

To the Graduate Council:

I am submitting herewith a thesis written by Lee Barry Straus entitled "A Data Acquisition System for Fast Ion-Atom Collision Experiments." I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Physics.

Marianne Breinig
Marianne Breinig, Major Professor

We have read this thesis
and recommend its acceptance:

Georget Condo

John J. Quinn

Accepted for the Council:

Lea Minkel
Associate Vice Chancellor and
Dean of The Graduate School

A Data Acquisition System for Fast Ion-Atom Collision Experiments

A Thesis
Presented for the
Master of Science
Degree
The University of Tennessee, Knoxville

Lee Barry Straus
August 1997

Acknowledgments

First and foremost, I would like to thank my advisor, Dr. Marianne Breinig. Her guidance and understanding have been so important to my graduate education; I hope she realizes just how much she is appreciated. I would also like to thank Dr. G.T. Condo and Dr. J.J. Quinn for their physics instruction and for their invaluable time serving on my committee. A special thank you goes to Dr. Dharmin Desai who, no matter how busy he was, always took the time to help me with the many problems I encountered. Finally, I would like to thank my parents for their endless love and support, but mere words cannot express how I truly cherish them both.

Abstract

A new data acquisition system has been created. It is designed to let the experimenter make full use of the capabilities of several multi-spectrometer apparatus. By accessing two digital I/O interfaces which have been installed in a personal computer, the new system is capable of simultaneously counting events from six different sources and can acquire two-dimensional spectra which may be measured in coincidence with four different events. The new system reads, processes, displays, and stores all of these data.

In order to demonstrate and test some of the capabilities of the new data acquisition system, we have measured the elastic scattering cross section for electrons incident upon argon gas and scattered into the angular range from 45° to 135° with respect to the beam axis for two different electron energies (500 eV, 800 eV). We have also measured the angular distribution of $L_{2,3}M_{2,3}M_{2,3}$ Auger electrons produced in coincidence with Ar^{2+} target recoil ions.

Our experimentally measured elastic scattering cross sections are compared with previously published cross sections and the agreement is quite good. The Auger electron-recoil ion results have poor statistics but prove that the new data acquisition system can successfully make coincidence measurements. The versatility of the new system will allow it to be used with many different experimental setups which may be required for future ion-atom collision experiments.

Table of Contents

1	Introduction	1
2	Theory	4
2.1	Elastic Scattering	4
2.2	Auger Effect	10
3	Experiment	14
3.1	Multi-Spectrometer Apparatus	15
3.1.1	Gas Cell	16
3.1.2	Electron Gun	17
3.1.3	Parallel-Plate Analyzer	18
3.1.4	Cylindrical Mirror Analyzer	21
3.1.5	Time-of-Flight Recoil Ion Analyzer	27
3.2	Electronic Signal Detection	30
4	Data Acquisition	35
5	Data Analysis and Results	46
5.1	Elastic Scattering Data	46
5.2	Coincidence Data	57

5.3	Summary	61
	References	63
	Appendices	66
	Appendix A	67
	Appendix B	71
	Appendix C	80
	Vita	112

List of Figures

Figure 2.1: Doubly differential elastic scattering cross section of electrons from Ar for electron energies of 50 to 1000 eV and for scattering angles from 5° to 180°.	9
Figure 2.2: Comparison of calculated cross sections with published cross sections for elastically scattered 200 and 800 eV electrons from Ar. Lines represent the calculations.	9
Figure 2.3: High-resolution K Auger spectra of multiply-ionized Ne produced by 33-MeV O ⁵⁺ . The e ⁻ + Ne spectrum is given for comparison. (from D.L. Matthews, B.M. Johnson, J.J. Mackey, and C.F. Moore, Phys. Rev. Lett. <u>31</u> , 1331 (1973))	12
Figure 2.4: The L _{2,3} MM Auger spectrum of argon. The spectrum and the numbers refer to lines resolved or partly resolved under higher resolution. (from L.O. Werme, T. Bergmark, and K. Siegbahn, Physica Scripta 8, 149 (1973))	13
Figure 3.1: Schematic diagram of the multi-spectrometer apparatus.	16
Figure 3.2: Schematic diagram of a parallel-plate analyzer.	19
Figure 3.3: Schematic diagram of the cylindrical mirror analyzer.	21
Figure 3.4: Schematic diagram of a single stage of the CMA.	22
Figure 3.5: Schematic drawing of the position-sensitive detector of the CMA.	27
Figure 3.6: Schematic diagram of the time-of-flight spectrometer.	28
Figure 3.7: Schematic diagram of the signal detection electronics.	33
Figure 3.8: A typical ADC spectrum shows the different Ar recoil-ion charge states appearing as different peaks resting on a background of random coincidences. (from F. Segner, Ph.D. Dissertation, Johann Wolfgang Goethe University, 1996 (unpublished))	33
Figure 4.1: The graphical user interface for <i>CRUNCHER.FRM</i> .	39

Figure 4.2: Diagram of level mode read handshake timing. (from National Instruments Corporation, <i>AT-DIO-32F User Manual</i> , (Austin, 1990))	42
Figure 4.3: The graphical user interface for <i>MCS.FRM</i> .	43
Figure 4.4: The graphical user interface for <i>COINS.FRM</i> .	45
Figure 5.1: Normalized yields for the elastic scattering of 500 eV electrons incident upon Ar for the target gas pressures of 0.0, 0.11, 0.22, and 0.62 mTorr.	47
Figure 5.2: Normalized yields for the elastic scattering of 800 eV electrons incident upon Ar for the target gas pressures of 0.0, 0.11, 0.22, and 0.62 mTorr.	48
Figure 5.3: Contour plot of the arbitrarily normalized distribution of 500 eV elastically scattered electrons incident upon Ar for a target gas pressure of 0.62 mTorr. The most forward direction is toward the left.	49
Figure 5.4: Contour plot of the arbitrarily normalized distribution of 800 eV elastically scattered electrons incident upon Ar for a target gas pressure of 0.62 mTorr. The most forward direction is toward the left.	50
Figure 5.5: Diagram of the 14 designated regions on the PSD used in mapping the average emission angles.	51
Figure 5.6: The number of electrons reaching region 9 on the PSD versus emission angle with respect to the beam axis.	51
Figure 5.7: Comparison of the published cross sections for elastically scattered 500 eV electrons from Ar with the experimental cross sections obtained by using a transmission function from the simulation program.	54
Figure 5.8: Comparison of the published cross sections for elastically scattered 800 eV electrons from Ar with the experimental cross sections obtained by using a transmission function from the simulation program.	55

Figure 5.9: Comparison of the published cross sections for elastically scattered 500 and 800 eV electrons from Ar with the experimental cross sections obtained with the calibrated CMA.	56
Figure 5.10: The ADC spectrum of the Ar^{2+} recoil-ion charge state peak resting on a background of random coincidences.	59
Figure 5.11: Contour plot of the Ar $L_{2,3}M_{2,3}M_{2,3}$ Auger electrons measured in coincidence with Ar^{2+} recoil ions after the subtraction of random coincidences. The most forward direction is toward the left.	60
Figure 5.12: The Ar $L_{2,3}M_{2,3}M_{2,3}$ Auger electron production cross section multiplied by the recoil ion detection efficiency as a function of emission angle with respect to the 800 eV incident electron beam.	61

Abbreviations

ADC	analog-to-digital converter
BPPA	backward 30° parallel-plate analyzer
CMA	cylindrical mirror analyzer
DAC	digital-to-analog converter
DLL	dynamic-link library
FPPA	forward 30° parallel-plate analyzer
GUI	graphical user interface
I/O	input / output
MCP	microchannelplate
MCS	multi-channel scalar
MSA	multi-spectrometer apparatus
PIA	Peripheral Interface Adapter
PSD	position-sensitive detector
RAE	resistive-anode encoder
SCA	single-channel analyzer
STC	System Timing Controller
TAC	time-to-amplitude converter
TOF	time-of-flight

1 Introduction

Ionizing collisions between ions, atoms, and molecules are fundamental processes of nature which play an important role in such diverse fields as fusion energy, atmospheric science, nuclear waste management, and radiation biology, to name a few. In fast ion-atom collisions ($v_{\text{projectile}} \gg 1$ atomic velocity unit) many reactions are possible, such as individual or simultaneous excitation and ionization of the electrons associated with either the target or the projectile. Target-electron transfer into bound or continuum states of the projectile is also possible. Many of these processes are often involved in the same ion-atom collision.

An ideal experiment would measure the initial and final momenta and charge states of projectile and target, as well as the momenta of all ejected electrons and photons produced in the collision. Conservation laws would make some of the information redundant which may serve as a check on the experiment. In an actual experiment only a limited amount of information is available and many constraints may limit the scope of an actual experiment. These may include restricted funding and beam time, limited efficiency and resolution of the experimental apparatus, low target density, small cross sections, and limitations of the data acquisition system.

During the last five years, our group has extensively studied the dynamics of collisional ionization in fast ion-atom collisions. In 1992, a new multi-spectrometer apparatus was designed and constructed to measure the energy and angular distributions of electrons ejected near 0° , between 45° - 135° , and near 180° relative to the incident beam

direction. These measurements can be made in coincidence with both the charge state distribution of the recoil ions and the exit charge state distribution of the projectile ions.

To use this new multi-spectrometer apparatus, our group was forced to upgrade our data acquisition system, which originally consisted of only a *LeCroy 3500* multi-channel analyzer. It was able to function as a system of multi-channel scalars and analog-to-digital converters. It allowed the measurement of electron energy distributions at fixed angles by scanning the various spectrometers, but it did not allow the measurement of angular distributions using the information from a two-dimensional position-sensitive detector. The *LeCroy 3500* could only handle one-dimensional spectra. A personal computer-based data acquisition system was developed as an add on to the *LeCroy 3500* system to allow the measurement of the two-dimensional angular distribution data.

The experiments which used this hybrid data acquisition system include a study of backward ejected electrons¹, cusp electrons² (which are ejected into the forward direction), and electron production at 55° relative to the incident beam direction³. All of these studies were restricted to measuring either the energy distribution of electrons ejected into a fixed angular range or the angular distribution of electrons emitted with a fixed energy. Obviously, the limited capabilities of this data acquisition system severely restrained the idea of an ideal experiment. Since the *LeCroy 3500* was a hardware-based system and could not be reprogrammed, it could not be modified for simultaneous energy and angular distribution measurements.

For this thesis, a new data acquisition program, *CRUNCHER*, has been created. It is designed to let the experimenter make full use of the many capabilities of a multi-spectrometer apparatus. Since *CRUNCHER* is fully PC-based, the *LeCroy 3500* may finally be retired. By accessing two digital I/O interfaces which have been installed in a personal computer, *CRUNCHER* is capable of simultaneously counting events from six different sources and can acquire two-dimensional spectra which may be measured in coincidence with four different events. *CRUNCHER* reads, processes, displays, and stores all of these data. The program is written so that energy and angular distributions of electrons ejected into the continuum may be measured simultaneously. Unlike the previous system, *CRUNCHER* is versatile enough to be adapted for use with many different spectrometer setups. In order to demonstrate and test some of the capabilities of *CRUNCHER*, we have measured the elastic scattering cross section for electrons incident upon argon gas and scattered into the angular range from 45° to 135° with respect to the beam axis for two different electron energies (500 eV, 800 eV). We have also measured the angular distribution of $L_{2,3}M_{2,3}M_{2,3}$ Auger electrons produced in coincidence with Ar^{2+} target recoil ions. The results of these experiments are compared to theory and previous experiments, and they are also used to calibrate the apparatus.

2 Theory

In this thesis we experimentally investigate two processes: the elastic scattering of electrons from neutral atoms and Auger electron production in electron-atom collisions. In order to demonstrate and test some of the capabilities of the newly designed data acquisition system, we have measured the elastic scattering cross section for electrons incident upon argon gas and scattered into the angular range from 45° to 135° with respect to the beam axis for two different electron energies (500 eV, 800 eV). We have also measured the angular distribution of $L_{2,3}M_{2,3}M_{2,3}$ Auger electrons produced in coincidence with Ar^{2+} target recoil ions. The theory of these two processes is briefly discussed in the next two sections.

2.1 Elastic Scattering

An electron approaching a stationary atomic target with a given velocity can elastically scatter off the target. Since the atomic target presents a finite range potential, the partial wave method can be used to calculate the differential scattering cross section. Numerical methods must be applied for such a calculation. Our group has written a *Quick Basic* program which calculates the differential cross section for the elastic scattering of electrons from neutral argon targets (see Appendix A). The partial wave method is used to solve the scattering problem. For different values of the orbital angular momentum, the program numerically integrates the radial part of the time-independent Schroedinger

equation using the Numerov method. This yields the phase shifts which are needed to calculate the cross sections.

The differential scattering cross section for electrons incident on a spherically symmetric finite range potential using the method of partial waves is given by

$$\sigma(\theta) = |f(\theta)|^2 = \frac{1}{k^2} \left| \sum_{l=0}^{\infty} (2l+1) e^{i\delta_l} \sin \delta_l P_l(\cos \theta) \right|^2 . \quad [2.1]$$

This expression can be rewritten in terms of only real numbers as

$$\sigma(\theta) = \left(\frac{f_r}{k} \right)^2 + \left(\frac{f_i}{k} \right)^2 , \quad [2.2]$$

where

$$f_r = \sum_l (2l+1) \cos \delta_l \sin \delta_l P_l(\cos \theta) , \quad [2.3]$$

and

$$f_i = \sum_l (2l+1) \sin^2 \delta_l P_l(\cos \theta) . \quad [2.4]$$

In equations 2.3 and 2.4, the letter l denotes an index that runs from zero to infinity; for large l one expects δ_l to be zero. The square root of the electron energy is proportional to

k . The $P_l(\cos \theta)$ functions are the Legendre polynomials, while the δ_l are known as the phase shifts for the different partial waves and are the only unknowns in equations 2.3 and 2.4. However, one does not have to calculate the infinite number of phase shifts between $l=0$ and $l=\infty$. The terms in the above sums for large l become very small, because the phase shifts for large l are much smaller than one and $\sin \delta_l$ is approximately zero. If l becomes greater than some number l_{max} , then the terms in the sum become so small that they can be neglected. One of the challenges of computing $\sigma(\theta)$ is to decide how many terms to include, i.e. to decide when a term is small enough to be neglected.

The Schroedinger equation is solved numerically to calculate the phase shifts. The radial part of the wave function of the partial wave with index l is $u_l(r)/r$, where r is the distance from the scattering center. The function $u_l(r)$ is a solution of

$$\left\{ \frac{d^2}{dr^2} + \frac{2m}{\hbar^2} \left[E - V(r) - \frac{\hbar^2 l(l+1)}{2mr^2} \right] \right\} u_l(r) = 0 . \quad [2.5]$$

For large r the solution of this differential equation has the form

$$u_l(r) = Akr \left(\cos \delta_l j_l(kr) + \sin \delta_l n_l(kr) \right) . \quad [2.6]$$

Here j_l and n_l are the regular and irregular spherical Bessel functions, respectively, of order l .

The Numerov method is used to find δ_l by numerically integrating equation 2.5.

The potential $V(r)$, which describes the forces between the scattering electron and the

atom, must be known in order to perform this integration. Garvey, Jackman, and Green have numerically calculated independent-particle-model potentials for atoms and ions using multi-configuration Hartree-Fock wave functions⁴. The appropriate $V(r)$ is obtained from their paper.

For the integration, the distance is divided into a large number of small segments of length dr , starting from the scattering center and extending to some distance r_{ext} which is greater than r_{max} . The $u_l(r)$ for each segment can be calculated as long as the $u_l(r)$ in the preceding two segments are known. If n denotes the n th segment, and if

$$g^2(r) = \frac{2m}{\hbar^2} \left[E - V(r) - \frac{\hbar^2 l(l+1)}{2mr^2} \right], \quad [2.7]$$

then

$$u_l(n) = \frac{2 - 10 \frac{dr^2}{12} g^2(n-1)}{1 + \frac{dr^2}{12} g^2(n)} u_l(n-1) - \frac{1 + \frac{dr^2}{12} g^2(n-2)}{1 + \frac{dr^2}{12} g^2(n)} u_l(n-2). \quad [2.8]$$

Setting $u_l(0)=0$ and $u_l(1)$ to some small number which determines A in equation 2.6, $u_l(2)$ can be calculated. The $u_l(3)$ term is calculated from $u_l(1)$ and $u_l(2)$, and so on until $u_l(n)$ is calculated for all n up to the n which corresponds to r_{ext} . Between r_{max} and r_{ext} , $u_l(r)$ is assumed to be of the form given in equation 2.6. Thus δ_l can be found from

$$\tan \delta_l = \frac{G j_l(r_{\max}) - j_l(r_{\text{ext}})}{G n_l(r_{\max}) - n_l(r_{\text{ext}})} \quad \text{with} \quad G = \frac{r_{\max} u_l(r_{\text{ext}})}{r_{\text{ext}} u_l(r_{\max})}. \quad [2.9]$$

One difficult task of the numerical calculation is to choose $r_{\max}$ correctly. If chosen too small, the calculation will be inaccurate. If chosen too large, numerical errors will likely be introduced into the calculation. Another challenge is to select the correct step size for the numerical integration of the Schroedinger equation. The potential $V(r)$ changes rapidly near the center of the atom, so a very small step size is needed to pick up these changes. Furthermore, farther away from the center of the atom the potential changes very slowly and a bigger step size is warranted. Therefore, some sort of compromising size must be selected.

Figure 2.1 shows the doubly differential elastic scattering cross section of electrons from Ar for electron energies of 50 to 1000 eV and for scattering angles from 5° to 180° . For large electron energies, the cross sections decrease smoothly with respect to scattering angle but for smaller energies they show considerable structure. The more the potential of the atom differs from the simple Coulomb potential of a bare nucleus, the more structured is the doubly differential electron scattering cross section.

For electrons with energies of 20 to 800 eV scattering from Ar targets, experimental data are readily available for comparison. In figure 2.2 the calculated cross sections at 200 eV and at 800 eV for Ar targets are compared with experimentally measured cross sections for the same electron energies and targets⁵. The calculations agree very well with the published data. Therefore, the results of DuBois and Rudd are used with confidence as a bench mark for checking the accuracy of the experimental data

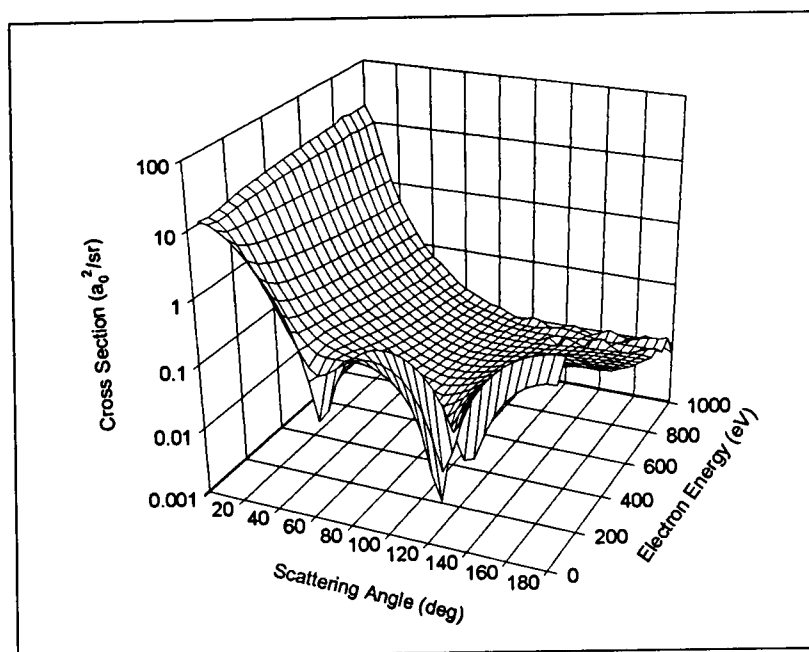


Figure 2.1: Doubly differential elastic scattering cross section of electrons from Ar for electron energies of 50 to 1000 eV and for scattering angles from 5° to 180°.

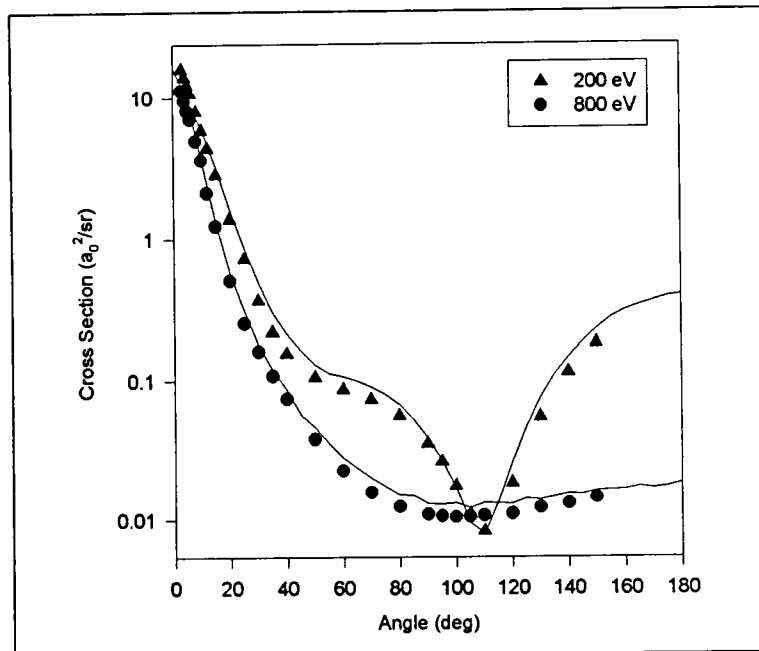


Figure 2.2: Comparison of calculated cross sections with published cross sections for elastically scattered 200 and 800 eV electrons from Ar. Lines represent the calculations.

measured for this thesis and the simulated data produced by a computer program designed to trace the trajectories of electrons through the electron spectrometer utilized in this thesis.

2.2 Auger Effect

In fast ion-atom collisions, an inner-shell electron of the target can be ionized. This leaves a vacancy in the inner shell which decays by the emission of photons or electrons. The term, Auger effect, is used to describe inner-shell vacancy decay by electron emission and transitions in which a single vacancy in an atomic shell leads to two vacancies in one or two different principal shells⁶.

First seen in 1925 by Pierre Auger⁷ and theoretically confirmed in 1927 by Wentzel⁸, the Auger effect is an auto-ionization process in which an inner-shell vacancy is filled by an outer-shell electron. The outer-shell electron loses energy in the transition, so energy is conserved through the ionization of another outer-shell electron. The freed electron has a characteristic kinetic energy equal to the difference in energy between the initial and final configuration of the bound electrons. Target Auger electrons often have an isotropic distribution and their characteristic kinetic energy can be observed as a well-defined peak in the electron energy distribution. Auger-electron spectroscopy has become a leading method of investigating collisional ionization mechanisms⁹.

Fast ions colliding with atomic targets are likely to produce multiple vacancies in the inner and outer shells. Auger spectra of atoms with multiple inner-shell holes can be divided into three main categories. Satellite Auger lines arise from atoms with a single

hole in an inner shell and additional holes in the outer shell. Hyper-satellites arise from multiple holes in an inner shell and no additional holes in outer shells. Satellites of hyper-satellites arise from multiple holes in both the inner and outer shells of the atom¹⁰. All of these processes contribute to a broad peak in the energy spectrum of emitted electrons¹¹.

Figure 2.3 compares two Ne K Auger spectra¹². The inner-shell hole is produced by a fast-ion beam and an electron beam, respectively. A broad peak due to overlapping satellite transitions, which have multiple L-shell vacancies, dominates the spectrum produced by the ion beam. Our group has previously measured a similarly broad Ar L Auger spectrum with an incident 1 MeV/u O^{q+} ($q = 3-8$) beam¹³. In our present experiment, however, an Ar L-hole is produced by 800 eV incident electrons, so we expect a much cleaner spectrum of Ar LMM Auger electrons as measured by Werme *et al*¹⁴ and shown in figure 2.4. The resolution of our electron spectrometer is not good enough to resolve all the lines shown in this figure, because the slit sizes of the spectrometer were chosen to increase transmission at the expense of resolution to measure small binary encounter electron production cross sections in ion-atom collisions. Therefore, our measurements include all of the L_{2,3} M_{2,3} M_{2,3} lines.

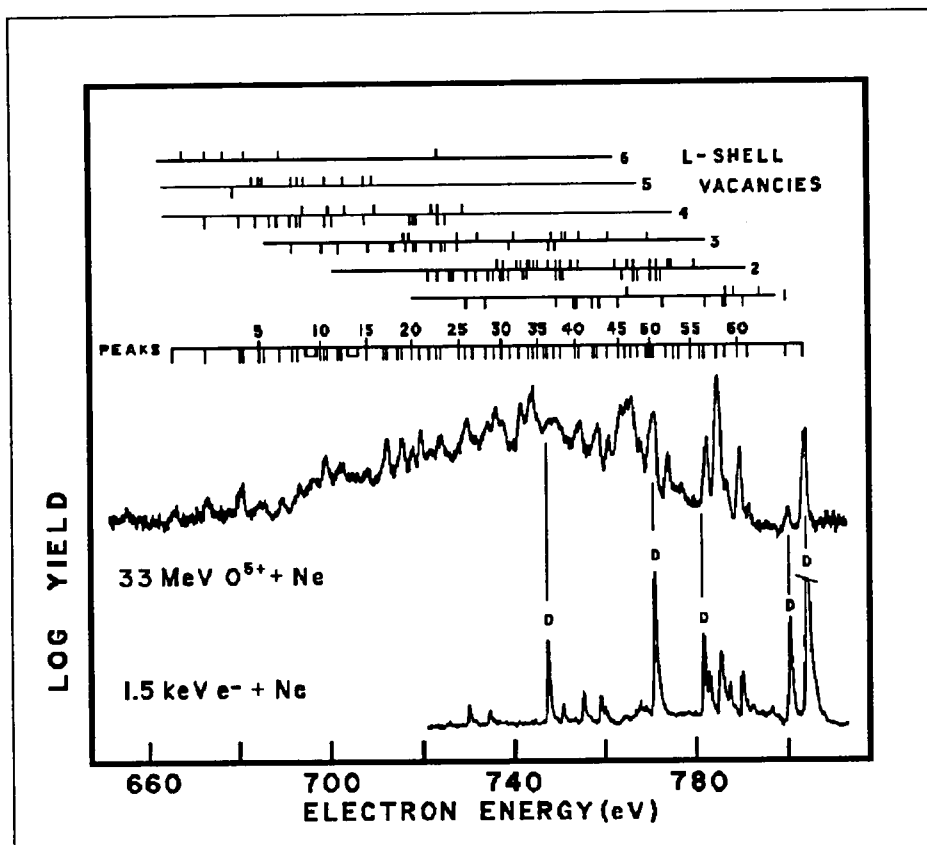


Figure 2.3: High-resolution K Auger spectra of multiply-ionized Ne produced by 33-MeV O^{5+} . The $e^- + Ne$ spectrum is given for comparison. (from D.L. Matthews, B.M. Johnson, J.J. Mackey, and C.F. Moore, *Phys. Rev. Lett.* 31, 1331 (1973))

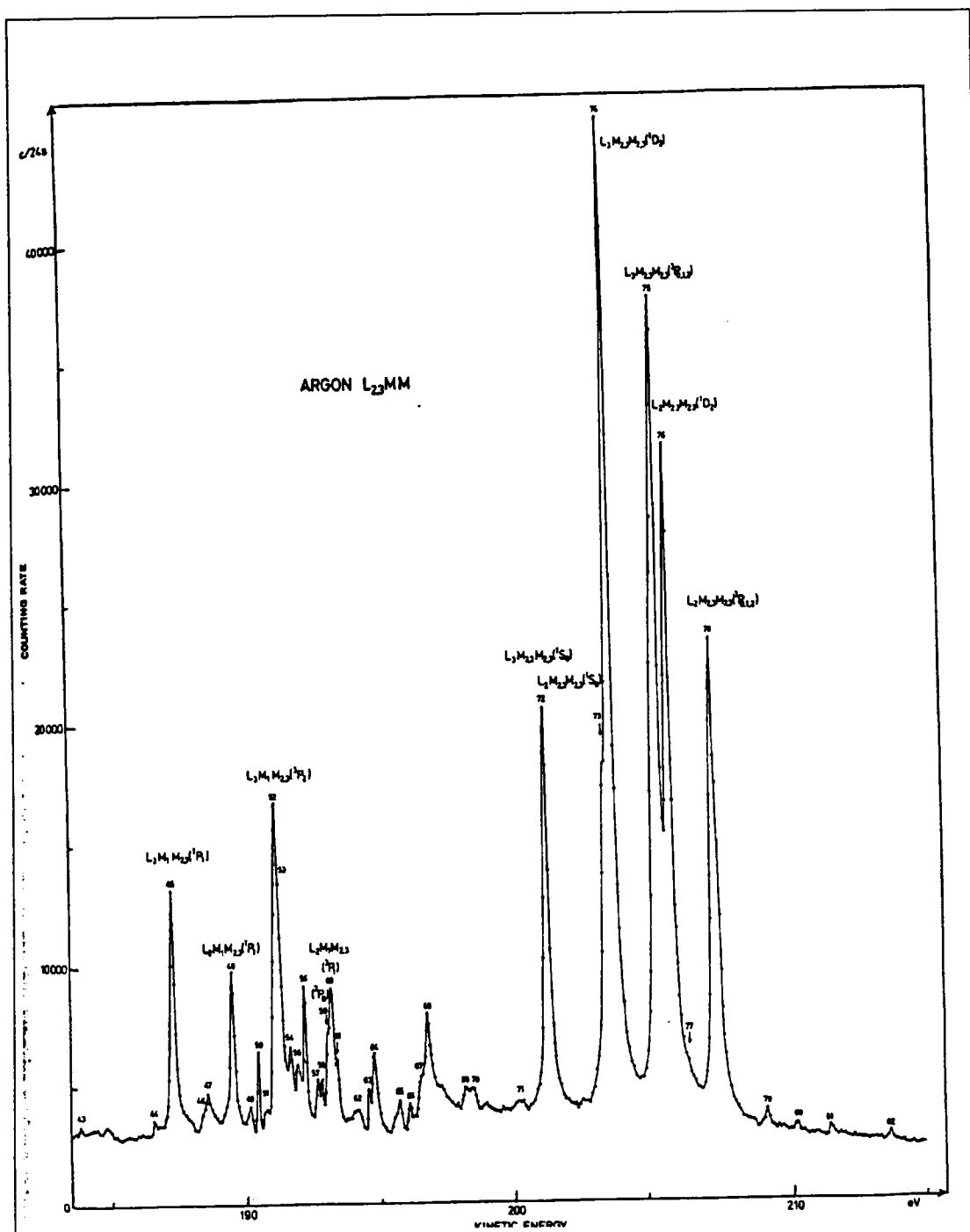


Figure 2.4: The $L_{2,3}MM$ Auger spectrum of argon. The spectrum and the numbers refer to lines resolved or partly resolved under higher resolution. (from L.O. Werme, T. Bergmark, and K. Siegbahn, *Physica Scripta* 8, 149 (1973))

3 Experiment

In order to demonstrate and test some of the capabilities of *CRUNCHER*, we have measured the elastic scattering cross section for electrons incident upon argon gas and scattered into the angular range from 45° to 135° with respect to the beam axis for two different electron energies (500 eV, 800 eV). An incident beam of electrons was chosen over an ion beam mainly because the EN Tandem Van de Graff accelerator located at the Oak Ridge National Laboratory was suddenly made unavailable due to restrained funding. We have also measured the angular distribution of $L_{2,3}M_{2,3}M_{2,3}$ Auger electrons produced in coincidence with Ar^{2+} target recoil ions. A particular problem of such measurements is to extract a recoil ion from the interaction region without losing information about the momenta of free electrons produced together with a detected recoil ion. The multi-spectrometer apparatus is designed to accomplish this goal. Electron analyzers are used to measure the energy and angular distributions of ejected electrons, while a time-of-flight analyzer is used to measure the charge state of the recoil ions.

Coincidence experiments often place conflicting demands on the experimental apparatus. Target recoil ions are born with very low energy and move slowly. They need to be accelerated by an electric field in the target region towards a recoil-ion detector. Stronger electric fields generally result in greater recoil-ion detection efficiencies. Electrons are very light particles and are easily affected by any electric field in the target region. Strong electric fields here will prevent the electrons from reaching their detector or will distort their trajectories through the electron analyzer itself. Therefore, only weak

electric fields (< 10 V/cm) may be used in the target region to accelerate recoil ions causing the detection efficiency to be quite low. Consequently very few experimenters have attempted to measure the energy distributions of ejected electrons in coincidence with the charge state of the target recoil ions. The following sections describe the various components that make up the multi-spectrometer apparatus used in this thesis.

3.1 Multi-Spectrometer Apparatus

The multi-spectrometer apparatus (MSA) was designed to measure energy and angular distributions of electrons produced in fast ion-atom collisions in coincidence with recoil and projectile ion charge states. The MSA consists of a gas cell surrounded by four different spectrometers. Three of the spectrometers are electron energy analyzers: a cylindrical mirror analyzer (CMA), a backward 30° parallel-plate analyzer (BPPA) with an electron gun attached to it, and a forward 30° parallel-plate analyzer (FPPA). The remaining spectrometer is a time-of-flight (TOF) recoil ion charge-state analyzer. With this setup, free electrons born in the gas cell region can be detected at angles near 0°, between 45°-135°, and near 180° relative to the incident beam direction. The MSA is surrounded by μ -metal for magnetic shielding and resides in a vacuum chamber which is pumped by a series of roughing and oil diffusion pumps. See figure 3.1 for a diagram of the MSA. In this work, the MSA is used to measure the elastic scattering cross section for electrons incident upon argon gas and scattered into the angular range from 45° to 135° with respect to the beam axis for two different electron energies (500 eV, 800 eV) and to measure the angular distribution of $L_{2,3}M_{2,3}M_{2,3}$ Auger electrons produced in

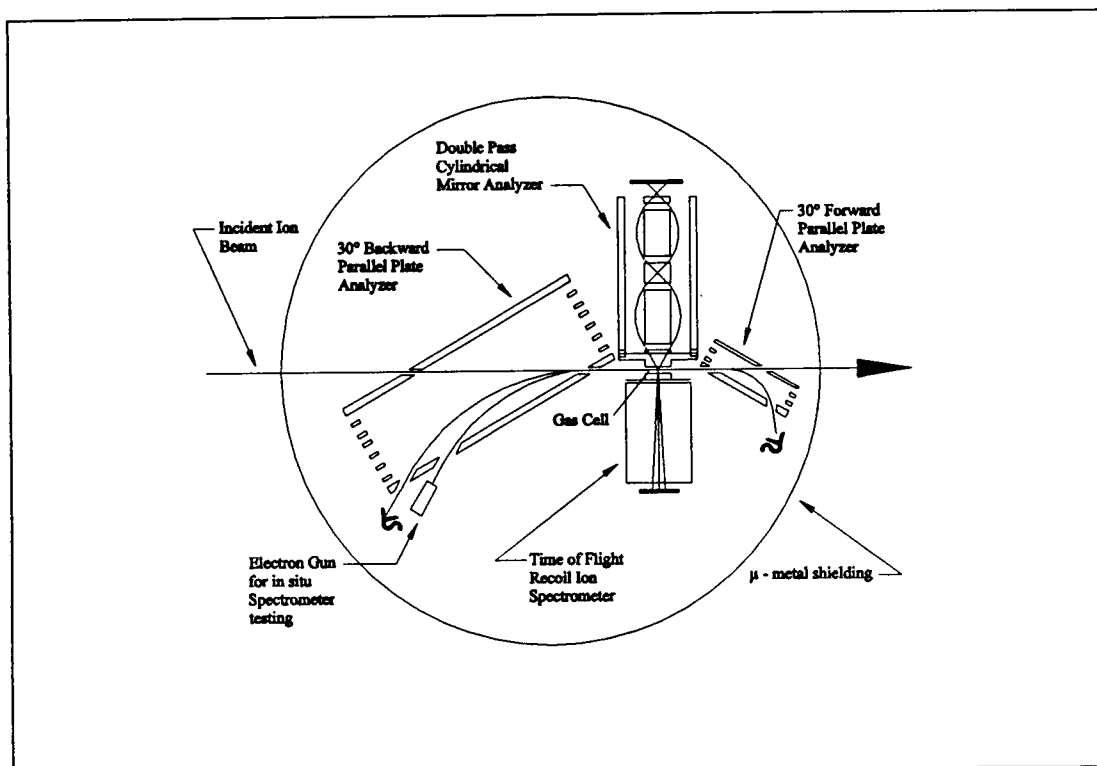


Figure 3.1: Schematic diagram of the multi-spectrometer apparatus.

coincidence with Ar^{2+} target recoil ions. Therefore, the forward parallel-plate analyzer shown in the diagram is replaced by a Faraday cup.

3.1.1 Gas Cell

The gas cell is a cylindrically shaped region 14 mm in diameter and 13 mm long. It has four apertures, two of which (2 mm in diameter) allow the incident beam to enter and exit. The other two apertures connect the gas cell to both the CMA and TOF. The TOF aperture is 2 mm in diameter, while the CMA aperture consists of a ring opening with an inner diameter of 6 mm and an outer diameter of 16 mm. Because of these apertures, the

gas density decreases immediately outside of the gas cell which causes the effective gas cell length to become larger than the actual gas cell length¹⁵. Accordingly, the effective gas cell length (l) is approximately 15.9 mm.

The gas flow into the cell is regulated with a feedback controlled Granville-Phillips Pressure/Flow controller and a Granville-Phillips Servo-Driven Valve Assembly. Gas is injected into a gas reservoir that is connected to the gas cell by a 14 mm wide channel situated above the gas cell. The pressure in the gas reservoir is monitored by a Baratron capacitive manometer. It should be noted that the gas pressure in the reservoir is higher but proportional to the gas pressure in the target area.

The sides of the gas cell adjacent to the CMA and to the TOF are electrically insulated from each other. Therefore, a bias voltage can be placed across the gas cell for the extraction of recoil ions. The bias voltage used is approximately 7.8 V across a distance of 1.3 cm which gives an electric field strength of ~ 6 V/cm. This field is small enough not to have any noticeable effect on the incident beam or the ejected electrons detected in this experiment.

3.1.2 Electron Gun

The experiments described in this thesis use an electron beam from an electron gun. Initially, the electron gun was attached to the BPPA. The BPPA was used to energy analyze and steer the electron beam. This electron gun (10 cm in length) is a device made by Frank Segner¹⁶ and consists of a tungsten tip electron emitter, a Pierce diode, and an attached lens system¹⁷. The design is similar to that of electron guns used in the cathode

ray tubes found in many oscilloscopes. The electron gun has been tested over an electron energy range from 300 eV to 3 keV. The maximum output varies from ~ 1 nA for 300 eV electrons to ~ 100 μ A for electrons with energies above 2 keV.

Because of the geometry and the distances involved, the resultant electron beam was not collimated sufficiently to pass through the gas cell without striking the apertures and to allow a consistent normalization. Thus, the BPPA was removed from the MSA and the electron gun was mounted so that it shot directly into the gas cell. The gun was shielded to prevent the emitted electrons from being affected by any of the voltages placed on the analyzers. Furthermore, two more apertures were added to the electron gun for increased collimation of the beam.

3.1.3 Parallel-Plate Analyzer

Past experiments^{18, 19} have detected electrons ejected at $180^\circ \pm 2^\circ$ with respect to the incident beam direction using a 30° parallel-plate analyzer (the BPPA). Electrons ejected at $0^\circ \pm 5^\circ$ with respect to the incident beam direction have also be detected using a 30° parallel-plate analyzer (the FPPA). Both analyzers are governed by the same equations. In preliminary experiments for this thesis, the BPPA was used to refocus electrons (of a chosen energy) emitted by the electron gun into the gas cell where they could elastically and inelastically scatter off a gas target and be detected using the CMA.

30° parallel-plate analyzers were first described by Yarnold and Bolton²⁰ and later by Green and Proca²¹. A schematic diagram of a parallel-plate analyzer is shown in figure

3.2. It consists of two flat, parallel aluminum plates placed a distance d apart. The potential difference between the two plates is V_d . Fringe plates are equally spaced between the two parallel plates and connected via a resistor chain to keep the electric field $E_d = \frac{V_d}{d}$ in the region between the plates as uniform as possible. The base plate of the analyzer is held at the same potential as the gas cell, while the deflecting plate is held at the voltage necessary for directing electrons of a chosen kinetic energy into the gas cell. In figure 3.2, $\beta = 30^\circ$, the base plate is the plane at $y = 0$, the deflecting plate is at $y = -d$, and the entrance focus is at $x = 0, y = +h$. The equation for the path of electrons of energy T_d deflected by an electric field E_d is

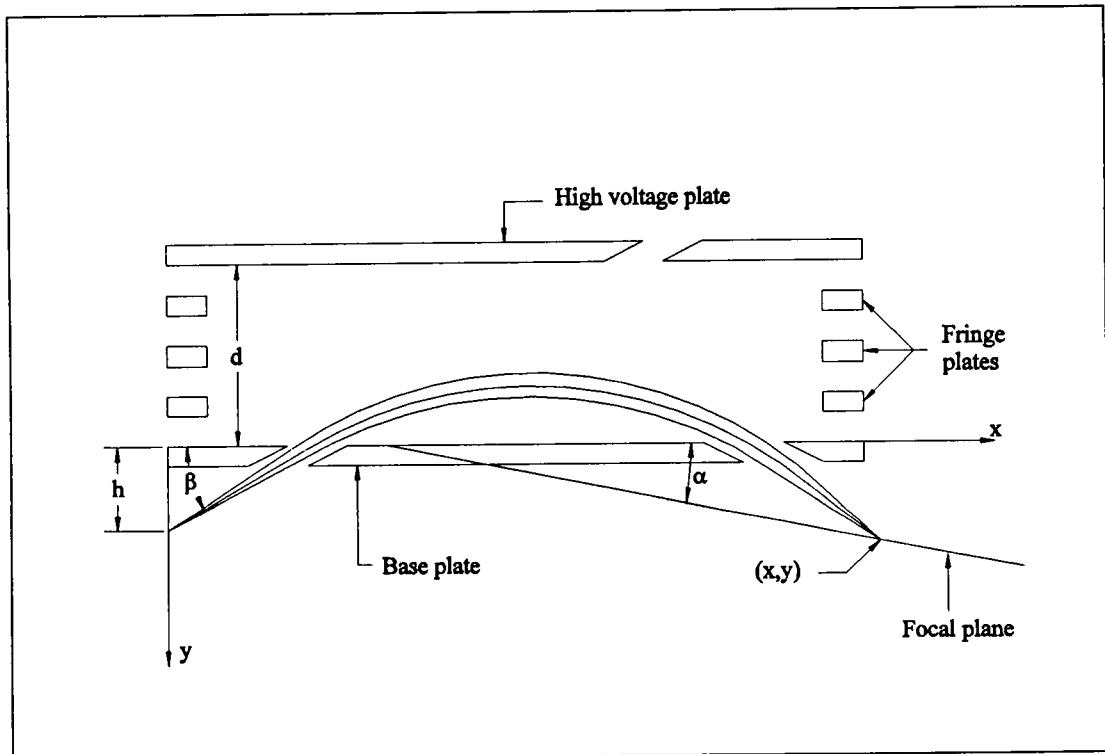


Figure 3.2: Schematic diagram of a parallel-plate analyzer.

$$x = (h + y) \cot \beta + \sin(2\beta) \left(\frac{2T_d}{E_d} \right) . \quad [3.1]$$

The condition for first-order focusing is found by differentiating this equation with respect to β for constant y and setting $\frac{dx}{d\beta} = 0$. The foci for various initial energies T_d lie on a straight line

$$x = (h + y) \cot \beta \left[1 + \left(\frac{1}{\cos(2\beta)} \right) \right] , \quad [3.2]$$

which defines the focal plane of the analyzer. This line makes an angle α with the x -axis where

$$\tan \alpha = \frac{\tan \beta \cos(2\beta)}{1 + \cos(2\beta)} . \quad [3.3]$$

For a 30° analyzer which focuses to second order, the focal plane makes an angle $\alpha = 10.9^\circ$ with the base plate of the analyzer. The electron gun was originally situated in this focal plane. As mentioned earlier, the BPPA was removed from the MSA and the electron gun was then mounted so that it shot directly into the gas cell.

3.1.4 Cylindrical Mirror Analyzer

Cylindrical mirror analyzers are widely used in the field of electron spectroscopy and their use and focusing properties are described in detail by several authors^{22, 23, 24, 25}.

The CMA uses cylindrical symmetry to achieve electron focusing over the whole azimuthal range for a given ejection angle with respect to the CMA axis. The CMA in this experiment consists of two coaxial cylinders with a potential difference of V_d between them. A schematic diagram of the CMA is shown in figure 3.3. The inner cylinder is at the same potential as the adjoining gas cell. Only electrons with initial velocities making an angle of $\sim 45^\circ$ with the cylinder axis can enter the CMA. If the appropriate V_d is used

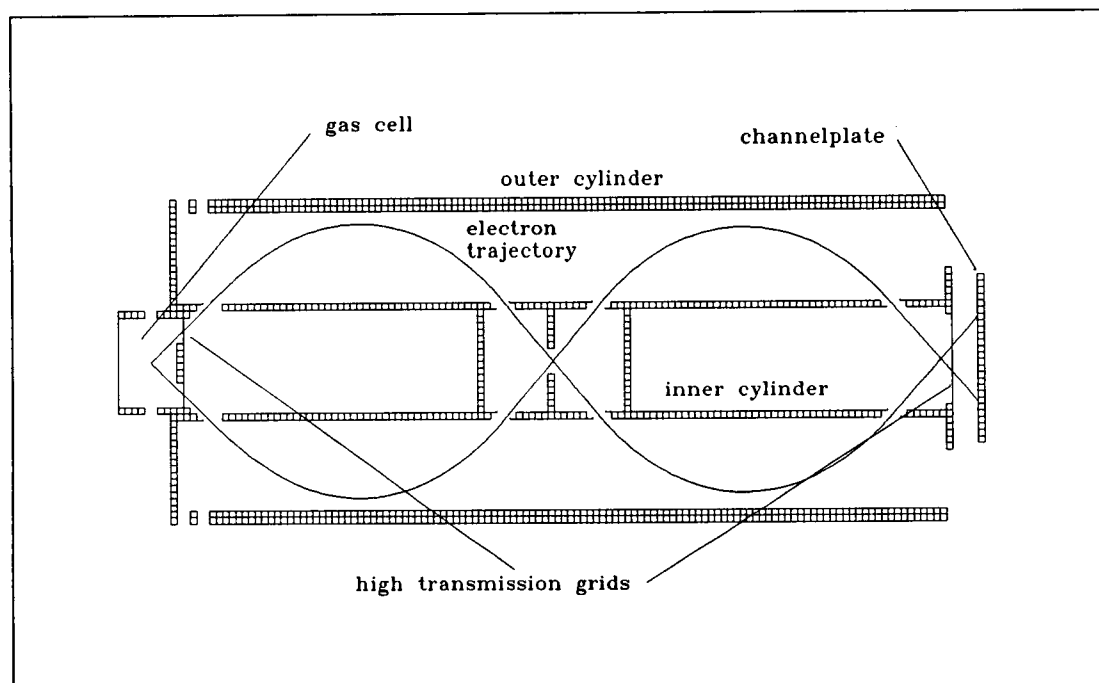


Figure 3.3: Schematic diagram of the cylindrical mirror analyzer.

in relation to the electron's kinetic energy T , then the electron's trajectory makes it through both stages of the CMA. All electrons with kinetic energy T , which are born on the CMA axis and whose initial velocities make the same angle with this axis, are focused again on the CMA axis after exiting the analyzer. The electrons continue with the same velocity they had when entering the CMA, so their angular distribution is preserved and can be measured. A double-pass CMA reduces the chance of a stray electron passing through all the slits and therefore reduces possible background contributions.

The cylinder axis is shown as the line connecting the points S (source) and I (image); see figure 3.4. The position and width of the slits S_1 and S_2 determine θ_0 and $\Delta\theta$. Assume a charged particle of mass m , charge $-e$, and kinetic energy T enters the region between the inner and outer cylinders on a trajectory making an angle θ with the cylinder axis.

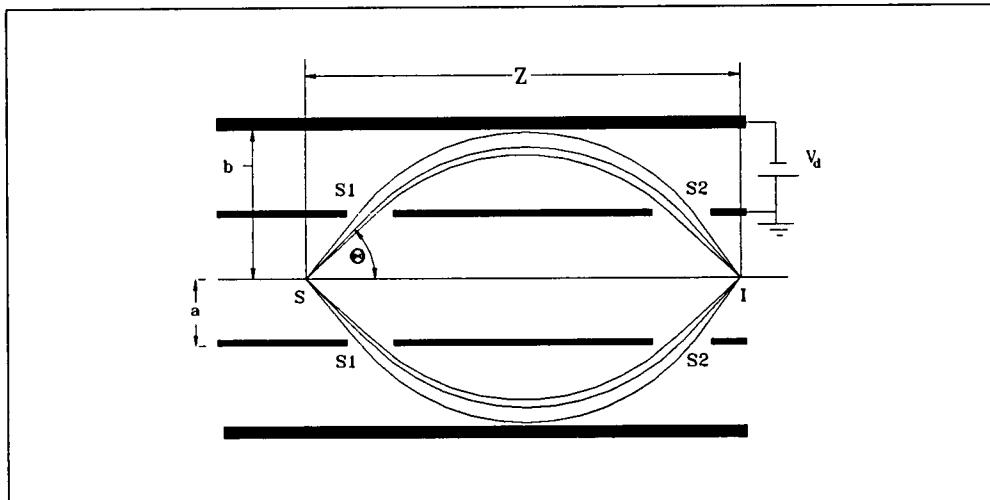


Figure 3.4: Schematic diagram of a single stage of the CMA.

The potential energy U of this particle varies with its radial distance r from the cylinder axis as

$$U = U_0 \ln\left(\frac{r}{a}\right) ; \quad a < r < b \quad [3.4]$$

where

$$U_0 = \frac{eV_d}{\ln\left(\frac{b}{a}\right)} . \quad [3.5]$$

Here a is the radius of the inner cylinder, b is the radius of the outer cylinder, and V_d is the potential difference between them. The equation of motion in cylindrical coordinates (r, ϕ, z) is

$$m\ddot{r} + \frac{U_0}{r} - \frac{l^2}{mr^3} = 0 , \quad [3.6]$$

where $l = mr^2\dot{\phi}$ and $p_z = m\dot{z} = mv_0 \cos\theta_0$ are constants of motion. If a particle is ejected in S , the angular momentum l about the cylinder axis is zero and equation 3.6 simplifies to

$$m\ddot{r} + \frac{U_0}{r} = 0 . \quad [3.7]$$

Energy conservation is expressed through the equation

$$T \sin^2 \theta_0 - U_0 \ln\left(\frac{r}{a}\right) = \frac{1}{2} m \dot{r}^2 . \quad [3.8]$$

Equation 3.8 yields the maximum value of r , $r_{\max}$, where $\dot{r}$ is zero;

$$r_{\max} = a e^{\frac{K^2}{2}} , \quad \text{with} \quad K^2 = 2 \frac{T}{U_0} \sin^2 \theta_0 . \quad [3.9]$$

For an electron moving from S to I , the trajectory from $z(r_{\max})$ to I is the mirror image of the trajectory from S to $z(r_{\max})$. To find the distance Z from S to I , equation 3.8 is solved for $\dot{r}$.

$$\dot{r} = \sqrt{\frac{U_0}{m} (K^2 - 2 \ln(\frac{r}{a}))} . \quad [3.10]$$

The time the electron spends between the cylinders is given by

$$t = 2 \sqrt{\frac{m}{U_0}} \int_a^{r_{\max}} \frac{dr}{\sqrt{K^2 - 2 \ln(\frac{r}{a})}} . \quad [3.11]$$

Z therefore is given by

$$Z = 2a \cot \theta_0 + \dot{z} t . \quad [3.12]$$

Substituting $u^2 = K^2 - 2 \ln(\frac{r}{a})$ into equation 3.11 for t yields

$$Z = 2a \cot \theta_0 \left[1 + K e^{\frac{K^2}{2}} \int_0^K e^{-\frac{u^2}{2}} du \right]. \quad [3.13]$$

The condition for first-order focusing is $\frac{dZ}{d\theta_0} = 0$,

and the condition for second-order focusing is $\frac{d^2Z}{d\theta_0^2} = 0$.

The trajectories of electrons not emitted from a point on the cylinder axis are found by numerically integrating their equations of motion. A computer program written in *Quick C* by our group computes these trajectories. The program determines if the electrons, produced at arbitrary positions with arbitrary velocities, are able to traverse the double-pass CMA and reach the position-sensitive detector (PSD). This information is used to analyze the measured PSD images. The source code of this program is found in Appendix B.

A first-order focus occurs for every ϕ_0 between 0° and 90° ; a second-order focus exists at $\phi_0 = 42.31^\circ$. The slits S_1 and S_2 are chosen such that only particles emitted with $\phi_0 = 42^\circ \pm 4^\circ$ are accepted by the CMA. The entrance and exit slits are covered with several very thin carbon fibers which are attached parallel to the cylinder axis. These grids reduce the inhomogeneities of the electric field between the two cylinders. All of the metal parts of the CMA are aluminum to ensure no residual, internal stray magnetic fields alter the trajectories of the electrons through the CMA.

Electrons which successfully traverse the two analyzing stages of the CMA are detected using a 25 mm diameter active-area chevroned microchannelplate (MCP) detector²⁶ equipped with a resistive-anode encoder (RAE) for position determination. The MCP consists of an array of micron-sized glass channels coated with electron-emissive material that emits several electrons when struck by an energetic particle. The applied 900 V across the channels results in an electric field which accelerates these secondary electrons down the channels where they collide with the walls and emit even more electrons. Two MCP's are chevroned so a particle cannot pass straight through the MCP assembly. Using this arrangement, a single input event will generate $\sim 10^7$ or more electrons at the output.

The electron shower produced by the MCP's strikes a roughly square-shaped RAE situated directly behind the MCP's. The RAE is biased at $\sim +300$ V above the back side of the second MCP. See figure 3.5 for a diagram of the PSD setup. The electric charge absorbed by the RAE due to the electron shower is transmitted to the corners of the RAE in proportions that are measured to determine the coordinates of the point where the electron shower struck the RAE.

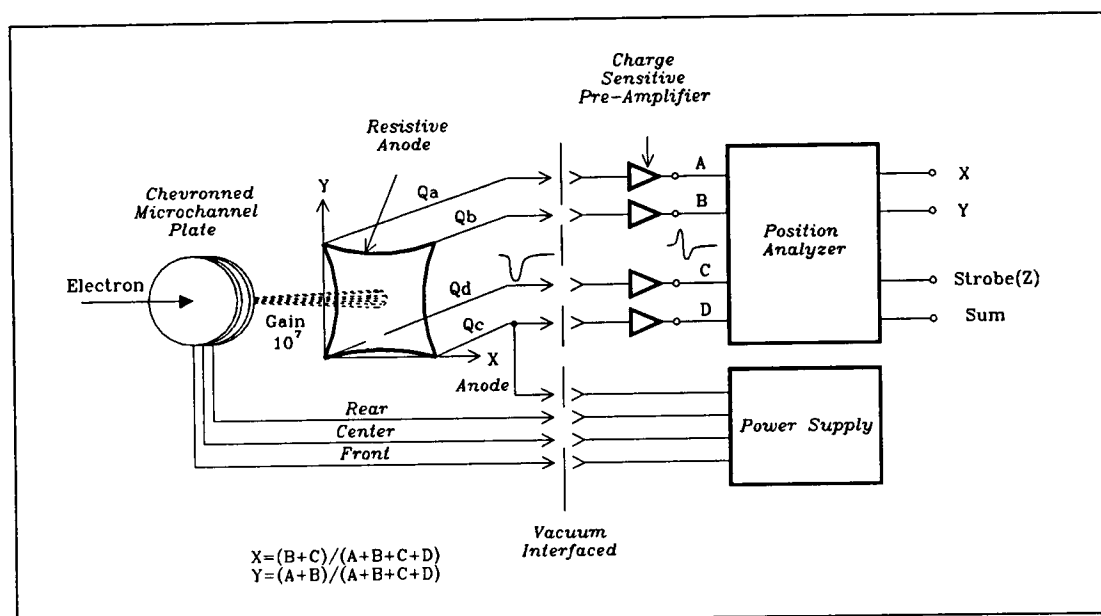


Figure 3.5: Schematic drawing of the position-sensitive detector of the CMA.

3.1.5 Time-of-Flight Recoil Ion Analyzer

A schematic diagram of the TOF recoil ion analyzer, which was first described by Wiley and McLaren²⁷ in 1955, is shown in figure 3.6. The analyzer consists of three regions: a source region (in our case this is the gas cell), an acceleration region, and a field-free drift region. By having different electric fields (E_s , E_d , $E = 0$) in these regions, ions of the same mass can be separated according to their charge q . As stated earlier, a small extraction voltage is applied across the gas cell which results in an electric field of ~ 6 V/cm. This field deflects slow recoil ions produced in the source region toward the acceleration region which is 5 mm long. Here the ions are accelerated by an electric field of ~ 11.55 V/cm. The use of two ion acceleration stages keeps the electric field needed in the gas cell low. Spatial focusing, where ions born at different positions in the source region arrive at the detector after the same flight time²⁸, is achieved by choosing the

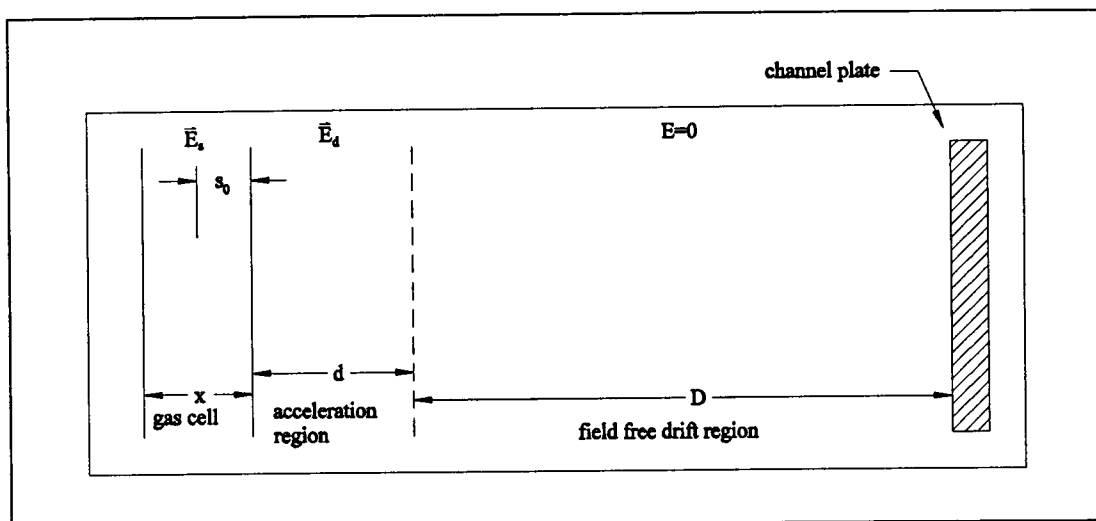


Figure 3.6: Schematic diagram of the time-of-flight spectrometer.

appropriate ratio between the voltages across the gas cell and the acceleration region. After passing through the acceleration region, the ions enter the field-free drift region. This region is separated from the acceleration region by carbon fiber grids with better than 99% transmission per grid. The ions drift until they strike a similar MCP detector setup as described before but equipped with a single copper plate anode. The front of the first MCP is held at -1500 V to accelerate the ions toward the MCP, while the back of the second MCP is held at +400 V. The copper anode, held at +700 V, collects the resulting electron shower from the MCP's when a recoil ion arrives.

Ions with a charge q are created with an initial energy U_0 in the source region a distance s from the entrance to the acceleration region. Due to the beam width, ions are produced in the gas cell with different values of s . The ions gain most of their energy when they enter the acceleration region as $\Delta U = qE_d d$. This energy gain is proportional to the charge q on the particle and differently charged ions gain different amounts of

energy. Upon entering the field-free region, the ions drift toward the detector with their final energy U given by

$$U = U_0 + qE_s s + qE_d d , \quad [3.14]$$

where E_s and E_d are the electric fields in the source and acceleration region, respectively.

The time of flight for the recoil ion from the source region to the detector is

$$t(U_0, q, s, d) = t_s + t_d + t_D , \quad [3.15]$$

where

$$t_s = \frac{\sqrt{2m}}{qE_s} \left(\sqrt{U_0 + q s E_s} \pm \sqrt{U_0} \right) \quad [3.16]$$

$$t_d = \frac{\sqrt{2m}}{qE_d} \left(\sqrt{U} - \sqrt{U_0 + q s E_s} \right) \quad [3.17]$$

$$t_D = \sqrt{2m} \frac{D}{2\sqrt{U}} . \quad [3.18]$$

The (+) and (-) signs in equation 3.16 correspond to ions whose initial velocity vector points either away or towards the detector, respectively. Ions created with the same mass and charge in different locations in the gas cell gain different amounts of energy. As the

ions pass through the TOF, the ions with greater energy catch up to the ions with less energy. The detector is located at the position where the ions pass each other. This location is obtained by setting $\left(\frac{dt}{ds}\right)_{0,s_0} = 0$, where $U_0 = 0$ and s_0 is the center of the gas cell. This yields the following relationship between the spectrometer dimensions and the applied electric fields in each region:

$$D = 2s_0\sqrt{k_0^3} \left[1 - \left(\frac{d}{s_0(k_0 + \sqrt{k_0})} \right) \right], \quad [3.19]$$

where

$$k_0 = 1 + \frac{E_d d}{E_s s_0}. \quad [3.20]$$

Equation 3.20 is used to determine E_d for the given dimensions of the TOF analyzer and the chosen value of E_s . The electric field strength and the dimensions of the analyzer used here are: $E_s = 6$ V/cm, $s_0 = 0.9$ cm, $d = 0.5$ cm, and $D = 7.62$ cm.

3.2 Electronic Signal Detection

In the experiments described in this thesis, electrons emitted between 45° and 135° with respect to the incident beam direction are detected with a PSD located at the exit of the CMA. A diagram of the PSD is shown in figure 3.5. The signals from the four

corners of the RAE are fed into a preamplifier box (Quantar Technology Inc. Model 24011). The preamplifier shapes and converts the low level charge pulses from the RAE into higher level bipolar pulses, which are then suitable as input for the position computer (Quantar Technology Inc. Model 2401). This position computer creates output voltages proportional to the position coordinates (X , Y) of the electron shower on the RAE by evaluating

$$X = \frac{Q_B + Q_C}{Q_A + Q_B + Q_C + Q_D} \quad [3.21]$$

and

$$Y = \frac{Q_A + Q_B}{Q_A + Q_B + Q_C + Q_D} \quad , \quad [3.22]$$

where Q_i is the amount of charge collected at corner i (i.e. A , B , C , or D) on the RAE. The position computer provides this information as analog signals, which can be visualized via an oscilloscope or processed as an eight bit digital output for both X and Y . Due to limitations in the spatial resolution of the CMA (caused by the extended source region), the resolution in our experiments is restricted to 64x64 pixels per image. The digital output is read, displayed, and stored by running a newly developed data acquisition program. This program, *CRUNCHER*, is run on an IBM-compatible personal computer and is discussed in detail in the next chapter.

The total beam current is collected in a Faraday cup, and from this current the total number of incident electrons is determined. The Faraday cup is electrically connected to a current digitizer which measures the charge incident upon the surface of the cup. The digitizer monitors the charge and sends out a pulse to *CRUNCHER* every time 10^{-13} Coulombs of charge are collected.

The detectors for the coincidence part of the experiment are the PSD of the CMA and the MCP's of the TOF spectrometer. Figure 3.7 shows a schematic diagram of the signal detection electronics. The TOF is used to determine the recoil ion charge state while the CMA detects the Auger electrons. The charge state of a recoil ion is determined by measuring the time difference between an electron detected with the CMA and an ion detected with the TOF. The signal from the CMA is the start pulse while the signal from the TOF is the stop pulse for the time-to- amplitude converter (TAC).

The TAC converts the time interval between the start and stop pulses into a voltage pulse whose height is proportional to the time between pulses. The voltage pulse from the TAC is sent to an analog-to-digital converter (ADC) module which records the number of pulses received versus the height of the pulses. The ADC converts pulse height to a channel number and stores the number of pulses (counts) versus channel number. A typical ADC spectrum is shown in figure 3.8²⁹.

Since each recoil ion charge state takes a different amount of time to pass through the TOF spectrometer, different charge states appear as different peaks in the ADC spectrum. The TAC signals are internally sent to the single-channel analyzer (SCA) of the TAC/SCA module. The SCA produces an output pulse if the pulse height of the input

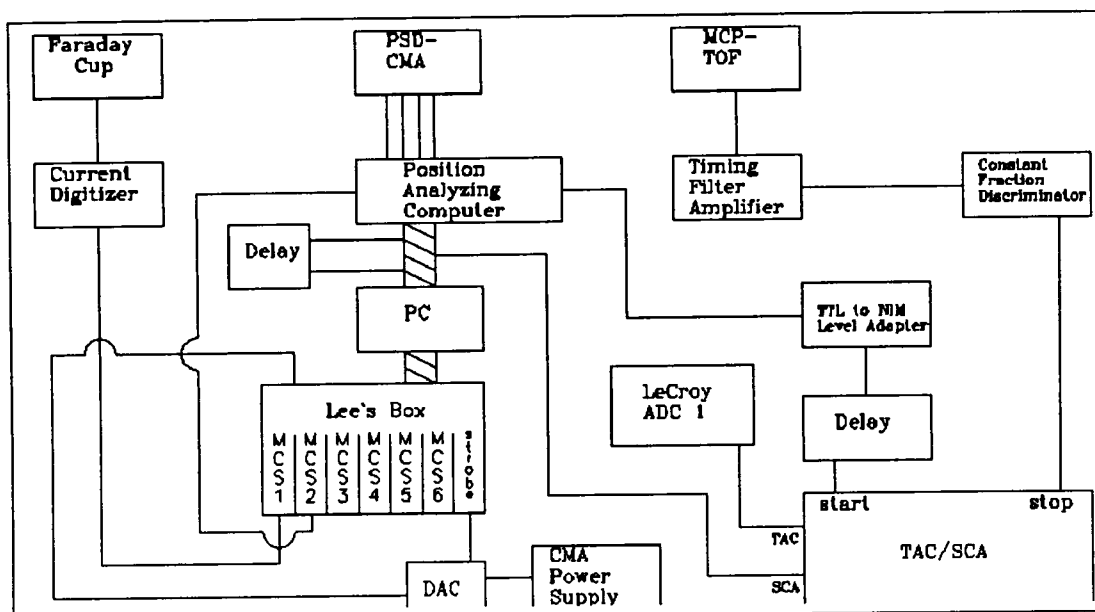


Figure 3.7: Schematic diagram of the signal detection electronics.

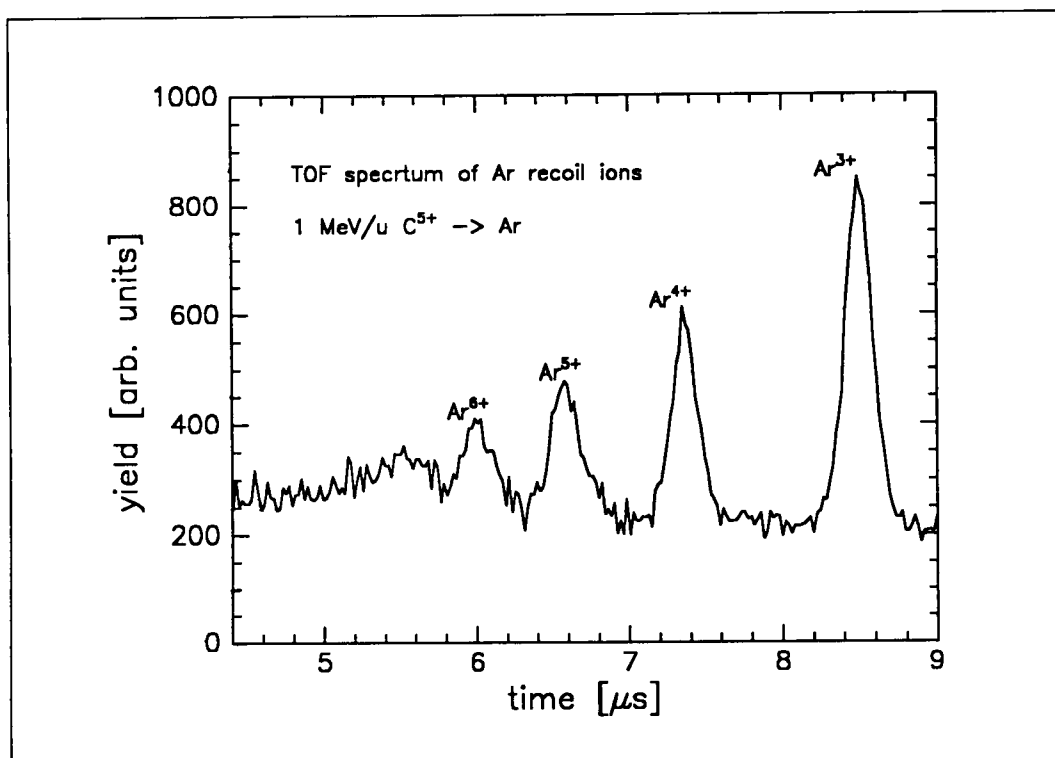


Figure 3.8: A typical ADC spectrum shows the different Ar recoil-ion charge states appearing as different peaks resting on a background of random coincidences. (from F. Segner, Ph.D. Dissertation, Johann Wolfgang Goethe University, 1996 (unpublished))

signal falls into a chosen window. To isolate different recoil ion charge states, windows of different SCA's can be positioned around different peaks in the ADC spectrum. When a pulse falls into a window, a signal is sent to *CRUNCHER* which then stores this coincidence information together with the position of the electron on the PSD as a function of electron energy. In this way, the number of electrons detected in coincidence with recoil ions of a given charge state is recorded.

4 Data Acquisition

The new data acquisition program, *CRUNCHER*, is the principal component of this thesis. The primary reason for the development of this program is to replace a prior hybrid system of a *LeCroy 3500* multi-channel analyzer and a PC-based data acquisition system. The *LeCroy 3500* was able to function as a system of multi-channel scalars and analog-to-digital converters. It allowed the measurement of electron energy distributions at fixed angles by scanning the various spectrometers, but it did not allow the measurement of angular distributions using the information from a two-dimensional position-sensitive detector. The *LeCroy 3500* could only handle one-dimensional spectra. A personal computer-based data acquisition system was developed as an add on to the *LeCroy 3500* system to allow the measurement of the two-dimensional angular distribution data.

Since the *LeCroy 3500* was a hardware-based system and could not be reprogrammed, it could not be modified for simultaneous energy and angular distribution measurements. Also, the transfer of data from the *LeCroy 3500* to a PC-based machine was not trivial and could be quite time-consuming. Therefore, the development of a data acquisition program which runs entirely on a personal computer fitted with the proper hardware is a logical improvement.

CRUNCHER is designed to use the *PC-TIO-10* and the *AT-DIO-32F* boards made by the National Instruments Corporation. The *PC-TIO-10* is used to count events. These events can be digitized pulses from a current digitizer connected to the Faraday cup, the

number of electrons reaching the PSD of the CMA, the number of recoil ions reaching the MCP's of the TOF, etc. The *AT-DIO-32F* is used to transfer the coordinates of the points where the electrons strike the PSD from the position computer to the personal computer.

The *PC-TIO-10* is a timing and digital I/O interface for the personal computer.

Two Advanced Micro Devices Am9513A System Timing Controllers (STC's) are used for the timing interface, and a Motorola MC6821 Peripheral Interface Adapter (PIA) is used for the digital I/O interface where each of the two 8-bit I/O ports is bit-configurable. The manual for the *PC-TIO-10* describes the Am9513A STC's as the heart of the *PC-TIO-10*. These chips have five individually-controlled 16-bit counters, each of which can be configured to operate in a number of different modes. Every counter has its own source, gate, and output connections. The STC's are clocked by an onboard 1-MHz crystal oscillator to give 1- μ sec timing resolution³⁰.

The *AT-DIO-32F* is a high-speed 32-bit parallel digital I/O interface for the personal computer. The 32 lines of digital I/O are organized into four 8-bit ports. These ports are further divided into two groups: *Group 1* contains *Ports A* and *B*, and *Group 2* contains *Ports C* and *D*. Each group can be independently programmed for input or output and a handshaking mode. Handshaking is simply a reference to the synchronization of the sending and receiving of data for each port. The *AT-DIO-32F* can operate in two modes: *Mode 0* and *Mode 1*. *Mode 0* is basic input and output where each group can be configured as a read or write group. *CRUNCHER* implements *Mode 1* which is strobed I/O that utilizes handshaking³¹.

The source code for *CRUNCHER* is written using Microsoft's *Visual Basic* programming language. This relatively new language is a powerful programming tool because it allows the user to fully exploit the graphical user interface (GUI) by creating applications specifically for use in the *Windows* environment. The GUI for an application is provided by drawing objects in a graphical manner. The properties of these objects can be set to refine their appearance and behavior. Then the interface is made to react to the user by writing code which responds to events that occur in the interface. Actually, the first prototype for *CRUNCHER* was written in *Quick Basic*. While the *PC-TIO-10* uses only 8-bit transfers, the *AT-DIO-32F* uses both 8-bit and 16-bit transfers. Since *Quick Basic* only supplies routines for 8-bit transfers, the prototype could not efficiently communicate with both boards. Also, the interface for the *Quick Basic* prototype was not graphical and therefore not very user-friendly. Thus for *CRUNCHER*, *Visual Basic* is the chosen programming language.

The source code for *CRUNCHER* is found in Appendix C. It should be noted that Appendix C does not include the settings for the GUI's. While these settings are important to the appearance and behavior of the GUI's, it is the source code which is most vital to the execution of *CRUNCHER*. The complete *CRUNCHER* program consists of three forms and two modules.

The first module (*VB-ASM.BAS*) includes the file *VBASM.TXT*. This file contains the declarations for all of the *VB-ASM* subroutines and functions. *VB-ASM* is a dynamic-link library (DLL) created by SoftCircuits Programming. It is written entirely in assembly language making it highly optimized. The DLL comprises many low-level

routines, but *CRUNCHER* requires the use of only four: *vbinp*, *vbout*, *vbinpw*, and *vboutw*. The first two read and write, respectively, a byte value from a specified I/O port, whereas the last two read and write, respectively, a word value from a specified I/O port. A byte is an 8-bit transfer and a word is a 16-bit transfer. Call statements found in *CRUNCHER* using these routines will have both the port address and some appropriate value enclosed in parentheses. The second module (*CRUNCHER.BAS*) consists of the declarations for all of the variables used in the program. These declarations are placed in a separate module and given global attributes so that they are available to all three of the forms which constitute the source code of *CRUNCHER*.

The most important of the forms is *CRUNCHER.FRM*. The GUI for this form is shown in figure 4.1. When *CRUNCHER* is initially launched, the *Form_Load* subroutine is executed. The first line of this subroutine, and of all the subroutines, is an error-trapping scheme that handles any mistakes made by the user or by the operation of the program. After clearing and resetting some arrays, the subroutine prompts the user for information: how many multi-channel scalars (MCS's) will be required, for how many seconds per channel will the MCS's pause to count, how many channels and complete scans are desired, and how sensitive should the display be for the PSD of the CMA. The results of these queries are then displayed on the form itself. In the upper left-hand corner of the form, there is a menu heading entitled, *Important Information*. Once the user clicks on this heading, four more headings appear (*Coincidence Display*, *Control Grid*, *Hardware and Forms*, *MCS Display*); each one linked to a display of text elaborating on important details concerning the execution of *CRUNCHER*. Located on the right-hand

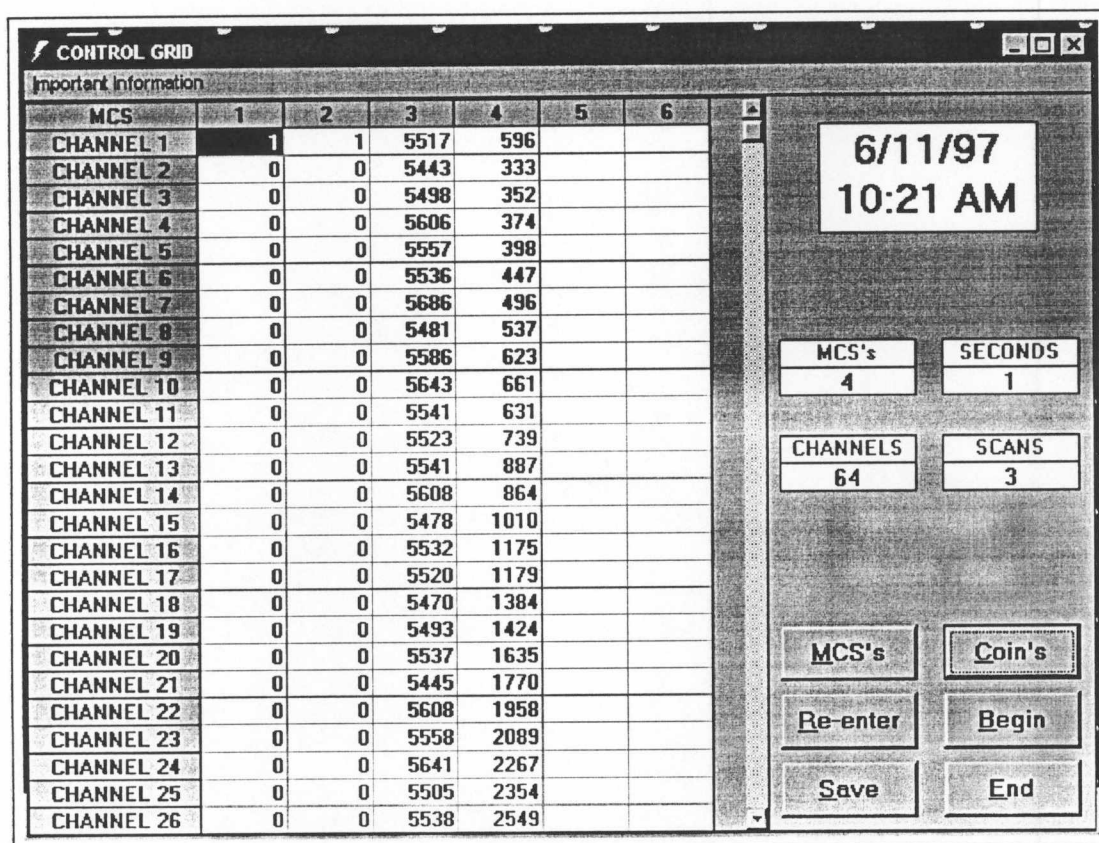


Figure 4.1: The graphical user interface for *CRUNCHER.FRM*.

side of the form are six command buttons. When the user clicks on any of these buttons, the respective subroutine is initiated.

The main subroutine (*cmdBegin_Click*) of *CRUNCHER.FRM* contains three important loops. The timer loop is where the data encoding the positions of the electrons striking the PSD of the CMA is actually acquired. Also during this loop, the events from up to six different sources are counted. Encompassing the timer loop is the channel loop, where all of the data is processed and displayed. Both of these loops are further encompassed by the scan loop, which allows the entire data acquisition process to be repeated over and over again. All three of the loops are described in detail.

The *cmdBegin_Click* subroutine starts by organizing the grid found on the left-hand side of the form. After clearing and resetting several arrays, the scan loop begins. The subroutine accesses the data direction register of the PIA and makes all of the corresponding I/O lines outputs. Then the output register is selected so that data can now be written to the PIA. Immediately following this initialization, the subroutine writes the zero channel number to the PIA which in turn sends the zero channel number to the digital-to-analog converter (DAC). The DAC is responsible for stepping the CMA voltage so that each channel of the spectrum corresponds to a particular electron energy. The grid is cleared, and a one and a zero are written to the PIA in order for a strobe to be sent to the DAC. The DAC does not acknowledge the channel number until the strobe is received.

The subsequent code initializes both of the STC's one at a time. First, a master reset is issued to the command register which essentially clears all the registers of the STC. The command register is accessed again in order to select the master mode control register. This register controls certain internal activities that are not regulated by the individual counter mode registers. Two call statements for the master mode preferences are required because the register is 16-bit but must be accessed through an 8-bit port, least significant byte first. Next, each counter mode register (already referred to as an MCS) is accessed in the same way to account for even more preferences. Originally for testing purposes, one preference was to use one of the five internal timebase clocks supplied by the *PC-TIO-10* board as a counting source, i.e. the clock signal was used as fake data which could be counted. Later, this was changed so that external events would be

counted. For event counting, each MCS is programmed to count rising edges applied to a certain STC source input. The counter value can then be read to determine the number of edges that have occurred.

The next three lines of code initialize the configuration registers on the *AT-DIO-32F* board. *Register 2* is assigned all zeros because *Group 2*, containing *Ports C* and *D*, is not needed by *CRUNCHER*. The expansion of *CRUNCHER* to use these ports is possible should future experiments require it. *Register 3* is programmed so that *Group 1*, containing *Ports A* and *B*, is configured for a read operation. *Register 1* is programmed to enable both double buffering and level mode handshaking for *Group 1*. Double buffering is when the incoming data is loaded into the first buffer of the port and held there until an additional signal arrives. Then the data is transferred to the second buffer which dumps the data to the digital I/O connector. In level mode handshaking (see figure 4.2)³², the incoming data produces a request (REQ) signal which in turn yields a data transfer ready (DRDY) signal. *CRUNCHER* reads (RD) the DRDY signal and has the option of a delay (TDELAY) before a handshaking acknowledgment (ACK) signal is sent to complete the data acquisition cycle. *CRUNCHER* is finally ready to acquire data. Now, the subroutine loads and arms all of the MCS's by accessing the command registers of both STC's, and thus the acquisition of the MCS data is begun.

The timer loop is where the data encoding the positions of the electrons striking the PSD of the CMA is actually acquired. The only two actions this loop performs are checking for data and reading the data. The exclusive nature of the timer loop optimizes the acquisition performance. *CRUNCHER* checks for data by looking for a one at the

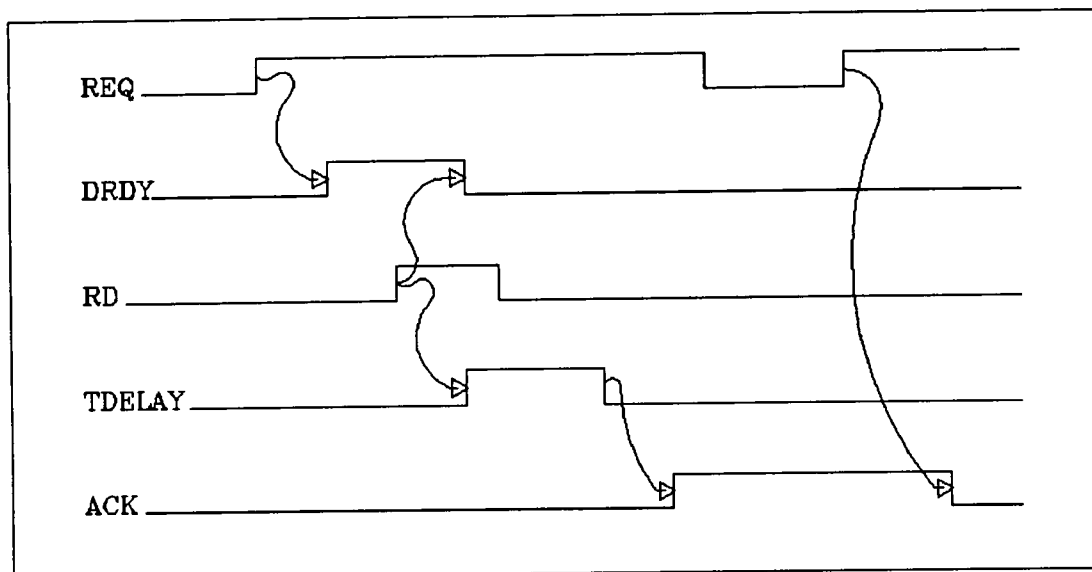


Figure 4.2: Diagram of level mode read handshake timing. (from National Instruments Corporation, *AT-DIO-32F User Manual*, (Austin, 1990))

DRDY bit of the *STAT* register on the *AT-DIO-32F* board. If data is present, it is read from the *Port A* register, which connects to the digital I/O connector, and is loaded into an array. Immediately following the conclusion of the timer loop, all of the MCS's are disarmed and their contents are saved through the command registers of both STC's. Also, a strobe is once again sent to the DAC to acknowledge the channel number which was sent to the DAC at the beginning of the channel loop.

While the channel loop does contain the timer loop where data is acquired, its major components are the several blocks of code which process and display both sets of data. First, the hold register for each MCS is selected so that the corresponding data may be accessed. Then, the 16 bits of data are read 8 bits at a time, loaded into an array, and finally displayed on the grid of *CRUNCHER.FRM*. Next, the data is placed on its respective plot of counts versus channel number found on *MCS.FRM* (see figure 4.3).

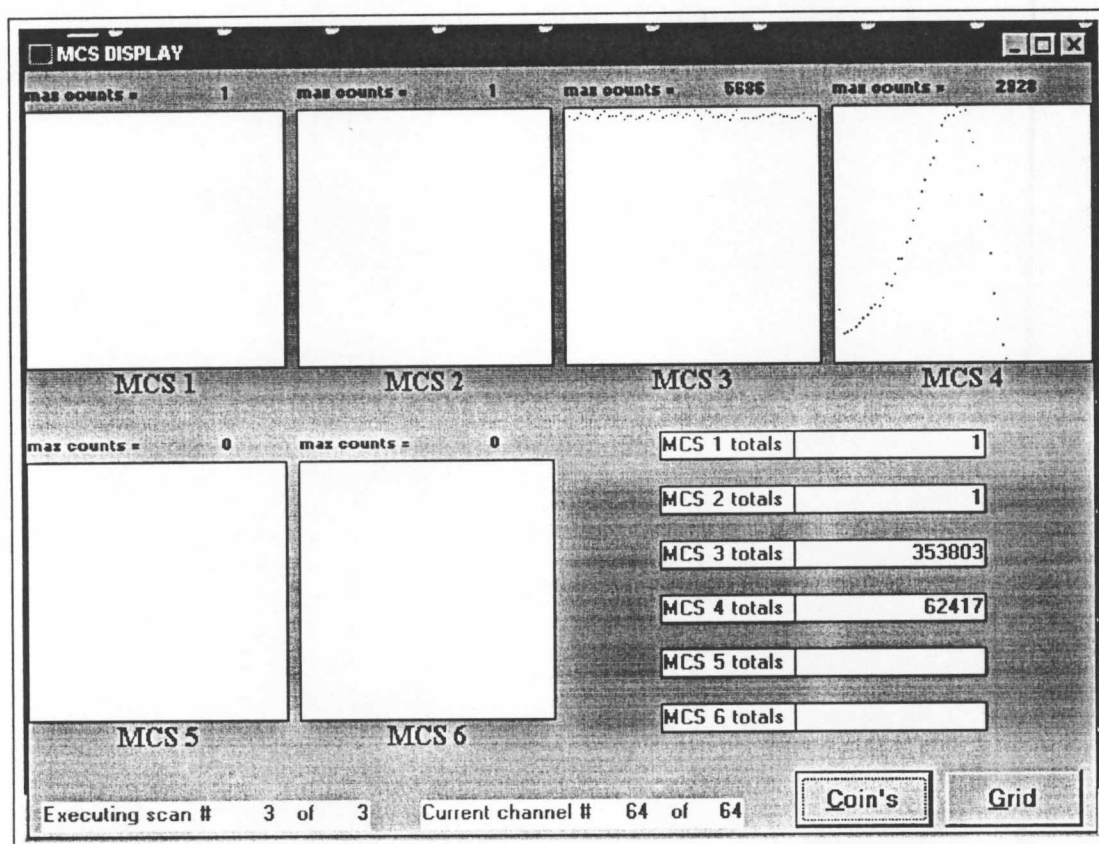


Figure 4.3: The graphical user interface for *MCS.FRM*.

Each plot is automatically scaled to reflect the maximum count value and the choice of total channel numbers. This form also shows the current scan number, the current channel number, and the sum of all the counts of all the channels for each plot, i.e. the totals. In the bottom right-hand corner are two command buttons that when clicked display either of the other two forms of *CRUNCHER*. This is the only code associated with *MCS.FRM* other than the settings for its GUI.

During the timer loop, the position data from the PSD of the CMA is already assigned to an array. Now each element of the array is broken down into three parts. The first part determines whether the incident electron is in coincidence with any recoil ion

detected by the TOF, and if so which one. The remaining two parts are the x and y coordinates of the impact on the PSD. The data is then placed on its respective plot found on *COINS.FRM* (see figure 4.4). This form also shows the sum of all the counts of all the coordinates for each plot, i.e. the totals. All of the plots feature an ascending color scale representing the relative number of counts for every coordinate. The sensitivity of the color scale may be changed at any time for each plot by clicking on the plot itself. This does not change the actual values of the data. In the bottom right-hand corner are two command buttons that when clicked display either of the other two forms of *CRUNCHER*. This, along with the sensitivity change, is the only code associated with *COINS.FRM* other than the settings for its GUI. Now that both sets of data have been processed and displayed, the subroutine reinitializes the registers of the *AT-DIO-32F* board and loads and arms all of the MCS's of the *PC-TIO-10* board. *CRUNCHER* is again ready to acquire data.

The remaining command buttons on *CRUNCHER.FRM* allow the user to end the program, reenter the information asked for when *CRUNCHER* initially loads, display either of the other two forms of *CRUNCHER*, and save the data to the hard drive. The data is saved sequentially and in the following order: number of scans, totals, and data. All of the command buttons are available to the user during the entire operation of *CRUNCHER*.

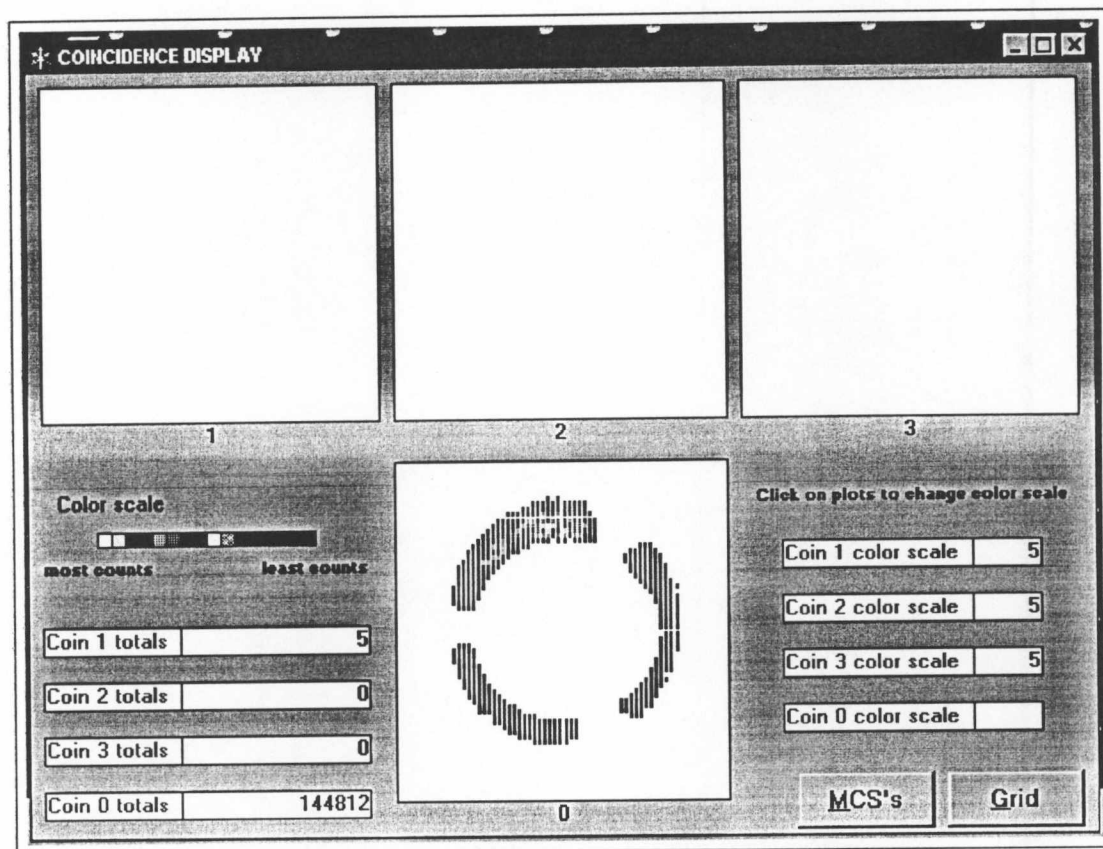


Figure 4.4: The graphical user interface for *COINS.FRM*.

5 Data Analysis and Results

In order to test the data acquisition system and to calibrate the spectrometer, we have measured the elastic scattering cross section of electrons on Ar as a function of scattering angle in the angular range from 45° to 135° with respect to the beam direction. We have also measured the angular distribution of Ar LMM Auger electrons produced by an 800 eV electron beam within the same angular range. For both measurements, the electron beam current passing through the gas cell is collected in a Faraday cup. The total charge collected in the cup is digitized and the digitizer pulses are then counted. All measurements are normalized to the total number of counted pulses.

5.1 Elastic Scattering Data

The elastic scattering cross section measurements are made with 500 and 800 eV electrons at target gas pressures of 0.11, 0.22, and 0.62 mTorr. Electrons elastically scattered into the acceptance cone of the CMA are detected with the PSD to reveal their angular distribution. The total number of electrons arriving at the detector and their arrival positions are recorded. *CRUNCHER* is used to acquire the data. Some electrons which traverse the CMA can be the result of electrons scattering off the apertures, gas cell, or other surfaces. Using a well-collimated electron beam and having a good alignment of the electron gun can reduce this sort of background. In this experiment, this kind of background is negligible for the elastic scattering spectra.

The spectra consist of the number of electrons detected, normalized to the amount of incident charge for different gas cell pressures, as a function of analyzer voltage. The amount of incident charge is converted to the number of incident electrons which pass through the gas cell. Each data set is divided by both the gas cell pressure and the total number of incident electrons. In figures 5.1 and 5.2, the different spectra are plotted and compared to see that they are identical within statistical error. This procedure proves there is no gas pressure dependent background contribution and the measurements are made under single collision conditions.

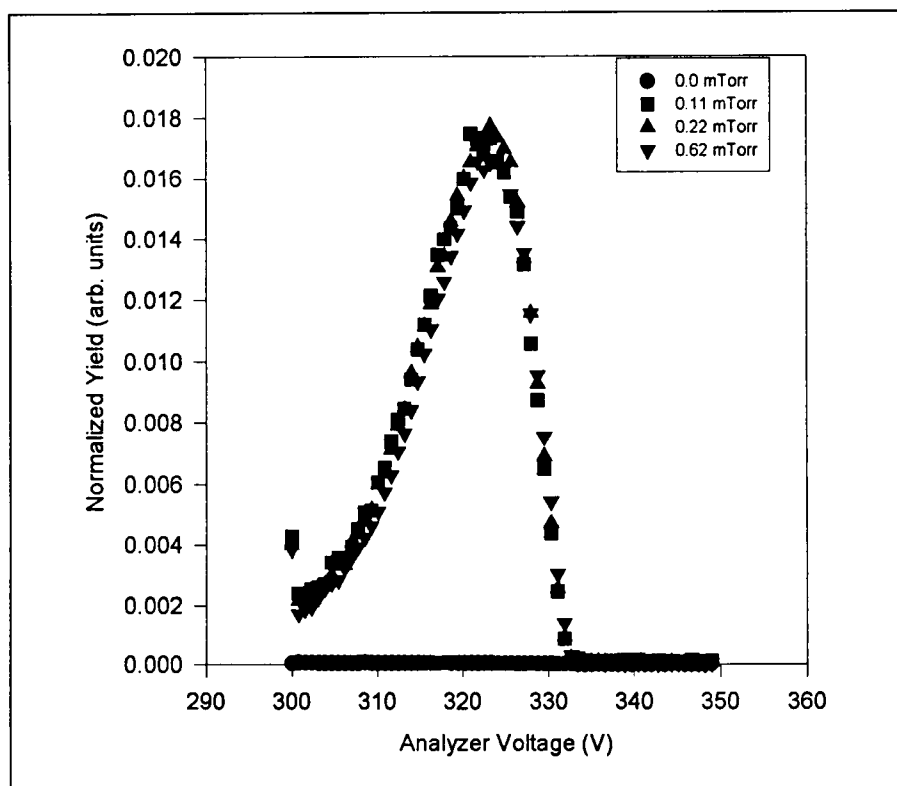


Figure 5.1: Normalized yields for the elastic scattering of 500 eV electrons incident upon Ar for the target gas pressures of 0.0, 0.11, 0.22, and 0.62 mTorr.

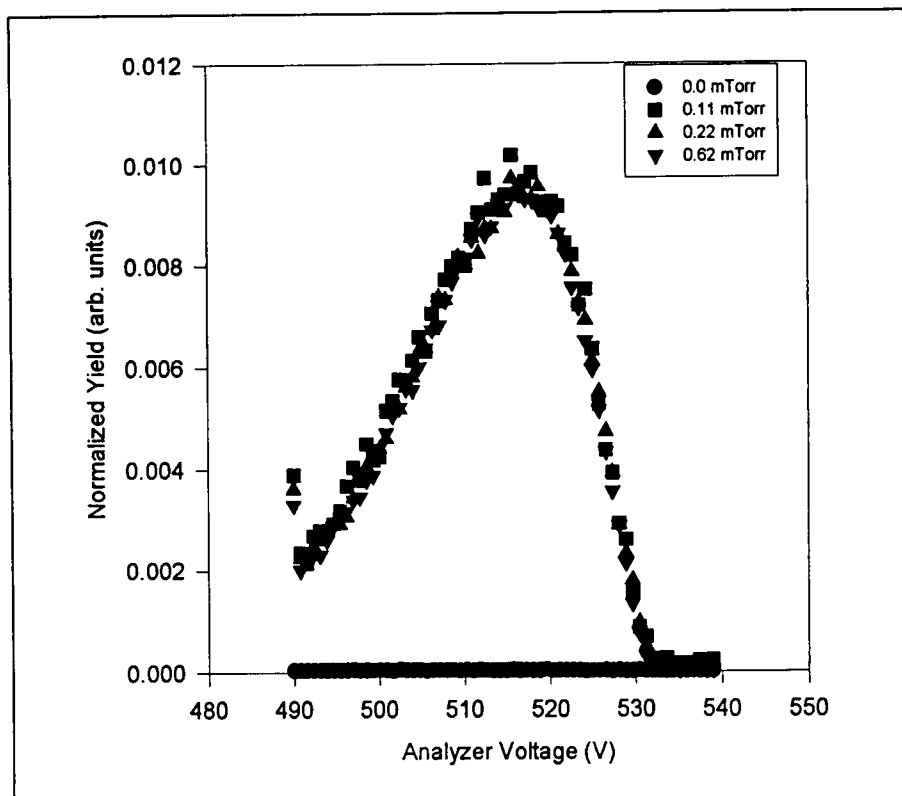


Figure 5.2: Normalized yields for the elastic scattering of 800 eV electrons incident upon Ar for the target gas pressures of 0.0, 0.11, 0.22, and 0.62 mTorr.

The process of obtaining the elastic scattering cross sections from the electron distributions on the PSD involves several steps. The first step requires the help of a simulation program which traces electron trajectories through the CMA (see Appendix B). Scattered electrons are born in the region where the electron beam interacts with the target gas. To calculate the probability that an electron produced in this region will reach the PSD, the program incorporates the exact dimensions of the analyzer including all apertures and slits. The simulation generates a uniform distribution of starting points in velocity space using a random number generator. The program assumes the electric field in the region between the cylinders of the CMA is equivalent to that of an infinitely long

cylindrical capacitor. It calculates the trajectory for each electron using the Runge-Kutta algorithm and determines if the electron will reach the PSD. The influence of fringe fields is accounted for by comparing selected trajectories with ones calculated by the *SIMION* program³³ and adding a correction term in order to make these trajectories agree.

The distributions of 500 and 800 eV elastically scattered electrons on the PSD are shown in figures 5.3 and 5.4, respectively. Three support posts are used to hold the inner cylinder of the CMA together. The presence of these posts results in three dead regions where no electrons can be observed.

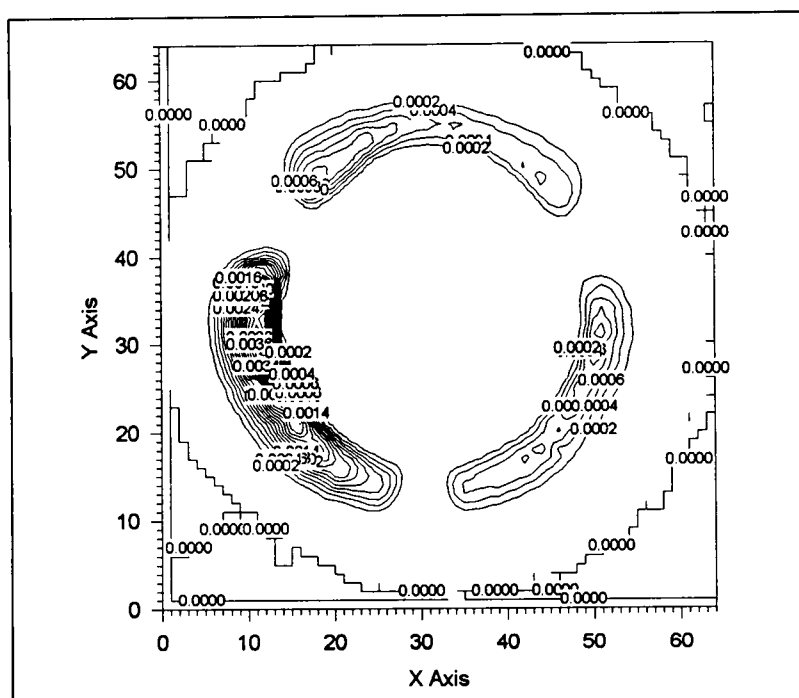


Figure 5.3: Contour plot of the arbitrarily normalized distribution of 500 eV elastically scattered electrons incident upon Ar for a target gas pressure of 0.62 mTorr. The most forward direction is toward the left.

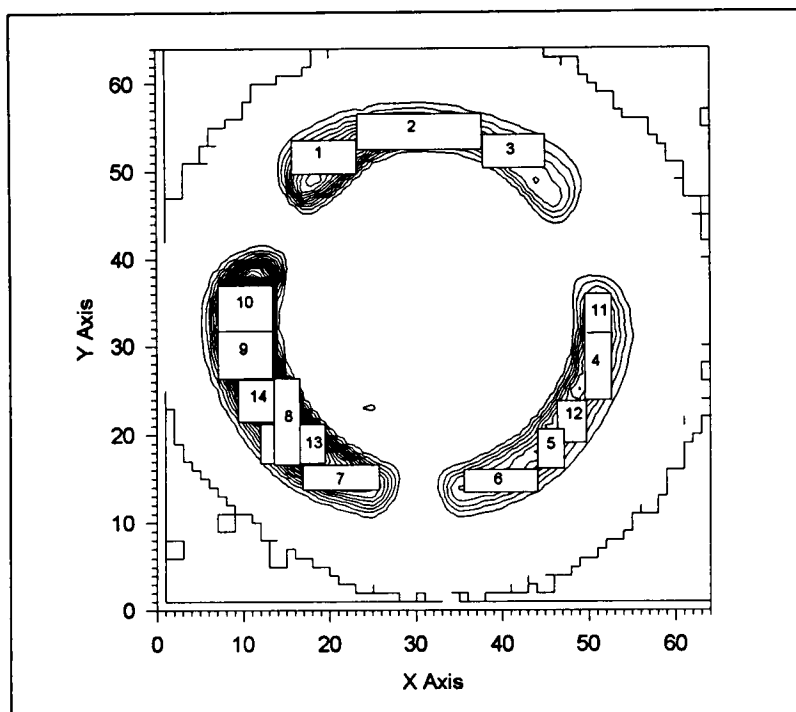


Figure 5.5: Diagram of the 14 designated regions on the PSD used in mapping the average emission angles.

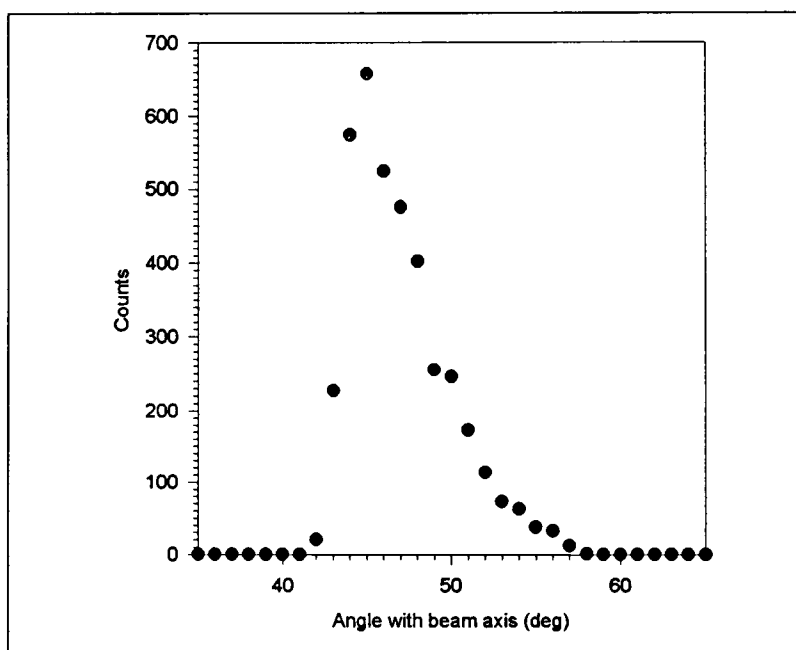


Figure 5.6: The number of electrons reaching region 9 on the PSD versus emission angle with respect to the beam axis.

In the simulation, the number of electrons which reach a chosen region of the detector is

$$N_j = N \int_{\Delta\Omega_{region}} P(\theta)T(\theta)d\Omega , \quad [5.1]$$

where N = the total number of trials

$T(\theta)$ = the transmission function of the analyzer

$P(\theta)d\Omega$ = the probability of an event being generated in the solid angle element $d\Omega$ around θ .

In the simulation $P(\theta) \Delta\Omega_{simulation} = 1$ and $P(\theta) = \text{constant}$, therefore $P(\theta)$ may come out of the integral so

$$N_j = N P(\theta) \int_{\Delta\Omega_{region}} T(\theta)d\Omega = \frac{N}{\Delta\Omega_{sim}} \int_{\Delta\Omega_{region}} T(\theta)d\Omega , \quad [5.2]$$

therefore

$$\int_{\Delta\Omega_{region}} T(\theta)d\Omega = \frac{N_j \Delta\Omega_{sim}}{N} = \epsilon . \quad [5.3]$$

In the experiment

$$N_{j\text{exp}} = N_i \int_{\Delta\Omega_{\text{region}}} P_{\text{exp}}(\theta) T(\theta) \eta(\theta) d\Omega , \quad [5.4]$$

where N_i = the number of electrons passing through the gas cell
 $P_{\text{exp}}(\theta)$ = the probability of an electron being scattered into a given
direction
 $\eta(\theta)$ = the electron detection efficiency.

Assuming $P_{\text{exp}}(\theta)$ and $\eta(\theta)$ vary little in the chosen region, they may come out of the
integral so

$$N_{j\text{exp}} = N_i P_{\text{exp}}(\theta) \eta(\theta) \int_{\Delta\Omega_{\text{region}}} T(\theta) d\Omega = N_i P_{\text{exp}}(\theta) \eta(\theta) \epsilon , \quad [5.5]$$

therefore

$$P_{\text{exp}}(\theta) = \frac{N_{j\text{exp}}}{N_i \eta(\theta) \epsilon} . \quad [5.6]$$

The relationship between the cross section $\frac{d\sigma}{d\Omega}$ and $P_{\text{exp}}(\theta)$ is given by

$$\frac{d\sigma}{d\Omega} = \frac{P_{\text{exp}}(\theta)}{n_0 p l} = \frac{N_{j\text{exp}}}{N_i \epsilon n_0 p l \eta(\theta)} ; \quad [5.7]$$

where n_0 = the number of target atoms per cm^3 per mTorr
 p = the pressure in mTorr
 l = the effective gas cell length in cm.

Using ε derived from the simulation, it should be possible to obtain the elastic scattering cross sections from the experimental data if $\eta(\theta) = \text{constant}$. Figures 5.7 and 5.8 compare these cross sections thus obtained with the published cross sections of DuBois and Rudd³⁴. The agreement is not very good. The assumption that $\eta(\theta) = \text{constant}$ over the PSD is probably not accurate, and using only the simulation program to calibrate the spectrometer is not advisable.

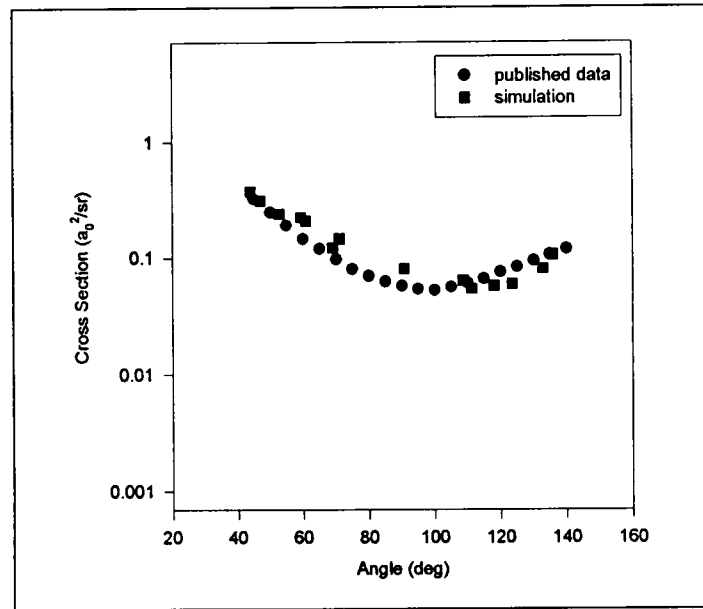


Figure 5.7: Comparison of the published cross sections for elastically scattered 500 eV electrons from Ar with the experimental cross sections obtained by using a transmission function from the simulation program.

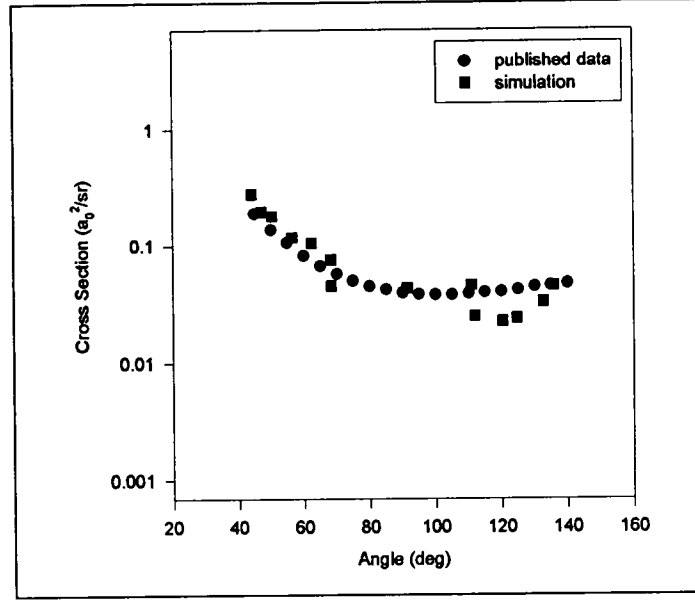


Figure 5.8: Comparison of the published cross sections for elastically scattered 800 eV electrons from Ar with the experimental cross sections obtained by using a transmission function from the simulation program.

We therefore decided to calibrate the CMA by using the published elastic scattering cross section data of DuBois and Rudd, which have already been shown to agree well with our computer-calculated cross sections. For each chosen region on the PSD corresponding to a scattering angle θ , the number of electrons detected in that region per mTorr of target gas pressure per incident electron is recorded. Using $\frac{d\sigma}{d\Omega} = \frac{N_{exp}}{p N_i} \xi$ with $\frac{d\sigma}{d\Omega}$ from DuBois and Rudd, ξ , which is the product of the transmission function of the CMA and the detection efficiency of the PSD, may be found. ξ is dependent on the geometry of the chosen region on the PSD. Using the PSD images of 500 eV elastically scattered electrons and the elastic scattering cross sections at 500 eV from DuBois and Rudd, we find ξ for each of our chosen regions on the PSD. This process is repeated for the 800 eV electrons so that an average ξ is determined for each region. Once the ξ 's are

established, the cross sections for the elastic scattering of electrons incident with different energies as well as the cross sections for other processes (e.g. free electron production in ion-atom collisions) may be determined. Using the ξ 's we obtain the measured elastic scattering cross sections at 500 and 800 eV. Figure 5.9 compares our measured cross sections with those of DuBois and Rudd and the agreement is quite good.

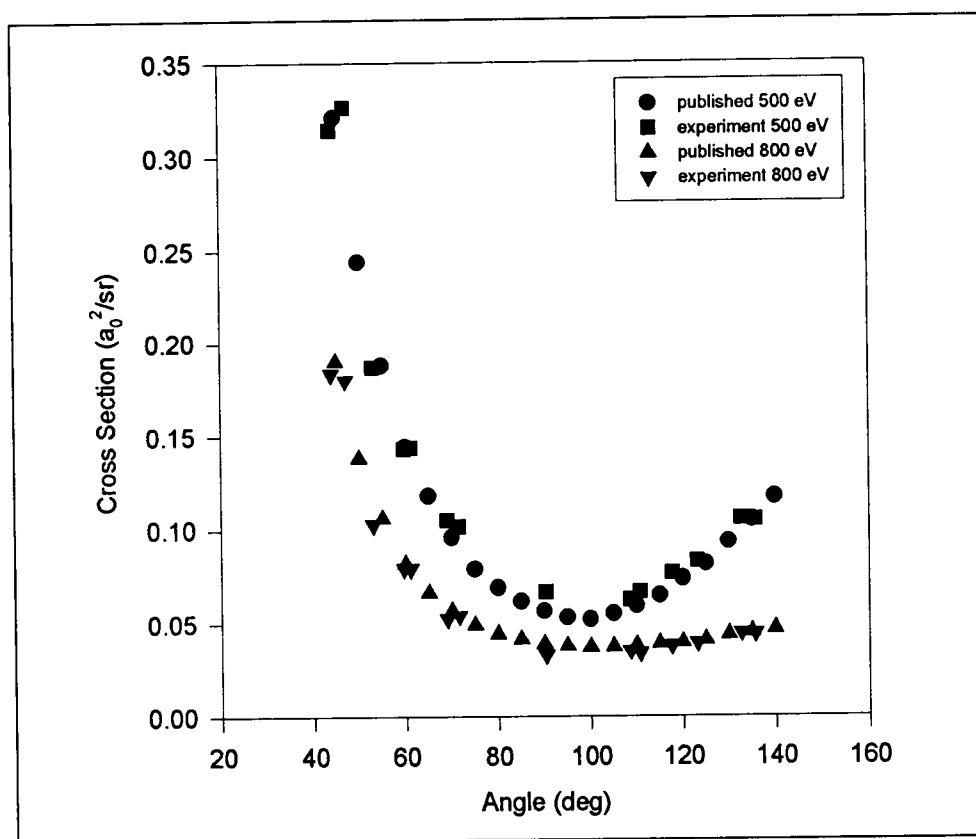


Figure 5.9: Comparison of the published cross sections for elastically scattered 500 and 800 eV electrons from Ar with the experimental cross sections obtained with the calibrated CMA.

5.2 Coincidence Data

CRUNCHER can measure two-dimensional spectra in coincidence with four different events. To test this capability of our new data acquisition system, we had planned to measure the energy and angular distribution of electrons ejected between 45° and 135° with respect to the beam direction in fast ion-atom collisions in coincidence with target recoil ions of different recoil ion charge states. However, due to funding constraints, the EN Tandem Van de Graff accelerator located at the Oak Ridge National Laboratory which was to provide the ion beam became unavailable during test time. Instead, the data acquisition system was tested using an electron beam.

Elastic scattering of electrons off neutral targets such as argon leaves the target undisturbed. Inelastic scattering often excites the target without ionizing it. The cross section for producing a recoil ion in coincidence with a particular scattered electron is very small. Target Auger electron production, however, always produces a recoil ion. An 800 eV electron incident on an Ar target can remove an L-shell electron. When this inner-shell hole is filled by an M-shell electron and another M-shell electron is ejected, an Ar^{2+} recoil ion is produced. It is possible that the incident electron removes an outer-shell electron in addition to the L-shell electron. The corresponding Auger decay results in an Ar^{3+} recoil ion, but this process is very unlikely.

The probability of even producing a single inner-shell hole is not very high in 800 eV electron-Ar collisions. However, when we detect the resulting Auger electron, we know that a recoil ion has also been produced. We therefore decided to test the capabilities of *CRUNCHER* to make coincidence measurements by measuring the angular

distributions of Ar $L_{2,3}M_{2,3}M_{2,3}$ Auger electrons in coincidence with Ar^{2+} recoil ions. The limited resolution of the CMA did not allow us to resolve the individual $L_{2,3}M_{2,3}M_{2,3}$ lines.

The coincidence measurements are made with 800 eV electrons incident on Ar at a target gas pressure of 2.3 mTorr. The angular distribution of emitted Ar $L_{2,3}M_{2,3}M_{2,3}$ Auger electrons is determined by the PSD of the CMA. The total number of electrons arriving at the detector and their arrival positions are recorded. *CRUNCHER* is used to acquire the data. The events are sorted by *CRUNCHER* into electrons that are detected in coincidence with a recoil ion that arrives after the characteristic flight time of an Ar^{2+} recoil ion and electrons for which a recoil ion does not arrive in the given time interval.

The main source of background for the coincidence data is due to random events, which are caused by electrons and recoil ions from different collisions starting and stopping the TAC, respectively. The number of random coincidences is determined from the corresponding ADC spectrum. The ADC spectrum shows the number of recoil ions observed after electron detection versus the time it takes the recoil ion to traverse the drift region of the TOF spectrometer. Figure 5.10 shows a typical ADC spectrum for this experiment. The large, broad peak corresponds to the recoil ion charge state of 2^+ . It rests on a nearly straight line background of random events. A small Ar^{3+} peak can also be seen resting on the background.

The energy spectrum of electrons measured in coincidence with Ar^{2+} recoil ions contains all events that fall into the window set with the SCA around the corresponding recoil ion peak in the ADC spectrum. The ratio of peak-to-background is determined by dividing the number of counts in the peak by the number of counts found in a chosen

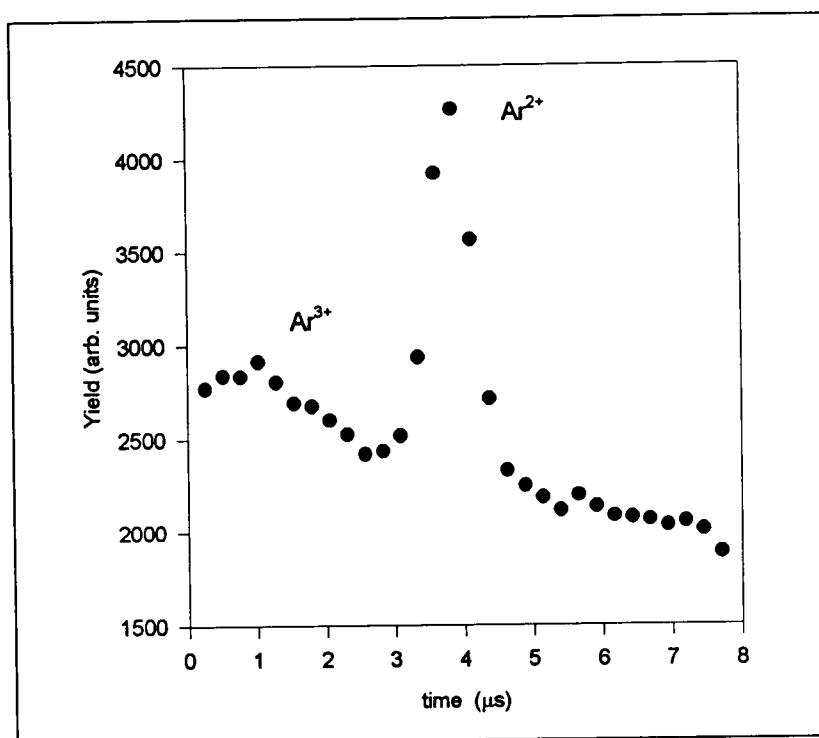


Figure 5.10: The ADC spectrum of the Ar^{2+} recoil-ion charge state peak resting on a background of random coincidences.

straight line section of the same width in the ADC spectrum. The PSD image of all electrons reaching the detector is normalized to the number of random coincidences and subtracted from the image of electrons detected in coincidence with a recoil ion in the chosen time interval. In this way, we obtain the spectrum of real coincidences.

Figure 5.11 shows a contour plot of a PSD image of $\text{Ar } L_{2,3}M_{2,3}M_{2,3}$ Auger electrons measured in coincidence with Ar^{2+} recoil ions after random coincidences have been subtracted. The spectrum has very poor statistics because a high number of random coincidences has been subtracted. In the energy distribution of ~ 200 eV electrons, the characteristic Ar LMM Auger peak rides on top of a large number of scattered electrons. In addition, many Ar^{1+} recoil ions are produced in different collisions. Since the MSA has

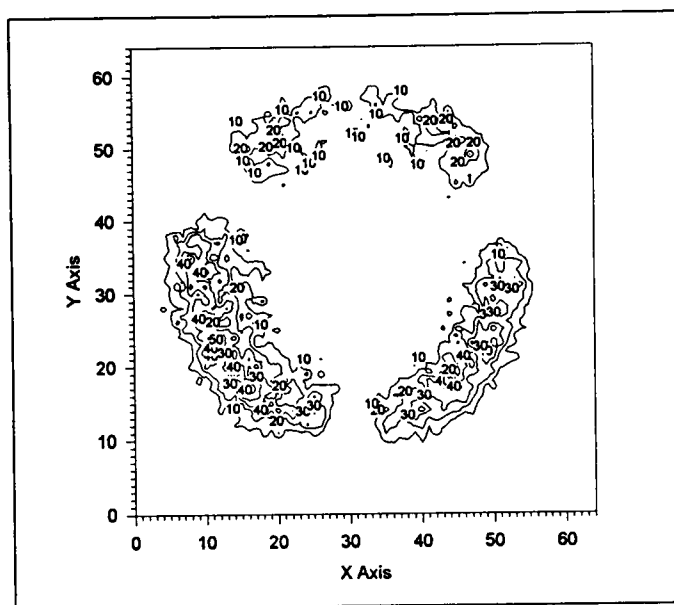


Figure 5.11: Contour plot of the Ar $L_{2,3}M_{2,3}M_{2,3}$ Auger electrons measured in coincidence with Ar^{2+} recoil ions after the subtraction of random coincidences. The most forward direction is toward the left.

been designed for measuring electron distributions from ion-atom collisions, it is not the ideal apparatus for making these measurements. However, the measurements do show that *CRUNCHER* can indeed handle coincidence experiments.

The Ar $L_{2,3}M_{2,3}M_{2,3}$ Auger electron production cross sections are determined from the PSD image as described in section 5.1 where the CMA is calibrated using previously measured elastic scattering cross sections. To find absolute coincidence cross sections, the recoil ion detection efficiency must also be known. It can be determined from our measurements by noting that each Auger electron is produced together with a recoil ion. The fraction of Auger electrons detected in coincidence with a recoil ion of any charge state therefore yields the recoil ion detection efficiency. However, to accurately subtract

the background of scattered electrons from the Auger electron peak, better statistics are needed. An apparatus which better discriminates against this background would also be helpful. Figure 5.12 therefore shows our measured Ar $L_{2,3}M_{2,3}M_{2,3}$ Auger electron production cross sections multiplied by the recoil ion detection efficiency. The data have poor statistics, but nonetheless suggest a distribution which is not isotropic.

5.3 Summary

A new data acquisition system has been created. It is designed to let the experimenter make full use of the capabilities of several multi-spectrometer apparatus. By accessing two digital I/O interfaces which have been installed in a personal computer, the

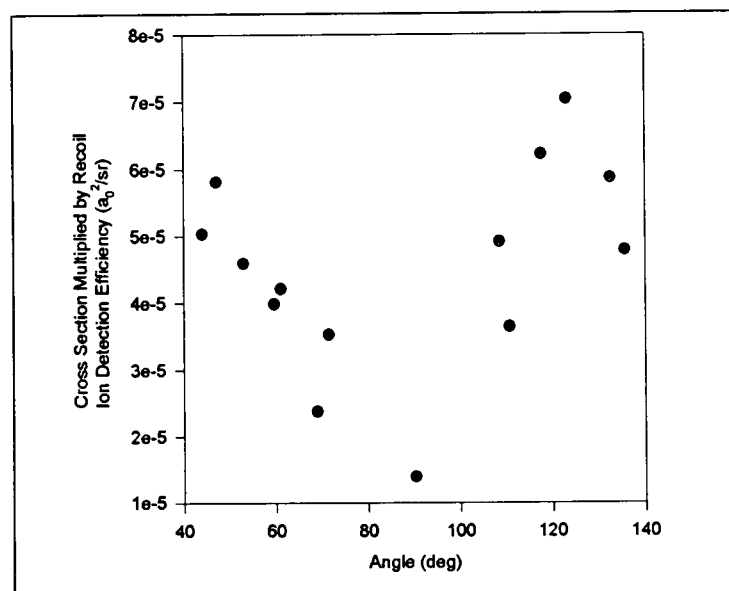


Figure 5.12: The Ar $L_{2,3}M_{2,3}M_{2,3}$ Auger electron production cross section multiplied by the recoil ion detection efficiency as a function of emission angle with respect to the 800 eV incident electron beam.

new system is capable of simultaneously counting events from six different sources and can acquire two-dimensional spectra which may be measured in coincidence with four different events. The new system reads, processes, displays, and stores all of these data.

The new data acquisition system has successfully been tested. We have measured the elastic scattering cross section for electrons incident upon argon and scattered into the angular range from 45° to 135° with respect to the beam axis for two different electron energies (500 eV, 800 eV). We have also measured the angular distribution of $L_{2,3}M_{2,3}M_{2,3}$ Auger electrons produced in coincidence with Ar^{2+} target recoil ions.

Since the previously measured elastic scattering cross sections of DuBois and Rudd were available, our measurements allowed us to calibrate the CMA of the MSA. Our measurements showed that calibrations using only the simulation program were not entirely satisfactory. While the statistics of the Auger electron-recoil ion coincidence results were poor, those measurements still proved the capability of the new data acquisition system to successfully make coincidence measurements. The versatility of the new system will allow it to be used with many different experimental setups which may be required for future ion-atom collision experiments.

References

1. M. Breinig, J.W. Berryman, F. Segner, and D.D. Desai, Phys. Rev. A 50, 4905 (1994)
2. A. Wig, Master's Thesis, University of Tennessee, 1995 (unpublished)
3. F. Segner, M. Breinig, D.D. Desai, A. Wig, and L. Straus, Phys. Rev. A 54, 1385 (1996)
4. R.H. Garvey, C.H. Jackman, and A.E.S. Green, Phys. Rev. A 12, 1144 (1975)
5. R.D. DuBois and M.E. Rudd, J. Phys. B 9, 2657 (1976)
6. E.J. McGuire, *Atomic Inner Shell Processes*, ed. by B. Crasemann, (Academic Press, New York, 1975) pp. 293
7. P. Auger, Compt. Rend. 177, 169 (1923) ; 180, 65 (1925) ; P. Auger, J. Phys. Radium 6, 205 (1925)
8. G. Wentzel, Z. Phys. 43, 254 (1927)
9. W. Mehlhorn and D. Stalhern, Z. Phys. 217, 294 (1968)
10. E.J. McGuire, *Atomic Inner Shell Processes*, ed. by B. Crasemann, (Academic Press, New York, 1975) pp. 293
11. P. Richard, *Atomic Inner Shell Processes*, ed. by B. Crasemann, (Academic Press, New York, 1975) pp. 73
12. D.L. Matthews, B.M. Johnson, J.J. Mackey, and C.F. Moore, Phys. Rev. Lett. 31, 1331 (1973)
13. M. Breinig, J.W. Berryman, F. Segner, and D.D. Desai, Phys. Rev. A 50, 4905 (1994)
14. L.O. Werme, T. Bergmark, and K. Siegbahn, Physica Scripta 8, 149 (1973)
15. M. Breinig, J.W. Berryman, F. Segner, and D.D. Desai, Phys. Rev. A 50, 4905 (1994)
16. F. Segner, Ph.D. Dissertation, Johann Wolfgang Goethe University, 1996 (unpublished)
17. J.H. Moore, C.C. Davis, and M.A. Coplan, *Building Scientific Apparatus*, (Addison-Wesley Publishing Company, Reading, 1989) pp. 321-323
18. J. Berryman, Master's Thesis, University of Tennessee, 1993 (unpublished)

19. M. Breinig, J.W. Berryman, F. Segner, and D.D. Desai, Phys. Rev. A 50, 4905 (1994)
20. G.D. Yarnold and H.C. Bolton, J. Sci. Instr. 26, 38 (1949)
21. T.S. Green and G.A. Proca, Rev. Sci. Instr. 41, 1409 (1970)
22. H.Z. Sar-El, Rev. Sci. Instr. 38, 1210 (1967)
23. S. Adsela, M. Karras, M. Pessa, and E. Suoninen, Rev. Sci. Instr. 41, 351 (1970)
24. J.S. Risley, Rev. Sci. Instr. 43, 95 (1972)
25. J.H. Moore, C.C. Davis, and M.A. Coplan, *Building Scientific Apparatus*, (Addison-Wesley Publishing Company, Reading, 1989) pp. 328
26. J.L. Wiza, Nucl. Instr. and Meth. 162, 587 (1979)
27. W.C. Wiley and I.H. McLaren, Rev. Sci. Instr. 26, 1150 (1955)
28. Ibid
29. F. Segner, Ph.D. Dissertation, Johann Wolfgang Goethe University, 1996 (unpublished)
30. National Instruments Corporation, *PC-TIO-10 User Manual*, (Austin, 1991)
31. National Instruments Corporation, *AT-DIO-32F User Manual*, (Austin, 1990)
32. Ibid
33. C. Gaither, Ph.D. Dissertation, University of Tennessee, 1992 (unpublished)
34. R.D. DuBois and M.E. Rudd, J. Phys. B 9, 2657 (1976)

Appendices

Appendix A

'This program calculates the differential scattering cross section for free
'electrons incident on a central potential. The potential is a model
'potential for an Ar target. The maximum electron energy e is 1000 eV.
'This program outputs the cross section for angles
'between 0 and 180 degrees in units of angstrom^2.

```
DECLARE SUB bessl ()
COMMON SHARED offset%, lmax%, x
'These variables are shared with the subroutine bessl.
CLS
```

```
'Here constants are declared.
pi = 3.14159: hbar = 7.62003:
'hbar = (hbar^2)/m in units of eV*angstrom^2
rmax = 3:      rext = 3.2:      dr = .000002
'The constants rmax, rext, and dr may be changed to test the program.
'The constants rmax and rext should be chosen bigger for bigger atoms.
nr = rmax / dr: nrext = rext / dr:      nang% = 36
e = 1000
k2 = 2 * e / hbar: k = SQR(k2)
lmax% = k * rmax / 2 + 4
'For l% > lmax% the phase shift delta(l%) will be zero.
lstart% = 0
```

```
'Here we declare arrays and double precision variables.
DIM pl(lmax%, nang%), dsigma(nang%)
DIM SHARED jl(2 * lmax% + 1), nl(2 * lmax% + 1), delta(lmax%)
DIM psi1 AS DOUBLE, psi2 AS DOUBLE, psi3 AS DOUBLE, ps AS DOUBLE
DIM psinr AS DOUBLE
```

'This is a routine to calculate and store the Legendre polynomials for all
'l% from zero to lmax% and for all angles from 0 to 180 degrees in 5 degree
'intervals. The routine uses a recursion formula.

```
FOR i% = 0 TO nang%
  theta = i% * pi / nang%
  x = COS(theta)
  pl(0, i%) = 1
  pl(1, i%) = x
  FOR l% = 1 TO lmax% - 1
    pli = (2 * l% + 1) * x * pl(l%, i%)
    pl(l% + 1, i%) = (pli - l% * pl(l% - 1, i%)) / (l% + 1)
  NEXT l%
NEXT i%
```

```
'This routine calculates the energy-dependent variables. Here you  
'can choose the energy.
e = 200
PRINT "energy = ", e
k2 = 2 * e / hbar: k = SQR(k2)
lmax% = k * rmax / 2 + 4
PRINT "lmax = ", lmax%
```

'Smaller energies get a smaller lmax%.

'Here we calculate the Bessel fuctions of rmax and rext for all l%
'from zero to lmax% by calling a subroutine. The subroutine uses
'a recursion formula.

```
x = k * rmax
offset% = 0
CALL bessel
x = k * rext
offset% = lmax% + 1
CALL bessel
```

'routine to integrate the Schroedinger equation using the Numerov method

```
cons = dr * dr / (hbarm * 6)
PRINT "l", "delta"
FOR l% = lstart% TO lmax%
  'We have to do a sepearte integration for each l%.
  'First we plant the seeds to start the Numerov method.
```

```
    ll = l% * (l% + 1) * hbarm / 2
    psi1 = 0: psi2 = 1D-25
    r = 4 * dr
    a1 = 3.657 * (EXP(1.809 * r) - 1) + 1
    v = 14.3998 * (18 * (1 - 1 / a1) - 18) / r
    v = v + ll / (r * r)
    kim1 = cons * (e - v)
    r = 5 * dr
    a1 = 3.657 * (EXP(1.809 * r) - 1) + 1
    v = 14.3998 * (18 * (1 - 1 / a1) - 18) / r
    v = v + ll / (r * r)
    ki = cons * (e - v)
```

'Now we integrate using the Numerov method.

```
    FOR i = 5 TO nnext - 1
        r = (i + 1) * dr
        IF i + 1 > nr THEN v = 0: GOTO outer
        a1 = 3.657 * (EXP(1.809 * r) - 1) + 1
        v = 14.3998 * (18 * (1 - 1 / a1) - 18) / r
outer:    v = v + ll / (r * r)
        kip1 = cons * (e - v)
        ps = ((2 - 10 * ki) * psi2 - (1 + kim1) * psi1)
        psi3 = ps / (1 + kip1)
        kim1 = ki: ki = kip1: psi1 = psi2: psi2 = psi3
        IF i + 1 = nr THEN psinr = psi3
    NEXT i
```

'Here we calculate the phase shifts. This is where we need the previously
'computed Bessel functions.

```
g = rmax * psi3 / (rext * psinr)
numer = g * jl(l%) - jl(l% + lmax% + 1)
denom = g * nl(l%) - nl(l% + lmax% + 1)
delta(l%) = ATN(numer / denom)
PRINT l%, delta(l%)
```

NEXT l%

'Now that we have all the phase shifts we can calculate the differential
'cross section for a given energy as a function of angle.

'This routine calculates dsigma/domega.

```
FOR i% = 0 TO nang%
  FREAL = 0: FIMAG = 0
  FOR l% = lstart% TO lmax%
    SD = SIN(delta(l%))
    CD = COS(delta(l%))
    FREAL = FREAL + (2 * l% + 1) * SD * CD * pl(l%, i%)
    FIMAG = FIMAG + (2 * l% + 1) * SD * SD * pl(l%, i%)
  NEXT l%
  FREAL = FREAL / k
  FIMAG = FIMAG / k
  dsigma(i%) = FREAL * FREAL + FIMAG * FIMAG
NEXT i%
```

'Here we write the scattering cross sections to a file.

```
a$ = "ar" + RIGHT$(STR$(e), 3) + ".dat"
OPEN a$ FOR OUTPUT AS #1
PRINT "The cross sections are written to the file ", a$
FOR i% = 0 TO nang%
  PRINT #1, i% / .2, dsigma(i%)
NEXT i%
CLOSE
```

SUBessel

'subroutine to calculate the spherical Bessel functions

```
nl(offset%) = -COS(x) / x
nl(1 + offset%) = -(COS(x) / (x * x)) - SIN(x) / x
FOR l% = (1 + offset%) TO (lmax% - 1 + offset%) 'forward recursion
  ltrue% = l% - offset%
  nl(l% + 1) = (2 * ltrue% + 1) * nl(l%) / x - nl(l% - 1)
NEXT l%
lupper% = x + 10
jp1 = 0
j = 9.999999E-21
FOR l% = (lupper% + offset%) TO (1 + offset%) STEP -1 'backward recursion
  ltrue% = l% - offset%
  jm1 = (2 * ltrue% + 1) * j / x - jp1
  IF (l% - 1) <= (lmax% + offset%) THEN jl(l% - 1) = jm1
  jp1 = j: j = jm1
NEXT l%
norm = SIN(x) / (x * jl(offset%))
FOR l% = (offset%) TO (lmax% + offset%)
  jl(l%) = norm * jl(l%)
NEXT l%
END SUB
```

Appendix B

```

#include <stdio.h>
#include <stdlib.h>
#include <float.h>
#include <math.h>
#include <graph.h>
#include <time.h>

/* (I'm trying to stick with SI-units) */

#define E_OVER_M 1.75881962e11
#define D double
#define MBIG 1000000000
#define MSEED 161803398
#define MZ 0
#define FAC (1.0/MBIG)

int gascell(D*,D*,D*,D*,D*,D*,D*,D*);
int nofield(D*,D*,D*,D*,D*,D*,float*,float*);
int field(D*,D*,D*,D*,D*,D*,D*,int*,int*,float*,float*);
int cma_exit(D*,D*,D*,D*,D*,D*,D*,D*);
float ran3(long*);

main()
{
D x,y,z;
D vx,vy,vz;
D ctheta,stheta,phi,v0;
D u_left,u_right;
D volt,u_front;
D x0,y0,z0,a0,a02,b0,a1,a12,si,co,dvr;
D thmax, thmin, vmax, vmin, zmax, zmin;
int steps,steps1,index;
float sltbotedg,slttopedg,data[160],number;
long seed;
time_t ltime;
FILE *ft;

steps=1000;
steps1=steps/16;
volt=-63.0;
u_left=0;
u_right=0;
u_front=-4;
a0=1e+6*(D)E_OVER_M*(u_left-u_right)/12.5;
a02=a0*a0;
b0=1e+6*(volt-u_left)*(D)E_OVER_M/.93827;
a1=1e+6*(D)E_OVER_M*(u_front-u_left)/7.5;
a12=a1*a1;
thmax=0;
thmin=1;
vmax=0;
vmin=1e+10;
zmax=-6;
zmin=6;

```

```

seed=-1*time(&lttime);
_setvideomode(_VRES16COLOR);
_clearscreen(_GCLEARSCREEN);
_setwindow( 1, -24.0,-18.0, 24.0, 18.0 );

again:
index=0;
number=0;
while (index < 160)
{
    fini:
    z=-3.4+6.8*ran3(&seed);
    x=-1+1.6*ran3(&seed);
    y=-1+2*ran3(&seed);
    if((x*x+y*y)>1)
        goto fini;
    v0=5.86+0.06*ran3(&seed);
    x0=v0*v0;
    y0=37.95*ran3(&seed);
    if(y0<x0)
        goto fini;
    number=number+1;
    x0=x;y0=y;z0=z;
    phi=6.2831853718*ran3(&seed);
    ctheta= .73+.07*ran3(&seed);
    stheta=sqrt(1-ctheta*ctheta);
    v0=v0*1e+9;
    vx=v0*ctheta;
    vz=v0*stheta*sin(phi);
    vy=v0*stheta*cos(phi);

    if(gascell(&x,&y,&z,&vx,&vy,&vz,&a0,&a02)!=0)
        goto fini;
    sltbotedg=8;slttopedg=10.8;
    if(nofield(&x,&y,&z,&vx,&vy,&vz,&sltbotedg,&slttopedg)!=0)
        goto fini;
    sltbotedg=48.2;slttopedg=50.8;
    if(field(&x,&y,&z,&vx,&vy,&vz,&b0,&steps,&steps1,&sltbotedg,&slttopedg)!=0)
        goto fini;
    sltbotedg=69.6;slttopedg=72.2;
    if(nofield(&x,&y,&z,&vx,&vy,&vz,&sltbotedg,&slttopedg)!=0)
        goto fini;
    sltbotedg=110.2;slttopedg=112.9;
    if(field(&x,&y,&z,&vx,&vy,&vz,&b0,&steps,&steps1,&sltbotedg,&slttopedg)!=0)
        goto fini;
    if(cma_exit(&x,&y,&z,&vx,&vy,&vz,&a1,&a12)!=0)
        goto fini;
    if(ctheta>thmax)
        thmax=ctheta;
    if(ctheta<thmin)
        thmin=ctheta;
    if(v0>vmax)

```

```

        vmax=v0;
    if(v0<vmin)
        vmin=v0;
    if(z0>zmax)
        zmax=z0;
    if(z0<zmin)
        zmin=z0;
    if(fabs(z0)>2.5)
    {
        _setcolor(13);
        _setpixel_w( y, z );
    }
    else
    {
        _setcolor(15);
        _setpixel_w( y, z );
    }
    _settextposition(26,1);
    printf("%f %f\n",thmin,thmax);
    _settextposition(27,1);
    printf("%f %f\n",vmin*1e-9,vmax*1e-9);
    _settextposition(28,1);
    printf("%f %f\n",zmin,zmax);
    data[index]=x0;
    index++;
    data[index]=y0;
    index++;
    data[index]=z0;
    index++;
    data[index]=v0;
    index++;
    data[index]=ctheta;
    index++;
    data[index]=phi;
    index++;
    data[index]=y;
    index++;
    data[index]=z;
    index++;
}
if( (ft=fopen("cut.500","ab"))!= NULL)
{
    fwrite(data, sizeof(data), 1, ft);
    fwrite(&number, 4, 1, ft);
    fclose (ft);
}
goto again;
}

int gascell(D*x,D*y,D*z,D*vx,D*vy,D*vz,D*a0,D*a02)
{
    D t0,r2;

```

```

    if(*a0>0)
    {
        t0=sqrt(*vx**vx/(*a02)-2*(x-5)/(*a0))-vx/(*a0);
        vx=vx+a0*t0;
    }
    else
    if(*a0<0)
    {
        t0=-sqrt(*vx**vx/(*a02)-2*(x-5)/(*a0))-vx/(*a0);
        vx=vx+a0*t0;
    }

    else
    {
        t0=(5-x)/(vx);
    }
    x=5;
    y=y+vy*t0;
    z=z+vz*t0;
    r2=y**y+z**z;
    if(r2>49||r2<9)
    return -1;
    return 0;
}

```

```

int nofield(D*x,D*y,D*z,D*vx,D*vy,D*vz,float*sltbotedg,float*slttopedg)
{
    D t1,p,q;
    p=(z**vz+y**vy)/(vz**vz+vy**vy);
    q=((y**y)+(z**z)-(81))/(vz**vz+vy**vy);
    t1=sqrt(p*p-q)-p;
    x=x+vx*t1; y=y+vy*t1; z=z+vz*t1;
    if(x<*sltbotedg||x>*slttopedg)
        return -1;
    return 0;
}

```

```

int field(D*x,D*y,D*z,D*vx,D*vy,D*vz,D*b0,int*steps,int*steps1,float*sltbotedg,float*slttopedg)
{
    D r2,t0,t1,dt;
    D var[5],oldvar[5],k1[5],k2[5],k3[5],k4[5],f[5];
    int ii,jj;
    t0=(sltbotedg-x)/(vx);
    dt=t0/(steps);
    var[1]=y;
    var[2]=z;
    var[3]=vy;
    var[4]=vz;
    r2=var[1]*var[1]+var[2]*var[2];
    for(ii=0;ii<*steps;ii++)

```

```

{
f[1]=var[3];
f[2]=var[4];
f[3]=*b0*var[1]/r2;
f[4]=*b0*var[2]/r2;
for(jj=1;jj<5;jj++)
{
oldvar[jj]=var[jj];
k1[jj]=dt*f[jj];
var[jj]=oldvar[jj]+k1[jj]/2;
}
r2=var[1]*var[1]+var[2]*var[2];
f[1]=var[3];
f[2]=var[4];
f[3]=*b0*var[1]/r2;
f[4]=*b0*var[2]/r2;
for(jj=1;jj<5;jj++)
{
k2[jj]=dt*f[jj];
var[jj]=oldvar[jj]+k2[jj]/2;
}
r2=var[1]*var[1]+var[2]*var[2];
f[1]=var[3];
f[2]=var[4];
f[3]=*b0*var[1]/r2;
f[4]=*b0*var[2]/r2;
for(jj=1;jj<5;jj++)
{
k3[jj]=dt*f[jj];
var[jj]=oldvar[jj]+k3[jj];
}
r2=var[1]*var[1]+var[2]*var[2];
f[1]=var[3];
f[2]=var[4];
f[3]=*b0*var[1]/r2;
f[4]=*b0*var[2]/r2;
for(jj=1;jj<5;jj++)
{
k4[jj]=dt*f[jj];
var[jj]=oldvar[jj]+(k1[jj]+2*k2[jj]+2*k3[jj]+k4[jj])/6;
}
r2=var[1]*var[1]+var[2]*var[2];
*x=*x+*vx*dt;
if(r2<81||r2>529||r2==0.0)
{
return -1;
}
}
t1=(*slttopdg-*x)/(*vx);
dt=t1/(*steps1);
for(ii=0;ii<*steps1;ii++)
{

```

```

f[1]=var[3];
f[2]=var[4];
f[3]=*b0*var[1]/r2;
f[4]=*b0*var[2]/r2;
for(jj=1;jj<5;jj++)
{
    oldvar[jj]=var[jj];
    k1[jj]=dt*f[jj];
    var[jj]=oldvar[jj]+k1[jj]/2;
}
r2=var[1]*var[1]+var[2]*var[2];
f[1]=var[3];
f[2]=var[4];
f[3]=*b0*var[1]/r2;
f[4]=*b0*var[2]/r2;
for(jj=1;jj<5;jj++)
{
    k2[jj]=dt*f[jj];
    var[jj]=oldvar[jj]+k2[jj]/2;
}
r2=var[1]*var[1]+var[2]*var[2];
f[1]=var[3];
f[2]=var[4];
f[3]=*b0*var[1]/r2;
f[4]=*b0*var[2]/r2;
for(jj=1;jj<5;jj++)
{
    k3[jj]=dt*f[jj];
    var[jj]=oldvar[jj]+k3[jj];
}
r2=var[1]*var[1]+var[2]*var[2];
f[1]=var[3];
f[2]=var[4];
f[3]=*b0*var[1]/r2;
f[4]=*b0*var[2]/r2;
for(jj=1;jj<5;jj++)
{
    k4[jj]=dt*f[jj];
    var[jj]=oldvar[jj]+(k1[jj]+2*k2[jj]+2*k3[jj]+k4[jj])/6;
}
r2=var[1]*var[1]+var[2]*var[2];
*x=*x+*vx*dt;
if(r2<=81)
{
    *y=var[1];
    *z=var[2];
    *vy=var[3];
    *vz=var[4];
    return 0;
}
}
return -1;

```

```

}

int cma_exit(D*x,D*y,D*z,D*vx,D*vy,D*vz,D*a1,D*a12)
{
    D r2,t0,t1;
    t0=(121.8-*x)/(*vx);
    *x=*x+*vx*t0;*y=*y+*vy*t0;*z=*z+*vz*t0;
    r2=*y*y+*z*z;
    if(r2>6.25)
    {
        return -1;
    }
    if(*a1==0.0)
    {
        t1=(131.1-*x)/(*vx);
        *y=*y+*vy*t1;*z=*z+*vz*t1;
    }
    else
    if (*a1>0.0)
    {
        t1=sqrt(*vx**vx/(*a12)-2*(*x-131.1)/(*a1))-*vx/(*a1);
        *y=*y+*vy*t1;*z=*z+*vz*t1;
    }
    else
    {
        t1=-sqrt(*vx**vx/(*a12)-2*(*x-131.1)/(*a1))-*vx/(*a1);
        *y=*y+*vy*t1;*z=*z+*vz*t1;
    }
    return 0;
}

float ran3(idum)
long *idum;
{
    static int inext,inextp,iff=0;
    static long ma[56];
    long mj,mk;
    int i,ii,k;

    if(*idum<0||iff==0)
    {
        iff=1;
        mj=MSEED-(*idum<0 ? -*idum : *idum);
        mj %= MBIG;
        ma[55]=mj;
        mk=1;
        for(i=1;i<=54;i++)
        {
            ii=(21*i)%55;
            ma[ii]=mk;
            mk=mj-mk;
            if(mk<MZ)

```

```

        mk+=MBIG;
        mj=ma[ii];
    }
    for(k=1;k<=4;k++)
        for(i=1;i<=55;i++)
            {
                ma[i]-=ma[1+(i+30) % 55];
                if(ma[i]<MZ)
                    ma[i]+=MBIG;
            }
        inext=0;
        inextp=31;
        *idum=1;
    }
    if(++inext==56)
        inext=1;
    if(++inextp==56)
        inextp=1;
    mj=ma[inext]-ma[inextp];
    if(mj<MZ)
        mj+=MBIG;
    ma[inext]=mj;
    return mj*FAC;
}

```

Appendix C

VB-ASM.BAS

' VB-ASM, Version 1.20
' Copyright (c) 1994 SoftCircuits Programming
' Redistributed by Permission.

' SoftCircuits Programming
' P.O. Box 16262
' Irvine, CA 92713
' CompuServe: 72134,263

' This program may be used and distributed freely on the
' condition that it is distributed in full and unchanged, and
' that no fee is charged for such use and distribution with the
' exception or reasonable media and shipping charges.

' You may also incorporate any or all portions of this program,
' and/or include the VB-ASM DLL, as part of your own programs
' and distribute such programs without payment of royalties on
' the condition that such program do not duplicate the overall
' functionality of VB-ASM and/or any of its demo programs, and
' that you agree to the following disclaimer.

' WARNING: Accessing the low-level services of Windows, DOS and
' the ROM-BIOS using VB-ASM is an extremely powerful technique
' that, if used incorrectly, can cause possible permanent
' damage and/or loss of data. You are responsible for
' determining appropriate use of any and all files included in
' this package. SoftCircuits will not be held liable for any
' damages resulting from the use of these files.

' SOFTCIRCUITS SPECIFICALLY DISCLAIMS ALL WARRANTIES,
' INCLUDING, WITHOUT LIMITATION, ALL IMPLIED WARRANTIES OF
' MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND
' NON-INFRINGEMENT OF THIRD PARTY RIGHTS.

' UNDER NO CIRCUMSTANCES WILL SOFTCIRCUITS BE LIABLE FOR
' SPECIAL, INCIDENTAL, CONSEQUENTIAL, INDIRECT, OR ANY OTHER
' DAMAGES OR CLAIMS ARISING FROM THE USE OF THIS PRODUCT,
' INCLUDING LOSS OF PROFITS OR ANY OTHER COMMERCIAL DAMAGES,
' EVEN IF WE HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH
' DAMAGES.

' Please contact SoftCircuits Programming if you have any
' questions concerning these conditions.

'VB-ASM DLL declarations

Type VBREGS

AX As Integer
BX As Integer
CX As Integer
DX As Integer
BP As Integer
SI As Integer
DI As Integer
Flags As Integer
DS As Integer
ES As Integer

End Type

'VBREGS Flags bit values

Global Const VBFLAGS_CARRY = &h1
Global Const VBFLAGS_PARITY = &h4
Global Const VBFLAGS_AUX = &h10
Global Const VBFLAGS_ZERO = &h40
Global Const VBFLAGS_SIGN = &h80

'vbGetDriveType return values

Global Const VBDRIIVE_REMOVABLE = &h1002
Global Const VBDRIIVE_FIXED = &h1003
Global Const VBDRIIVE_REMOTE = &h1004
Global Const VBDRIIVE_CDROM = &h1005
Global Const VBDRIIVE_FLOPPY = &h1006
Global Const VBDRIIVE_RAMDISK = &h1007
Global Const VBDRIIVE_UNKNOWN = &h1010
Global Const VBDRIIVE_INVALID = &h0

Declare Function vbGetCtrlHwnd Lib "VBASM.DLL" (Ctrl As Any) As Integer
Declare Function vbGetCtrlModel Lib "VBASM.DLL" (Ctrl As Any) As Long
Declare Function vbGetCtrlName Lib "VBASM.DLL" (Ctrl As Any) As String
Declare Sub vbGetData Lib "VBASM.DLL" (ByVal Pointer As Long, Variable As Any, ByVal nCount As Integer)
Declare Function vbGetDriveType Lib "VBASM.DLL" (ByVal drive As String) As Integer
Declare Function vbGetLongPtr Lib "VBASM.DLL" (nVariable As Any) As Long
Declare Function vbHiByte Lib "VBASM.DLL" (ByVal nValue As Integer) As Integer
Declare Function vbHiWord Lib "VBASM.DLL" (ByVal nValue As Long) As Integer
Declare Function vbInp Lib "VBASM.DLL" (ByVal nPort As Integer) As Integer
Declare Function vbInpw Lib "VBASM.DLL" (ByVal nPort As Integer) As Integer
Declare Sub vbInterrupt Lib "VBASM.DLL" (ByVal IntNum As Integer, InRegs As VBREGS, OutRegs As VBREGS)
Declare Sub vbInterruptX Lib "VBASM.DLL" (ByVal IntNum As Integer, InRegs As VBREGS, OutRegs As VBREGS)
Declare Function vbLoByte Lib "VBASM.DLL" (ByVal nValue As Integer) As Integer
Declare Function vbLoWord Lib "VBASM.DLL" (ByVal nValue As Long) As Integer
Declare Function vbMakeLong Lib "VBASM.DLL" (ByVal nLoWord As Integer, ByVal nHiWord As Integer) As Long
Declare Function vbMakeWord Lib "VBASM.DLL" (ByVal nLoByte As Integer, ByVal nHiByte As Integer) As Integer

```

Declare Sub vbOut Lib "VBASM.DLL" (ByVal nPort As Integer, ByVal nData As Integer)
Declare Sub vbOutw Lib "VBASM.DLL" (ByVal nPort As Integer, ByVal nData As Integer)
Declare Function vbPeek Lib "VBASM.DLL" (ByVal nSegment As Integer, ByVal nOffset As Integer) As Integer
Declare Function vbPeekw Lib "VBASM.DLL" (ByVal nSegment As Integer, ByVal nOffset As Integer) As Integer
Declare Sub vbPoke Lib "VBASM.DLL" (ByVal nSegment As Integer, ByVal nOffset As Integer, ByVal nValue As Integer)
Declare Sub vbPokew Lib "VBASM.DLL" (ByVal nSegment As Integer, ByVal nOffset As Integer, ByVal nValue As Integer)
Declare Function vbRealModeIntX Lib "VBASM.DLL" (ByVal IntNum As Integer, InRegs As VBREGS, OutRegs As VBREGS) As Integer
Declare Function vbRecreateCtrl Lib "VBASM.DLL" (Ctrl As Any) As Integer
Declare Function vbSAdd Lib "VBASM.DLL" (Variable As String) As Integer
Declare Sub vbSetData Lib "VBASM.DLL" (ByVal Pointer As Long, Variable As Any, ByVal nCount As Integer)
Declare Function vbShiftLeft Lib "VBASM.DLL" (ByVal nValue As Integer, ByVal nBits As Integer) As Integer
Declare Function vbShiftRight Lib "VBASM.DLL" (ByVal nValue As Integer, ByVal nBits As Integer) As Integer
Declare Function vbSSeg Lib "VBASM.DLL" (Variable As String) As Integer
Declare Function vbVarPtr Lib "VBASM.DLL" (Variable As Any) As Integer
Declare Function vbVarSeg Lib "VBASM.DLL" (Variable As Any) As Integer

```

CRUNCHER.BAS

Option Explicit

Global COUNTS() As Long
Global DATUM(32000) As Integer

Global COIN_0(64, 64) As Long
Global COIN_1(64, 64) As Long
Global COIN_2(64, 64) As Long
Global COIN_3(64, 64) As Long

Global COUNTERS As Integer
Global MCS_1234 As Integer
Global MCS_56 As Integer
Global SECONDS As Integer
Global CHANNELS As Integer
Global SCANS As Integer
Global COINS As Integer

Global m As Integer
Global z(64, 64) As Integer
Global a As Integer
Global s As Integer
Global d(64, 64) As Integer
Global i As Integer
Global l As Integer
Global u(64, 64) As Integer
Global v As Long
Global h As Integer
Global o(64, 64) As Integer
Global c As Long
Global k As Integer
Global e As Long
Global y As Integer
Global j As Integer
Global x As Integer
Global r As Integer
Global w As Integer
Global p As Integer
Global q As Integer

Global mcs1_tot As Long
Global mcs2_tot As Long
Global mcs3_tot As Long
Global mcs4_tot As Long
Global mcs5_tot As Long
Global mcs6_tot As Long

Global coin0_tot As Long
Global coin1_tot As Long
Global coin2_tot As Long
Global coin3_tot As Long

Global START
Global FINISH
Global TOTAL

Global STATUS As Integer
Global EVENT As Integer

Global FILE1 As String
Global FILE2 As String

Global MSG As String
Global TITLE As String
Global ANSWER As Integer

CRUNCHER.FRM

Option Explicit

Sub cmdBegin_Click ()

On Error GoTo AWSHUCKS

' sets up the grid

```
grdDisplay.ColWidth(0) = 1500
grdDisplay.Row = 0
grdDisplay.Col = 0
grdDisplay.FixedAlignment(0) = 2
grdDisplay.Text = "MCS"
```

For i = 1 To 6

```
grdDisplay.Row = 0
grdDisplay.Col = i
grdDisplay.Text = i
grdDisplay.ColWidth(i) = 750
grdDisplay.ColAlignment(i) = 1
grdDisplay.FixedAlignment(i) = 2
```

Next i

For i = 1 To 256

```
grdDisplay.Row = i
grdDisplay.Col = 0
grdDisplay.Text = "CHANNEL " & i
```

Next i

' clears and resets the MCS displays

```
frmMCS.lblMTotals1.Caption = ""
frmMCS.picDisplay1.Cls
frmMCS.picDisplay1.ScaleTop = 0
frmMCS.lblMCS1max.Caption = 0
```

```
frmMCS.lblMTotals2.Caption = ""
frmMCS.picDisplay2.Cls
frmMCS.picDisplay2.ScaleTop = 0
frmMCS.lblMCS2max.Caption = 0
```

```
frmMCS.lblMTotals3.Caption = ""
frmMCS.picDisplay3.Cls
frmMCS.picDisplay3.ScaleTop = 0
```

```

frmMCS.lblMCS3max.Caption = 0

frmMCS.lblMTotals4.Caption = ""
frmMCS.picDisplay4.Cls
frmMCS.picDisplay4.ScaleTop = 0
frmMCS.lblMCS4max.Caption = 0

frmMCS.lblMTotals5.Caption = ""
frmMCS.picDisplay5.Cls
frmMCS.picDisplay5.ScaleTop = 0
frmMCS.lblMCS5max.Caption = 0

frmMCS.lblMTotals6.Caption = ""
frmMCS.picDisplay6.Cls
frmMCS.picDisplay6.ScaleTop = 0
frmMCS.lblMCS6max.Caption = 0

' clears and sets the MCS array

ReDim COUNTS(CHANNELS, COUNTERS)
MCS_56 = 0
MCS_1234 = COUNTERS

' clears the coincidence displays

frmCOINS.picDisplay0.Cls
frmCOINS.lblCTotals0.Caption = ""
frmCOINS.picDisplay1.Cls
frmCOINS.lblCTotals1.Caption = ""
frmCOINS.picDisplay2.Cls
frmCOINS.lblCTotals2.Caption = ""
frmCOINS.picDisplay3.Cls
frmCOINS.lblCTotals3.Caption = ""

' clears the coincidence arrays and sets the coincidence counters

For i = 0 To 63

    For j = 0 To 63

        COIN_0(i, j) = 0
        COIN_1(i, j) = 0
        COIN_2(i, j) = 0
        COIN_3(i, j) = 0

        z(i, j) = 1
        d(i, j) = 1
        u(i, j) = 1
        o(i, j) = 1

    Next j

```

Next i

' resets the MCS and COIN totals

```
mcs1_tot = 0
mcs2_tot = 0
mcs3_tot = 0
mcs4_tot = 0
mcs5_tot = 0
mcs6_tot = 0
```

```
coin0_tot = 0
coin1_tot = 0
coin2_tot = 0
coin3_tot = 0
```

cmdReenterStop.Caption = "S&top"

For s = 1 To SCANS ' scan loop

```
Call vbout(421, 0) ' select the data direction reg
Call vbout(420, 255) ' make all the I/O lines outputs
Call vbout(421, 4) ' select the output reg
```

```
Call vbout(423, 0) ' select the data direction reg
Call vbout(422, 255) ' make all the I/O lines outputs
Call vbout(423, 4) ' select the output reg
```

```
Call vbout(420, 0) ' write zero channel # to output
```

frmMCS.lblScanning.Caption = s

cmdSave.Enabled = True ' begin Save "window"

' clears the grid

For i = 1 To 256

grdDisplay.Row = i

For j = 1 To 6

```
grdDisplay.Col = j
grdDisplay.Text = ""
```

Next j

Next i

```
Call vbout(422, 1) ' write to output for strobe on
Call vbout(422, 0) ' write to output for strobe off
```

' the main program

If COUNTERS > 4 Then

MCS_56 = COUNTERS + 1
MCS_1234 = 4

Call vbout(419, 255) ' master reset
Call vbout(419, 23) ' select master mode w/ control
Call vbout(418, 0) ' write to master mode (part 1)
Call vbout(418, 80) ' write to master mode (part 2)

For i = 1 To (MCS_56 - 5)

Call vbout(419, i) ' select counter mode reg
Call vbout(418, 41) ' write to mode reg (part 1)
Call vbout(418, i) ' change to i for source

Next i

End If

Call vbout(417, 255) ' master reset
Call vbout(417, 23) ' select master mode w/ control
Call vbout(416, 0) ' write to master mode (part 1)
Call vbout(416, 80) ' write to master mode (part 2)

For i = 1 To MCS_1234

Call vbout(417, i) ' select counter mode reg
Call vbout(416, 41) ' write to mode reg (part 1)
Call vbout(416, i) ' change to i for source

Next i

Call vboutw(578, 0) ' initializes CFG2 reg
Call vboutw(580, 0) ' initializes CFG3 reg
Call vboutw(576, 1696) ' initializes CFG1 reg

Call vbout(417, 127) ' load and arm all counters
Call vbout(419, 127) ' load and arm all counters

For i = 1 To CHANNELS ' channel loop

Call vbout(420, i) ' write channel # to output

frmMCS.lblCurrent.Caption = i

r = 1 ' resets data counter

TOTAL = 0 ' resets timer

```

START = Timer

Do While TOTAL < SECONDS      ' timer loop

    FINISH = Timer

    STATUS = 1 And vbinp(577)  ' checks for data

    If STATUS = 1 Then

        DATUM(r) = vbinpw(582) ' reads data

        r = r + 1              ' data counter

    End If

    If FINISH < START Then      ' timer fix

        FINISH = FINISH + 86400

    End If

    TOTAL = FINISH - START      ' computes total time elapsed

Loop

Call vbout(417, 159)           ' disarm and save all counters
Call vbout(419, 159)           ' disarm and save all counters

Call vbout(422, 1)             ' write to output for strobe on
Call vbout(422, 0)             ' write to output for strobe off

' MCS Analysis (for MCS 1, 2, 3, 4)

For j = 17 To (MCS_1234 + 16)

    DoEvents                    ' allows interaction

    cmdSave.Enabled = False     ' end Save "window"

    Call vbout(417, j)          ' select hold reg

    c = vbinp(416)              ' read hold reg for low byte
    e = vbinp(416)              ' read hold reg for high byte
    v = e * 256 + c             ' organize the bytes

    COUNTS(i, j - 16) = COUNTS(i, j - 16) + v      ' fill array

    grdDisplay.Row = i
    grdDisplay.Col = j - 16
    grdDisplay.Text = COUNTS(i, j - 16)            ' fill grid

```

Select Case j

Case 17 ' MCS 1

mcs1_tot = mcs1_tot + v

frmMCS.lblMTotals1.Caption = mcs1_tot

frmMCS.lblMCS1max.Caption = frmMCS.picDisplay1.ScaleTop

If COUNTS(i, j - 16) > frmMCS.picDisplay1.ScaleTop Then ' auto-scaling

frmMCS.picDisplay1.Cls

frmMCS.picDisplay1.ScaleTop = COUNTS(i, j - 16)

frmMCS.picDisplay1.ScaleHeight = -COUNTS(i, j - 16)

frmMCS.lblMCS1max.Caption = frmMCS.picDisplay1.ScaleTop

For k = 1 To i

frmMCS.picDisplay1.PSet (k, COUNTS(k, j - 16)), QBColor(15)

Next k

End If

frmMCS.picDisplay1.PSet (i, COUNTS(i, j - 16)), QBColor(15) ' plots counts

Case 18 ' MCS 2

mcs2_tot = mcs2_tot + v

frmMCS.lblMTotals2.Caption = mcs2_tot

frmMCS.lblMCS2max.Caption = frmMCS.picDisplay2.ScaleTop

If COUNTS(i, j - 16) > frmMCS.picDisplay2.ScaleTop Then ' auto-scaling

frmMCS.picDisplay2.Cls

frmMCS.picDisplay2.ScaleTop = COUNTS(i, j - 16)

frmMCS.picDisplay2.ScaleHeight = -COUNTS(i, j - 16)

frmMCS.lblMCS2max.Caption = frmMCS.picDisplay2.ScaleTop

For k = 1 To i

frmMCS.picDisplay2.PSet (k, COUNTS(k, j - 16)), QBColor(14)

Next k

```

End If

frmMCS.picDisplay2.PSet (i, COUNTS(i, j - 16)), QBColor(14) ' plots counts

Case 19                                ' MCS 3

mcs3_tot = mcs3_tot + v

frmMCS.lblMTotals3.Caption = mcs3_tot

frmMCS.lblMCS3max.Caption = frmMCS.picDisplay3.ScaleTop

If COUNTS(i, j - 16) > frmMCS.picDisplay3.ScaleTop Then ' auto-scaling

    frmMCS.picDisplay3.Cls

    frmMCS.picDisplay3.ScaleTop = COUNTS(i, j - 16)
    frmMCS.picDisplay3.ScaleHeight = -COUNTS(i, j - 16)

    frmMCS.lblMCS3max.Caption = frmMCS.picDisplay3.ScaleTop

    For k = 1 To i

        frmMCS.picDisplay3.PSet (k, COUNTS(k, j - 16)), QBColor(13)

    Next k

End If

frmMCS.picDisplay3.PSet (i, COUNTS(i, j - 16)), QBColor(13) ' plots counts

Case 20                                ' MCS 4

mcs4_tot = mcs4_tot + v

frmMCS.lblMTotals4.Caption = mcs4_tot

frmMCS.lblMCS4max.Caption = frmMCS.picDisplay4.ScaleTop

If COUNTS(i, j - 16) > frmMCS.picDisplay4.ScaleTop Then ' auto-scaling

    frmMCS.picDisplay4.Cls

    frmMCS.picDisplay4.ScaleTop = COUNTS(i, j - 16)
    frmMCS.picDisplay4.ScaleHeight = -COUNTS(i, j - 16)

    frmMCS.lblMCS4max.Caption = frmMCS.picDisplay4.ScaleTop

    For k = 1 To i

        frmMCS.picDisplay4.PSet (k, COUNTS(k, j - 16)), QBColor(12)

```

```

        Next k

    End If

    frmMCS.picDisplay4.PSet (i, COUNTS(i, j - 16)), QBColor(12) ' plots counts

End Select

Next j

' MCS Analysis (for MCS 5, 6)

For j = 17 To ((MCS_56 - 5) + 16)

    DoEvents          ' allows interaction

    Call vbout(419, j)    ' select hold reg

    c = vbinp(418)        ' read hold reg for low byte
    e = vbinp(418)        ' read hold reg for high byte
    v = e * 256 + c       ' organize the bytes

    COUNTS(i, j - 12) = COUNTS(i, j - 12) + v    ' fill array

    grdDisplay.Col = j - 12
    grdDisplay.Text = COUNTS(i, j - 12)          ' fill grid

    Select Case j

        Case 17          ' MCS 5

            mcs5_tot = mcs5_tot + v

            frmMCS.lblMTotals5.Caption = mcs5_tot

            frmMCS.lblMCS5max.Caption = frmMCS.picDisplay5.ScaleTop

            If COUNTS(i, j - 12) > frmMCS.picDisplay5.ScaleTop Then    ' auto-scaling

                frmMCS.picDisplay5.Cls

                frmMCS.picDisplay5.ScaleTop = COUNTS(i, j - 12)
                frmMCS.picDisplay5.ScaleHeight = -COUNTS(i, j - 12)

                frmMCS.lblMCS5max.Caption = frmMCS.picDisplay5.ScaleTop

            For k = 1 To i

                frmMCS.picDisplay5.PSet (k, COUNTS(k, j - 12)), QBColor(11)

            Next k

```

```

End If

frmMCS.picDisplay5.PSet (i, COUNTS(i, j - 12)), QBColor(11) ' plots counts

Case 18                                ' MCS 6

mcs6_tot = mcs6_tot + v

frmMCS.lblMTotals6.Caption = mcs6_tot

frmMCS.lblMCS6max.Caption = frmMCS.picDisplay6.ScaleTop

If COUNTS(i, j - 12) > frmMCS.picDisplay6.ScaleTop Then    ' auto-scaling

    frmMCS.picDisplay6.Cls

    frmMCS.picDisplay6.ScaleTop = COUNTS(i, j - 12)
    frmMCS.picDisplay6.ScaleHeight = -COUNTS(i, j - 12)

    frmMCS.lblMCS6max.Caption = frmMCS.picDisplay6.ScaleTop

    For k = 1 To i

        frmMCS.picDisplay6.PSet (k, COUNTS(k, j - 12)), QBColor(10)

    Next k

End If

frmMCS.picDisplay6.PSet (i, COUNTS(i, j - 12)), QBColor(10) ' plots counts

End Select

Next j

' Coincidence Analysis

For w = 1 To (r - 1)

    DoEvents                                ' allows interaction

    EVENT = Int(DATUM(w) / 4096)

    Select Case EVENT

        Case 0                                ' COIN 0

            coin0_tot = coin0_tot + 1

            x = Int((DATUM(w) - (EVENT * 4096)) / 64)
            y = Int(DATUM(w) - (EVENT * 4096) - (x * 64))

```

```

COIN_0(x, y) = COIN_0(x, y) + 1          ' coin counter

If COIN_0(x, y) >= z(x, y) * (2 ^ m) Then  ' checks for matches

    z(x, y) = z(x, y) + 1

    If z(x, y) > 15 Then                    ' avoids color error

        frmCOINS.picDisplay0.PSet (x, y), QBColor(15)

        GoTo PHILLY

    End If

    frmCOINS.picDisplay0.PSet (x, y), QBColor(z(x, y) - 1) ' plots data

End If

Case 1                                     ' COIN 1

    coin1_tot = coin1_tot + 1

    x = Int((DATUM(w) - (EVENT * 4096)) / 64)
    y = Int(DATUM(w) - (EVENT * 4096) - (x * 64))

    COIN_1(x, y) = COIN_1(x, y) + 1        ' coin counter

    If COIN_1(x, y) >= d(x, y) * (2 ^ a) Then  ' checks for matches

        d(x, y) = d(x, y) + 1

        If d(x, y) > 15 Then                ' avoids color error

            frmCOINS.picDisplay1.PSet (x, y), QBColor(15)

            GoTo PHILLY

        End If

        frmCOINS.picDisplay1.PSet (x, y), QBColor(d(x, y) - 1) ' plots data

    End If

Case 2                                     ' COIN 2

    coin2_tot = coin2_tot + 1

    x = Int((DATUM(w) - (EVENT * 4096)) / 64)
    y = Int(DATUM(w) - (EVENT * 4096) - (x * 64))

    COIN_2(x, y) = COIN_2(x, y) + 1        ' coin counter

```

```

If COIN_2(x, y) >= u(x, y) * (2 ^ l) Then    ' checks for matches

    u(x, y) = u(x, y) + 1

    If u(x, y) > 15 Then                        ' avoids color error

        frmCOINS.picDisplay2.PSet (x, y), QBColor(15)

        GoTo PHILLY

    End If

    frmCOINS.picDisplay2.PSet (x, y), QBColor(u(x, y) - 1) ' plots data

End If

Case 4                                     ' COIN 3

    coin3_tot = coin3_tot + 1

    x = Int((DATUM(w) - (EVENT * 4096)) / 64)
    y = Int(DATUM(w) - (EVENT * 4096) - (x * 64))

    COIN_3(x, y) = COIN_3(x, y) + 1          ' coin counter

    If COIN_3(x, y) >= o(x, y) * (2 ^ h) Then    ' checks for matches

        o(x, y) = o(x, y) + 1

        If o(x, y) > 15 Then                    ' avoids color error

            frmCOINS.picDisplay3.PSet (x, y), QBColor(15)

            GoTo PHILLY

        End If

        frmCOINS.picDisplay3.PSet (x, y), QBColor(o(x, y) - 1) ' plots data

    End If

PHILLY: End Select

Next w

frmCOINS.lblCTotals0.Caption = coin0_tot
frmCOINS.lblCTotals1.Caption = coin1_tot
frmCOINS.lblCTotals2.Caption = coin2_tot
frmCOINS.lblCTotals3.Caption = coin3_tot

Call vboutw(578, 0)    ' initializes CFG2 reg
Call vboutw(580, 0)    ' initializes CFG3 reg

```

```

    Call vboutw(576, 1696)    ' initializes CFG1 reg

    Call vbout(417, 127)    ' load and arm all counters
    Call vbout(419, 127)    ' load and arm all counters

Next i

Next s

    cmdReenterStop.Caption = "&Re-enter"

    cmdSave.Enabled = True

    TITLE = "ATTENTION !!!"
    MSG = "The scans are complete.  Save your data!"

    MsgBox MSG, 48, TITLE

Exit Sub

AWSHUCKS:

    cmdReenterStop.Caption = "&Re-enter"

    TITLE = "ATTENTION !!!"
    MSG = "The previous program has terminated" & Chr(10)
    MSG = MSG & "because: " & Error

    MsgBox MSG, 16, TITLE

Exit Sub

End Sub

Sub cmdCOIN_Click ()

    frmCOINS.Show

End Sub

Sub cmdEnd_Click ()

    TITLE = "ATTENTION !!!"
    MSG = "Did you remember to save your data?"
    ANSWER = MsgBox(MSG, 292, TITLE)

    If ANSWER = 6 Then

        End

    End If

```

```

Exit Sub

End Sub

Sub cmdMCS_Click ()

    frmMCS.Show

End Sub

Sub cmdReenterStop_Click ()

    On Error GoTo MESSUP

    i = 5150      ' causes "Subscript Out of Range" error to stop program

    If cmdReenterStop.Caption = "S&top" Then

        cmdSave.Enabled = False

        GoTo STOPPED

    End If

    ' sets up the Input Boxes

    TITLE = "MCS's (1 to 6)"
    MSG = "How many MCS's?"
    COUNTERS = InputBox(MSG, TITLE)

    lblCounters.Caption = COUNTERS

    TITLE = "SECONDS"
    MSG = "For how many seconds per channel will the MCS's count?"
    SECONDS = InputBox(MSG, TITLE)

    lblSeconds.Caption = SECONDS

    TITLE = "CHANNELS (1 to 256)"
    MSG = "How many channels?"
    CHANNELS = InputBox(MSG, TITLE)

    lblChannels.Caption = CHANNELS
    frmMCS.lblChanmax.Caption = CHANNELS
    frmMCS.lblCurrent.Caption = ""

    ' scales the MCS displays

    frmMCS.picDisplay1.ScaleWidth = CHANNELS + 1
    frmMCS.picDisplay2.ScaleWidth = CHANNELS + 1
    frmMCS.picDisplay3.ScaleWidth = CHANNELS + 1
    frmMCS.picDisplay4.ScaleWidth = CHANNELS + 1

```

```
frmMCS.picDisplay5.ScaleWidth = CHANNELS + 1
frmMCS.picDisplay6.ScaleWidth = CHANNELS + 1
```

```
TITLE = "SCANS"
MSG = "How many scans?"
SCANS = InputBox(MSG, TITLE)
```

```
lblScans.Caption = SCANS
frmMCS.lblScanmax.Caption = SCANS
frmMCS.lblScanning.Caption = ""
```

```
TITLE = "COIN COLOR SCALE (1 to 100)"
MSG = "Indicate desired color scale." & Chr(10) & "(1 is most sensitive)"
MSG = MSG & Chr(10) & "Choose Cancel to keep old values."
m = InputBox(MSG, TITLE)
```

```
a = m
l = m
h = m
```

```
frmCOINS.lblSense0.Caption = m
frmCOINS.lblSense1.Caption = a
frmCOINS.lblSense2.Caption = l
frmCOINS.lblSense3.Caption = h
```

```
STOPPED: cmdReenterStop.Caption = "&Re-enter"
```

```
Exit Sub
```

```
MESSUP:
```

```
TITLE = "ATTENTION !!!"
MSG = "You either chose a wrong value or selected cancel."
```

```
MsgBox MSG, 64, TITLE
```

```
Exit Sub
```

```
End Sub
```

```
Sub cmdSave_Click ()
```

```
On Error GoTo WHOOPS
```

```
TITLE = "SAVING MCS's"
MSG = "Save as: (path and filename)"
FILE1 = InputBox(MSG, TITLE)
```

```
Open FILE1 For Output As #1
```

```
Write #1, s - 1           ' saves the number of scans
```

```

Write #1,

Write #1, mcs1_tot      ' saves the MCS 1 totals

Write #1, mcs2_tot      ' saves the MCS 2 totals

Write #1, mcs3_tot      ' saves the MCS 3 totals

Write #1, mcs4_tot      ' saves the MCS 4 totals

Write #1, mcs5_tot      ' saves the MCS 5 totals

Write #1, mcs6_tot      ' saves the MCS 6 totals

Write #1,

' saves the MCS counts

For j = 1 To MCS_1234

    For i = 1 To CHANNELS

        Write #1, COUNTS(i, j)

    Next i

    Write #1,

Next j

For k = 5 To (MCS_56 - 1)

    For i = 1 To CHANNELS

        Write #1, COUNTS(i, k)

    Next i

    Write #1,

Next k

Close #1

' saves the COIN counts

TITLE = "SAVING COIN'S"
MSG = "How many Coin's to save? (1 to 4)"
COINS = InputBox(MSG, TITLE)

If COINS >= 1 Then

```

```
MSG = "Save as: (path and filename)"  
FILE2 = InputBox(MSG, TITLE)
```

```
Open FILE2 For Output As #2
```

```
Write #2, s - 1           ' saves the number of scans
```

```
Write #2,
```

```
Write #2, coin0_tot       ' saves the COIN 0 totals
```

```
Write #2, coin1_tot       ' saves the COIN 1 totals
```

```
Write #2, coin2_tot       ' saves the COIN 2 totals
```

```
Write #2, coin3_tot       ' saves the COIN 3 totals
```

```
Write #2,
```

```
For i = 0 To 63
```

```
    For j = 0 To 63
```

```
        Write #2, COIN_0(i, j)
```

```
    Next j
```

```
Next i
```

```
Write #2,
```

```
End If
```

```
If COINS >= 2 Then
```

```
    For i = 0 To 63
```

```
        For j = 0 To 63
```

```
            Write #2, COIN_1(i, j)
```

```
        Next j
```

```
    Next i
```

```
Write #2,
```

```
End If
```

```
If COINS >= 3 Then
```

```
    For i = 0 To 63
```

```

        For j = 0 To 63
            Write #2, COIN_2(i, j)
        Next j
    Next i

    Write #2,

End If

If COINS >= 4 Then
    For i = 0 To 63
        For j = 0 To 63
            Write #2, COIN_3(i, j)
        Next j
    Next i

    Write #2,

End If

Close #2

Exit Sub

WHOOPS:

    TITLE = "ATTENTION !!!"
    MSG = "You either messed up or selected cancel."

    MsgBox MSG, 64, TITLE

    Exit Sub

End Sub

Sub Form_Load ()

    On Error GoTo MISTAKE

' clears the coincidence arrays and sets the coincidence counters

    For i = 0 To 63

```

```

For j = 0 To 63

    COIN_0(i, j) = 0
    COIN_1(i, j) = 0
    COIN_2(i, j) = 0
    COIN_3(i, j) = 0

    z(i, j) = 1
    d(i, j) = 1
    u(i, j) = 1
    o(i, j) = 1

Next j

Next i

' sets up the Input Boxes

TITLE = "MCS's (1 to 6)"
MSG = "How many MCS's?"
COUNTERS = InputBox(MSG, TITLE)

lblCounters.Caption = COUNTERS

TITLE = "SECONDS"
MSG = "For how many seconds per channel will the MCS's count?"
SECONDS = InputBox(MSG, TITLE)

lblSeconds.Caption = SECONDS

TITLE = "CHANNELS (1 to 256)"
MSG = "How many channels?"
CHANNELS = InputBox(MSG, TITLE)

lblChannels.Caption = CHANNELS
frmMCS.lblChanmax.Caption = CHANNELS
frmMCS.lblCurrent.Caption = ""

' scales the MCS displays

frmMCS.picDisplay1.ScaleWidth = CHANNELS + 1
frmMCS.picDisplay2.ScaleWidth = CHANNELS + 1
frmMCS.picDisplay3.ScaleWidth = CHANNELS + 1
frmMCS.picDisplay4.ScaleWidth = CHANNELS + 1
frmMCS.picDisplay5.ScaleWidth = CHANNELS + 1
frmMCS.picDisplay6.ScaleWidth = CHANNELS + 1

TITLE = "SCANS"
MSG = "How many scans?"
SCANS = InputBox(MSG, TITLE)

lblScans.Caption = SCANS

```

```
frmMCS.lblScanmax.Caption = SCANS
frmMCS.lblScanning.Caption = ""
```

```
ReDim COUNTS(CHANNELS, COUNTERS)      ' clears and sets the MCS array
```

```
TITLE = "COIN COLOR SCALE (1 to 100)"
MSG = "Indicate desired color scale." & Chr(10) & "(1 is most sensitive)"
m = InputBox(MSG, TITLE)
```

```
a = m
l = m
h = m
```

```
frmCOINS.lblSense0.Caption = m
frmCOINS.lblSense1.Caption = a
frmCOINS.lblSense2.Caption = l
frmCOINS.lblSense3.Caption = h
```

```
Exit Sub
```

MISTAKE:

```
TITLE = "ATTENTION !!!"
MSG = "You either chose a wrong value or selected cancel."
```

```
MsgBox MSG, 64, TITLE
```

```
Exit Sub
```

End Sub

```
Sub mnuCOIN_Click ()
```

```
TITLE = "Coincidence Display"

MSG = " Each Coincidence display represents the x and y coordinates on a "
MSG = MSG & "position sensitive detector from (0,0) to (63,63). "
MSG = MSG & "The 'totals' is the sum of all the "
MSG = MSG & "counts of all the coordinates for each display. At any time, "
MSG = MSG & "the user may click on a display and change "
MSG = MSG & "the color scale. 'Coin 0' represents the non-coincidence data."
```

```
MsgBox MSG, 64, TITLE
```

End Sub

```
Sub mnuControl_Click ()
```

```
TITLE = "Control Grid"

MSG = " The data acquisition does not start until the 'Begin' button is "
MSG = MSG & "clicked. It should be noted that when this button is clicked, "
```

```

MSG = MSG & "all arrays and displays are CLEARED and all variables are RESET. "
MSG = MSG & "Therefore, any previous data should be saved BEFORE this time."
MSG = MSG & Chr(10) & Chr(10) & " The 'Re-enter' button allows the user to "
MSG = MSG & "change the displayed input values without exiting CRUNCHER."
MSG = MSG & Chr(10) & Chr(10)
MSG = MSG & " The 'Stop' button causes a 'Subscript Out of Range' error which "
MSG = MSG & "is used to terminate program execution without exiting CRUNCHER. "
MSG = MSG & "The error itself can be ignored." & Chr(10) & Chr(10)
MSG = MSG & " The 'Save' button allows the user to save both the MCS and "
MSG = MSG & "Coincidence data. Be sure to indicate the full path when saving. "
MSG = MSG & "This button also causes an error which can be ignored, but the "
MSG = MSG & "program is stopped and must be restarted by 'Begin'."
MSG = MSG & Chr(10) & Chr(10)
MSG = MSG & " CRUNCHER saves the data sequentially and in the following order: "
MSG = MSG & "number of scans, totals, and data. The MCS data starts with the "
MSG = MSG & "first channel of the first MCS, then the next channel of the first "
MSG = MSG & "MCS until all the channels are done, before moving to the next MCS. "
MSG = MSG & "The Coincidence data starts with coordinate (0,0), then (0,1), "
MSG = MSG & "then (0,2)...until all (63,63) are done, before moving to the next "
MSG = MSG & "Coincidence." & Chr(10) & Chr(10) & " The 'Save' button can "
MSG = MSG & "ONLY be clicked after the last channel of a scan is displayed or "
MSG = MSG & "after all the scans are complete."

```

MsgBox MSG, 64, TITLE

End Sub

Sub mnuHard_Click ()

TITLE = "Hardware and Forms"

```

MSG = " The CRUNCHER data acquisition program is designed to use the "
MSG = MSG & "PC-TIO-10 and AT-DIO-32F boards made by National Instruments "
MSG = MSG & "Corporation." & Chr(10) & Chr(10)
MSG = MSG & " The PC-TIO-10 is a timing and digital I/O interface for "
MSG = MSG & "the PC. Two AMD Am9513A STC's are used for the timing interface, "
MSG = MSG & "and a Motorola MC6821 PIA is used for the digital I/O interface."
MSG = MSG & Chr(10) & Chr(10) & " The AT-DIO-32F is a high-speed 32-bit parallel "
MSG = MSG & "digital I/O interface designed for high-performance data "
MSG = MSG & "acquisition for the PC." & Chr(10) & Chr(10)
MSG = MSG & " CRUNCHER consists of three forms: Control Grid, MCS Display, "
MSG = MSG & "and Coincidence Display. These forms may be minimized during "
MSG = MSG & "execution, but should NOT be unloaded because their information "
MSG = MSG & "will be LOST!" & Chr(10) & Chr(10)
MSG = MSG & " Once one of the forms is maximized, the other forms "
MSG = MSG & "may be retrieved by clicking the appropriate button on the maximized "
MSG = MSG & "form and pressing the 'Enter' key."

```

MsgBox MSG, 64, TITLE

End Sub

```
Sub mnuMCS_Click ()
```

```
    TITLE = "MCS Display"
```

```
    MSG = "  Each MCS display is 'Counts vs. Channels' where the 'max counts' "  
    MSG = MSG & "is the current maximum count value of the display. The 'totals' "  
    MSG = MSG & "is the sum of all the counts of all the channels for each display."
```

```
    MsgBox MSG, 64, TITLE
```

```
End Sub
```

```
Sub tmrClock_Timer ()
```

```
    lblClock.Caption = Format(Now, "m/d/yy h:mm AM/PM")    ' displays the current date and time
```

```
End Sub
```

MCS.FRM

Option Explicit

Sub cmdCOIN_Click ()

 frmCOINS.Show

End Sub

Sub cmdGrid_Click ()

 frmControl.Show

End Sub

COINS.FRM

Option Explicit

Sub cmdGrid_Click ()

 frmControl.Show

End Sub

Sub cmdMCS_Click ()

 frmMCS.Show

End Sub

Sub picDisplay0_Click ()

 On Error GoTo AWSHUCKS

 TITLE = "COIN 0 COLOR SCALE (1 to 100)"

 MSG = "Indicate desired color scale." & Chr(10) & "(1 is most sensitive)"

 MSG = MSG & Chr(10) & "To only clear the display, choose the same color scale."

 m = InputBox(MSG, TITLE)

 lblSense0.Caption = m

 picDisplay0.Cls

 For p = 0 To 63

 For q = 0 To 63

 z(p, q) = 1

 Next q

 Next p

 Exit Sub

AWSHUCKS:

 TITLE = "ATTENTION !!!"

 MSG = "You either chose a wrong value or selected cancel."

 MsgBox MSG, 64, TITLE

 Exit Sub

End Sub

Sub picDisplay1_Click ()

On Error GoTo MESSUP

TITLE = "COIN 1 COLOR SCALE (1 to 100)"

MSG = "Indicate desired color scale." & Chr(10) & "(1 is most sensitive)"

MSG = MSG & Chr(10) & "To only clear the display, choose the same color scale."

a = InputBox(MSG, TITLE)

lblSense1.Caption = a

picDisplay1.Cls

For p = 0 To 63

For q = 0 To 63

d(p, q) = 1

Next q

Next p

Exit Sub

MESSUP:

TITLE = "ATTENTION !!!"

MSG = "You either chose a wrong value or selected cancel."

MsgBox MSG, 64, TITLE

Exit Sub

End Sub

Sub picDisplay2_Click ()

On Error GoTo WHOOPS

TITLE = "COIN 2 COLOR SCALE (1 to 100)"

MSG = "Indicate desired color scale." & Chr(10) & "(1 is most sensitive)"

MSG = MSG & Chr(10) & "To only clear the display, choose the same color scale."

l = InputBox(MSG, TITLE)

lblSense2.Caption = l

picDisplay2.Cls

For p = 0 To 63

```

    For q = 0 To 63

        u(p, q) = 1

    Next q

Next p

Exit Sub

WHOOPS:

    TITLE = "ATTENTION !!!"
    MSG = "You either chose a wrong value or selected cancel."

    MsgBox MSG, 64, TITLE

    Exit Sub

End Sub

Sub picDisplay3_Click ()

    On Error GoTo MISTAKE

    TITLE = "COIN 3 COLOR SCALE (1 to 100)"
    MSG = "Indicate desired color scale." & Chr(10) & "(1 is most sensitive)"
    MSG = MSG & Chr(10) & "To only clear the display, choose the same color scale."
    h = InputBox(MSG, TITLE)

    lblSense3.Caption = h

    picDisplay3.Cls

    For p = 0 To 63

        For q = 0 To 63

            o(p, q) = 1

        Next q

    Next p

    Exit Sub

MISTAKE:

    TITLE = "ATTENTION !!!"
    MSG = "You either chose a wrong value or selected cancel."

```

MsgBox MSG, 64, TITLE

Exit Sub

End Sub

Vita

Lee Barry Straus was born in Lansdale, Pennsylvania on February 9, 1970. After moving to Lilburn, Georgia in August of 1978, he graduated from Brookwood High School in May of 1988. He received his Bachelor of Science degree in Physics from Clemson University in December of 1992 and entered the Master's program at the University of Tennessee, Knoxville in August of 1993. During his graduate career, Lee has received authorship on two journal articles:

"Recoil-ion charge-state resolved electron production cross sections at 55° for 1 MeV/u C^{5+} on He and Ar", F. Segner, M. Breinig, D.D. Desai, A. Wig, and L. Straus, Phys. Rev. A 54, 1385 (1996)

"Energy and angular distribution of binary encounter electrons in coincidence with recoil ion and projectile ion exit charge states for 0.5 MeV/u $C^{2+} + H_2$ collisions", D.D. Desai et al., Am. Phys. Soc. 41, 1651 (1996)