integer;var flname:flbuf);
  procedure ibrpp (bd:integer;var ppr);
  procedure ibrsc (bd:integer;v:integer);
  procedure ibrsp (bd:integer;var spr);
  procedure ibrsv (bd:integer;v:integer);
  procedure ibsad (bd:integer;v:integer);
  procedure ibsic (bd:integer);
  procedure ibsre (bd:integer;v:integer);
  procedure ibstop (bd:integer);
  procedure ibtmo (bd:integer;v:integer);
  procedure ibtrap (mask:word;v:integer);
  procedure ibtrg (bd:integer);
  procedure ibwait (bd:integer;mask:word);
  procedure ibwrt (bd:integer;var wrt;cnt:word);
  procedure ibwrta (bd:integer;var wrt;cnt:word);
  procedure ibwrtf (bd:integer;var flname:flbuf);


implementation

var
       found : integer;      (* flag set after first successful ibfind *)
    our_lcv : integer;       (* local loop control variable     *)

(* The GPIB board functions declared public by TPIB.OBJ:          *)
```

```
  procedure ibbna (bd:integer;var bname:nbuf); external;
  procedure ibcac (bd:integer;v:integer); external;
  procedure ibclr (bd:integer); external;
  procedure ibcmd (bd:integer; var cmd;cnt:word); external;
  procedure ibcmda (bd:integer; var cmd;cnt:word); external;
  procedure ibdiag (bd:integer;var rd;cnt:word); external;
  procedure ibdma (bd:integer;v:integer); external;
  procedure ibeos (bd:integer;v:integer); external;
  procedure ibeot (bd:integer;v:integer); external;
  function  ibfind (var bdname:nbuf):integer; external;
  procedure ibgts (bd:integer;v:integer); external;
  procedure ibist (bd:integer;v:integer); external;
  procedure ibloc (bd:integer); external;
  procedure ibonl (bd:integer;v:integer); external;
  procedure ibpad (bd:integer;v:integer); external;
  procedure ibpct (bd:integer); external;
  procedure ibppc (bd:integer;v:integer); external;
  procedure ibrd  (bd:integer;var rd;cnt:word); external;
  procedure ibrda (bd:integer;var rd;cnt:word); external;
  procedure ibrdf (bd:integer;var flname:flbuf); external;
  procedure ibrpp (bd:integer;var ppr); external;
  procedure ibrsc (bd:integer;v:integer); external;
  procedure ibrsp (bd:integer;var spr); external;
  procedure ibrsv (bd:integer;v:integer); external;
  procedure ibsad (bd:integer;v:integer); external;
  procedure ibsic (bd:integer); external;
  procedure ibsre (bd:integer;v:integer); external;
  procedure ibstop (bd:integer); external;
  procedure ibtmo (bd:integer;v:integer); external;
  procedure ibtrap (mask:word;v:integer); external;
  procedure ibtrg (bd:integer); external;
  procedure ibwait (bd:integer;mask:word); external;
  procedure ibwrt (bd:integer;var wrt;cnt:word); external;
  procedure ibwrta (bd:integer;var wrt;cnt:word); external;
  procedure ibwrtf (bd:integer;var flname:flbuf); external;
begin
        found:=0;          (* initialize successful ibfind flag *)
        ibsta:=0;          (*  initialize global status variables *)
        iberr:=0;
        ibcnt:=0;
        for our_lcv:=1 to nbufsize do       (* blank fill name buffers *)
    begin
                bname[our_lcv]:=' ';
         bdname[our_lcv]:=' ';
    end;
        for our_lcv:=1 to flbufsize do
                flname[our_lcv]:=' ';
end.
```

Appendix B

TPS — The Control and Data Acquisition Program for Photodissociation Dynamics Experiments Involving Time- and Position-Sensitive Detection

This Appendix contains TPS, the data acquisition program used to collect the relative time-of-flight and the measured charge on each of the 6 conductors of the coincidence wedge-and-strip anode used in the time- and position-sensitive detector, for each dissociation event. The program saves all the raw data for later treatment on the Sun workstations, as well as displaying the build-up of dissociation fragment signal on the detector face as the experiment progresses. Routines are included that allow the user to vary all of the critical parameters in data collection (i.e., hardware and software ADC discrimination thresholds) and display (i.e., multiplicative, additive and crosstalk coefficients in the position finding algorithm). Memory management is particularly important in this program; a large heap is desirable so that as much data can be kept in memory as possible, limiting down time that accompanies transfers to disk. Please refer to the Borland Turbo Pascal manuals for information on the use of dynamic memory.

Units that are *identical* in both FRBM and TPS appear in Appendix A will not be repeated here. They are insgrdrv.pas, plotutil.pas, grafutil.pas. codes.pas appears in this program courtesy of Dr. John Price.

1 Description of Files

tps.pas	Contains a list of units to be built into the program, and calls initialization routines followed by the main menu.
inti_ovr.pas	Initializes Turbo Pascal overlay manager to automatically overlay all units in tps.pas marked with the compiler directive $O+.

tps_init.pas	Initializes all parameters and hardware (i.e., the ADC's) used by the program.
tps_mm.pas	Presents the user with an array of menu items, calling the routine corresponding to the selection made by the user.
tps_glob.pas	Declares all variables, constants, data types, strings etc. that are shared among units
insgrdrv.pas	Installs various graphics drivers.
tps_util.pas	This is a catch-all unit which includes somewhat common functions and procedures that can be called from other units. See utility.pas of Appendix A.
codes.pas	ASCII codes are assigned constant designations for readability.
tpsparam.pas	Searches for and reads in a setup file containing initial parameters for experiments. If such a file is not found, it prompts the user for this information and allows such a setup file to then be written.
tps_radc.pas	radc stands for 'Read ADC', and this is primarily the function of this unit.
tps_exp.pas	Controls the actual time and position data acquisition, storage and display.
dfplot.pas	Allows an image of the detector face with the build-up of coincidence photofragment events to be displayed on the computer screen as the experiment is in progress.
tps_dcc.pas	Allows the input of additive and multiplicative factors for the position algorithm.
tps_ctf.pas	Allows the input of cross-talk factors for the position algorithm.
tps_lld.pas	Allows the user to enter new hardware lower level discrimination settings for the ADC channels (upper level hardware discrimination is not possible).

tps_swd.pas Allows the user to enter new lower *and* upper software discrimination settings to guard against spurious ADC readings.

2 Source Code

2.1 tps.pas

```
{ tps.pas                                                    }
{ ************************************************************************** }
{ main program TPS                                           }
program TPS (input, output);
{$F+}
{$M 16000,140000,300000}
uses overlay,init_ovr,tps_init;
{$O tps_init}
{$O tps_glob}
{$O insgrdrv}
{$O tps_util}
{$O plotutil}
{$O grafutil}
{$O codes}
{$O tpsparam}
{$O tps_radc}
{$O tps_exp}
{$O dfplot}
{$O tps_dcc}
{$O tps_ctf}
{$O tps_lld}
{$O tps_swd}
{$O tps_mm}
{ ************************************************************************** }
begin { main program TPS }
 initialize_overlay;
 initialize_tps_program;
end. { main program TPS }
```

2.2 init_ovr.pas

```
{ init_ovr.pas                                    }

{ contents :                                      }

{ procedure initialize_overlay                          }
{     initializes overlay                          }

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

unit init_ovr;

{$F+}

interface

uses overlay;

procedure initialize_overlay;

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

implementation

procedure initialize_overlay;
const
 OvrMaxSize = 40000;
var
 y : integer;
 OvrName : String[79];
 Size : LongInt;
begin
 OvrName := 'tps.ovr';
 repeat
  OvrInit(OvrName);
  if OvrResult = ovrNotFound then
   begin
    writeln ('Overlay file not found: ', Ovrname);
    write ('Enter correct overlay file name: ');
    readln (OvrName);
   end;
 until OvrResult <> ovrNotFound;
 if OvrResult <> ovrOK then
  begin
   writeln ('Overlay Manager Error!');
   if OvrResult = ovrNoMemory then
    begin
     writeln ('Not enought memory for overlay buffer');
    end;
   Halt(1);
  end;
```

```
  OvrSetBuf(OvrMaxSize);
  if OvrResult <> ovrOk then
    begin
     case OvrResult of
       ovrError:    writeln ('Overlay manager error');
       ovrNoMemory: writeln ('Not enough memory for extra overlay buffer size');
     end;
     write ('OvrResult = ', OvrResult, '. Press <enter> to continue ...');
     readln;
    end;
end; { procedure initialize_overlay }

end. { unit init_ovr }
```

2.3 tps_init.pas

```
{ tps_init.pas                                          }

{ contents :                                            }

{ procedure initialize_tps_program                                  }
{     initializes everything (except for overlay manager)           }

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

unit tps_init;

{$O+}
{$F+}

interface

uses tps_glob,tps_util,tpsparam,tps_lld,tps_dcc,insgrdrv,grafutil,tps_mm,CRT,DOS;

procedure initialize_tps_program;

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

implementation

procedure initialize_tps_program;
const
  all = 8;
  upper = 9;
  lower = 10;
var
  y, axis : integer;
begin
  setcbreak (TRUE);
  experiment_type := tps;
```

```
wedge_1_channel := 0;
strip_1_channel := 1;
z_1_channel     := 2;
TAC_channel     := 3;
wedge_2_channel := 0;
strip_2_channel := 1;
z_2_channel     := 2;

InstallScreenGraphicsDriver;

GraphicsOn := FALSE;

window (1, 1, 80, 24); clrscr;

full_parameter_listing := FALSE;
list_gate_settings := FALSE;
initial_read := TRUE;
if not read_set_up then
 begin
  with tps_stp do
   begin
    TAC_delay := 250;
    blocking_strip_full_width := 4;
    Ortec_ADC_channel [wedge_1] := 1;
    Ortec_ADC_channel [strip_1] := 2;
    Ortec_ADC_channel [z_1]     := 3;
    Ortec_ADC_channel [TAC]     := 4;
    Ortec_ADC_channel [wedge_2] := 5;
    Ortec_ADC_channel [strip_2] := 6;
    Ortec_ADC_channel [z_2]     := 7;
    lld_setting [wedge_1] := 500;
    lld_setting [strip_1] := 500;
    lld_setting [z_1]     := 500;
    lld_setting [TAC]     := 500;
    lld_setting [wedge_2] := 500;
    lld_setting [strip_2] := 500;
    lld_setting [z_2]     := 500;
    dmc_setting [wedge_1] := 1;
    dmc_setting [strip_1] := 1;
    dmc_setting [z_1]     := 1;
    dmc_setting [TAC]     := 1;
    dmc_setting [wedge_2] := 1;
    dmc_setting [strip_2] := 1;
    dmc_setting [z_2]     := 1;
    dac_setting [wedge_1]  := 0;
    dac_setting [strip_1]  := 0;
    dac_setting [wedge_2]  := 0;
    dac_setting [strip_2]  := 0;
    ctf_setting [upper_ctf, ws] := 0;
    ctf_setting [upper_ctf, wz] := 0;
    ctf_setting [upper_ctf, sz] := 0;
    ctf_setting [lower_ctf, ws] := 0;
    ctf_setting [lower_ctf, wz] := 0;
    ctf_setting [lower_ctf, sz] := 0;
```

```
      end;
      show_mm_parameters;
      y := wherey;
      window (1, y, 80, 24);
      clrscr;
      get_TAC_delay;
      get_blocking_strip_full_width;
      get_Ortec_ADC_channels;
      get_lower_level_discriminator_settings (tps_stp.lld_setting, 'A');
      get_detector_multiplier_constant_settings (tps_stp.dmc_setting,
                                                 tps_stp.dac_setting,
                                                 'A');
      write_set_up;
    end;
  initial_read := FALSE;

  clrscr;

  temp_data_path := 'e:\temp';
  tps_data_path := 'c:\turbo\frbm\dyn';

  tps_accumulate_mode := TRUE;
  tps_shot_number := 100000;
  tps_refresh_limit := 201;
  df_display_lower_discriminator := 1;

  df_radius := 20000;
  df_plot_max := 15;
  df_plot_min := 0;

  set_up_Ortec_ADCs (tps_stp.lld_setting);

  main_menu;

end; { procedure initialize_tps_program }

end. { unit tps_init }
```

2.4 tps_mm.pas

```
{ tps_mm.pas                                                    }

{ contents :                                                    }

{ procedure main_menu                                           }
{     menu of things to do before and after dynamics experiment }

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

unit tps_mm;
```

```
{$O+}
{$F+}

interface

uses tps_glob,tps_util,tpsparam,tps_radc,tps_exp,tps_lld,CRT;

procedure main_menu;

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

implementation

procedure main_menu;
var
 ch : char;
 y : integer;
begin { procedure main_menu }
 show_mm_parameters;
 repeat
  window (1, 1, 36, 24);
  clrscr; textcolor (15);
  writeln;
  writeln;
  writeln ('E : Set Experimental Parameters');
  writeln;
  writeln ('R : Read Ortec ADCs');
  writeln ('T : Test Detector Spatial Response');
  writeln;
  writeln;
  writeln ('Q : Quit Dynamics Menu');
  writeln;
  writeln ('-----------------------------------');
  writeln; y := wherey; textcolor (14);
  repeat
   window (1, y, 80, 24); clrscr;
   write ('? > '); ch := Readkey; ch := upcase(ch); writeln (ch);
   writeln;
  until ch in ['E', 'R', 'T', 'Q'];
  case ch of
   'E' : begin
          tps_experimental_parameters;
          show_mm_parameters;
         end;
   'R' : begin
          read_adc;
          show_mm_parameters;
         end;
   'T' : begin
          tps_menu;
          show_mm_parameters;
         end;
  end; { case ch }
 until (ch = 'Q');
```

```
 if not ask_for_boolean
        ('Are you sure you want to quit DYN menu? [Y/N] : ') then
  begin
   main_menu;
  end;
 window (1, 1, 80, 24); textcolor (15); clrscr;
end; { procedure main_menu }

end. { unit tps_mm }
```

2.5 tps_glob.pas

```
{ tps_glob.pas
 this unit contains all global variables neccessary for tps.pas        }

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

unit tps_glob;

{$O+}
{$F+}

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

interface

uses graphadd;

const

 { type of experiment in progress }

 tps = 0;
 dsr = 4;

 counts = 0;

 wedge_1 : integer = 1;
 strip_1 : integer = 2;
 z_1     : integer = 3;
 TAC     : integer = 4;
 wedge_2 : integer = 5;
 strip_2 : integer = 6;
 z_2     : integer = 7;

 ws = 1;
 wz = 2;
 sw = 3;
 sz = 4;
 zw = 5;
 zs = 6;
```

```
full_detector          : integer =  8;
upper_detector_semicircle : integer =  9;
lower_detector_semicircle : integer = 10;

upper_ctf :  integer = 8;
lower_ctf :  integer = 9;

upper = 0;
lower = 1;

w1_col = 1;
s1_col = 2;
z1_col = 3;
TAC_col = 4;
w2_col = 5;
s2_col = 6;
z2_col = 7;

rd_size_limit = 4600;

tps_set_up_file_name = 'tps_set.up';

{ CAMAC crate stations for various electronics }
Ortec_ADC_1 : integer = 10;  { CAMAC crate station of Ortec AD413 ADC }
Ortec_ADC_2 : integer = 13;  { CAMAC crate station of Ortec AD413 ADC }

{ CAMAC address for for module specific instructions }
module     : integer =  0;

{ descriptive constants for CAMAC function calls }
read_channel         : integer =  0; { CAMAC F0 }
read_control_register  : integer =  0; { CAMAC F0 }
read_lld_setting       : integer =  1; { CAMAC F1 }
read_ADC_conversion    : integer =  2; { CAMAC F2 }
test_LAM             : integer =  8; { CAMAC F8 }
write_control_register : integer = 16; { CAMAC F16 }
write_lld_setting      : integer = 17; { CAMAC F17 }

dataway_z            : integer =  1; { CAMAC Initialize }
dataway_c            : integer =  2; { CAMAC Clear }
dataway_i            : integer =  4; { CAMAC Inhibit }

CRIACL               : integer = 64; { Clear Request Inhibit
                                       & ACL detection registers }

{ Ortec control register locations }
control_register_1_location  :  integer = 0;
control_register_2_location  :  integer = 1;

{ Ortec control register settings }
{ control register 1 }
 control_register_1_Ortec_1  :  integer = -3328;
 control_register_1_Ortec_2  :  integer = -3327;
```

```
{ in control register 1 the bits have the following functions :
  bits 1..8 :  Virtual Station Number : set to 0 and 1 respectively for
               each ADC
               (Ortec #1 value : 0, Ortec #2 value : 1)
  bit 9     : Zero Suppression Enable : set to 1 for sequential or
               random access)
               (value : 256)
  bit 10    : ECL Port Enable : set to 1 to enable CAMAC readout
               (value : 512)
  bit 11    : Not Used
  bit 12    : Not Used
  bit 13    : Coincidence/Singles Mode : set to 1 for singles mode
               (value : 4096)
  bit 14    : CAMAC Random Access Enable : set to 1 for addressed
               readout
               (value : 8192)
  bit 15    : CAMAC LAM Enable : set to 1 so LAM is asserted when data
               is ready to be read out on the CAMAC port
               (value : 16384)
  bit 16    : Overflow-Suppression Enable : set to 1 so all pulses above
               the upper level discriminator are reported with a value
               between 8064 and 8191
               (value : makes the integer value negative if set to 1)

               total integer value (excluding VSN) : -21248          }

 PathToGraphics : string = 'c:\turbo\graphics';

type
 string_255 = string[255];
 string_80 = string[80];
 string_8 = string[8];
 string_2 = string[2];

 tps_int_column_array = array [1..7] of integer;
 swd_int_column_array = array [1..7, 0..1] of integer;
 dcc_real_column_array = array [1..7] of real;
 ctf_real_column_array = array [8..9, 1..6] of real;

 tps_raw_data_pointer = ^tps_raw_data_array;
 tps_raw_data_array = array [1..rd_size_limit, w1_col..z2_col] of integer;

 tps_energy_and_angle_pointer = ^tps_energy_and_angle_array;
 tps_energy_and_angle_array = array [0..325, 0..9, 0..9] of word;

 tps_df_data_pointer = ^tps_df_data_array;
 tps_df_data_array = array [-120..119, -120.. 119] of byte;

 tps_set_up = record
    blocking_strip_full_width : real;
    TAC_delay : real;
    z_scale_factor : real;
    Ortec_ADC_channel : tps_int_column_array;
    gate_ADCs : boolean;
```

```
    lld_setting : tps_int_column_array;
    swd_setting : swd_int_column_array;
    dmc_setting : dcc_real_column_array;
    dac_setting : dcc_real_column_array;
    ctf_setting : ctf_real_column_array;
   end; { record tps_set_up }

var

 D, Q, X : integer;

 wedge_1_channel  : integer;
 strip_1_channel  : integer;
 z_1_channel      : integer;
 TAC_channel      : integer;
 wedge_2_channel  : integer;
 strip_2_channel  : integer;
 z_2_channel      : integer;

{ control register 2 }
 control_register_2 : integer;
{ in control register 2 of both ADCs bits 1..5 are enable/disable (0/1)
 gate 1...master gate respectively.  Therefore useful settings for us are
 $1F (disable all gates) and
 $10 (disable master gate, enable individual gates                    }

 Year, Month, Day, DayofWeek, Hour, Minute, Second, Sec100 : word;

 tps_data_path : string;
 temp_data_path : string;

 experiment_type : integer;
 detector_element : integer;

 tps_stp : tps_set_up;

 tps_df_data : tps_df_data_pointer;
 tps_energy_and_angle_data : tps_energy_and_angle_pointer;

 file_name : string_80;
 upper_file_name : string_80;
 lower_file_name : string_80;
 comp_file_name : string_80;
 data_file_name : string_80;
 log_file_name : string_80;
 data_file : text;
 log_file : text;

 initial_read : boolean;
 upper_left_corner_y : integer;
 full_parameter_listing : boolean;
 list_gate_settings : boolean;
```

```
  sw_discriminator : swd_int_column_array;

  df_min, df_max : word;
  df_plot_min, df_plot_max : word;
  tps_accumulate_mode : boolean;
  tps_shot_number : longint;
  tps_data_count : longint;
  tps_refresh_count : word;
  tps_refresh_limit : word;
  tps_data_read_cycles : longint;
  rec_data_read_cycles : word;
  zero_data_count, low_data_count : longint;
  high_data_count, overflow_count : longint;
  outside_active_area_count, non_coincidence_count : longint;
  rec_zero_data_count, rec_low_data_count : word;
  rec_high_data_count, rec_overflow_count : word;
  rec_outside_active_area_count, rec_non_coincidence_count : word;
  df_radius, df_display_lower_discriminator : integer;

  IOErrorCode : integer;
  active_directory : string;
  bin_data_file : file;
  int_data_file : file of integer;
  text_data_file : text;

  h, m, s, s100 : word;
  initial_start_time, start_time : real;
  count_time, total_count_time : real;
  total_exp_time : word;
  count_rate, average_count_rate : real;

  GraphDriver : integer;  { The Graphics device driver }
  GraphMode   : integer;  { The Graphics mode value }
  ErrorCode   : integer;  { Reports any graphics errors }
  MaxColor    : word;     { The maximum color value available }
  OldExitProc : Pointer;  { Saves exit procedure address }
  GraphicsOn  : Boolean;  { Saves status of CRT mode (graphics or text) }
  HP7470Plotter : integer;
  PostScriptLaserWriterFile : integer;
  status : word;
  magic : ^byte;
  wordptr : ^word;
  hold : byte;

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

implementation

end. { unit tps_glob }
```

2.6 tps_util.pas

```
{ tps_util.pas                                          }

{ contents :                                            }

{ i. math                                               }
{ function int_power (base : real;                              }
{                     power : integer) : real                   }
{     raises any real 'base' to positive integer 'power'        }

{ ii. user input                                        }

{ ii.a. yes/no                                          }
{ function ask_for_boolean (prompt : string_80) : boolean       }
{     user must answer yes/no question, 'prompt'                 }

{ ii.b. continue/quit                                   }
{ function ask_continue : boolean                               }
{     user must hit <enter> to continue or Esc to quit          }

{ ii.c. numbers                                         }
{ these functions convert strings to numbers and ignore inappropriate    }
{ characters -- beware of misinterpretation of invalid input             }
{ true if user enters 'response' between 'min' and 'max' or              }
{                     '<' which sets 'response' to 'min' or              }
{                     '>' which sets 'response' to 'max'                 }
{ false if user just presses 'enter'; 'response' is unchanged            }
{ 'prompt' is repeated if user enters something else                     }

{ function ask_for_integer (var response : integer;                      }
{                           prompt : string_80;                          }
{                           min, max : integer) : boolean                }

{ function ask_for_real (var response : real;                            }
{                        prompt : string_80;                             }
{                        min, max : integer) : boolean                   }

{ ii.d. wavelength to wavenumber conversion and vise-versa               }

{ function wl2wn (wl : real) : real;                                     }

{ function wn2wl (wn : real) : real;                                     }

{ ii.e. file names                                      }

{ function exist (file_name : string_80) : boolean                       }
{     true if file called 'file_name' is present on disk                 }

{ function legal_name (file_name : string_80) : boolean                  }
{     true if file name 'file_name' is a legal one                       }
```

```
{ procedure get_out_file_name (var file_name : string_80;              }
{                       var overwrite : boolean                 }
{                       var got_good_out_file_name : boolean      }
{    true if user enters a new 'file_name' or wants to overwrite an    }
{    existing file; if overwrite, sets overwrite to TRUE             }

{ function get_in_file_name (var file_name : string_80;               }
{                       prompt : string_80) : boolean             }
{    true if user entered 'file_name' of an existing file           }


{ iii. save and read data                                          }

{ procedure save_data (experiment_type : integer);                    }
{    save data to individual'.dat' file, append scan parameter info    }
{    to one 'date.log' file per day                                 }


{ iv. miscellaneous                                                }

{ procedure reserved;                                              }
{    displays message : 'reserved for future development'           }
{                                                                                    }
{ function real_convert (var a : integer_24) : real;                 }
{    converts DSP output to a real number                          }

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

unit tps_util;

{SO+}
{SF+}

interface

uses tps_glob,codes,DOS,CRT;

function int_power (base : real; power : integer) : real;
function ask_for_boolean (prompt : string_80) : boolean;
function ask_continue : boolean;
function ask_for_byte (var response : byte; prompt : string_80;
                       min, max : byte) : boolean;
function ask_for_integer (var response : integer; prompt : string_80;
                       min, max : integer) : boolean;
function ask_for_long_integer (var response : longint; prompt : string_80;
                       min, max : longint) : boolean;
function ask_for_real (var response : real; prompt : string_80;
                    min, max : real) : boolean;
function wl2wn (wl : real) : real;
function wn2wl (wn : real) : real;
function I2S (Val, Digit : Integer) : String;
function W2S (Val, Digit : Word) : String;
function R2S (Val : real; Digit, Decimal : Integer) : String;
function exist (file_name : string_80) : boolean;
```

```
function legal_name (file_name : string_80) : boolean;
procedure get_out_file_name (var file_name : string_80; var overwrite : boolean;
                    experiment_type : integer;
                    var got_good_out_file_name : boolean);
function get_in_file_name (var file_name : string_80) : boolean;
procedure form_date_and_time_strings (var date_string : string;
                          var time_string : string);
procedure clock_on (var start_clock : real);
function elapsed_time (start_clock : real) : real;
procedure save_data_file;
procedure read_data_file;
function compress (file_name : string_80) : boolean;
function uncompress (comp_file_name : string_80) : boolean;
procedure reserved;

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

implementation

function int_power (base : real;
              power : integer) : real;
begin { function int_power }
 if (power = 0) then
  int_power := 1.0
 else
  int_power := base * int_power (base, (power - 1));
end;  { function int_power }

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

function ask_for_boolean (prompt : string_80) : boolean;
var
 ch : char;
begin { function ask_for_boolean }
 ch := ' ';
 repeat
  write (prompt);  ch := ReadKey;  ch := upcase (ch);  writeln (ch);
  if ((ch <> 'Y') and (ch <> 'N')) then
   writeln ('Please type "Y" or "N"');
 until ((ch = 'Y') or (ch = 'N'));
 ask_for_boolean := (ch = 'Y');
end;  { function ask_for_boolean }

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

function ask_continue : boolean;
var
 ch : char;
begin { function ask_continue }
 repeat
  writeln;
  write ('Hit <enter> to continue or Esc to quit : ');
  ch := readkey;  ch := upcase (ch);  write (ch);
  if ch = ESC then ask_continue := false;
```

```
    if ch = #13 then ask_continue := true;
  until ((ch = ESC) or (ch = #13));
end; { function ask_continue }

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

function ask_for_byte (var response : byte;
                       prompt : string_80;
                       min, max : byte) : boolean;
var
  answer, i : byte;
  code : integer;
  input_string : string[8];
begin { function ask_for_byte }
  answer := 0;
  code := 1;
  repeat
    write (prompt); readln (input_string);
    if (input_string = '') then
      begin
        ask_for_byte := FALSE;
        exit;
      end;
    if (input_string = '<') then
      begin
        ask_for_byte := TRUE;
        response := min;
        exit;
      end;
    if (input_string = '>') then
      begin
        ask_for_byte := TRUE;
        response := max;
        exit;
      end;
    i := 1;
    repeat
      if not (input_string [i] in ['0'..'9']) then
        delete (input_string, i, 1)
      else
        i := i + 1;
    until (i = (length (input_string) + 1));
    val (input_string, answer, code);
  until ((input_string <> '') and (answer >= min) and (answer <= max) and
         (code = 0));
  ask_for_byte := TRUE;
  response := answer;
end;  { function ask_for_byte }

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

function ask_for_integer (var response : integer;
                          prompt : string_80;
                          min, max : integer) : boolean;
```

```
var
  answer, code, i : integer;
  input_string : string[8];
begin { function ask_for_integer }
  answer := 0;
  code := 1;
  repeat
    write (prompt); readln (input_string);
    if (input_string = '') then
      begin
        ask_for_integer := FALSE;
        exit;
      end;
    if (input_string = '<') then
      begin
        ask_for_integer := TRUE;
        response := min;
        exit;
      end;
    if (input_string = '>') then
      begin
        ask_for_integer := TRUE;
        response := max;
        exit;
      end;
    i := 1;
    repeat
      if not (input_string [i] in ['0'..'9']) then
        delete (input_string, i, 1)
      else
        i := i + 1;
    until (i = (length (input_string) + 1));
    val (input_string, answer, code);
  until ((input_string <> '') and (answer >= min) and (answer <= max) and
        (code = 0));
  ask_for_integer := TRUE;
  response := answer;
end;  { function ask_for_integer }

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

function ask_for_long_integer (var response : longint;
                    prompt : string_80;
                    min, max : longint) : boolean;
var
  answer : longint;
  code, i : integer;
  input_string : string[8];
begin { function ask_for_long_integer }
  answer := 0;
  code := 1;
  repeat
    write (prompt); readln (input_string);
    if (input_string = '') then
```

```
    begin
      ask_for_long_integer := FALSE;
      exit;
    end;
  if (input_string = '<') then
    begin
      ask_for_long_integer := TRUE;
      response := min;
      exit;
    end;
  if (input_string = '>') then
    begin
      ask_for_long_integer := TRUE;
      response := max;
      exit;
    end;
  i := 1;
  repeat
    if not (input_string [i] in ['0'..'9']) then
      delete (input_string, i, 1)
    else
      i := i + 1;
  until (i = (length (input_string) + 1));
  val (input_string, answer, code);
 until ((input_string <> '') and (answer >= min) and (answer <= max) and
       (code = 0));
 ask_for_long_integer := TRUE;
 response := answer;
end;  { function ask_for_long_integer }

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

function ask_for_real (var response : real;
                       prompt : string_80;
                       min, max : real) : boolean;
var
 answer : real;
 code, i : integer;
 input_string : string[10];
begin { function ask_for_real }
 answer := 0.0;
 code := 1;
 repeat
  write (prompt); readln (input_string);
  if (input_string = '') then
    begin
      ask_for_real := FALSE;
      exit;
    end;
  if (input_string = '<') then
    begin
      ask_for_real := TRUE;
      response := min;
      exit;
```

```
    end;
   if (input_string = '>') then
    begin
     ask_for_real := TRUE;
     response := max;
     exit;
    end;
   i := 1;
   repeat
    if not (input_string [i] in ['0'..'9', '.', 'e', 'E', '-', '+']) then
     delete (input_string, i, 1)
    else
     i := i + 1;
   until (i = (length (input_string) + 1));
   val (input_string, answer, code);
 until ((input_string <> '') and (answer >= min) and (answer <= max) and
       (code = 0));
 ask_for_real := TRUE;
 response := answer;
end;  { function ask_for_real }

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

function wl2wn (wl : real) : real;
begin
wl2wn := (10000000 / wl);
end;

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

function wn2wl (wn : real) : real;
begin
wn2wl := (10000000 / wn);
end;

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

function I2S (Val, Digit : Integer) : String;
var
 buffer : string;
begin
 str(Val:Digit, Buffer);
 I2S := Buffer;
end;

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

function W2S (Val, Digit : Word) : String;
var
 buffer : string;
begin
 str (Val:Digit, Buffer);
 W2S := Buffer;
end;
```

```
{ ************************************************************************ }

function R2S (Val : real; Digit, Decimal : Integer) : String;
var
 buffer : string;
begin
 str(Val:Digit:Decimal, Buffer);
 R2S := Buffer;
end;

{ ************************************************************************ }
function exist (file_name : string_80) : boolean;
var
 test_file : file;
begin { function exist }
 assign (test_file, file_name);
 {$I-}
 reset (test_file);
 close (test_file);
 {$I+}
 exist := (IOresult = 0);
end;  { function exist }

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

function legal_name (file_name : string_80) : boolean;
var
 test_file : file;
begin { function legal_name }
 assign (test_file, file_name);
 {$I-}
 rewrite (test_file);
 {$I+}
 legal_name := (IOresult = 0);
 close (test_file);
end; { function legal_name }

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

procedure get_out_file_name (var file_name : string_80;
                    var overwrite : boolean; experiment_type : integer;
                    var got_good_out_file_name : boolean);
var
 upper_extension, lower_extension, prompt : string;
 i : integer;
begin { function get_out_file_name }
 case experiment_type of

 tps :
  begin
   upper_extension := '.upp';
   lower_extension := '.low';
   prompt := 'Output data file name? [No extension] : '
```

```
  end;
end;
file_name := '';
upper_file_name := '';
lower_file_name := '';
overwrite := FALSE;
write (prompt); readln (file_name);
if (file_name = '') then
  begin
    writeln;
    if ask_for_boolean
          ('Do you really want to NOT save this file? [Y/N] : ') then
      begin
        got_good_out_file_name := FALSE;
        exit;
      end
    else
      begin
        get_out_file_name (file_name, overwrite, experiment_type,
                    got_good_out_file_name);
      end;
  end;
if pos ('.', file_name) = 0 then
  begin
    upper_file_name := upper_file_name + upper_extension;
    lower_file_name := lower_file_name + lower_extension;
  end
else
  begin
    writeln ('I said, no extension!!!');
    get_out_file_name (file_name, overwrite, experiment_type,
                    got_good_out_file_name);
  end;
if exist (upper_file_name) then
  begin
    writeln; writeln; write ('            ');
    if ask_for_boolean ('Overwrite '+ upper_file_name + '? [Y/N] : ') then
      begin
        overwrite := TRUE;
      end
    else
      begin
        writeln;
        get_out_file_name (file_name, overwrite, experiment_type,
                    got_good_out_file_name);
      end;
  end
else if exist (lower_file_name) then
  begin
    writeln; writeln; write ('            ');
    if ask_for_boolean ('Overwrite '+ lower_file_name + '? [Y/N] : ') then
      begin
        overwrite := TRUE;
      end
```

```
    else
     begin
      writeln;
      get_out_file_name (file_name, overwrite, experiment_type,
                  got_good_out_file_name);
     end;
   end
  else if not legal_name (upper_file_name) then
   begin
    writeln ('Not a legal name for a file!');
    get_out_file_name (file_name, overwrite, experiment_type,
                got_good_out_file_name);
   end
  else if not legal_name (lower_file_name) then
   begin
    writeln ('Not a legal name for a file!');
    get_out_file_name (file_name, overwrite, experiment_type,
                got_good_out_file_name);
   end
  else
   got_good_out_file_name := TRUE;
end; { function get_out_file_name }

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

function get_in_file_name (var file_name : string_80) : boolean;
var
 extension, prompt : string;
 i : integer;
begin { function get_in_file_name }
 get_in_file_name := FALSE;
 case experiment_type of

 tps :
  begin
   extension := '';
   chdir (tps_data_path);
   prompt := 'Enter file name to retrieve [No extension] : '
  end;
 end;
 file_name := '';
 write (prompt); readln (file_name);
 if pos ('.', file_name) <> 0 then
  begin
   exit;
  end;
 if (file_name = extension) then
  begin
   exit;
  end
 else if not exist (file_name) then
  begin
   writeln (file_name + ' does not exist.');
   get_in_file_name := get_in_file_name (file_name);
```

```
   end
  else
   get_in_file_name := TRUE;
end;  { function get_in_file_name }

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

procedure form_date_and_time_strings (var date_string, time_string : string);
var
 Yearstr, Monthstr, Daystr, Hourstr, Minstr, Secstr : string;
begin
 GetTime (Hour, Minute, Second, Sec100);
 time_string := '';
 Hourstr := I2S (Hour,0);
 if length (Hourstr) = 1 then
  begin
   Hourstr := '0' + Hourstr;
  end;
 Minstr := I2S (Minute,0);
 if length (Minstr) = 1 then
  begin
   Minstr := '0' + Minstr;
  end;
 Secstr := I2S (Second,0);
 if length (Secstr) = 1 then
  begin
   Secstr := '0' + Secstr;
  end;
 time_string := Hourstr + ':' + Minstr + ':' + Secstr;

 GetDate (Year, Month, Day, DayofWeek);
 date_string := '';
 Yearstr := I2S (Year,0);
 delete (Yearstr,1,2);
 Monthstr := I2S (Month,0);
 if length (Monthstr) = 1 then
  begin
   Monthstr := '0' + Monthstr;
  end;
 Daystr := I2S (Day,0);
 if length (Daystr) = 1 then
  begin
   Daystr := '0' + Daystr;
  end;
 date_string := Yearstr + Monthstr + Daystr;
end;

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

procedure clock_on (var start_clock : real);
begin { procedure clock_on }
 GetTime (h, m, s, s100);
 start_clock := h * 3600 + m * 60 + s + s100 / 100;
end; { procedure clock_on }
```

```
{ ************************************************************************** }

function elapsed_time (start_clock : real) : real;
begin { function elapsed_time }
 GetTime (h, m, s, s100);
 elapsed_time := (h * 3600 + m * 60 + s + s100 / 100) - start_clock;
end; { function elapsed_time }

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

procedure save_data_file;
var
 i, j, k, drive, remainder : integer;
 DiskDriveFreeMemory : longint;
 overwrite, got_good_out_file_name : boolean;
 active_directory, data_path, extension : string;
 Yearstr, Monthstr, Daystr, Hourstr, Minstr, Secstr : string;
 ms_time : real;
begin { procedure save_data }

 window (1, 1, 80, 24);
 clrscr;
 GoToXY (1, 8);

 getdir (0, active_directory);
 case experiment_type of

  tps :
   begin
    {$I-}
    chdir (tps_data_path);
    IOErrorCode := IOResult;
    if IOErrorCode <> 0 then
     begin
      writeln ('Is RamDrive enabled in config.sys ?');
      exit;
     end;
    {$I+}
   end;
  end;

 get_out_file_name (data_file_name, overwrite, experiment_type,
              got_good_out_file_name);

 if not got_good_out_file_name then
  begin
   chdir (active_directory);
   exit;
  end;

 assign (data_file, 'c:\turbo\frbm\tps\allupdat.txt');
 rename (data_file, data_file_name + '.upp');
 assign (data_file, 'c:\turbo\frbm\tps\alllwdat.txt');
```

```
 rename (data_file, data_file_name + '.low');

end;  { procedure save_data }

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

procedure read_data_file;
var
 i, j, k, remainder : integer;
 normalized_detached_electron_signal : real;
 normalized_laser_power : real;
 active_directory : string;
begin { procedure read_data_file }
 getdir (0, active_directory);
 window (1, 1, 80, 24);
 clrscr;
 GoToXY (1, 8);
 writeln ('WARNING : This action will overwrite current data in RAM.');
 writeln;
 writeln ('If you have not done so already, save the ',
       'current data after exiting now.');
 writeln;
 if not ask_continue then
  begin
   exit;
  end;
 writeln;  writeln;
 if not get_in_file_name (data_file_name) then
  begin
   chdir (active_directory);
   exit;
  end;

 case experiment_type of

  tps :
   begin

   end;
  end;
 chdir (active_directory);
end;  { procedure read_data_file }


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

function compress (file_name : string_80) : boolean;
var
 i, l : integer;
 command_string : string_80;
begin { function compress }
 l := length (file_name);
 data_file_name := file_name;
 i := pos ('.', file_name);
```

```
   if i > 0 then
    begin
     delete (file_name, i, (l - i + 1));
    end;
   comp_file_name := file_name + '.lzh';
  SwapVectors;
  command_string := '/c c:\comp.bat ' + comp_file_name + ' ' + data_file_name;
  Exec('c:\dos\command.com', command_string);
  SwapVectors;
  if DosError <> 0 then
   begin
    TextColor (14 + Blink);
    GraphicsOn := FALSE;
    GoToXY (10,12);
    writeln ('Dos Error # ', DosError);
    write ('Hit <enter> to continue ...');
    readln;
    compress := FALSE;
   end
  else
   begin
    GoToXY (1,10);
    writeln ('Compression of file ', data_file_name, ' successful.');
    delay (2000);
    compress := TRUE;
   end;

end; { function compress }

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

procedure reserved;
begin { procedure reserved }
 writeln ('This key is reserved');
 writeln ('for future development.');
 delay (2000);
end;  { procedure reserved }

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

end. { of unit tps_util }
```

2.7 codes.pas

```
UNIT Codes;

{$O+}
{$F+}

{ This unit is simply a list of the ascii codes for the
 various editing and function keys.  Any program
```

```
 employing this unit can use the constant designations in
 a readkey statement.  -- I got sick of looking up the
 codes. --  JMP.}

INTERFACE

CONST
   { Editing Key Codes }

   ESC = #27;
   RETURN = #13;
   UPARROW = #72;
   DOWNARROW = #80;
   LEFTARROW = #75;
   RIGHTARROW = #77;
   HOME = #71;
   PGUP = #73;
   ENDKEY = #79;
   PGDN = #81;
   INS = #82;
   DEL = #83;

   { Function Key Codes }

   F1  = #59;
   F2  = #60;
   F3  = #61;
   F4  = #62;
   F5  = #63;
   F6  = #64;
   F7  = #65;
   F8  = #66;
   F9  = #67;
   F10 = #68;

   ALTF1  = #104;
   ALTF2  = #105;
   ALTF3  = #106;
   ALTF4  = #107;
   ALTF5  = #108;
   ALTF6  = #109;
   ALTF7  = #110;
   ALTF8  = #111;
   ALTF9  = #112;
   ALTF10 = #113;

   CTRLF1 = #94;
   CTRLF2 = #95;
   CTRLF3 = #96;
   CTRLF4 = #97;
   CTRLF5 = #98;
   CTRLF6 = #99;
   CTRLF7 = #100;
   CTRLF8 = #101;
```

```
    CTRLF9 = #102;
    CTRLF10 = #103;
    CTRLF11 = #137;
    CTRLF12 = #138;

    SHIFTF1  = #84;
    SHIFTF2  = #85;
    SHIFTF3  = #86;
    SHIFTF4  = #87;
    SHIFTF5  = #88;
    SHIFTF6  = #89;
    SHIFTF7  = #90;
    SHIFTF8  = #91;
    SHIFTF9  = #92;
    SHIFTF10 = #93;

IMPLEMENTATION

END.
```

2.8 tpsparam.pas

```
{ tpsparam.pas                                          }

{ contents :                                            }

{ procedure tps_experimental_parameters                 }
{     menu for changing experimental set_up parameters  }

{ procedure get_Ortec_ADC_channels                      }

{ procedure get_lower_level_discriminator_settings      }

{ function read_set_up : boolean                        }
{     initialize dynamics experimental set-up parameters from a file }

{ procedure write_set_up                                }
{     write dynamics experimental set_up parameters to a file }

{ procedure show_mm_parameters                          }
{     display experimental set_up parameters on right hand side of screen }

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

unit tpsparam;

{$O+}
{$F+}

interface
```

```
uses tps_glob,tps_util,tps_lld,tps_swd,DOS,CRT;

procedure show_mm_parameters;
procedure get_blocking_strip_full_width;
procedure get_TAC_delay;
procedure get_Ortec_ADC_channels;
procedure get_gate_ADCs;
function read_set_up : boolean;
procedure write_set_up;
procedure tps_experimental_parameters;

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

implementation

procedure show_mm_parameters;
var
 wind_y : integer;
begin { procedure show_mm_parameters }
 wind_y := Hi(WindMin) + 1;
 window (37, 1, 80, 24);
 clrscr; textcolor (11);
 with tps_stp do
  begin
   write (' Blocking Strip Full Width : ');
   writeln (blocking_strip_full_width:10:2);
   write ('              TAC Delay : ');
   writeln (TAC_delay:9:1);
   write ('  Scale Factor for Z Data : ');
   writeln (z_scale_factor:10:2);
   write ('         Gate Ortec ADCs : ');
   if tps_stp.gate_ADCs then
    begin
     writeln ('   YES')
    end
   else
    begin
     writeln ('    NO');
    end;
   write ('     Upper Wedge Channel : ');
   writeln (Ortec_ADC_channel [wedge_1]:7);
   write ('     Upper Strip Channel : ');
   writeln (Ortec_ADC_channel [strip_1]:7);
   write ('         Upper Z Channel : ');
   writeln (Ortec_ADC_channel [z_1]:7);
   write ('             TAC Channel : ');
   writeln (Ortec_ADC_channel [TAC]:7);
   write ('     Lower Wedge Channel : ');
   writeln (Ortec_ADC_channel [wedge_2]:7);
   write ('     Lower Strip Channel : ');
   writeln (Ortec_ADC_channel [strip_2]:7);
   write ('         Lower Z Channel : ');
   writeln (Ortec_ADC_channel [z_2]:7);
  end;
```

```
 window (1, wind_y, 80, wind_y + 1);
 textcolor (14);
end;  { procedure show_mm_parameters }

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

procedure get_blocking_strip_full_width;
begin { procedure get_blocking_strip_full_width }
 if ask_for_real (tps_stp.blocking_strip_full_width,
          'Blocking Strip Full Width [mm] : ', 0, 10) then
   show_mm_parameters;
end; { procedure blocking_strip_full_width }

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

procedure get_TAC_delay;
begin { procedure get_TAC_delay }
 if ask_for_real (tps_stp.TAC_delay, 'TAC Delay [ns] : ', 0, 1000) then
   show_mm_parameters;
end; { procedure TAC_delay }

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

procedure get_z_scale_factor;
begin { procedure get_z_scale_factor }
 writeln ('   Z Data Scaling Factor : ', tps_stp.z_scale_factor:3:2);
 if ask_for_real (tps_stp.z_scale_factor, 'New Z Data Scaling Factor : ',
          -10, 10) then
   show_mm_parameters;
end; { procedure get_z_scale_factor }

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

procedure get_Ortec_ADC_channels;
var
 y : integer;
begin {procedure get_Ortec_ADC_channels}
 y := wherey + Hi(WindMin);
 window ( 1, y, 80, (y + 1));
 writeln (' Upper Wedge Signal in Ortec ADC Channel : ',
            tps_stp.Ortec_ADC_channel [wedge_1]:1);
 if ask_for_integer (tps_stp.Ortec_ADC_channel [wedge_1],
            'New Upper Wedge Signal Ortec ADC Channel : ', 1, 7) then
              wedge_1_channel := (wedge_1 - 1) mod 4;
              show_mm_parameters;
 writeln (' Upper Strip Signal in Ortec ADC Channel : ',
          tps_stp.Ortec_ADC_channel [strip_1]:1);
 if ask_for_integer (tps_stp.Ortec_ADC_channel [strip_1],
            'New Upper Strip Signal Ortec ADC Channel : ', 1, 7) then
              strip_1_channel := (strip_1 - 1) mod 4;
              show_mm_parameters;
 writeln ('   Upper Z Signal in Ortec ADC Channel : ',
          tps_stp.Ortec_ADC_channel [z_1]:1);
 if ask_for_integer (tps_stp.Ortec_ADC_channel [z_1],
```

```
                ' New Upper Z Signal Ortec ADC Channel : ', 1, 7) then
                  z_1_channel := (z_1 - 1) mod 4;
                  show_mm_parameters;
  writeln ('      TAC Signal in Ortec ADC Channel : ',
           tps_stp.Ortec_ADC_channel [TAC]:1);
  if ask_for_integer (tps_stp.Ortec_ADC_channel [TAC],
                '    New TAC Signal Ortec ADC Channel : ', 1, 7) then
                  TAC_channel := (TAC - 1) mod 4;
                  show_mm_parameters;
  writeln (' Lower Wedge Signal in Ortec ADC Channel : ',
              tps_stp.Ortec_ADC_channel [wedge_2]:1);
  if ask_for_integer (tps_stp.Ortec_ADC_channel [wedge_2],
             'New Lower Wedge Signal Ortec ADC Channel : ', 1, 7) then
                  wedge_2_channel := (wedge_2 - 1) mod 4;
                  show_mm_parameters;
  writeln (' Lower Strip Signal in Ortec ADC Channel : ',
           tps_stp.Ortec_ADC_channel [strip_2]:1);
  if ask_for_integer (tps_stp.Ortec_ADC_channel [strip_2],
             'New Lower Strip Signal Ortec ADC Channel : ', 1, 7) then
                  strip_2_channel := (strip_2 - 1) mod 4;
                  show_mm_parameters;
  writeln ('    Lower Z Signal in Ortec ADC Channel : ',
              tps_stp.Ortec_ADC_channel [z_2]:1);
  if ask_for_integer (tps_stp.Ortec_ADC_channel [z_2],
                ' New Lower Z Signal Ortec ADC Channel : ', 1, 7) then
                  z_2_channel := (z_2 - 1) mod 4;
                  show_mm_parameters;
end { procedure get_Ortec_ADC_channels };

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

procedure get_gate_ADCs;
begin { procedure get_gate_ADCs }
  tps_stp.gate_ADCs := ask_for_boolean ('Gate Ortec ADCs ? : ');
  if tps_stp.gate_ADCs then
    begin
      control_register_2 := $10;
    end
  else
    begin
      control_register_2 := $1F;
    end;
  show_mm_parameters;
end {procedure get_gate_ADCs };

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

function read_set_up : boolean;
var
  i, coupling : integer;
  shg, gating, ch : char;
begin { function read_set_up }
  if not initial_read then
  begin
```

```
  writeln ('!!!!!  Are you sure you want to read old set up file?  !!!!!');
  if not ask_continue then exit;
end;
if not exist (tps_set_up_file_name) then
  begin
    read_set_up := FALSE;
    exit;
  end;
assign (data_file, tps_set_up_file_name);
reset (data_file);
with tps_stp do
  begin
    readln (data_file, blocking_strip_full_width);
    readln (data_file, TAC_delay);
    readln (data_file, z_scale_factor);
    for detector_element := wedge_1 to strip_2 do
      begin
        read (data_file, Ortec_ADC_channel [detector_element]);
      end;
    readln (data_file, Ortec_ADC_channel [z_2]);
    readln (data_file, gating);
      if gating = 'T' then
        begin
          gate_ADCs := TRUE;
          control_register_2 := $10;
        end
      else
        begin
          gate_ADCs := FALSE;
          control_register_2 := $1F;
        end;
    for detector_element := wedge_1 to strip_2 do
      begin
        read (data_file, lld_setting [detector_element]);
      end;
    readln (data_file, lld_setting [z_2]);
    for detector_element := wedge_1 to strip_2 do
      begin
        read (data_file, swd_setting [detector_element, upper]);
      end;
    readln (data_file, swd_setting [z_2, upper]);
    for detector_element := wedge_1 to strip_2 do
      begin
        read (data_file, swd_setting [detector_element, lower]);
      end;
    readln (data_file, swd_setting [z_2, lower]);
    readln (data_file, dmc_setting [wedge_1]);
    readln (data_file, dmc_setting [strip_1]);
    readln (data_file, dmc_setting [z_1]);
    readln (data_file, dmc_setting [TAC]);
    readln (data_file, dmc_setting [wedge_2]);
    readln (data_file, dmc_setting [strip_2]);
    readln (data_file, dmc_setting [z_2]);
    readln (data_file, dac_setting [wedge_1]);
```

```
    readln (data_file, dac_setting [strip_1]);
    readln (data_file, dac_setting [wedge_2]);
    readln (data_file, dac_setting [strip_2]);
    for coupling := ws to zs do
     begin
      read (data_file, ctf_setting [upper_ctf, coupling]);
     end;
    readln (data_file);
    for coupling := ws to zs do
     begin
      read (data_file, ctf_setting [lower_ctf, coupling]);
     end;
    readln (data_file);
   end;
 close (data_file);
 read_set_up := TRUE;
end;  { function read_set_up }

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

procedure write_set_up;
var
 i, coupling : integer;
 shg, gating : char;
begin { procedure write_set_up }
 assign  (data_file, tps_set_up_file_name);
 rewrite (data_file);
 with tps_stp do
  begin
   writeln (data_file, blocking_strip_full_width:12:2);
   writeln (data_file, TAC_delay:11:1);
   writeln (data_file, z_scale_factor:12:2);
   for detector_element := wedge_1 to strip_2 do
    begin
     write (data_file, Ortec_ADC_channel [detector_element]:9);
    end;
   writeln (data_file, Ortec_ADC_channel [z_2]:9);
   if gate_ADCs = TRUE then
    begin
     gating := 'T';
    end
   else
    begin
     gating := 'F';
    end;
   writeln (data_file, '       ', gating);
   for detector_element := wedge_1 to strip_2 do
    begin
     write (data_file, lld_setting [detector_element]:9);
    end;
   writeln (data_file, lld_setting [z_2]:9);
   for detector_element := wedge_1 to strip_2 do
    begin
     write (data_file, swd_setting [detector_element, upper]:9);
```

```
      end;
    writeln (data_file, swd_setting [z_2, upper]:9);
    for detector_element := wedge_1 to strip_2 do
     begin
      write (data_file, swd_setting [detector_element, lower]:9);
     end;
    writeln (data_file, swd_setting [z_2, lower]:9);
    writeln (data_file, dmc_setting [wedge_1]:13:3);
    writeln (data_file, dmc_setting [strip_1]:13:3);
    writeln (data_file, dmc_setting [z_1]:13:3);
    writeln (data_file, dmc_setting [TAC]:13:3);
    writeln (data_file, dmc_setting [wedge_2]:13:3);
    writeln (data_file, dmc_setting [strip_2]:13:3);
    writeln (data_file, dmc_setting [z_2]:13:3);
    writeln (data_file, dac_setting [wedge_1]:13:3);
    writeln (data_file, dac_setting [strip_1]:13:3);
    writeln (data_file, dac_setting [wedge_2]:13:3);
    writeln (data_file, dac_setting [strip_2]:13:3);
    for coupling := ws to zs do
     begin
      write (data_file, ctf_setting [upper_ctf, coupling]:13:3);
     end;
    writeln (data_file);
    for coupling := ws to zs do
     begin
      write (data_file, ctf_setting [lower_ctf, coupling]:13:3);
     end;
    writeln (data_file);
   end;
 close (data_file);
end; { procedure write_set_up }

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

procedure tps_experimental_parameters;
var
 ch : char;
 y : integer;
 min_lambda, max_lambda : real;
  procedure show_mm_menu;
   begin { procedure show_mm_menu }
    window (1, 1, 36, 24);
    clrscr; textcolor (15);
    writeln ('B : Blocking Strip Full Width');
    writeln ('T : TAC Delay');
    writeln ('Z : Change Z Data Scaling Factor');
    writeln ('G : Gate Ortec ADC Channels');
    writeln;
    writeln;
    writeln;
    writeln ('C : Change Ortec ADC Channels');
    writeln;
    writeln;
    writeln;
```

```
      writeln ('L : Set CAMAC LL Discriminator');
      writeln ('S : Set Software Discriminator');
      writeln;
      writeln ('R : Read Set-up from File');
      writeln ('W : Write Current Set-up to File');
      writeln;
      writeln ('Q : Quit DDD Set-up Menu');
      writeln;
      writeln ('-----------------------------------');
      textcolor (14);
      upper_left_corner_y := wherey;
    end;  { procedure show_mm_menu }

begin { procedure tps_experimental_parameters }
 show_mm_menu;
 full_parameter_listing := TRUE;
 show_mm_parameters;
 repeat
  window (1, upper_left_corner_y, 80, 24);  clrscr;
  write ('? > ');
  ch := Readkey;  ch := upcase (ch);  writeln (ch);  writeln;
  case ch of
    'B' : get_blocking_strip_full_width;
    'T' : get_TAC_delay;
    'Z' : get_z_scale_factor;
    'G' : get_gate_ADCs;
    'C' : get_Ortec_ADC_channels;
    'L' : begin
            lld_menu;
            show_mm_menu;
            show_mm_parameters;
          end;
    'S' : begin
            swd_menu;
            show_mm_menu;
            show_mm_parameters;
          end;
    'R' : begin
            if read_set_up then
              show_mm_parameters;
          end;
    'W' : write_set_up;
  end; { case ch }
 until (ch = 'Q');
 full_parameter_listing := FALSE;
 show_mm_parameters;
end;  { procedure tps_experimental_parameters }

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

end. { unit tpsparam }
```

2.9 tps_radc.pas

```
{ tps_radc.pas                                                    }

unit tps_radc;

{$O+}
{$F+}

interface

uses tps_glob,tps_util,codes,tpsparam,tps_lld,DOS,CRT;


procedure read_adc;

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

implementation

{$F-}

{$I camturbo.v4}

{$F+}

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

procedure read_adc;
var
  ch : char;
  w, y, i, Q1, Q2 : integer;
  wedge_1_charge, strip_1_charge, z_1_charge : integer;
  wedge_2_charge, strip_2_charge, z_2_charge : integer;
  TAC_voltage : integer;
  tps_delta_lambda, lambda, old_lambda : real;
begin { procedure read_adc }
  show_mm_parameters;
  repeat
    window (1, 1, 36, 24);
    clrscr; textcolor (15);
    writeln;
    writeln ('R : Read Ortec ADCs');
    writeln;
    writeln ('L : Set Lower Level Discriminator');
    writeln;
    writeln;
    writeln;
    writeln;
    writeln ('Q : Quit');
    writeln;
    writeln ('-----------------------------'); writeln;
```

```
textcolor(14); w := wherey;
repeat
 window (1, w, 80, 24); clrscr;
 write ('? > '); ch := Readkey; ch := upcase (ch); writeln (ch);
 writeln;
until ch in ['R', 'L', 'Q'];
case ch of
 'R' : begin
       set_up_Ortec_ADCs (tps_stp.lld_setting);
       CAMCL (dataway_c);
       y := wherey;
       gotoxy (1, y);
       write ('TESTING LAMs ...');
       repeat
        CAMI (Ortec_ADC_1, test_LAM, module, D, Q1, X);
        CAMI (Ortec_ADC_2, test_LAM, module, D, Q2, X);
        if keypressed then
         ch := readkey;
         if ch = ESC then
          begin
           DelLine;
           exit;
          end;
       until (Q1 = 1) or (Q2 = 1);
       DelLine;
       gotoxy (1, y);
       write ('READING ADCs ...');
       CAMI (Ortec_ADC_1, read_ADC_conversion, wedge_1_channel,
                          wedge_1_charge, Q, X);
       CAMI (Ortec_ADC_1, read_ADC_conversion, strip_1_channel,
                          strip_1_charge, Q, X);
       CAMI (Ortec_ADC_1, read_ADC_conversion, z_1_channel,
                          z_1_charge, Q, X);
       z_1_charge := round (z_1_charge * tps_stp.z_scale_factor);
       CAMI (Ortec_ADC_1, read_ADC_conversion, TAC_channel,
                          TAC_voltage, Q, X);
       CAMI (Ortec_ADC_2, read_ADC_conversion, wedge_2_channel,
                          wedge_2_charge, Q, X);
       CAMI (Ortec_ADC_2, read_ADC_conversion, strip_2_channel,
                          strip_2_charge, Q, X);
       CAMI (Ortec_ADC_2, read_ADC_conversion, z_2_channel,
                          z_2_charge, Q, X);
       z_2_charge := round (z_2_charge * tps_stp.z_scale_factor);
       gotoxy (1, y);
       writeln ('Upper Detector Wedge Signal : ', wedge_1_charge:4);
       writeln ('Upper Detector Strip Signal : ', strip_1_charge:4);
       writeln ('   Upper Detector Z Signal : ', z_1_charge:4);
       writeln ('              TAC Signal : ', TAC_voltage:4);
       writeln ('Lower Detector Wedge Signal : ', wedge_2_charge:4);
       writeln ('Lower Detector Strip Signal : ', strip_2_charge:4);
       writeln ('   Lower Detector Z Signal : ', z_2_charge:4);
       writeln;
       write  ('                                ');
       write ('press < enter > to continue ...');
```

```
          readln;
        end;

    'L' : begin
          lld_menu;
          show_mm_parameters;
        end;

    end; { case ch }
  until (ch = 'Q');
  window (1, 1, 80, 24); textcolor (15); clrscr;
end;  { procedure read_adc }

end. { unit tps_radc }
```

2.10 tps_exp.pas

```
{ tps_exp.pas                                    }

{ contents :                                     }

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

unit tps_exp;

{$O+}
{$F+}

interface

uses tps_glob,tps_util,tpsparam,tps_lld,tps_dcc,tps_ctf,
   dfplot,tpsplot,graph,plotutil,codes,CRT,DOS;

function test_data (half : integer; tps_data : tps_int_column_array) : boolean;
procedure obtain_tps_data;
procedure show_tps_parameters;
procedure tps_menu;

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

implementation

{$F-}

{$I camturbo.v4}

{$F+}

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

function test_data (half : integer; tps_data : tps_int_column_array) : boolean;
```

```
var
 start_element, end_element : integer;
begin
 case half of
  upper : begin
    start_element := wedge_1;
    end_element := z_1;
   end;
  lower : begin
    start_element := wedge_2;
    end_element := z_2;
   end;
  end;
 for detector_element := start_element to end_element do
  begin
   if (tps_data [detector_element] = 0.0) then
    begin
     zero_data_count := zero_data_count + 1;
     rec_zero_data_count := rec_zero_data_count + 1;
     test_data := FALSE;
     exit;
    end
   else if (tps_data [detector_element] > 0 ) and
    (tps_data [detector_element] <
     tps_stp.swd_setting [detector_element, lower]) then
    begin
     low_data_count := low_data_count + 1;
     rec_low_data_count := rec_low_data_count + 1;
     test_data := FALSE;
     exit;
    end
   else if (tps_data [detector_element] >
     tps_stp.swd_setting [detector_element, upper]) and
     (tps_data [detector_element] < 8064)
     then
    begin
     high_data_count := high_data_count + 1;
     rec_high_data_count := rec_high_data_count + 1;
     test_data := FALSE;
     exit;
    end
   else if (tps_data [detector_element] > 8064) then
    begin
     overflow_count := overflow_count + 1;
     rec_overflow_count := rec_overflow_count + 1;
     test_data := FALSE;
     exit;
    end;
  end;
 test_data := TRUE;
end; { procedure test_data }

{ ************************************************************************** }
```

```
procedure obtain_tps_data;
var
 ch, df_ch : char;
 i, j, k, y, detector_element : integer;
 wedge_1_data, strip_1_data, z_1_data, TAC_data : integer;
 wedge_2_data, strip_2_data, z_2_data : integer;
 wedge_fraction, strip_fraction, z_fraction : real;
 total_charge : word;
 upper_x_pos, upper_y_pos, lower_x_pos, lower_y_pos : real;
 x_bin, y_bin : integer;
 Q1, Q2 : integer;
 result : word;
 tps_raw_data : tps_raw_data_pointer;
 tps_data : tps_int_column_array;
 test_data_ok, done : boolean;
 e_drive_file, c_drive_file : file;
 numread, numwritten : word;
 block_buffer : array [1..4098] of char;
 test : longint;

begin
 clock_on (initial_start_time);
 ch := #13;
 set_up_Ortec_ADCs (tps_stp.lld_setting);
 done := FALSE;

 tps_refresh_count := 1;
 rec_zero_data_count    := 0;
 rec_low_data_count     := 0;
 rec_high_data_count    := 0;
 rec_overflow_count     := 0;
 rec_outside_active_area_count := 0;
 rec_non_coincidence_count := 0;
 zero_data_count    := 0;
 low_data_count     := 0;
 high_data_count    := 0;
 overflow_count     := 0;
 outside_active_area_count := 0;
 non_coincidence_count := 0;
 tps_data_count     := 0;
 total_exp_time     := 0;
 total_count_time   := 0;

 df_max := 1;
 df_plot_max := 15;
 df_min := 0;
 df_plot_min := 0;

 tps_data_read_cycles := 0;
 rec_data_read_cycles := 0;
 draw_detector_face;
 write_tps_df_legend;
 write_tps_ms_info;
 update ('Initializing');
```

```
new (tps_raw_data);
for i := 1 to rd_size_limit do
 begin
  for j := w1_col to z2_col do
   begin
    tps_raw_data^ [i, j] := 0;
   end;
 end;
new (tps_df_data);
for i := -120 to 119 do
 begin
  for j := -120 to 119 do
   begin
    tps_df_data^ [i, j] := 0;
   end;
 end;

        {

new (tps_energy_and_angle_data);
for i := 0 to 325 do
 begin
  for j := 0 to 9 do
   begin
    for k := 0 to 9 do
     begin
      tps_energy_and_angle_data^ [i, j, k] := 0;
     end;
   end;
 end;}

update ('Acquiring Data');
clock_on (start_time);
repeat  { until done }

 repeat {until }
  test_data_ok := FALSE;
  CAMCL (dataway_c);
  tps_data_read_cycles := tps_data_read_cycles + 1;
  rec_data_read_cycles := rec_data_read_cycles + 1;
  for detector_element := wedge_1 to z_2 do
   begin
    tps_data [detector_element] := 0;
   end;

  repeat
   CAMI (Ortec_ADC_1, test_LAM, module, D, Q1, X);
   CAMI (Ortec_ADC_2, test_LAM, module, D, Q2, X);
   if keypressed then
    begin
     ch := readkey;
     ch := upcase (ch);
     if (ch = ESC) or (ch = 'Q') then
```

```
      begin
       done := TRUE;
      end
    else if (ch in [LEFTARROW, RIGHTARROW, DOWNARROW, UPARROW]) then
      begin
       df_ch := ch;
       update ('Changing Disp Disc');
       repeat
        if (df_ch = RIGHTARROW) and
          (df_display_lower_discriminator < df_plot_max * 14 div 15) then
          begin
           i := round ((df_display_lower_discriminator) * 15
                    / df_plot_max);
           df_display_lower_discriminator := round (df_plot_max
                                    / 15 * (i + 1) + 1);
          end
        else if (df_ch = LEFTARROW) and
             (df_display_lower_discriminator > df_plot_max div 15) then
          begin
           i := round ((df_display_lower_discriminator) * 15
                    / df_plot_max);
           df_display_lower_discriminator := round (df_plot_max
                                    / 15 * (i - 1));
          end
        else if (df_ch = UPARROW) and
             (df_display_lower_discriminator < df_plot_max) then
          begin
           df_display_lower_discriminator :=
                    df_display_lower_discriminator + 1;
          end
        else if (df_ch = DOWNARROW) and
             (df_display_lower_discriminator > 1) then
          begin
           df_display_lower_discriminator :=
                    df_display_lower_discriminator - 1;
          end;
        write_tps_df_legend;
        repeat
          begin
           df_ch := readkey;
          end;
        until df_ch in [LEFTARROW, RIGHTARROW, DOWNARROW, UPARROW,
                                    ESC, 'Q', RETURN];
       until df_ch in [ESC, 'Q', RETURN];
       plot_tps_ms_data;
      end
    else if ch = 'L' then
      begin
       update ('Reserved Key');
      end;
    update ('Acquiring Data');
   end;
until (Q1 = 1) or (Q2 = 1) or done;
```

```
  if not done then
   begin
    delay (1);
    CAMI (Ortec_ADC_1, read_ADC_conversion,
                TAC_channel, tps_data [TAC], Q, X);
{ To run in non-coincidence mode, i.e. with no TAC 'enforcement' for flat
 field acquisition, replace the following if statement with the 'if(1=1) then
 statement following it which is normally commented out}
    if (tps_data [TAC] >= tps_stp.swd_setting [TAC, lower]) and
      (tps_data [TAC] <= tps_stp.swd_setting [TAC, upper]) then
    {if (1 = 1) then}
      begin
       CAMI (Ortec_ADC_1, read_ADC_conversion, wedge_1_channel,
                              tps_data [wedge_1], Q, X);
       CAMI (Ortec_ADC_1, read_ADC_conversion, strip_1_channel,
                              tps_data [strip_1], Q, X);
       CAMI (Ortec_ADC_1, read_ADC_conversion, z_1_channel,
                              tps_data [z_1], Q, X);
       if test_data (upper, tps_data) then
        begin
         tps_data [z_1] := round (tps_data [z_1] * tps_stp.z_scale_factor);
         tps_raw_data^ [tps_refresh_count, w1_col] := tps_data [wedge_1];
         tps_raw_data^ [tps_refresh_count, s1_col] := tps_data [strip_1];
         tps_raw_data^ [tps_refresh_count, z1_col] := tps_data [z_1];
         tps_raw_data^ [tps_refresh_count, TAC_col] := tps_data [TAC];

         CAMI (Ortec_ADC_2, read_ADC_conversion, wedge_2_channel,
                              tps_data [wedge_2], Q, X);
         CAMI (Ortec_ADC_2, read_ADC_conversion, strip_2_channel,
                              tps_data [strip_2], Q, X);
         CAMI (Ortec_ADC_2, read_ADC_conversion, z_2_channel,
                              tps_data [z_2], Q, X);
         if test_data (lower, tps_data) then
          begin
           tps_data [z_2] := round (tps_data [z_2] * tps_stp.z_scale_factor);
           tps_raw_data^ [tps_refresh_count, w2_col] := tps_data [wedge_2];
           tps_raw_data^ [tps_refresh_count, s2_col] := tps_data [strip_2];
           tps_raw_data^ [tps_refresh_count, z2_col] := tps_data [z_2];
           tps_refresh_count := tps_refresh_count + 1;
           tps_data_count := tps_data_count + 1;
           test_data_ok := TRUE;
          end;
        end;
      end
     else
      begin
       non_coincidence_count := non_coincidence_count + 1;
       rec_non_coincidence_count := rec_non_coincidence_count + 1;
      end;
    end;

  until test_data_ok or done;
  if not done then
   begin
```

```
total_charge   := tps_data [wedge_1] + tps_data [strip_1] +
                  tps_data [z_1];
wedge_fraction := tps_data [wedge_1] / total_charge;
strip_fraction := tps_data [strip_1] / total_charge;
   z_fraction := tps_data [z_1] / total_charge;
with tps_stp do
 begin
  upper_x_pos := 28399 + dac_setting [wedge_1] * 1000
                 - round (113596 * dmc_setting [wedge_1] *
                 ( tps_data [wedge_1] - tps_data [z_1] *
                 ( ( ctf_setting [upper_ctf,ws] - ctf_setting [upper_ctf,wz] )
                 / ( 1.0 - 3.0 * ctf_setting [upper_ctf,wz] ) ) )
                 / total_charge);
  upper_y_pos := - 4563 + dac_setting [strip_1] * 1000
                 + round (63750 * dmc_setting [strip_1] *
                 ( tps_data [strip_1] - tps_data [z_1] *
                 ( ( ctf_setting [upper_ctf,sw] - ctf_setting [upper_ctf,sz] )
                 / ( 1.0 - 3.0 * ctf_setting [upper_ctf,sz] ) ) )
                 / total_charge);
 end;
x_bin := round (upper_x_pos / 167);
y_bin := 12 + round (upper_y_pos / 184);
{ The above constant is
 df_radius [20000 in um] / 120 = 167 um per bin }
if (abs(x_bin) <= 120) and (y_bin <= 120) and
    (y_bin >= 0) then
 begin
  tps_df_data^ [x_bin, y_bin] := tps_df_data^ [x_bin, y_bin] + 1;
 end
else
 begin
  outside_active_area_count := outside_active_area_count + 1;
  rec_outside_active_area_count := rec_outside_active_area_count + 1;
 end;
total_charge   := tps_data [wedge_2] + tps_data [strip_2] +
                  tps_data [z_2];
wedge_fraction := tps_data [wedge_2] / total_charge;
strip_fraction := tps_data [strip_2] / total_charge;
   z_fraction := tps_data [z_2] / total_charge;
with tps_stp do
 begin
  lower_x_pos := - 28399 - dac_setting [wedge_2] * 1000
                 + round (113596 * dmc_setting [wedge_2] *
                 ( tps_data [wedge_2] - tps_data [z_2] *
                 ( ( ctf_setting [lower_ctf,ws] - ctf_setting [lower_ctf,wz] )
                 / ( 1.0 - 3.0 * ctf_setting [lower_ctf,wz] ) ) )
                 / total_charge);
  lower_y_pos := - 4563 + dac_setting [strip_2] * 1000
                 + round (63750 * dmc_setting [strip_2] *
                 ( tps_data [strip_2] - tps_data [z_2] *
                 ( ( ctf_setting [lower_ctf,sw] - ctf_setting [lower_ctf,sz] )
                 / ( 1.0 - 3.0 * ctf_setting [lower_ctf,sz] ) ) )
                 / total_charge);
 end;
```

```
    x_bin := round (lower_x_pos / 167);
    y_bin := -12 - round (lower_y_pos / 184);
    { The above constant is
      df_radius [20000 in um] / 120 = 167 um per bin }
    if (abs(x_bin) <= 120) and (y_bin >= -120) and
        (y_bin <= 0) then
      begin
       tps_df_data^ [x_bin, y_bin] := tps_df_data^ [x_bin, y_bin] + 1;
      end
    else
      begin
       outside_active_area_count := outside_active_area_count + 1;
       rec_outside_active_area_count := rec_outside_active_area_count + 1;
      end;
    if tps_accumulate_mode then
      begin
       if keypressed then
         begin
          done := TRUE;
         end;
      end
    else if (tps_data_count >= tps_shot_number) then
      begin
       done := TRUE;
      end;
  end;

if (tps_refresh_count >= tps_refresh_limit) or done then
  begin
    count_time := elapsed_time (start_time);
    total_count_time := total_count_time + count_time;
    total_exp_time := round (elapsed_time (initial_start_time));
    update ('New Data to e:');
    assign (bin_data_file, 'e:\temp\tps_rd.bin');
    {SI-}
    reset (bin_data_file, 2);
    IOErrorCode := IOResult;
    if (IOErrorCode <> 0) or (tps_data_count = tps_refresh_count - 1) then
      begin
       rewrite (bin_data_file, 2);
       reset (bin_data_file, 2);
      end;
    {SI+}
    test := filesize(bin_data_file);
    seek (bin_data_file, filesize(bin_data_file));
    blockwrite (bin_data_file, tps_raw_data^, (tps_refresh_count - 1) * 7, result);
    close (bin_data_file);

    plot_tps_ms_data;
    write_tps_df_legend;
    write_tps_ms_info;

    if diskfree(5) < (rd_size_limit * 14) then
      begin
```

```
        done := TRUE;
       end
     else if not done then
       begin
        tps_refresh_count := 1;
        rec_data_read_cycles   := 0;
        rec_zero_data_count    := 0;
        rec_low_data_count     := 0;
        rec_high_data_count    := 0;
        rec_overflow_count     := 0;
        rec_outside_active_area_count := 0;
        rec_non_coincidence_count := 0;
        clock_on (start_time);
        update ('Acquiring Data');
       end;
   end;
until done;
dispose (tps_raw_data);
if (DiskFree(0) > 60000) then
 begin
  update ('Detector Face to c:');
  assign (bin_data_file, 'c:\turbo\swdev\tps\tps_df.bin');
  {$I-}
  reset (bin_data_file, 1);
  IOErrorCode := IOResult;
  if IOErrorCode <> 0 then
   begin
    rewrite (bin_data_file, 1);
    reset (bin_data_file, 1);
   end;
  {$I+}
  blockwrite (bin_data_file, tps_df_data^, 240 * 240, result);
  close (bin_data_file);
  dispose (tps_df_data);
 end;


update ('Write Raw Data');
assign (int_data_file, 'e:\temp\tps_rd.bin');
reset (int_data_file);
if DiskFree(0) > (filesize(int_data_file) * 6) then
 begin
  assign (text_data_file, 'c:\turbo\swdev\tps\tps_rd.txt');
  rewrite (text_data_file);
  repeat
   if not eof (int_data_file) then
     begin
      read (int_data_file, wedge_1_data, strip_1_data, z_1_data,
                TAC_data, wedge_2_data, strip_2_data, z_2_data);
      writeln (text_data_file, wedge_1_data:4, #9, strip_1_data:4, #9,
                        z_1_data:4, #9, TAC_data:4, #9,
                        wedge_2_data:4, #9, strip_2_data:4, #9,
                        z_2_data:4);
     end;
```

```
    until eof (int_data_file);
    close (int_data_file);
    close (text_data_file);
   end
  else
   begin
    close (int_data_file);
   end;

                          (*

  if (DiskFree(0) > 2000000) then
   begin
    update ('Write E & A Data');
    assign (text_data_file, 'c:\frbm\dyn\tps_ea.txt');
    rewrite (text_data_file);
    new (tps_energy_and_angle_data);
    for i := 0 to 325 do
     begin
      writeln (text_data_file, 'Energy = ', i:3);
      for j := 0 to 9 do
       begin
        writeln (text_data_file, 'Theta = ', j:2);
        writeln (text_data_file, ' 0-9  9-18  18-27  27-36 36-45  45-54',
                          '  54-63  63-72  72-81  81-90  ');
        for k := 0 to 9 do
         begin
          write (text_data_file, tps_energy_and_angle_data^ [i, j, k]:4);
         end;
        writeln (text_data_file);
       end;
      writeln (text_data_file);
     end;
    close (text_data_file);
    dispose (tps_energy_and_angle_data);
   end;
                      *)
  update ('!!! DONE !!!');
  set_done_flag;
  RestoreCRTMode;
  GraphicsOn := FALSE;
end;

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

procedure get_tps_shot_number;
begin { procedure get_tps_shot_number }
  if ask_for_long_integer (tps_shot_number, 'Shots per Scan : ',
    1, 2000000000) then
    show_tps_parameters;
end { procedure get_tps_shot_number };

{ ************************************************************************** }
```

```
procedure show_tps_parameters;
var
 wind_y : byte;
begin { procedure show_txs_plot_parameters }
 wind_y := Hi(WindMin);
 window (37, 1, 80, 24);  clrscr;
 clrscr; textcolor (9);
 GoToXY (1,2);
 if tps_accumulate_mode = FALSE then
  begin
   writeln ('            Multiple Shot');
  end
 else
  begin
   writeln ('            Accumulate ...');
  end;
 GoToXY (1,7);   textcolor (LightRed);
 if tps_accumulate_mode = FALSE then
  begin
   write ('       Shot Number :  ');
   writeln (tps_shot_number:12);
  end
 else
  begin
   writeln ('       Shot Number : Until Stopped');
  end;
 writeln;
 writeln ('     Lower Display');
 writeln ('     Discriminator : ', df_display_lower_discriminator:12);
 window (1, wind_y, 80, wind_y);
 end;  { procedure show_tps_parameters }


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

procedure show_tps_menu;
 begin
  if not GraphicsOn then
   begin
    window (1, 1, 37, 24);
    clrscr; textcolor (15);
    writeln ('M : Multiple Shots (see Shot #)');
    writeln ('U : Until Next Key is Pressed');
    writeln;
    writeln ('T : Begin Dynamics Experiment');
    writeln;
    writeln ('N : Set Shot Number');
    writeln;
    writeln;
    writeln ('D : Adjust Detector Constants');
    writeln ('C : Adjust Detector CT Factors');
    writeln;
    writeln ('L : Change Display Discriminator');
    writeln;
```

```
      writeln ('P : Plot Menu');
      writeln;
      writeln ('S : Save DSR Data File');
      writeln ('R : Read DSR Data File');
      writeln;
      writeln ('Q : Quit Test DSR Menu');
      writeln;
      writeln ('----------------------------------');
    end;
 end;

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

procedure tps_menu;
var
 ch : char;
 i, y, axis : integer;
begin { procedure tps_menu }
 experiment_type := tps;
 writeln (MemAvail, ' bytes available');
 writeln ('Largest free block is ', MaxAvail, ' bytes');
 writeln (Disksize(5) div 1024, ' kilobytes capacity on e: drive');
 writeln (DiskFree(5) div 1024, ' kilobytes free on e: drive');
 delay (1000);

 df_plot_max := 15;
 repeat
  if not tps_accumulate_mode then
   begin
    if GraphicsOn then
     begin
      RestoreCRTMode;
      GraphicsOn := FALSE;
      show_tps_parameters;
     end;
   end;
  if not GraphicsOn then
   begin
    show_tps_parameters;
    show_tps_menu;
    y := wherey;
    repeat
     window (1, y, 80, 24); clrscr;
     write ('? > ');
     ch := Readkey; ch := upcase(ch); writeln (ch); writeln;
    until ch in ['U', 'M', 'T', 'N', 'F', 'D', 'C', 'L',
                 'P', 'S', 'R', #13, 'Q'];
   end
  else
   begin
    repeat
     begin
      ch := Readkey; ch := upcase (ch);
     end;
```

```
  until ch in ['U', 'M', 'T', 'N', 'F', 'D', 'C', 'L',
               'P', 'S', 'R', #13, 'Q'];
 end;
case ch of
 'M' :
  begin
   if not GraphicsOn then
    begin
     tps_accumulate_mode := FALSE;
    end;
  end;
 'U' :
  begin
   if not GraphicsOn then
    begin
     tps_accumulate_mode := TRUE;
    end;
  end;
 'T' :
  begin
   if tps_accumulate_mode then
    begin
     obtain_tps_data;
    end;
  end;
 'N' :
  begin
   if not GraphicsOn then
    begin
     get_tps_shot_number;
    end;
  end;
 'D' :
  begin
   dcc_menu;
   show_tps_parameters;
  end;
 'C' :
  begin
   ctf_menu;
   show_tps_parameters;
  end;
 'L' :
  begin
   repeat
    y := wherey + Hi(WindMin);
    window (1, y, 80, y);
    textcolor (14);
    begin
     write ('Press ', #24, '/', #25,
            'to raise/lower DSR Display Lower Discriminator : ');
     repeat
      begin
       ch := readkey;
```

```
        end;
      until ch in [LEFTARROW, RIGHTARROW, DOWNARROW, UPARROW,
                          ESC, 'Q', #13];
      if (ch = RIGHTARROW) and
        (df_display_lower_discriminator < df_plot_max * 14 div 15) then
        begin
         write (#24);  delay (100);
         i := round ((df_display_lower_discriminator) * 15
                  / df_plot_max);
         df_display_lower_discriminator := round (df_plot_max
                                  / 15 * (i + 1) + 1);
        end
      else if (ch = LEFTARROW) and
           (df_display_lower_discriminator > df_plot_max div 15) then
        begin
         write (#25);  delay (100);
         i := round ((df_display_lower_discriminator) * 15
                  / df_plot_max);
         df_display_lower_discriminator := round (df_plot_max
                                  / 15 * (i - 1));
        end
      else if (ch = UPARROW) and
           (df_display_lower_discriminator < df_plot_max) then
        begin
         write (#24);  delay (100);
         df_display_lower_discriminator :=
                  df_display_lower_discriminator + 1;
        end
      else if (ch = DOWNARROW) and
           (df_display_lower_discriminator > 1) then
        begin
         write (#25);  delay (100);
         df_display_lower_discriminator :=
                  df_display_lower_discriminator - 1;
        end;
     end;
     show_tps_parameters;
    until ch in [ESC, 'Q', RETURN];
  end;
'P' :
  begin
   tps_plot_menu;
  end;
'S' :
  begin
   reserved;  {save_data_file;}
  end;
'R' :
  begin
   reserved;  {read_data_file;}
  end;
#13 :
  begin
   if GraphicsOn then
```

```
        begin
         RestoreCRTMode;
         GraphicsOn := FALSE;
        end;
       show_tps_parameters;
      end;
    'Q' :
     begin
      if GraphicsOn then
       begin
        RestoreCRTMode;
        GraphicsOn := FALSE;
       end;
     end;
   end;
 until (ch = 'Q');

 experiment_type := tps;
 window (1, 1, 80, 24); textcolor (15); clrscr;
end; { procedure tps_menu }

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

end. { unit tps_exp }
```

2.11 dfplot.pas

```
{ dfplot.pas                                    }

unit dfplot;

{$O+}
{$F+}

interface

uses tps_glob,tps_util,plotutil,graph,grafutil,CRT,DOS;

procedure update (display_str : string_80);
procedure draw_detector_face;
procedure plot_tps_ms_data;
procedure write_tps_df_legend;
procedure write_tps_ms_info;

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

implementation

procedure update (display_str : string_80);
begin
if GraphicsOn = FALSE then
```

```
    begin
      exit;
    end;
  DefineWindow (750, 0, 1000, 100);
  DefineUserCoordinates (1, 1, 20, 5);
  SetFillStyle (EmptyFill, EmptyFill);
  MoveUser (1, 1);
  DrawBarUser (20, 2);
  SetColor (Magenta);
  bSetTextStyle (Font, HorizDir, 1);
  SetTextJustify (LeftText, TopText);
  OutTextXYUser (1, 1, display_str);
end;

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

procedure draw_detector_face;
const
  field_of_view_radius = 19000;
var
  i : integer;
  blocker_half_width : real;
begin
  if GraphicsOn = FALSE then
    begin
      SetGraphMode (GraphMode);
      GraphicsOn := TRUE;
    end;
  SetColor (3);
  SetFillStyle (InterleaveFill, 3);
  DefineWindow (250, 0, 1000, 1000);
  DefineUserCoordinates (-df_radius, df_radius, df_radius, -df_radius);
  DrawCircleUser (0, 0, df_radius);
  DrawCircleUser (0, 0, field_of_view_radius);
  blocker_half_width := tps_stp.blocking_strip_full_width * 500;
  MoveUser (-field_of_view_radius, -blocker_half_width);
  DrawBarUser (field_of_view_radius,  blocker_half_width);
end;

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

procedure plot_tps_ms_data;
var
  i, j, FillColor : integer;
begin
  update ('Plotting Data');
  SetPalette (1,  Magenta);
  SetPalette (2,  LightMagenta);
  SetPalette (3,  DarkGray);
  SetPalette (4,  Blue);
  SetPalette (5,  LightBlue);
  SetPalette (6,  Cyan);
  SetPalette (7,  LightCyan);
  SetPalette (8,  Green);
```

```
  SetPalette (9, LightGreen);
  SetPalette (10, Brown);
  SetPalette (11, Yellow);
  SetPalette (12, Red);
  SetPalette (13, LightRed);
  SetPalette (14, LightGray);
  SetPalette (15, White);
  DefineWindow (250, 0, 1000, 1000);
  DefineUserCoordinates (-120, 120, 120, -120);
  for i := -120 to 119 do
    begin
      for j := -120 to 119 do
        begin
          if tps_df_data^ [i, j] >= df_display_lower_discriminator then
            begin
              FillColor := trunc ((tps_df_data^ [i, j]
                          / (df_plot_max + 1)) * 15) + 1;
              SetFillStyle (SolidFill, FillColor);
              MoveUser (i, j);
              DrawBarUser (i + 1, j + 1);
            end;
        end;
    end;
end;

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

procedure write_tps_df_legend;
var
  higher_color_value_label, lower_color_value_label : string;
  color_value_label_string : string;
  tps_display_lower_discriminator_label : string;
  higher_color_value, lower_color_value : word;
  i, lowest_visible_color : integer;
begin  { procedure write_tps_df_legend }
  if GraphicsOn = FALSE then
    begin
      SetGraphMode (GraphMode);
      GraphicsOn := TRUE;
    end;
  DefineWindow (0, 500, 40, 1000);
  DefineUserCoordinates (0, 16, 1, 1);
  SetPalette (1,  Magenta);
  SetPalette (2,  LightMagenta);
  SetPalette (3,  DarkGray);
  SetPalette (4,  Blue);
  SetPalette (5,  LightBlue);
  SetPalette (6,  Cyan);
  SetPalette (7,  LightCyan);
  SetPalette (8,  Green);
  SetPalette (9,  LightGreen);
  SetPalette (10, Brown);
  SetPalette (11, Yellow);
  SetPalette (12, Red);
```

```
  SetPalette (13, LightRed);
  SetPalette (14, LightGray);
  SetPalette (15, White);
  lowest_visible_color := round (df_display_lower_discriminator * 15
                        / df_plot_max);
  for i := lowest_visible_color to 15 do
   begin
    SetFillStyle (SolidFill, i);
    MoveUser (0, i);
    DrawBarUser (1, (i + 1));
   end;
  { label colors }
  bSetTextStyle (Font, HorizDir, 1);
  SetTextJustify (LeftText, TopText);
  SetFillStyle (EmptyFill, EmptyFill);
  Bar (40, MaxY, 155, MaxY div 2);
  for i := lowest_visible_color to 15 do
   begin
    SetColor (i);
    higher_color_value := round (df_plot_max / 15 * i + 1);
    str (higher_color_value:5, higher_color_value_label);
    lower_color_value := round (df_plot_max / 15 * (i - 1) + 1);
    str (lower_color_value:6, lower_color_value_label);
    color_value_label_string := lower_color_value_label + ' -' +
                        higher_color_value_label;
    OutTextXY (30, (MaxY div 31 * (32 - i) - i + 5),
              color_value_label_string);
   end;
  str (df_display_lower_discriminator:1, tps_display_lower_discriminator_label);
  tps_display_lower_discriminator_label := 'Disc : ' +
                              tps_display_lower_discriminator_label;
  bSetTextStyle (Font, HorizDir, 1);
  SetTextJustify (LeftText, TopText);
  SetFillStyle (EmptyFill, EmptyFill);
  Bar (175, MaxY, 300, MaxY * 46 div 48);
  SetColor (lowest_visible_color);
  OutTextXY (175, (MaxY - 9), tps_display_lower_discriminator_label)
end;  { procedure write_tps_df_legend }

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

procedure write_tps_ms_info;
var
  validnumstr, goodstr, zerostr, lowstr, highstr : string;
  overstr, oaastr, ncostr : string;
  recentvalidnumstr, recentgoodstr, recentzerostr : string;
  recentlowstr, recenthighstr : string;
  recentoverstr, recentoaastr, recentncostr : string;
  hstr, mstr, sstr : string;
  x_pos, y_pos : string;
  refresh_limit_str : string;
  count_rate_str, average_count_rate_str : string;
  count_time_str, total_count_time_str, total_exp_time_str : string;
  good, zero, low, high, over, oaa, nco : real;
```

```
 rgood, rzero, rlow, rhigh, rover, roaa, rnco : real;
begin  { procedure write_tps_ms_info }
 if GraphicsOn = FALSE then
  begin
   SetGraphMode (GraphMode);
   GraphicsOn := TRUE;
  end;
 DefineWindow (0, 0, 300, 500);
 DefineUserCoordinates (1, 1, 24, 25);
 SetColor (LightRed);
 bSetTextStyle (Font, HorizDir, 1);
 SetTextJustify (LeftText, TopText);
 OutTextXY (1, 1, 'Events  Last[%]  Cum[%]');
 if rec_data_read_cycles = 0 then
  begin
   rec_data_read_cycles := 1;
   if tps_data_read_cycles = 0 then
    begin
     tps_data_read_cycles := 1;
    end;
  end;
 good := tps_data_count / tps_data_read_cycles * 100;
 zero := zero_data_count / tps_data_read_cycles * 100;
 low := low_data_count / tps_data_read_cycles * 100;
 high := high_data_count / tps_data_read_cycles * 100;
 over := overflow_count / tps_data_read_cycles * 100;
 oaa := outside_active_area_count / tps_data_read_cycles * 100;
 nco := non_coincidence_count / tps_data_read_cycles * 100;
 rgood := (tps_refresh_count - 1) / rec_data_read_cycles * 100;
 rzero := rec_zero_data_count / rec_data_read_cycles * 100;
 rlow := rec_low_data_count / rec_data_read_cycles * 100;
 rhigh := rec_high_data_count / rec_data_read_cycles * 100;
 rover := rec_overflow_count / rec_data_read_cycles * 100;
 roaa := rec_outside_active_area_count / rec_data_read_cycles * 100;
 rnco := rec_non_coincidence_count / rec_data_read_cycles * 100;

 str (good:5:1, goodstr);
 str (zero:5:1, zerostr);
 str (low:5:1, lowstr);
 str (high:5:1, highstr);
 str (over:5:1, overstr);
 str (oaa:5:1, oaastr);
 str (nco:5:1, ncostr);
 str (rgood:5:1, recentgoodstr);
 str (rzero:5:1, recentzerostr);
 str (rlow:5:1, recentlowstr);
 str (rhigh:5:1, recenthighstr);
 str (rover:5:1, recentoverstr);
 str (roaa:5:1, recentoaastr);
 str (rnco:5:1, recentncostr);

 goodstr := ' Valid : ' + recentgoodstr + '   ' + goodstr;
 zerostr := '  Zero : ' + recentzerostr + '   ' + zerostr;
 lowstr := '   Low : ' + recentlowstr + '   ' + lowstr;
```

```
highstr := '  High : ' + recenthighstr + '   ' + highstr;
overstr := '  O.F. : ' + recentoverstr + '   ' + overstr;
oaastr := 'O.A.A. : ' + recentoaastr + '   ' + oaastr;
ncostr := 'Non Co : ' + recentncostr + '   ' + ncostr;
SetFillStyle (EmptyFill, EmptyFill);
MoveUser (1, 2);
DrawBarUser (24, 13);
SetColor (LightRed);
OutTextXYUser (1, 3, goodstr);
OutTextXYUser (1, 4, zerostr);
OutTextXYUser (1, 5, lowstr);
OutTextXYUser (1, 6, highstr);
OutTextXYUser (1, 7, overstr);
OutTextXYUser (1, 8, oaastr);
OutTextXYUser (1, 9, ncostr);
if (tps_data_count > 0) and (tps_refresh_count > 1) then
  begin
    str (count_time:5:1, count_time_str);
    count_time_str := 'Times  : ' + count_time_str + ' s';
    OutTextXYUser (1, 11, count_time_str);
    count_rate := tps_refresh_count / count_time;
    str (count_rate:5:1, count_rate_str);
    count_rate_str := 'Rates  : ' + count_rate_str + ' Hz';
    OutTextXYUser (1, 12, count_rate_str);
  end;

SetFillStyle (EmptyFill, EmptyFill);
MoveUser (1, 18);
DrawBarUser (16, 24);

str ((tps_refresh_limit - 1):1, refresh_limit_str);
refresh_limit_str := 'Refresh : ' + refresh_limit_str;
str (tps_data_count:1, validnumstr);
validnumstr := ' Points : ' + validnumstr;
h := total_exp_time div 3600;
m := (total_exp_time mod 3600) div 60;
s := (total_exp_time mod 3600) mod 60;
hstr := W2S (h, 0);
if length (hstr) = 1 then
begin
  hstr := '0' + hstr;
end;
mstr := W2S (m, 0);
if length (mstr) = 1 then
begin
  mstr := '0' + mstr;
end;
sstr := I2S (s, 0);
if length (sstr) = 1 then
begin
  sstr := '0' + sstr;
end;
total_exp_time_str := '    ' + hstr + ':' + mstr + ':' + sstr;
```

```
  SetColor (LightGreen);
  bSetTextStyle (Font, HorizDir, 1);
  SetTextJustify (LeftText, TopText);
  OutTextXYUser (1, 17, refresh_limit_str);
  OutTextXYUser (1, 19, 'Valid Data');
  OutTextXYUser (1, 20, validnumstr);
  OutTextXYUser (1, 22, 'Elapsed Time');
  OutTextXYUser (1, 23, total_exp_time_str);

  SetColor (LightGreen);
  bSetTextStyle (Font, HorizDir, 1);
  SetTextJustify (LeftText, TopText);
  OutTextXY (480, (MaxY - 9), 'Hit Esc to Quit ...');
end;  { procedure write_tps_ms_info }

end. { unit dfplot }
```

2.12 tps_dcc.pas

```
{ tps_dcc.pas                                                    }

unit tps_dcc;

{$O+}
{$F+}

interface

uses tps_util,tps_glob,DOS,CRT;

procedure show_dcc_parameters (dmc_setting : dcc_real_column_array;
                               dac_setting : dcc_real_column_array);
procedure get_detector_multiplier_constant_settings
        (var dmc_setting : dcc_real_column_array;
             dac_setting : dcc_real_column_array;
             ch : char);
procedure get_detector_additive_constant_settings
            (dmc_setting : dcc_real_column_array;
          var dac_setting : dcc_real_column_array;
             ch : char);
procedure dcc_menu;

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

implementation

procedure show_dcc_parameters (dmc_setting : dcc_real_column_array;
                               dac_setting : dcc_real_column_array);
var
  wind_y : byte;
begin { procedure show_dcc_parameters }
```

```
 wind_y := Hi(WindMin) + 1;
 window (37, 1, 80, 24);
 clrscr; textcolor (11);
 with tps_stp do
  begin
   write ('      Upper Wedge Channel : ');
   writeln (Ortec_ADC_channel [wedge_1]:7);
   write ('      Upper Strip Channel : ');
   writeln (Ortec_ADC_channel [strip_1]:7);
   write ('         Upper Z Channel : ');
   writeln (Ortec_ADC_channel [z_1]:7);
   write ('            TAC Channel : ');
   writeln (Ortec_ADC_channel [TAC]:7);
   write ('      Lower Wedge Channel : ');
   writeln (Ortec_ADC_channel [wedge_2]:7);
   write ('      Lower Strip Channel : ');
   writeln (Ortec_ADC_channel [strip_2]:7);
   write ('         Lower Z Channel : ');
   writeln (Ortec_ADC_channel [z_2]:7);
  end;
 writeln;
 write ('   Upper Wedge DMC Setting : ');
 writeln (dmc_setting [wedge_1]:11:3);
 write ('   Upper Strip DMC Setting : ');
 writeln (dmc_setting [strip_1]:11:3);
 write ('      Upper Z DMC Setting : ');
 writeln (dmc_setting [z_1]:11:3);
 write ('         TAC DMC Setting : ');
 writeln (dmc_setting [TAC]:11:3);
 write ('   Lower Wedge DMC Setting : ');
 writeln (dmc_setting [wedge_2]:11:3);
 write ('   Lower Strip DMC Setting : ');
 writeln (dmc_setting [strip_2]:11:3);
 write ('      Lower Z DMC Setting : ');
 writeln (dmc_setting [z_2]:11:3);
 writeln;
 write ('   Upper Wedge DAC Setting : ');
 writeln (dac_setting [wedge_1]:11:3);
 write ('   Upper Strip DAC Setting : ');
 writeln (dac_setting [strip_1]:11:3);
 write ('   Lower Wedge DAC Setting : ');
 writeln (dac_setting [wedge_2]:11:3);
 write ('   Lower Strip DAC Setting : ');
 writeln (dac_setting [strip_2]:11:3);
 window (1, wind_y, 80, (wind_y + 1));
end; { procedure show_dcc_parameters }

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

procedure get_detector_multiplier_constant_settings
      (var dmc_setting : dcc_real_column_array;
          dac_setting : dcc_real_column_array;
          ch : char);
var
```

```
 y : integer;
begin { procedure get_detector_multiplier_constant_settings }
 case ch of
  'A': begin
    y := wherey + Hi(WindMin);
    window ( 1, y, 80, y);
    if ask_for_real (dmc_setting [wedge_1],
      ' New Upper Wedge Detector Multiplier Constant Setting : ', 0, 10) then
                begin
                 show_dcc_parameters (dmc_setting, dac_setting);
                 textcolor (14);
                end;
    if ask_for_real (dmc_setting [strip_1],
      ' New Upper Strip Detector Multiplier Constant Setting : ', 0, 10) then
                begin
                 show_dcc_parameters (dmc_setting, dac_setting);
                 textcolor (14);
                end;
    if ask_for_real (dmc_setting [z_1],
      '    New Upper Z Detector Multiplier Constant Setting : ', 0, 10) then
                begin
                 show_dcc_parameters (dmc_setting, dac_setting);
                 textcolor (14);
                end;
    if ask_for_real (dmc_setting [TAC],
      '       New TAC Detector Multiplier Constant Setting : ', 0, 10) then
                begin
                 show_dcc_parameters (dmc_setting, dac_setting);
                 textcolor (14);
                end;
    if ask_for_real (dmc_setting [wedge_2],
      ' New Lower Wedge Detector Multiplier Constant Setting : ', 0, 10) then
                begin
                 show_dcc_parameters (dmc_setting, dac_setting);
                 textcolor (14);
                end;
    if ask_for_real (dmc_setting [strip_2],
      ' New Lower Strip Detector Multiplier Constant Setting : ', 0, 10) then
                begin
                 show_dcc_parameters (dmc_setting, dac_setting);
                 textcolor (14);
                end;
    if ask_for_real (dmc_setting [z_2],
      '    New Lower Z Detector Multiplier Constant Setting : ', 0, 10) then
                begin
                 show_dcc_parameters (dmc_setting, dac_setting);
                 textcolor (14);
                end;
    end;

  '1' : begin
    if ask_for_real (dmc_setting [wedge_1],
      ' New Upper Wedge Detector Multiplier Constant Setting : ', 0, 10) then
                show_dcc_parameters (dmc_setting, dac_setting);
```

```
      end;
  '2' : begin
   if ask_for_real (dmc_setting [strip_1],
     '   New Upper Strip Detector Multiplier Constant Setting : ', 0, 10) then
           show_dcc_parameters (dmc_setting, dac_setting);
      end;
  '3' : begin
   if ask_for_real (dmc_setting [z_1],
     '      New Upper Z Detector Multiplier Constant Setting : ', 0, 10) then
           show_dcc_parameters (dmc_setting, dac_setting);
      end;
  '4' : begin
   if ask_for_real (dmc_setting [TAC],
     '         New TAC Detector Multiplier Constant Setting : ', 0, 10) then
           show_dcc_parameters (dmc_setting, dac_setting);
      end;
  '5' : begin
   if ask_for_real (dmc_setting [wedge_2],
     '   New Lower Wedge Detector Multiplier Constant Setting : ', 0, 10) then
           show_dcc_parameters (dmc_setting, dac_setting);
      end;
  '6' : begin
   if ask_for_real (dmc_setting [strip_2],
     '   New Lower Strip Detector Multiplier Constant Setting : ', 0, 10) then
           show_dcc_parameters (dmc_setting, dac_setting);
      end;
  '7' : begin
   if ask_for_real (dmc_setting [z_2],
     '      New Lower Z Detector Multiplier Constant Setting : ', 0, 10) then
           show_dcc_parameters (dmc_setting, dac_setting);
      end;
  end;
end { procedure get_detector_multiplier_constant_settings };

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

procedure get_detector_additive_constant_settings
        (dmc_setting : dcc_real_column_array;
       var dac_setting : dcc_real_column_array;
         ch : char);
var
 y : integer;
begin { procedure get_detector_additive_constant_settings }
 case ch of
  'A': begin
   y := wherey + Hi(WindMin);
   window ( 1, y, 80, y);
   if ask_for_real (dac_setting [wedge_1],
     '   New Upper Wedge Detector Additive Constant Setting : ', -1e9, 1e9) then
           begin
            show_dcc_parameters (dmc_setting, dac_setting);
            textcolor (14);
           end;
   if ask_for_real (dac_setting [strip_1],
```

```
      ' New Upper Strip Detector Additive Constant Setting : ', -1e9, 1e9) then
              begin
               show_dcc_parameters (dmc_setting, dac_setting);
               textcolor (14);
              end;
    if ask_for_real (dac_setting [wedge_2],
      ' New Lower Wedge Detector Additive Constant Setting : ', -1e9, 1e9) then
              begin
               show_dcc_parameters (dmc_setting, dac_setting);
               textcolor (14);
              end;
    if ask_for_real (dac_setting [strip_2],
      ' New Lower Strip Detector Additive Constant Setting : ', -1e9, 1e9) then
              begin
               show_dcc_parameters (dmc_setting, dac_setting);
               textcolor (14);
              end;
    end;

  '1' : begin
    if ask_for_real (dac_setting [wedge_1],
      ' New Upper Wedge Detector Additive Constant Setting : ', -1e9, 1e9) then
              show_dcc_parameters (dmc_setting, dac_setting);
     end;
  '2' : begin
    if ask_for_real (dac_setting [strip_1],
      ' New Upper Strip Detector Additive Constant Setting : ', -1e9, 1e9) then
              show_dcc_parameters (dmc_setting, dac_setting);
     end;
  '5' : begin
    if ask_for_real (dac_setting [wedge_2],
      ' New Lower Wedge Detector Additive Constant Setting : ', -1e9, 1e9) then
              show_dcc_parameters (dmc_setting, dac_setting);
     end;
  '6' : begin
    if ask_for_real (dac_setting [strip_2],
      ' New Lower Strip Detector Additive Constant Setting : ', -1e9, 1e9) then
              show_dcc_parameters (dmc_setting, dac_setting);
     end;
  end;
end { procedure get_detector_additive_constant_settings };

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

procedure dcc_menu;
var
 dmc_setting : dcc_real_column_array;
 dac_setting : dcc_real_column_array;
 ch, dcc_ch : char;
 y : integer;
begin { procedure dcc_menu }
 dmc_setting := tps_stp.dmc_setting;
 dac_setting := tps_stp.dac_setting;
 show_dcc_parameters (dmc_setting, dac_setting);
```

```
repeat
 window (1, 1, 36, 24);
 clrscr; textcolor (15);
 write  ('Change the D');
 textcolor (12);
 write  ('M');
 textcolor (15);
 writeln ('C Setting On : ');
 writeln ('A : All Channels');
 writeln ('1 : Upper Wedge');
 writeln ('2 : Upper Strip');
 writeln ('3 : Upper Z');
 writeln ('4 : TAC');
 writeln ('5 : Lower Wedge');
 writeln ('6 : Lower Strip');
 writeln ('7 : Lower Z');
 writeln;
 write  ('Change the D');
 textcolor (12);
 write  ('A');
 textcolor (15);
 writeln ('C Setting On : ');
 writeln ('A : All Channels');
 writeln ('1 : Upper Wedge');
 writeln ('2 : Upper Strip');
 writeln ('5 : Lower Wedge');
 writeln ('6 : Lower Strip');
 writeln;
 writeln ('Q : Quit DCC Menu');
 writeln;
 writeln ('-----------------------------------');
 writeln; y := wherey; textcolor (14);
 repeat
  window (1, y, 80, 24); clrscr;
  write ('? > '); ch := Readkey; ch := upcase(ch); write (ch);
 until ch in ['M', 'A', 'Q'];
 if ch <> 'Q' then
  begin
   if ch = 'M' then
    begin
     repeat
      window (1, y, 80, 24); clrscr;
      write ('Which DMC do you want to change ? > ');
      dcc_ch := Readkey; dcc_ch := upcase(dcc_ch); writeln (dcc_ch);
      writeln;
     until dcc_ch in ['A', '1', '2', '3', '4', '5', '6', '7', 'Q'];
     if dcc_ch <> 'Q' then
      begin
       get_detector_multiplier_constant_settings (tps_stp.dmc_setting,
                                                  tps_stp.dac_setting,
                                                  dcc_ch);
      end
     else
      begin
```

```
            ch := 'Q';
           end;
         end;
       if ch = 'A' then
        begin
         repeat
          window (1, y, 80, 24); clrscr;
          write ('Which DAC do you want to change ? > ');
          dcc_ch := Readkey;  dcc_ch := upcase(dcc_ch);  writeln (dcc_ch);
          writeln;
         until dcc_ch in ['A', '1', '2', '5', '6', 'Q'];
         if dcc_ch <> 'Q' then
          begin
           get_detector_additive_constant_settings (tps_stp.dmc_setting,
                                                    tps_stp.dac_setting,
                                                    dcc_ch);
          end
         else
          begin
           ch := 'Q';
          end;
        end;
     end;
  until (ch = 'Q');
  window (1, 1, 80, 24); textcolor (15); clrscr;
end; { procedure dcc_menu }

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

end. { unit tps_dcc }
```

2.13 tps_ctf.pas

```
{ tps_ctf.pas                                    }

unit tps_ctf;

{$O+}
{$F+}

interface

uses tps_util,tps_glob,DOS,CRT;

procedure show_ctf_parameters (ctf_setting : ctf_real_column_array);
procedure get_detector_crosstalk_factor_settings
          (var ctf_setting : ctf_real_column_array;
           ch : char);
procedure ctf_menu;

{ ************************************************************************** }
```

```
implementation

procedure show_ctf_parameters (ctf_setting : ctf_real_column_array);
var
  wind_y : byte;
begin { procedure show_ctf_parameters }
  wind_y := Hi(WindMin) + 1;
  window (37, 1, 80, 24);
  clrscr; textcolor (11);
  write ('Upper Wedge-Strip CT Factor : ');
  writeln (ctf_setting [upper_ctf, ws]:11:3);
  write ('   Upper Wedge-Z CT Factor : ');
  writeln (ctf_setting [upper_ctf, wz]:11:3);
  write ('Upper Strip-Wedge CT Factor : ');
  writeln (ctf_setting [upper_ctf, sw]:11:3);
  write ('   Upper Strip-Z CT Factor : ');
  writeln (ctf_setting [upper_ctf, sz]:11:3);
  write ('   Upper Z-Wedge CT Factor : ');
  writeln (ctf_setting [upper_ctf, zw]:11:3);
  write ('   Upper Z-Strip CT Factor : ');
  writeln (ctf_setting [upper_ctf, zs]:11:3);
  writeln;
  write ('Lower Wedge-Strip CT Factor : ');
  writeln (ctf_setting [lower_ctf, ws]:11:3);
  write ('   Lower Wedge-Z CT Factor : ');
  writeln (ctf_setting [lower_ctf, wz]:11:3);
  write ('Lower Strip-Wedge CT Factor : ');
  writeln (ctf_setting [lower_ctf, sw]:11:3);
  write ('   Lower Strip-Z CT Factor : ');
  writeln (ctf_setting [lower_ctf, sz]:11:3);
  write ('   Lower Z-Wedge CT Factor : ');
  writeln (ctf_setting [lower_ctf, zw]:11:3);
  write ('   Lower Z-Strip CT Factor : ');
  writeln (ctf_setting [lower_ctf, zs]:11:3);
  window (1, wind_y, 80, (wind_y + 1));
end;  { procedure show_ctf_parameters }

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

procedure get_detector_crosstalk_factor_settings
          (var ctf_setting : ctf_real_column_array;
           ch : char);
var
  y : integer;
begin { procedure get_detector_crosstalk_factor_settings }
  y := wherey + Hi(WindMin);
  window (1, y, 80, y);
  case ch of
    'U' : begin
      if ask_for_real (ctf_setting [upper_ctf, ws],
        'New Upper Detector Wedge-Strip Crosstalk Subtraction Factor : ',
            -100, 100) then
              begin
```

```
          show_ctf_parameters (ctf_setting);
          textcolor (14);
         end;
 if ask_for_real (ctf_setting [upper_ctf, wz],
   '  New Upper Detector Wedge-Z Crosstalk Subtraction Factor : ',
      -100, 100) then
         begin
          show_ctf_parameters (ctf_setting);
          textcolor (14);
         end;
 if ask_for_real (ctf_setting [upper_ctf, sw],
   'New Upper Detector Strip-Wedge Crosstalk Subtraction Factor : ',
      -100, 100) then
         begin
          show_ctf_parameters (ctf_setting);
          textcolor (14);
         end;
 if ask_for_real (ctf_setting [upper_ctf, sz],
   '  New Upper Detector Strip-Z Crosstalk Subtraction Factor : ',
      -100, 100) then
         begin
          show_ctf_parameters (ctf_setting);
          textcolor (14);
         end;
 if ask_for_real (ctf_setting [upper_ctf, zw],
   '  New Upper Detector Z-Wedge Crosstalk Subtraction Factor : ',
      -100, 100) then
         begin
          show_ctf_parameters (ctf_setting);
          textcolor (14);
         end;
 if ask_for_real (ctf_setting [upper_ctf, zs],
   '  New Upper Detector Z-Strip Crosstalk Subtraction Factor : ',
      -100, 100) then
         begin
          show_ctf_parameters (ctf_setting);
          textcolor (14);
         end;
 end;
'L' : begin
 if ask_for_real (ctf_setting [lower_ctf, ws],
   'New Lower Detector Wedge-Strip Crosstalk Subtraction Factor : ',
      -100, 100) then
         begin
          show_ctf_parameters (ctf_setting);
          textcolor (14);
         end;
 if ask_for_real (ctf_setting [lower_ctf, wz],
   '  New Lower Detector Wedge-Z Crosstalk Subtraction Factor : ',
      -100, 100) then
         begin
          show_ctf_parameters (ctf_setting);
          textcolor (14);
         end;
```

```
    if ask_for_real (ctf_setting [lower_ctf, sw],
      'New Lower Detector Strip-Wedge Crosstalk Subtraction Factor : ',
          -100, 100) then
           begin
            show_ctf_parameters (ctf_setting);
            textcolor (14);
           end;
    if ask_for_real (ctf_setting [lower_ctf, sz],
      '   New Lower Detector Strip-Z Crosstalk Subtraction Factor : ',
          -100, 100) then
           begin
            show_ctf_parameters (ctf_setting);
            textcolor (14);
           end;
    if ask_for_real (ctf_setting [lower_ctf, zw],
      '   New Lower Detector Z-Wedge Crosstalk Subtraction Factor : ',
          -100, 100) then
           begin
            show_ctf_parameters (ctf_setting);
            textcolor (14);
           end;
    if ask_for_real (ctf_setting [lower_ctf, zs],
      '   New Lower Detector Z-Strip Crosstalk Subtraction Factor : ',
          -100, 100) then
           begin
            show_ctf_parameters (ctf_setting);
            textcolor (14);
           end;
   end;
 end;
end; { procedure get_detector_crosstalk_factor_settings }
{ ************************************************************************** }

procedure ctf_menu;
var
 ctf_setting : ctf_real_column_array;
 ch, ctf_ch : char;
 y : integer;
begin { procedure ctf_menu }
 ctf_setting := tps_stp.ctf_setting;
 show_ctf_parameters (ctf_setting);
 repeat
  window (1, 1, 36, 24);
  clrscr; textcolor (15);
  writeln;  writeln;
  writeln ('Change the CT Factors On : ');
  writeln;
  writeln;
  writeln ('U : Upper Detector');
  writeln;
  writeln ('L : Lower Detector');
  writeln;
  writeln;
  writeln ('Q : Quit CTF Menu');
```

```
    writeln;
    writeln ('-----------------------------------');
    writeln; y := wherey; textcolor (14);
    repeat
      window (1, y, 80, 24); clrscr;
      write ('? > '); ch := Readkey; ch := upcase(ch); write (ch);
    until ch in ['U', 'L', 'Q'];
    if ch <> 'Q' then
      begin
        get_detector_crosstalk_factor_settings (tps_stp.ctf_setting, ch);
      end;
  until (ch = 'Q');
  window (1, 1, 80, 24); textcolor (15); clrscr;
end; { procedure ctf_menu }

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

end. { unit tps_ctf }
```

2.14 tps_lld.pas

```
{ tps_lld.pas                                                    }

unit tps_lld;

{$O+}
{$F+}

interface

uses tps_util,tps_glob,DOS,CRT;

procedure show_lld_parameters (lld_setting : tps_int_column_array);
procedure get_lower_level_discriminator_settings
        (var lld_setting : tps_int_column_array; ch : char);
procedure set_up_Ortec_ADCs (lld_setting : tps_int_column_array);
procedure lld_menu;

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

implementation

{$F-}

{$I camturbo.v4}

{$F+}

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

procedure show_lld_parameters (lld_setting : tps_int_column_array);
```

```
var
 wind_y : byte;
begin { procedure show_lld_parameters }
 wind_y := Hi(WindMin) + 1;
 window (37, 1, 80, 24);
 clrscr; textcolor (11);
 with tps_stp do
  begin
   write ('      Upper Wedge Channel : ');
   writeln (Ortec_ADC_channel [wedge_1]:7);
   write ('      Upper Strip Channel : ');
   writeln (Ortec_ADC_channel [strip_1]:7);
   write ('         Upper Z Channel : ');
   writeln (Ortec_ADC_channel [z_1]:7);
   write ('            TAC Channel : ');
   writeln (Ortec_ADC_channel [TAC]:7);
   write ('      Lower Wedge Channel : ');
   writeln (Ortec_ADC_channel [wedge_2]:7);
   write ('      Lower Strip Channel : ');
   writeln (Ortec_ADC_channel [strip_2]:7);
   write ('         Lower Z Channel : ');
   writeln (Ortec_ADC_channel [z_2]:7);
   writeln;
  end;
 write ('  Upper Wedge LLD Setting : ');
 writeln (lld_setting [wedge_1]:7);
 write ('  Upper Strip LLD Setting : ');
 writeln (lld_setting [strip_1]:7);
 write ('      Upper Z LLD Setting : ');
 writeln (lld_setting [z_1]:7);
 write ('         TAC LLD Setting : ');
 writeln (lld_setting [TAC]:7);
 write ('  Lower Wedge LLD Setting : ');
 writeln (lld_setting [wedge_2]:7);
 write ('  Lower Strip LLD Setting : ');
 writeln (lld_setting [strip_2]:7);
 write ('      Lower Z LLD Setting : ');
 writeln (lld_setting [z_2]:7);
 window (1, wind_y, 80, (wind_y + 1));
end;  { procedure show_lld_parameters }

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

procedure get_lower_level_discriminator_settings
        (var lld_setting : tps_int_column_array; ch : char);
var
 y, new_setting, detector_element : integer;
begin { procedure get_lower_level_discriminator_settings }
 case ch of
  'A': begin
   y := wherey + Hi(WindMin);
   window ( 1, y, 80, y);
   if ask_for_integer (new_setting,
     'New Lower Level Discriminator Settings [mV] : ', 0, 512) then
```

```
                begin
                 for detector_element := wedge_1 to z_2 do
                   begin
                    lld_setting [detector_element] := new_setting;
                   end;
                 show_lld_parameters (lld_setting);
                 textcolor (14);
                end;
      end;
    '1' : begin
     if ask_for_integer (lld_setting [wedge_1],
       'New Upper Wedge Lower Level Discriminator Setting [mV] : ', 0, 512) then
                show_lld_parameters (lld_setting);
      end;
    '2' : begin
     if ask_for_integer (lld_setting [strip_1],
       'New Upper Strip Lower Level Discriminator Setting [mV] : ', 0, 512) then
                show_lld_parameters (lld_setting);
      end;
    '3' : begin
     if ask_for_integer (lld_setting [z_1],
       '   New Upper Z Lower Level Discriminator Setting [mV] : ', 0, 512) then
                show_lld_parameters (lld_setting);
      end;
    '4' : begin
     if ask_for_integer (lld_setting [TAC],
       '      New TAC Lower Level Discriminator Setting [mV] : ', 0, 512) then
                show_lld_parameters (lld_setting);
      end;
    '5' : begin
     if ask_for_integer (lld_setting [wedge_2],
       'New Lower Wedge Lower Level Discriminator Setting [mV] : ', 0, 512) then
                show_lld_parameters (lld_setting);
      end;
    '6' : begin
     if ask_for_integer (lld_setting [strip_2],
       'New Lower Strip Lower Level Discriminator Setting [mV] : ', 0, 512) then
                show_lld_parameters (lld_setting);
      end;
    '7' : begin
     if ask_for_integer (lld_setting [z_2],
       '   New Lower Z Lower Level Discriminator Setting [mV] : ', 0, 512) then
                show_lld_parameters (lld_setting);
      end;
   end;
end { procedure get_lower_level_discriminator_settings };

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

procedure set_up_Ortec_ADCs (lld_setting : tps_int_column_array);
var
 detector_element : integer;
begin
 CAMCL (dataway_z);
```

```
  CAMCL (dataway_i);
  CAMO (Ortec_ADC_1, write_control_register, control_register_1_location,
              control_register_1_Ortec_1, Q, X);
  CAMO (Ortec_ADC_1, write_control_register, control_register_2_location,
              control_register_2, Q, X);
  CAMO (Ortec_ADC_2, write_control_register, control_register_1_location,
              control_register_1_Ortec_2, Q, X);
  CAMO (Ortec_ADC_2, write_control_register, control_register_2_location,
              control_register_2, Q, X);

  for detector_element := wedge_1 to z_2 do
   begin
    lld_setting [detector_element] := lld_setting [detector_element] div 2;
   end;

  CAMO (Ortec_ADC_1, write_lld_setting, wedge_1_channel,
              lld_setting [wedge_1], Q, X);
  CAMO (Ortec_ADC_1, write_lld_setting, strip_1_channel,
              lld_setting [strip_1], Q, X);
  CAMO (Ortec_ADC_1, write_lld_setting, z_1_channel,
              lld_setting [z_1], Q, X);
  CAMO (Ortec_ADC_1, write_lld_setting, TAC_channel,
              lld_setting [TAC], Q, X);
  CAMO (Ortec_ADC_2, write_lld_setting, wedge_2_channel,
              lld_setting [wedge_2], Q, X);
  CAMO (Ortec_ADC_2, write_lld_setting, strip_2_channel,
              lld_setting [strip_2], Q, X);
  CAMO (Ortec_ADC_2, write_lld_setting, z_2_channel,
              lld_setting [z_2], Q, X);
end;

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

procedure lld_menu;
var
  lld_setting : tps_int_column_array;
  ch : char;
  y : integer;
begin { procedure lld_menu }
  lld_setting := tps_stp.lld_setting;
  show_lld_parameters (lld_setting);
  repeat
    window (1, 1, 36, 24);
    clrscr; textcolor (15);
    writeln ('Change the LLD Setting On : ');
    writeln;
    writeln ('A : All Channels');
    writeln;
    writeln ('1 : Upper Wedge');
    writeln ('2 : Upper Strip');
    writeln ('3 : Upper Z');
    writeln ('4 : TAC');
    writeln ('5 : Lower Wedge');
    writeln ('6 : Lower Strip');
```

```
    writeln ('7 : Lower Z');
    writeln;
    writeln ('Q : Quit LLD Menu');
    writeln;
    writeln ('-----------------------------------');
    writeln; y := wherey; textcolor (14);
    repeat
      window (1, y, 80, 24); clrscr;
      write ('? > '); ch := Readkey; ch := upcase(ch); writeln (ch);
      writeln;
    until ch in ['A', '1', '2', '3', '4', '5', '6', '7', 'Q'];
    if ch <> 'Q' then
      begin
        get_lower_level_discriminator_settings (tps_stp.lld_setting, ch);
        set_up_Ortec_ADCs (tps_stp.lld_setting);
      end;
  until (ch = 'Q');
  window (1, 1, 80, 24); textcolor (15); clrscr;
end; { procedure lld_menu }

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

end. { unit tps_lld }
```

2.15 tps_swd.pas

```
{ tps_swd.pas                                          }

unit tps_swd;

{$O+}
{$F+}

interface

uses tps_util,tps_glob,DOS,CRT;

procedure show_swd_parameters (swd_setting : swd_int_column_array);
procedure get_software_discriminator_settings
        (var swd_setting : swd_int_column_array; ch : char);
procedure swd_menu;

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

implementation

{$F-}

{$I camturbo.v4}

{$F+}
```

```
{ ************************************************************************** }

procedure show_swd_parameters (swd_setting : swd_int_column_array);
var
 wind_y : byte;
begin { procedure show_swd_parameters }
 wind_y := Hi(WindMin) + 1;
 window (37, 1, 80, 24);
 clrscr; textcolor (11);
 with tps_stp do
  begin
   write ('          Upper Wedge Channel : ');
   writeln (Ortec_ADC_channel [wedge_1]:7);
   write ('          Upper Strip Channel : ');
   writeln (Ortec_ADC_channel [strip_1]:7);
   write ('            Upper Z Channel : ');
   writeln (Ortec_ADC_channel [z_1]:7);
   write ('              TAC Channel : ');
   writeln (Ortec_ADC_channel [TAC]:7);
   write ('          Lower Wedge Channel : ');
   writeln (Ortec_ADC_channel [wedge_2]:7);
   write ('          Lower Strip Channel : ');
   writeln (Ortec_ADC_channel [strip_2]:7);
   write ('            Lower Z Channel : ');
   writeln (Ortec_ADC_channel [z_2]:7);
  end;
 write ('  Upper Wedge Upper SWD Setting : ');
 writeln (swd_setting [wedge_1, upper]:7);
 write ('  Upper Strip Upper SWD Setting : ');
 writeln (swd_setting [strip_1, upper]:7);
 write ('    Upper Z Upper SWD Setting : ');
 writeln (swd_setting [z_1, upper]:7);
 write ('       TAC Upper SWD Setting : ');
 writeln (swd_setting [TAC, upper]:7);
 write ('  Lower Wedge Upper SWD Setting : ');
 writeln (swd_setting [wedge_2, upper]:7);
 write ('  Lower Strip Upper SWD Setting : ');
 writeln (swd_setting [strip_2, upper]:7);
 write ('    Lower Z Upper SWD Setting : ');
 writeln (swd_setting [z_2, upper]:7);
 write ('  Upper Wedge Lower SWD Setting : ');
 writeln (swd_setting [wedge_1, lower]:7);
 write ('  Upper Strip Lower SWD Setting : ');
 writeln (swd_setting [strip_1, lower]:7);
 write ('    Upper Z Lower SWD Setting : ');
 writeln (swd_setting [z_1, lower]:7);
 write ('       TAC Lower SWD Setting : ');
 writeln (swd_setting [TAC, lower]:7);
 write ('  Lower Wedge Lower SWD Setting : ');
 writeln (swd_setting [wedge_2, lower]:7);
 write ('  Lower Strip Lower SWD Setting : ');
 writeln (swd_setting [strip_2, lower]:7);
 write ('    Lower Z Lower SWD Setting : ');
```

```
  writeln (swd_setting [z_2, lower]:7);
  window (1, wind_y, 80, (wind_y + 1));
end;  { procedure show_swd_parameters }

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

procedure get_software_discriminator_settings
        (var swd_setting : swd_int_column_array; ch : char);
var
  y, new_setting, detector_element : integer;
begin { procedure get_software_discriminator_settings }
  y := wherey + Hi(WindMin);
  window ( 1, y, 80, y);
  case ch of
   'U': begin
    if ask_for_integer (new_setting,
      'New Software Upper Discriminator Settings [ADC Channels] : '
         , 0, 8064) then
          begin
           for detector_element := wedge_1 to z_2 do
             begin
              swd_setting [detector_element, upper] := new_setting;
             end;
           show_swd_parameters (swd_setting);
           textcolor (14);
          end;
      end;
   'L': begin
    if ask_for_integer (new_setting,
      'New Software Lower Discriminator Settings [ADC Channels] : ',
        0, 8064) then
          begin
           for detector_element := wedge_1 to z_2 do
             begin
              swd_setting [detector_element, lower] := new_setting;
             end;
           show_swd_parameters (swd_setting);
           textcolor (14);
          end;
      end;
   '1' : begin
    if ask_for_integer (swd_setting [wedge_1, upper],
      'New Upper Wedge Software Upper Discriminator Setting [ADC Channels] : ',
        0, 8064) then
          begin
           show_swd_parameters (swd_setting);
           textcolor (14);
          end;
    if ask_for_integer (swd_setting [wedge_1, lower],
      'New Upper Wedge Software Lower Discriminator Setting [ADC Channels] : ', 0, 8064) then
                show_swd_parameters (swd_setting);
     end;
   '2' : begin
    if ask_for_integer (swd_setting [strip_1, upper],
```

```
    'New Upper Wedge Software Upper Discriminator Setting [ADC Channels] : ',
       0, 8064) then
        begin
         show_swd_parameters (swd_setting);
         textcolor (14);
        end;
  if ask_for_integer (swd_setting [strip_1, lower],
    'New Upper Wedge Software Lower Discriminator Setting [ADC Channels] : ', 0, 8064) then
              show_swd_parameters (swd_setting);
     end;
'3' : begin
  if ask_for_integer (swd_setting [z_1, upper],
    'New Upper Wedge Software Upper Discriminator Setting [ADC Channels] : ',
       0, 8064) then
        begin
         show_swd_parameters (swd_setting);
         textcolor (14);
        end;
  if ask_for_integer (swd_setting [z_1, lower],
    'New Upper Wedge Software Lower Discriminator Setting [ADC Channels] : ', 0, 8064) then
              show_swd_parameters (swd_setting);
     end;
'4' : begin
  if ask_for_integer (swd_setting [TAC, upper],
    'New Upper Wedge Software Upper Discriminator Setting [ADC Channels] : ',
       0, 8064) then
        begin
         show_swd_parameters (swd_setting);
         textcolor (14);
        end;
  if ask_for_integer (swd_setting [TAC, lower],
    'New Upper Wedge Software Lower Discriminator Setting [ADC Channels] : ', 0, 8064) then
              show_swd_parameters (swd_setting);
     end;
'5' : begin
  if ask_for_integer (swd_setting [wedge_2, upper],
    'New Upper Wedge Software Upper Discriminator Setting [ADC Channels] : ',
       0, 8064) then
        begin
         show_swd_parameters (swd_setting);
         textcolor (14);
        end;
  if ask_for_integer (swd_setting [wedge_2, lower],
    'New Upper Wedge Software Lower Discriminator Setting [ADC Channels] : ', 0, 8064) then
              show_swd_parameters (swd_setting);
     end;
'6' : begin
  if ask_for_integer (swd_setting [strip_2, upper],
    'New Upper Wedge Software Upper Discriminator Setting [ADC Channels] : ',
     0, 8064) then
      begin
         show_swd_parameters (swd_setting);
         textcolor (14);
        end;
```

```
    if ask_for_integer (swd_setting [strip_2, lower],
      'New Upper Wedge Software Lower Discriminator Setting [ADC Channels] : ', 0, 8064) then
                show_swd_parameters (swd_setting);
       end;
   '7' : begin
    if ask_for_integer (swd_setting [z_2, upper],
      'New Upper Wedge Software Upper Discriminator Setting [ADC Channels] : ',
       0, 8064) then
         begin
            show_swd_parameters (swd_setting);
            textcolor (14);
          end;
    if ask_for_integer (swd_setting [z_2, lower],
      'New Upper Wedge Software Lower Discriminator Setting [ADC Channels] : ', 0, 8064) then
                show_swd_parameters (swd_setting);
       end;
  end;
end { procedure get_software_discriminator_settings };

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

procedure swd_menu;
var
 swd_setting : swd_int_column_array;
 ch : char;
 y : integer;
begin { procedure swd_menu }
 swd_setting := tps_stp.swd_setting;
 show_swd_parameters (swd_setting);
 repeat
  window (1, 1, 36, 24);
  clrscr; textcolor (15);
  writeln;
  writeln ('Change the SWD Settings On : ');
  writeln;
  writeln;
  writeln ('U : All Upper SWD Levels');
  writeln;
  writeln ('L : All Lower SWD Levels');
  writeln;
  writeln;
  writeln ('1 : Upper Wedge');
  writeln ('2 : Upper Strip');
  writeln ('3 : Upper Z');
  writeln ('4 : TAC');
  writeln ('5 : Lower Wedge');
  writeln ('6 : Lower Strip');
  writeln ('7 : Lower Z');
  writeln;
  writeln;
  writeln ('Q : Quit SWD Menu');
  writeln;
  writeln ('-----------------------------------');
  writeln; y := wherey; textcolor (14);
```

```
    repeat
      window (1, y, 80, 24); clrscr;
      write ('? > '); ch := Readkey; ch := upcase(ch); writeln (ch);
      writeln;
    until ch in ['U', 'L', '1', '2', '3', '4', '5', '6', '7', 'Q'];
    if ch <> 'Q' then
      begin
        get_software_discriminator_settings (tps_stp.swd_setting, ch);
      end;
  until (ch = 'Q');
  window (1, 1, 80, 24); textcolor (15); clrscr;
end; { procedure swd_menu }

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

end. { unit tps_swd }
```

LAWRENCE BERKELEY LABORATORY
UNIVERSITY OF CALIFORNIA
TECHNICAL INFORMATION DEPARTMENT
BERKELEY, CALIFORNIA 94720