Experimental Studies of Sputtering on Zirconium Analyzed using Modified Roosandaal Sanders Theory

by

Paul Robert Schomber

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

University of Washington

1995

Approved by _____

(Chairperson of Supervisory Committee)

_____

_____

Program Authorized to Offer Degree _____

Date _____

Doctoral Dissertation

In presenting this dissertation in partial fulfillment of the requirements for the Doctoral degree at the University of Washington, I agree that the Library shall make its copies freely available for inspection. I further agree that extensive copying of this dissertation is allowable only for scholarly purposes, consistent with "fair use" as prescribed in the U.S. Copyright Law. Requests for copying or reproduction of this dissertation may be referred to University Microfilms, 1490 Eisenhower Place, P.O. Box 975, Ann Arbor MI 48106, to whom the author has granted "the right to reproduce and sell (a)copies of the manuscript in microform and/or (b) printed copies of the manuscript made from microform."

Signature _____

Date _____

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE May 95	3. REPORT TYPE AND DATES COVERED

4. TITLE AND SUBTITLE
Experimental studies of sputtering on Zirconium Analyzed using modified Roosendaal Sanders theory

5. FUNDING NUMBERS

6. AUTHOR(S)
Paul P. Schamber

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
AFIT Students Attending:

Univ of Washington

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/CI/CIA

95-003D

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
DEPARTNEMT OF THE AIR FORCE
AFIT/CI
2950 P STREET, BDLG 125
WRIGHT-PATTERSON AFB OH 45433-7765

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for Public Release IAW AFR 190-1
Distribution Unlimited
BRIAN D. GAUTHIER, MSgt, USAF
Chief Administration

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

DTIC QUALITY INSPECTED 5

14. SUBJECT TERMS			15. NUMBER OF PAGES 287
			16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT

University of Washington

Abstract

Experimental Studies of Sputtering on Zirconium Analyzed using
Modified Roosandaal Sanders Theory

By Paul Robert Schomber

Chairperson of The Supervisory Committee: Professor Robert O. Watts
Department of Chemistry

An ion optics system utilizing a wein filter velocity selector has been
modeled and characterized for use as an ion source for an instrument to
measure high resolution angular distributions of sputtered neutral atoms.
Laser induced fluorescence detection techniques are used to measure ground
state and first excited state sputtering angular distributions on a
polycrystalline zirconium foil using argon and nitrogen sputter gases. The
incident ion beam impact angle has been varied from 15 ° to 75° as measured
from surface normal and the wein filter velocity selector has been used to
select N_2^+ and N^+ ion beams from the nitrogen ion beam.

The experimental data gathered are compared to Roosandaal Sanders
analytical sputtering theory along with data on xenon and neon. Roosandaal
Sanders theory reproduces the near surface normal sputtering behavior but
rapidly breaks down as the incident ion beam impact angle moves toward the
surface. Modifications to the Roosandaal Sanders equation to introduce

adjustable fitting parameters and non-linear least squares fitting of the experimental data to these parameters has been accomplished. The results are discussed relating the fitting parameters to physical constants based in Roosandaal Sanders Theory. Discrepancies in the theory are addressed with extensive discussion on ion surface interaction.

Accesion For	
NTIS CRA&I	☑
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution /	
Availability Codes	
Dist	Avail and / or Special
A-1	

TABLE OF CONTENTS

ii

List of Figures

List of Tables

ACKNOWLEDGMENTS

To my parents and my wife, Beth who always had faith in me

Chapter 1. Introduction and Theoretical Models of Sputtering

Sputtering is described as the "removal of surface atoms due to energetic particle bombardment."[1] Sputter erosion is not the only observable effect of particle bombardment nor can all erosion caused by particle bombardment be classified as sputtering. Sigmund[2] states four criteria for an event to be classified as sputtering:

1. *Sputtering is a class of erosion phenomena observed on a material surface as a consequence of (external or internal) particle bombardment;*
2. *Sputtering is observable in the limit of small incident-particle current;*
3. *Sputtering is observable in the limit of small incident-particle fluence;*
4. *Sputtering is observable on target materials of homogeneous composition.*

The first criterion defines the type of events that could be classified as sputtering. The second criterion distinguishes sputtering from macroscopic heating and subsequent evaporation of target atoms by high intensity ion beams. The third criterion allows a sputtering event to be caused by a single particle as opposed to blistering,[3] which requires a threshold fluence to be observed. The last criterion distinguishes sputtering from collision-induced desorption. These criteria are generally accepted as the conditions which determine if a sputtering event occurs.

The great majority of sputtering experiments to date have used noble gas ion beams as projectile sources. There are only a handful of papers that report the use of diatomic sputter gases, and these measure the total sputter yield as opposed to angular distributions.[4,5] Previous workers[6,7] in this laboratory have used laser-induced fluorescence to detect the angular distributions of zirconium atoms sputtered using argon, neon, and xenon

gases. Zirconium was chosen as the target because its fluorescence spectrum is well-characterized and easily reached using a standard dye laser. We wish to expand this work to diatomic gas species and negative halide ion sources in order to compare the excited state angular distributions with the ground state distributions. The analytical theories of sputtering are built around the theory of elastic collisions, yet there are a significant fraction of atoms sputtered in excited states, providing evidence of inelasticity. If the electron on a negatively charged ion is stripped away from the incoming ion upon hitting the target surface, the energy of this electron will be localized in the surface layers and the excited state yield detected should be greater than that observed from a positively charged ion.

History and Early Theory

Sputtering was first observed experimentally by Grove[8] in 1853 as the build up of a metallic deposit on the glass walls of a discharge tube. The mechanism for this deposit was first postulated as the heating and subsequent evaporation of the cathode in the discharge. This was disproved experimentally with the observation that the sputtering rate did not depend on the temperature as long as the cathode was well below its melting point. Fifty years later, Goldstein[9] proved conclusively that the deposit was caused by positive ions in the discharge hitting the cathode. Since that date, several theoretical and experimental investigations have examined the sputter process in an attempt to elucidate the various mechanisms responsible.

Stark[10] is recognized as the first to propose a model of an individual sputtering event occurring on an atomic scale. His 'hot spot' model treated

the sputtering event as evaporation of target atoms from a microscopically small high temperature region instigated by individual ion bombardment. Subsequently, he developed a collision theory model viewing the sputtering event as a series of binary collisions initiated by one ion at a time.[11] In his collision theory, Stark applied the conservation laws of elastic scattering along with the ideas of collisional cross sections to interpret the observed energy dependence of the sputtering yield, Y, on hydrogen ions bombarding metal targets. Y was observed to increase with increasing energy at low incident ion energies. The behavior changed at higher energies resulting in a plateau in the yield curve. Stark ascribed the low energy behavior to increasing transfer of energy between the incident ions and the target surface atoms. At higher ion energies, the incident ion penetrated deeper into the target as a function of increasing energy (smaller cross section) resulting in a smaller sputtering effect. This caused the decreasing sputter yield and subsequent plateau observed in experiment. In the 1950's, the use of accelerators became common and the sputtering yield was proven to *always decrease* at high enough energies, lending support to this description.

Stark considered his hot-spot model and his collision theory of sputtering as two different views of one and the same process. Other investigators viewed them as contradictory models and in 1923 Kingdon and Langmuir[12] used Stark's collisional theory to describe the ion-induced desorption of monolayers. Unfortunately, this led to the erroneous impression that collision theory implied that sputtering was a single-collision process leading to a strongly peaked angular distribution of sputtered particles. This impression, along with the experimental observation of the angular distribution following the Knudson cosine law,[13] was considered

unambiguous evidence for rejection of the collisional theory of sputtering. It took Wehner's[14] observations of crystal structure effects in the angular distributions from single crystals both to prove that local evaporation alone could not explain the sputtering phenomena and to reinvigorate the interest in a collisional theory of sputtering.

With experimental evidence to support both theories of sputtering, Lamar and Compton[15] pointed out the predominance of binary collision processes in light ion sputtering and the predominance of local evaporation processes in heavy ion sputtering. This led with a little rephrasing to the modern qualitative classifications of the sputtering process. Elastic collisions, so-called knock-on sputtering, are the basis for the current models for the sputtering event.

Elastic collisions are most important for understanding sputtering from metallic targets. A 10 keV Argon ion has one thousandth the velocity of the speed of light. It takes approximately 10^{-13} seconds for this ion to travel 100 Å in vacuum. This is much longer than the relaxation times of conduction electrons, about 10^{-19} seconds. Therefore, in inelastic collisions where some or most of the energy of the incoming ion is used to excite the electrons in the target atoms, the energy transferred to the conduction electrons will immediately be shared and dissipated by the other electrons, preventing any atoms from escaping.

Three Regimes of Sputtering

Examination of the behavior of sputtering events has led to the qualitative separation into three sputtering regimes: 1) the single knock-on regime, 2) the linear cascade regime, and 3) the spike regime (Figure 1). The single knock-on regime (Figure 1a) is characteristic of low energy, low fluence events. In this regime, the incident ion transfers energy to target atoms and after a small number of collisions, a few surface atoms are ejected if they receive enough energy to overcome surface binding forces. In the linear cascade regime (Figure 1b) and the spike regime (Figure 1c), the recoil atoms have enough energy to initiate second generation and higher recoils. These secondary collision 'cascades' can also provide energy to eject surface atoms. The difference between the linear cascade regime and the spike regime is determined by the spatial density of moving target atoms in the cascade volume. In the linear cascade regime, all the target atoms are at rest before suffering a collision with either an incident ion or a secondary recoil atom. In the spike regime, the density of collisions is so great that the atoms in the cascade volume are already moving as the cascade develops and subsequent collisions occur. The easiest way to differentiate between the linear cascade regime and the spike regime is to observe the sputtering behavior of diatomic molecules. A diatomic molecule will dissociate almost immediately after hitting the surface. If the sputtering yield is twice the yield of the individual ions then it is considered to be in the linear cascade regime. The subsequent cascades generated by the two atoms are relatively dilute and can be treated as a linear superposition of the individual ions. In the spike regime, however, the cascades are so dense that twice the energy is shared by all the atoms in the cascade volume as the individual ion bombardment. Depending on how

this energy is distributed, sputtering yields can be observed that will be substantially higher than twice the yield of the individual ions. All the theories and experiments summarized in this thesis will deal with the linear cascade regime.
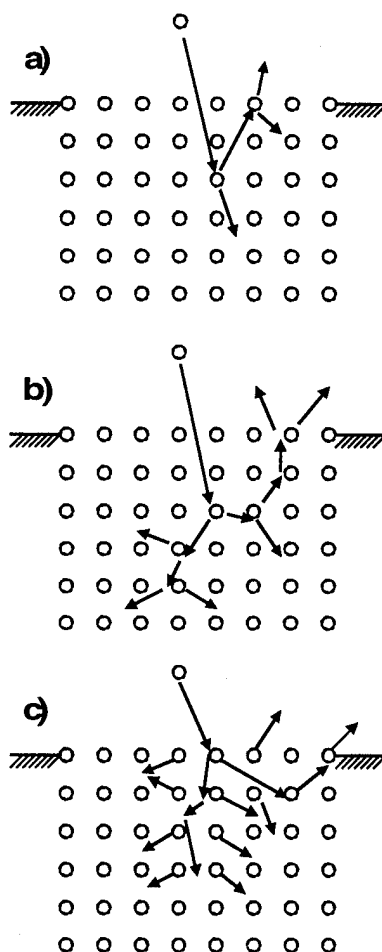
Figure 1. The three sputtering regimes: (a) The single Knock-on regime; (b) The linear cascade regime; (c) The spike regime

Collisional Theory

Since sputtering theory has borrowed much of its formalism from collisional theory, an examination of some of the common tools and analytical techniques used in the theory is useful. The approach used in this thesis will mirror approaches used by both Sigmund[16] and Li.[7] Sputtering involves high energy collisions occurring on an atomic scale; thus, classical mechanics is used to describe the behavior of the colliding particles. In elastic binary collisions, the motion of two different particles interacting through a central potential $V(r_{12})$ can be transformed into an equivalent representation involving the center of mass motion and the relative motion of the two particles. In the absence of external fields, the center of mass moves at a constant velocity and the dynamics of the collision can be determined completely by the relative motion of the two particles.[17,18] This motion is equivalent to the motion of a single particle of mass $\mu = \dfrac{M_1 M_2}{M_1 + M_2}$ moving in a coordinate system fixed on the center of mass and subject to a central force.
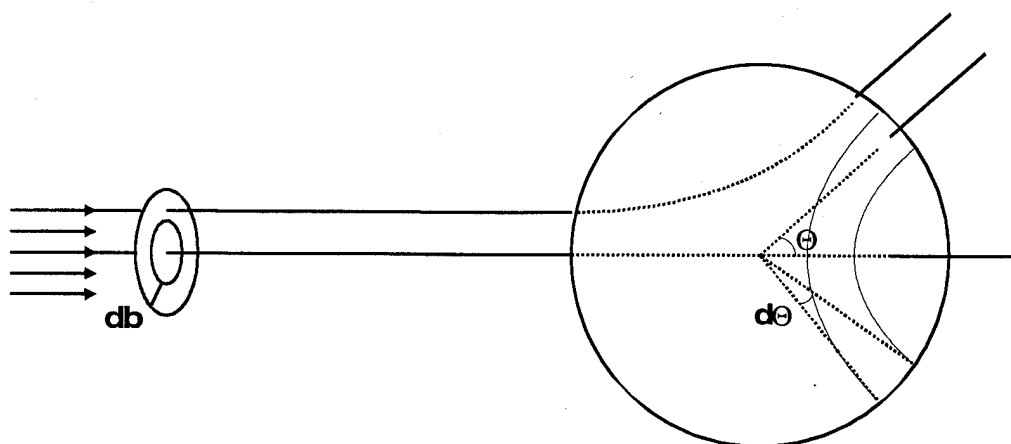
Figure 2. Scattering of particles by a central potential.

To examine the behavior of particles scattered by a central field, that is, to follow the motion of these particles, we need to use a fundamental concept of scattering theory, the cross section (Figure 2). Under an atomic collision process involving projectile and target particles, the average fraction of projectile beam particles hitting a target of thickness x, density N and experiencing a collision can be characterized by its cross section σ such that equation 1 holds.

$$(\text{fraction beam particles colliding with target}) = N \times \sigma \qquad (1)$$

Additionally, when, $N \times \sigma \ll 1$ equation 1 represents the probability that a collision will occur between beam atoms and homogeneous randomly distributed target atoms before the beam atoms penetrate a path length x into the target. Analogously, the probability of a particle being scattered into a solid angle element $d\Omega$ at Ω can be written as

$$\sigma(\Omega)d\Omega = \frac{dN(\#\ \text{of particles scattered into } d\Omega \text{ at } \Omega \text{ per unit time})}{I(\text{incident flux density})} \qquad (2)$$

$\sigma(\Omega)d\Omega$ is called the *solid angle differential scattering cross section or the differential scattering cross section*. Examining Figure 2, it is obvious that $\sigma(\Omega)$ is independent of the azimuthal angle ϕ. Therefore, a *polar angle differential scattering cross section* is defined as

$$\sigma(\Theta)d\Theta = \int \sigma(\Omega)\sin\Theta d\Theta d\phi = 2\pi\sigma(\Omega)\sin\Theta d\Theta \qquad (3)$$

The number of particles scattered into a ring $d\Theta$ centered at Θ is

$$dN = I2\pi b db = i\sigma(\Theta)d\Theta \qquad (4)$$

where b is the impact parameter, the distance of closest approach of the two particles if they cannot interact with each other.[19] Combining equations 3 and 4, the differential cross section can be related to the impact parameter by

$$\sigma(\Theta)d\Theta = 2\pi b \frac{db}{d\Theta}d\Theta \qquad (5)$$

The impact parameter and the center of mass scattering angle can be related to the potential using the conservation of energy and angular momentum to obtain the classical deflection function:[20]

$$\Theta = \pi - 2b \int_{r_{min}}^{\infty} r^{-2}\left(1 - \frac{V(r)}{E_r} - \frac{b^2}{r^2}\right)^{-\frac{1}{2}} dr \qquad (6)$$

where E_r is the relative kinetic energy of the collision and r_{min} is the classical turning point determined by calculating the largest positive root to the equation

$$1 - \frac{V(r)}{E_r} - \frac{b^2}{r^2} = 0 \qquad (7)$$

Thomas-Fermi Potential

The potential energy function used most extensively in analytical theories is the power approximation to the Thomas-Fermi potential. The Thomas-Fermi potential is based on the Thomas-Fermi theory of atoms,[21,22] where the electrons in an atom are distributed according to the Fermi-Dirac distribution function:

$$f = \left(e^{\frac{(E-\varepsilon)}{kT}} + 1\right)^{-1} \qquad (8)$$

where ε is the chemical potential, T is the temperature, and k is the Boltzmann constant. It is instructive to examine this equation more closely in the zero temperature limit. At absolute zero, all states with energy less than ε will be occupied and all states with energy above ε will be unoccupied. ε has

the properties of a cutoff energy and the Pauli exclusion principle forces the electrons to occupy all states from the ground state to the state with energy ε. The total energy of an electron is composed of the sum of its potential and kinetic energies. It can be written as

$$\varepsilon = V_1(r) + \frac{P_F^2}{2M} \tag{9}$$

where $V_1(r)$ is the electrostatic potential energy of a test charge located a distance r from the nucleus and P_F is the Fermi momentum, the maximum momentum of the electrons. Treating ε as a constant, the potential energy can be written as $V(r)=V_1(r)-\varepsilon$. The total number of electrons is equal to the integral of the density of states in wave vector space from $k = 0$ to k_f. This allows the Fermi momentum to be connected to the number density of electrons through

$$P_F(r) = \left(3\hbar^3\pi^2\rho(r)\right)^{\frac{1}{3}} \tag{10}$$

Poisson's equation can be used to relate the charge density $-e\rho$ to the electrostatic potential $-(1/e)V(r)$

$$\nabla^2 V(r) = -4\pi e^2 \rho(r) \tag{11}$$

Equations 9 - 11 can be combined to form a differential equation for the potential energy function

$$\frac{1}{r}\frac{d^2}{dr^2}(rV) = -\frac{4e^2}{3\pi\hbar^2}(2M)^{3/2}(-V)^{3/2} \tag{12}$$

Upon making the following change of variables,

$$r = \left[\frac{(3\pi)^{2/3}}{2^{7/3}}\frac{\hbar}{Me^2}Z^{-1/3}\right]X = 0.885a_0Z^{-1/3}X = bX \tag{13}$$

$$rV = -Ze^2\Phi \tag{14}$$

the so-called Thomas-Fermi equation results

$$\frac{d^2\Phi}{dX^2} = \frac{1}{\sqrt{X}}\Phi^{2/3} \tag{15}$$

with the boundary conditions

$$\Phi(0) = 1, \Phi(\infty) = \Phi'(\infty) = 0 \tag{16}$$

Equation 15 cannot be solved analytically but can be solved numerically. Using equation 14, the Thomas-Fermi potential for an isolated atom can be determined:

$$V(r) = -\frac{Ze^2}{r}\Phi\left(\frac{r}{b}\right) \tag{17}$$

Equation 17 has the form of a coulomb potential multiplied by a screening function Φ. The function Φ represents the screening of the test charge from the nucleus by the other electrons in the atom with b representing the screening radius. A tabulation of the numerical solution to $\Phi\left(\dfrac{r}{b}\right)$ can be found in Torren's book.[23] A plot of $\Phi\left(\dfrac{r}{b}\right)$ along with various power law approximations to the function is included as Figure 3.
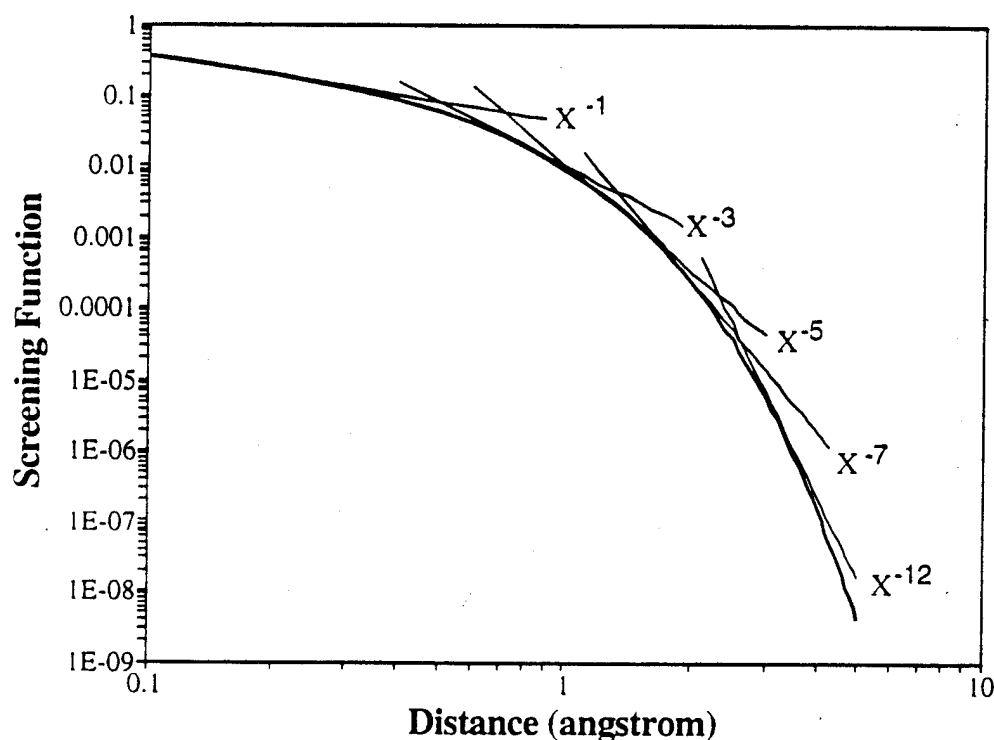
Figure 3: **The Thomas Fermi screening function, $\Phi(X)$, and the power law approximations. $X=r/a$ where a is the screening radius. Each straight line is a power approximation to $\Phi(X)$ and is valid only for a short distance.**

The potential energy function calculated above works for the case of an atom and an isolated electron. In sputtering, we are interested in the potential between two atoms. Firsov[23] applied Thomas-Fermi theory to diatomic systems and found that if the screening radius is written as

$$a = 0.8853a_0 \left(Z_1^{1/2} + Z_2^{1/2} \right)^{-2/3} \tag{18}$$

the potential can be approximated to first order by the Thomas-Fermi equation (equation 15). Therefore, for a diatomic system the Thomas-Fermi potential can be written as

$$V(r) = \frac{Z_1 Z_2 e^2}{r} \Phi\left(\frac{r}{a}\right) \tag{19}$$

where r is now the interatomic distance. This equation must also be solved numerically. The one caveat to this approach is that the potential is only valid for short interatomic separations, r < 1.0 Å.

Analytical Approximations to the Thomas-Fermi Potential

Because the Thomas-Fermi potential is very cumbersome to solve numerically for most applications, several analytical approximations for the screening function have been developed. Some of the more common ones include the Sommerfield approximation, the Bohr screening function, the

Moliere screening function, the Lenz-Jensen screening function, the Krypton-Carbon(KR-C) screening function, and various power law approximations. A full description of these functions are given elsewhere.[7,23]

All analytical theories of sputtering to date use a power law approximation to the Thomas-Fermi potential. In this potential, the interatomic separation of the two atoms are divided into several segments and each segment is approximated by a power function of (r/a);

$$\Phi\left(\frac{r}{a}\right) = \frac{k_s}{s}\left(\frac{r}{a}\right)^{-(s-1)}$$

(20)

where k_s and s are constants that depend on the interatomic distance r. This is the potential plotted in Figure 3 along with the numerical solution to the Thomas Fermi potential.

Using the power law potential in equation 6, the integration can be solved exactly[24] to give

$$\Theta = \frac{\gamma_s k_s}{\varepsilon}\left(\frac{a}{b}\right)^s$$

(21)

where $\gamma_s = \frac{1}{2} B\left(\frac{1}{2}, \frac{s+1}{2}\right) \cong \frac{1}{s}\sqrt{\frac{3s-1}{2}}$ with B(m,n) corresponding to the Beta function,[25] $\varepsilon = \frac{aE_r}{Z_1 Z_2 e^2}$ and $a = 0.8853 a_0 \left(Z_1^{2/3} + Z_2^{2/3}\right)^{-1/2}$ (a is the Bohr radius $= 0.529\text{Å}$). Equation 21 can then be inserted into equation 5 to give

$$d\sigma(\Theta) = 2\pi b \frac{db}{d\Theta} = \frac{const \cdot a^2}{\varepsilon^{2/s} \Theta^{1+2/s}} \tag{22}$$

Kinetic Energy Transfer to Target

In sputtering, we are interested in the transfer of energy from bombarding particle to target. In other words, we want to calculate the *energy loss cross section*. This can be calculated from the angular cross section once a relationship between the transferred energy and the center of mass scattering angle is established. In the special case of a moving atom (1) colliding with a stationary atom (2), it can be shown[26] that the maximum energy transferred from atom 1 to atom 2 is given by

$$T_m = \frac{4 M_1 M_2}{(M_1 + M_2)^2} E = \gamma E \tag{23}$$

after undergoing a head-on collision. The transferred kinetic energy T and the center of mass scattering angle can be related through this energy as

$$T = T_m \, sin^2 \left(\frac{\Theta}{2} \right) \tag{24}$$

When $T \ll T_m$, this equation points to Θ being small, and T can be approximated by

$$T \cong T_m \frac{\Theta^2}{4} \qquad (25)$$

With this relationship and the angular cross section in equation 22, the energy loss cross section can be easily calculated.

$$d\sigma(T)dT = d\sigma(\Theta(T)) \cdot \left(\frac{d\Theta}{dT}\right)dT = \frac{C_m}{E_m T^{m+1}}dT \qquad (26)$$

where m=1/s and C_m is given[27] by

$$C_m = \frac{\pi}{2}\lambda a^2 \left(\frac{M_1}{M_2}\right)^m \left(\frac{2Z_1 Z_2 e}{a}\right)^{2m} \qquad (27)$$

λ_m is a dimensionless function of the parameter m that varies from high energies; s = 1, m = 1, λ = 1/2, to very low energies; s = ∞, m $\approx$ 0, λ = 24. Several values of λ_m are collected in Table 1.

Table 1. Values of λ_m used in Equation 27.

λ_m	0.500	0.327	1.309	2.92	15	24
m	1.000	0.500	0.333	0.191	0.055	0.000

Equation 26 was derived in the limit of small angle scattering corresponding to soft collisions. Using the power law potential, it has been shown to be remarkably accurate even up to high energies.

Energy Loss Cross Section

Two additional quantities needed to develop the theory of the linear cascade are the *nuclear stopping cross section* and the *nuclear stopping power*. The mean energy dE lost to collisions over a path length dx is

$$dE = Ndx \int d\sigma(T)TdT \equiv NdxS_n(E) \tag{28}$$

where $S_n(E)$ is defined identically as the nuclear stopping cross section. Examining equation 28 and dividing both sides by dx gives the nuclear stopping power $\dfrac{dE}{dx} = NS_n(E)$. Combining this relation with equation 26 gives the nuclear stopping power calculated using a power law potential:

$$\frac{dE}{dx} = N \int_0^{\gamma E} Td\sigma(T)dT = N \int_0^{\gamma E} T \frac{C_m}{E^m T^{m+1}} dT = \frac{N\gamma^{1-m}C_m}{1-m} E^{1-2m} \tag{29}$$

With the calculation of the energy loss cross section, we have the tools necessary to treat the phenomena observed in the linear cascade regime. The theoretical approach used to investigate the linear cascade regime starts with the separation of the sputtering event into two phenomena. First is the creation of the primary recoil atoms by the incident ion. Next, the secondary

collision cascades develop until some of the atoms are ejected through the surface. This is the approach followed by Almen and Bruce,[28] Thompson,[29,30] and Sigmund.[31] Thompson's derivation will be followed in this thesis.

Thompson's Theory of Sputtering

Thompson based his theory on atomic scattering theory and results from radiation damage studies. He made four initial assumptions: (1) the target was completely amorphous with randomly distributed atoms; (2) the target was infinite for the purpose of developing collision cascades and semi-infinite with respect to atom ejection; (3) only elastic binary collisions occur; and (4) the momentum distribution of high generation recoil atoms is isotropic. With these assumptions, Thompson was able to derive a theory that produces the explicit angular and energy distributions of sputtered atoms.

Thompson started with the idea of an ion source supplying a flux, Ψ ions per second, of energy E_1 to an infinite imaginary surface in an infinite target. The density of primary recoils created per unit time with recoil energy in the range dt at T can be expressed as

$$\dot{\rho}(E_1, T)dT = N\Psi d\sigma(E_1, T)dT \tag{30}$$

where N is the number density of target atoms. The above equation assumes that the ions pass through the surface only once. This assumption is valid for target ion interactions where the mass of the ion is much large that the target,

$M_1 >> M_2$. In light ion sputtering of heavy target atoms, back reflection of ions can contribute significantly to the sputtering yield and equation 28 must then be integrated over ion energy and direction.

Every primary recoil will generate secondary collisions and subsequent cascades if the energy is high enough. Neglecting the initial surge of ions produced when starting the ion beam and assuming a stable ion beam is used, the ion source will rapidly form a stationary distribution of moving atoms initiated by primary recoils. In other words, there will be a constant number of atoms with a kinetic energy within a specified range at any time. The mean number of recoil atoms with energy in the interval (E', dE') initiated by primary recoils of energy T is defined as n(T,E')dE'. The total number of these atoms per unit time and distance can be calculated from

$$g(E_1, E')dE' = dE' \int_{E'}^{\gamma E_1} n(T, E')\rho(E_1, T)dT \tag{31}$$

where γE_1 is the maximum energy that can be transferred in a collision and $g(E_1, E')$ represents the number of atoms generated in a unit energy interval around E' per unit time and distance. The quantity of interest to us is G(E,E'), the total number of atoms in the energy interval E' at *any* time regardless of when they were generated. This number should be proportional to the lifetime, τ, of the atom in energy state E'. τ can be defined as

$$\tau = \frac{1}{dE'/dt} = \frac{1}{v'\,dE'/dx} \tag{32}$$

where v′ is the velocity of a target atom with energy E′. Multiplying g(E,E′) by τ gives the quantity we were looking for:

$$G(E_1, E') = g(E_1, E')\tau dE' = \frac{dE'}{v'\,dE'/dx} \int_{E'}^{\gamma E_1} n(T, E')\dot{\rho}(E_1, T)\,dT \qquad (33)$$

Examining equation 33, we now need expressions for both n(T,E′) and dE′/dx. The radiation damage function which calculates the number of atoms displaced by one primary recoil is similar to n(T, E′) and can be written as

$$n(T, E') = \Gamma_m \frac{T}{E'} \qquad (34)$$

The constant Γ_m varies with the potential similarly to the constant λ_m earlier and is compiled in Table 2.

Table 2. Values of Γ_m.

m	0.500	0.333	0.250	0.00
Γ_m	0.361	0.452	0.491	0.608

The quantity dE′/dx can be calculated from equation 29 substituting E′ for E. Combining this quantity with equations 30 and 34 and substituting them into equation 33 results in

$$G(E_1, E')dE' = \frac{(1-m)\Gamma_m}{C_m \gamma^{1-m} N} \frac{\Psi dE'}{v' E'^{2-2m}} N \int_{E'}^{\gamma E_1} T d\sigma(E_1, T) dT \qquad (35)$$

In most cases, $E' \ll \gamma E_1$ and for small E' the lower limit of the integral can be set to zero. Recalling the definition of the nuclear stopping power, the nuclear stopping power of the ion, $S_n^{(i)}$ is just N times the integral. Because $S_n^{(i)}$ is a constant for any ion target combination, and using the above approximations, we can write

$$G(E_1, E')dE' = A_m \Psi S_n^{(i)} \frac{dE'}{v' E'^{2-2m}} \qquad (36)$$

where. $A_m = \frac{(1-m)\sqrt{m}}{C_m \gamma^{1-m} N}$. $G(E_1, E')dE'$ determines the number of high generation recoils in the energy range (E', dE). Thompson's fourth initial assumption assumes that these recoils are isotropic. In other words,

$$G(E_1, E', \Omega')dE' d\Omega' = G(E_1, E')dE' \frac{d\Omega'}{4\pi} = A_m \Psi S_n^{(i)} \frac{dE' d\Omega'}{4\pi \, v' E'^{2-2m}} \qquad (37)$$

It is important to remember that the above equation is still differential in distance. This allows us to calculate the current of recoil atoms. Current is by definition the number of particles per volume element moving with a certain speed. The current of recoil atoms in the energy range (E', dE') moving in the area element $(\theta', d\theta')$ is

$$v'G(E_1, E', \Omega')dE' d\Omega' = A_m \Psi S_n^{(i)} \frac{dE' d\Omega'}{4\pi \, E'^{2-2m}} \qquad (38)$$

The flux of recoil atoms through a plane perpendicular to the flow of particles is proportional to this current. The flux across any plane not perpendicular to the flow of particles is proportional to the perpendicular flux multiplied by the cosine of the angles between the two planes. Inserting an arbitrary plane parallel to the surface inside the target and calculating the flux passing through this plane gives

$$\Phi(E_1, E', \theta') dE' d\Omega' = A_m \Psi S_n^{(i)} \frac{\cos\theta'}{4\pi\, E'^{2-2m}} dE' d\Omega' \qquad (39)$$

with θ' defined as the angle between a plane perpendicular to the direction Ω' of the recoil particles and the surface plane. By geometry, this angle and the angle between the surface normal and the direction of the recoil particles are equivalent.

Up to this point all calculations have been solved inside the target, hence the primes on the calculated values. Unprimed quantities from here on will refer to values outside the sample surface. To exit the bulk, the particles must pass through a surface and overcome surface binding forces. Thompson used a planer binding potential to simulate these forces. The planer binding potential assumes all the equipotential surfaces felt by a particle leaving the bulk and entering the vacuum are both planer and parallel to the sample surface. This will have the effect of decreasing the vertical velocity component of the ejecting particle upon passing through the surface. The horizontal component will be unaffected. The vertical velocity component will be decreased by $(2U/M_2)^{1/2}$ where U is defined as the surface binding energy. The surface binding energy is the minimum energy a particle needs

to escape the surface. Due to a lack of alternatives, U is most often approximated using the heat of sublimation.

Inserting a surface using the planer binding potential gives the following relations between the energy and the sputtering angle inside and outside the target:

$$E' = E + U \tag{40}$$

$$\cos\theta' = \frac{\sqrt{E\cos^2\theta + U}}{\sqrt{E + U}} \tag{41}$$

Using these relations, the Jacobian to transform from the volume element $dE'd\cos\theta'd\phi'$ to $dEd\cos\theta d\phi$ is

$$J = \begin{Vmatrix} \dfrac{dE'}{dE} & \dfrac{dE'}{d\cos\theta} & \dfrac{dE'}{d\phi} \\[2mm] \dfrac{d\cos\theta'}{dE} & \dfrac{d\cos\theta'}{d\cos\theta} & \dfrac{d\cos\theta'}{d\phi} \\[2mm] \dfrac{d\phi'}{dE} & \dfrac{d\phi'}{d\cos\theta} & \dfrac{d\phi'}{d\phi} \end{Vmatrix} = \frac{E\cos\theta}{(E+U)\cos\theta'} \tag{42}$$

Incorporating equations 40 - 42 into equation 39 and dividing by the flux of incoming ions, we get

$$Y(E,\theta) = A_m S_n^{(i)} \frac{E\cos\theta}{4\pi\,(E+U)^{3-2m}} \tag{43}$$

which gives the number of sputtered atoms per ion, per unit energy, and per steradian. Examining this equation, we see that the energy and angular distributions are independent of each other with the energy spectrum reaching a maximum at

$$E_{max} = \frac{U}{2(1-m)} \tag{44}$$

The angular distribution is cosine, peaking at surface normal, $\theta = 0$.

Roosandaal Sanders Theory

Using a normally incident ion beam, peaking of the sputtering yield at surface normal has been observed by some researchers.[32] Unfortunately, as the incident ion beam moves away from surface normal, the peak in the distribution also shifts away from surface normal as long as the energy of the incident ion is not too large. Returning to the initial assumptions of Thompson, this off normal peaking is thought to result from anisotropy in the momentum distribution of the recoil atoms and the subsequent cascades. This anisotropy would result in a "memory" of the incident ions angular direction being partially retained by the collision cascade. Roosandaal and Sanders[33,34] examined this anisotropy and derived angular and energy distributions for four different cases. Other than discarding the idea of an isotropic momentum distribution, they used the same assumptions that Thompson used in linear cascade theory.

The four cases considered by Roosendaal and Sanders are as follows:

(1) The incident ion creates primary recoils and these recoils form subsequent cascades. The ion is ignored completely after it has finished creating primary recoils. This follows the assumptions Thompson used to calculate the flux of sputtered particles. The interaction of the ion with the primary recoil can be described using a power potential with the probability of an ion of energy E_1 producing a recoil with energy T at polar angle θ and azimuthal angle ϕ measured with respect to the incoming ion as

$$d\sigma\,(T,\theta,\phi) = A_m\,(E)T^{-1-m}\delta\left(\cos\theta - \sqrt{\frac{T}{dE_1}}\right)dT\cos\theta\,\frac{d\phi}{2\pi} \qquad (45)$$

The momentum density of these recoils can be written as

$$S(T,E',\cos\varepsilon',\phi') = c\left(\frac{T}{2E'^2} + \frac{3}{2}\frac{T^{\frac{1}{2}}}{E_1'^{\frac{3}{2}}}\sqrt{\frac{M_1}{M_2}}\cos\varepsilon'\right) \qquad (46)$$

where T is the energy of the primary particle, E' is the recoil energy, ε' is the angle with respect to the incident particle, and ϕ' is the azimuthal angle of the recoil. In the case of primary recoils, $M_1 = M_2$ and higher generation recoils are characterized by the angles ε' and ϕ' measured with respect to the primary recoil.

(2) The incident ion can be back scattered toward the surface by target atoms and kick out surface atoms thus acting as the primary recoil. This is especially important for light ion sputtering when $M_1 < M_2$. Analogous to

case 1, the target-ion interaction can be described by a power potential to give the following cross section:

$$d\sigma\,(T,\theta,\phi\,) = A_m\,(E)T^{-1-m}\delta\left(\cos\theta\,-\frac{M_1+M_2}{2M_1}\sqrt{\frac{E-T}{E}}-\frac{M_1-M_2}{2M_1}\sqrt{\frac{E}{E-T}}\,\right)dTd\cos\theta\,\frac{d\phi}{2\pi}$$

(47)

(3) When the incident ion is of low energy, all the energy will be dissipated in the region of the surface. All ions therefore behave as primary recoil atoms. The momentum density will follow equation 46.

(4) The last case is limited to oblique incidence collisions where the incident ion produces direct recoils in a specified momentum interval and no collision cascades will develop. The ion-target interaction is given by equation 45.

For cases 1-3, the energy integrated angular distribution for an ion of energy E_1 impacting a surface at θ_1 and transferring energy in excess of the surface binding energy U, T'>>U is

$$Y(E,\theta_i,\theta,\phi\,) \propto B(E_1)\cos\theta\left[1+6\sqrt{U}C(E_1)(-\cos\theta_i\cdot F(\theta\,)+\tfrac{3}{8}\pi\,\sin\theta\,\cos\phi\,)\right]$$

$$for\;\theta\neq 0,\;F(\theta\,)=\frac{2-3\cos^2\theta}{2\sin^2\theta}-\frac{\cos^2\theta}{2\sin\theta}\left(1+\frac{\cos^2\theta}{4\sin^2\theta}\right)\ln\left(\frac{1-\sin\theta}{1+\sin\theta}\right)$$

$$and\;Y(E,\theta_i,\theta,\phi\,)\propto B(E_1)\left[1-8\sqrt{U}C(E_1)\cos\theta_i\right]for\;\theta=0 \qquad (48)$$

$B(E_1)$ is a global constant that depends on the energy of the incoming ion. We are most interested in the shape of this distribution, so the exact value of B is

unimportant for us. The constant $C(E_1)$ does affect the shape and has the following form for the cases above:

$$C(E_1) = \frac{1}{\sqrt{\gamma E_1}} \qquad\qquad \text{Case I}$$

$$C(E_1) = \sqrt{\frac{M_1}{M_2 E_1}}\, g(E_1, T) \qquad \text{Case II}$$

$$C(E_1) = \sqrt{\frac{M_1}{M_2 E_1}} \qquad\qquad \text{Case III} \qquad\qquad (49)$$

$$\text{where } g(E_1, T) = \frac{M_1 + M_2}{2M_1} + \frac{M_1 - M_2}{2M_1} \frac{\dfrac{\gamma E_1}{T'} - 1}{\dfrac{\gamma E_1}{T'} - 1 - \gamma\, \ln\!\left(\dfrac{\gamma E_1}{T'}\right)}$$

The Roosandaal-Sanders theory accurately predicts the off normal peaking of the yield curve. Unfortunately, it also inaccurately predicts a cosine shape which is not observed in experiments. The shape in general shows over-cosine behavior, e.g. it follows a $\cos^n\theta$ with $1<n<3$.[35] This over-cosine behavior is seen in both cases of normal and off normal incidence ions. Whitaker,[6] Li,[7] and several other experimentalists[36,37,38,39] have empirically modified the Roosandaal Sanders equation to include a $\cos^n\theta$ term:

$$Y(E, \theta_i, \theta, \phi) \propto B(E_1)\cos^n\theta\left[1 + 6\sqrt{U}C(E_1)(-\cos\theta_i \cdot F(\theta) + \tfrac{3}{8}\pi\,\sin\theta\,\cos\phi)\right]$$

$$\text{for } \theta \neq 0,\; F(\theta) = \frac{2 - 3\cos^2\theta}{2\sin^2\theta} - \frac{\cos^2\theta}{2\sin\theta}\left(1 + \frac{\cos^2\theta}{4\sin^2\theta}\right)\ln\!\left(\frac{1 - \sin\theta}{1 + \sin\theta}\right)$$

$$\text{and } Y(E, \theta_i, \theta, \phi) \propto B(E_1)\left[1 - 8\sqrt{U}C(E_1)\cos\theta_i\right] \text{ for } \theta = 0 \qquad (50)$$

The new equation is fit to the experimentally obtained data using $B(E_1)$, n, and U as fitting parameters. This approach provides reasonable fits to experiment although the physical significance of the fitting parameters is lost. Regardless, the convention of using $B(E_1)$, n , and U as fitting parameters will be followed to evaluate the experimental data in this thesis.

Chapter 2. Experimental Characterization and Procedures

The sputtering apparatus can be segmented into four basic systems: (1) the ion source and its associated optics, (2) the laser induced fluorescence(LIF) detection system, (3) the ultra high vacuum (UHV) sample chamber including the surface characterization/residual gas analysis components, and (4) the master control computers with their assorted software.

Ion Systems - Hardware Specifications

The ion optics system consists of an ion source, 2 einsel lens, a wein filter, and several apertures and deflection plate electrodes (Figure 4). The core component of the ion optics system is a commercial Colutron ion source Model 101 - Q (Figure 5). The main components are a tungsten filament cathode and a stainless steel disk anode with a 1 mm diameter central hole. These components are mounted in a quartz glass holder assembly which directs the flow of gas molecules across the filament. The cathode-anode spacing is set to 6 mm and maintained by tension created by two support wires. The filament is voltage regulated at 15 volts, 19 amps using an HP 6261B DC power supply. The anode is also run in voltage regulation mode at 75 volts, 0.5 amps using an HP6448B DC power supply. The entire ion source and two HP power supplies are floated at the energy of interest by an Fluke Model 415B high voltage power supply.
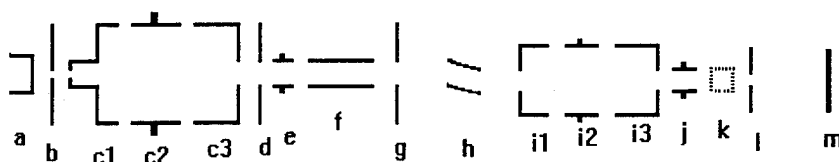
a) cathode
b) anode
c) first einsel lens
d) apeture(ground)
e) vertical deflection plate
 f) wein filter
g) apeture
h) 10 degree bend deflection plates
i) second einsel lens
j) vertical deflection plate
 k) horizontal deflection plate
l) apeture
m) sample

Figure 4. Ion Optics System
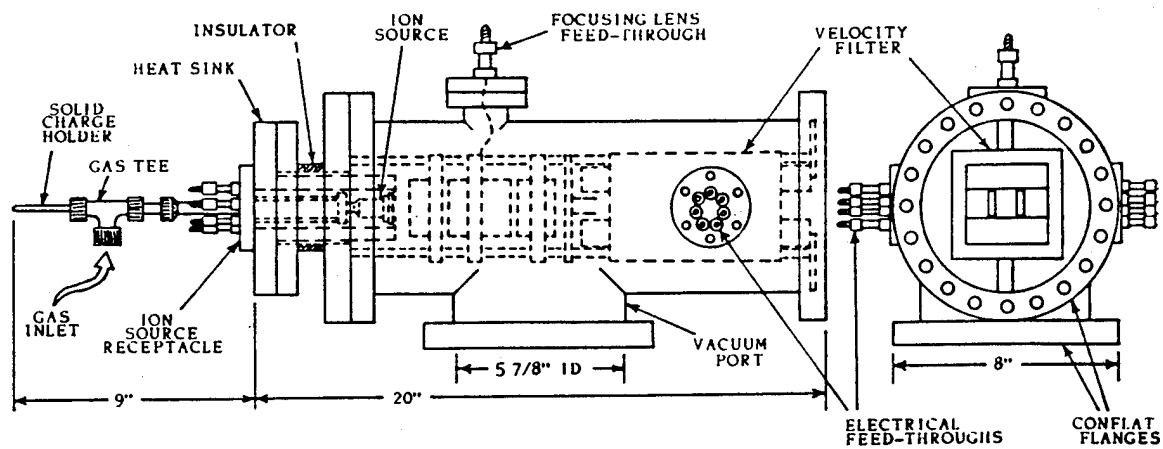
Figure 5. Colutron Ion Source.

The sputter gas is introduced slowly into the source until an arc initiates and a stable plasma forms between the cathode and anode. A stable discharge is maintained with gas pressures between 2×10^{-6} - 1×10^{-5} torr as measured in the ion beam chamber (30 - 50 μ measured at the gas inlet leak valve). Standard operating conditions and optimum long term stability of the ion beam were obtained using an ion beam chamber gas pressure of 2.8×10^{-6} torr. The first einsel lens is used to extract the ions from the ion source and focus the beam to the first aperture. The einsel lens consists of three cylindrical stainless steel electrodes. The first and third electrodes are held at ground while the voltage on the center electrode is varied. This produces a voltage gradient and corresponding electric field that bends divergent ions toward the center axis of the lens. The ions leave the first einsel lens and the center section of the resulting beam is selected by the first aperture. The ions then encounter a vertical deflection plate that centers the beam into the entrance of the wein filter. The wein filter is composed of electromagnets and variable electrodes in a configuration that produces crossed magnetic and electric fields. The ions enter a region of constant magnetic field and are bent at right angles to this field. Heavier ions moving at lower velocities are diverted more by the field due to their increased residence time inside the field. In the experiments described in this thesis, the wein filter electrodes were set to 100 volts and the magnetic field was tuned to the mass of interest by varying the current entering the coils of the electromagnets. The variation of magnetic field as a function of current was measured using a hall probe (Figure 6). The ions leave the wein filter and encounter an aperture plate held at ground. By varying the voltage on the electrodes, ions of a certain velocity(mass) can be selected to pass unobstructed through the aperture plate. An experimental scan of a mixed N_2^+ and N^+ ion beam is included as Figure 7 . The ions are

then deflected through a 10° bend and enter the UHV chamber. The ions are refocused by a second einsel lens and deflected by a pair of vertical and horizontal electrodes to pass through a final aperture plate before hitting the sample. All variable voltages to the electrodes are controlled by many turn potential meters and recorded in the lab manual on the day of the run. These readings are compared with simulations run using the Simion ion trajectory code which will be discussed in a later section.
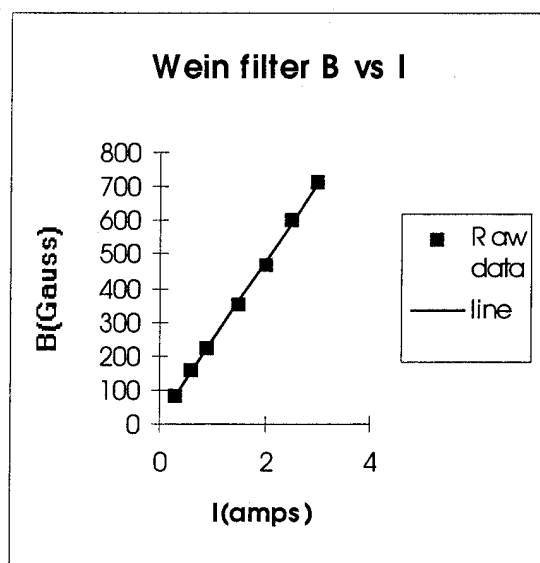
Figure 6. Wein Filter Field vs. Coil Current.
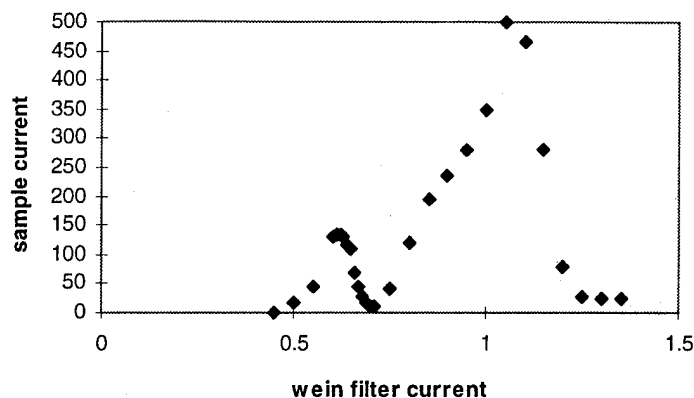
Figure 7. Wein filter scan of mixed N_2^+ and N^+ ion beam.

The entire ion optic system up to the 10° bend is mounted in a high vacuum(HV) chamber, base pressure 10^{-8} - 10^{-7} torr. This pressure is achieved by using a liquid nitrogen trap on a Varian 6" diffusion pump backed by a Edwards model E-2M two stage roughing pump. A Varian Turbo V-60 turbopump is connected at the bend to provide a differential pumping stage to prevent neutral atoms from contaminating the ultra high vacuum of the main sample chamber. The neutral atoms are unaffected by the deflection voltages and collect at the bend elbow where they are removed by the turbopump. The voltage source to the bend deflector plates is connected to a medium voltage switch controlled by the TTL pulse from a Camac DAC controller. This provides the 'flag' for the ion beam during an experiment.

Simion Simulations

Sputtering requires a well-defined and well-characterized ion source. It is very difficult to develop a feel for the effects of varying the potentials on the many adjustable electrodes present in this system. A simulation of the entire system detailed enough to concentrate on individual components was necessary to characterize the ion source fully. The simulation was conducted for two purposes: (1) the parameter space of all the ion elements was examined to find the optimum operating conditions which would provide the maximum beam intensity on the sample, and (2) the behavior of the wein filters was examined to verify the conditions necessary to mass resolve the ions in a mixed beam, e.g., N^+ vs N_2^+.

The Simion PC version 4^{40} electrostatic lens analysis code was used to simulate the ion optics system. With this code, the user defines a 16,000 pt array of electrode and nonelectrode points to simulate the ion optics of interest. The array is then numerically refined using overrelaxation methods until the potential between points converges to user selectable limits. This array is stored in a fast adjust file to allow easy variation of electrode potentials and subsequent calculation of voltage contours and ion trajectories. The Simion code does not treat space charge effects so the results achieved will not correspond exactly to experiment but are used to find order of magnitude corrections.

Extraction conditions

The first components examined using the Simion code were the ion source and first extraction optic. Whitaker[6] and Li[7] measured a maximum ion flux of 200 nA on the sample for light ion sources (e.g., Ne, Ar) without using the wein filter. Wolfram Maring, a postdoc in the Watts group, suggested the low beam current was caused by inefficient extraction conditions. The extraction region was modeled using Simion and ion trajectories initiated using starting points recommended by Coultron, the ion source manufacturer. The potentials on the electrodes were set using the values measured in the lab (Figure 8). As can be seen in Figure 8, the extraction regime was very inefficient and many ions did not propagate through the first einsel lens. The extraction optic was too far from the ion source anode. Adding an extension piece to the extraction optic would bring the extraction region closer to the ion source anode. The ion beam would still be tightly compressed and subsequently focused through the aperture (Figure 9). Using these results as a basis, a variable adjusting extension screw was designed for the first extraction optic. Using a separation of 10 mm between the anode and extraction optic, a doubling of the ion current was observed on the sample.
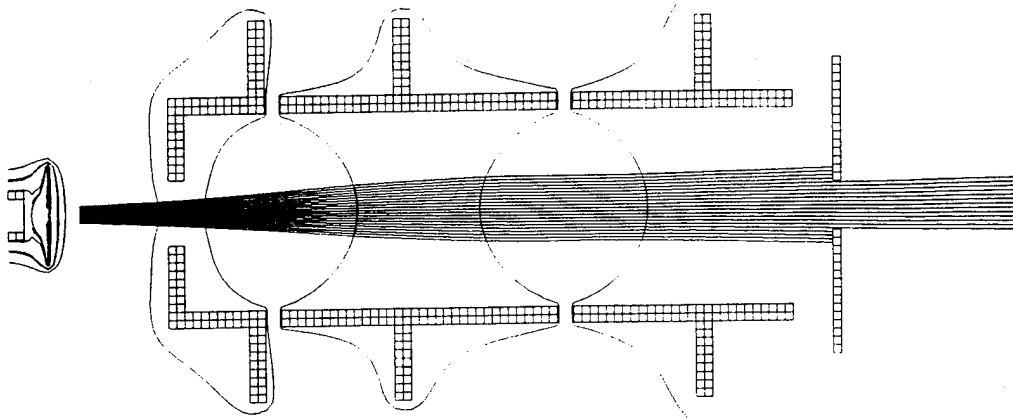
Figure 8. Ion Extraction Region.
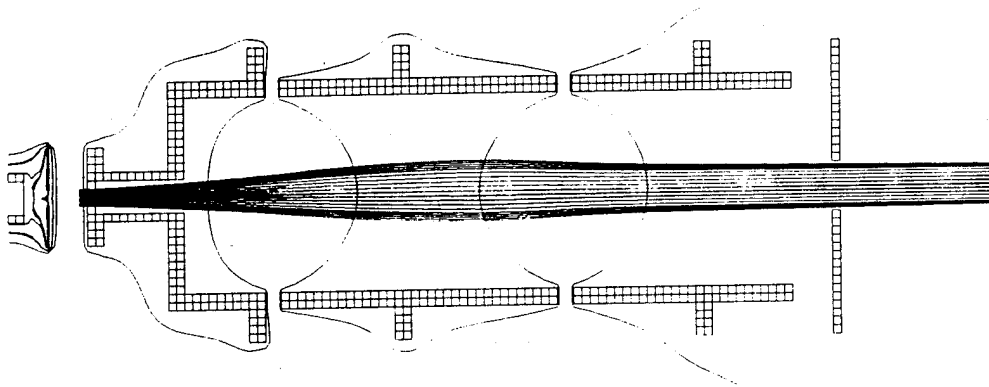
Figure 9. Extending Extraction Optic.

Wein Filter parameters

Whitaker and Li used single component ion beams so did not use a wein filter. Rather, they operated the ion source with all the wein filter components grounded. A wein filter can be used to mass resolve the mixed beam resulting from the ionization of a diatomic species or a halide salt. During the course of mass separation, the ion beam spread out with a corresponding decrease in intensity measured on the sample. The operating parameters required to mass separate the ion beam with minimal impact on the intensity transmitted to the sample had to be calculated. This data could be efficiently gathered using the Simion program.

The limitations on the maximum array size that could be input into the program forced us to divide the optics simulation into two parts to achieve the resolution necessary to accurately follow the ion trajectories to the sample. The most obvious dividing point was the 10° bend, with the first stage consisting of the Colutron ion source, the first einsel lens, the first vertical deflector, the wein filter and the associated apertures (Figure 10). The second stage starts with the first aperture after the 10° bend and includes the second einsel lens, a set of vertical and horizontal deflection electrodes, the final aperture, and the sample target (Figure 11).
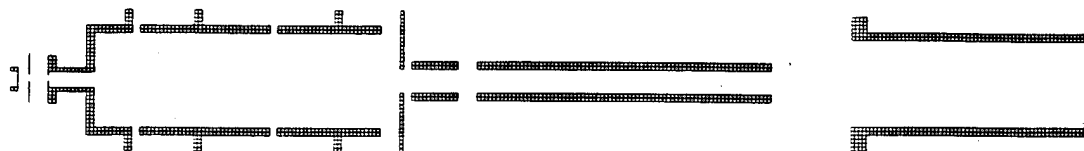
Figure 10. First Stage of Ion Optics.
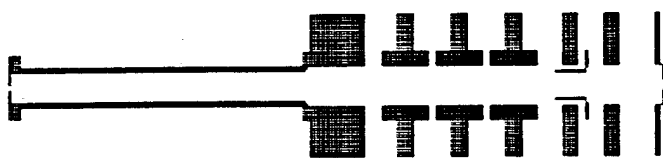
Figure 11. Second Stage of Ion Optics.

The behavior of the wein filter was simulated using the first stage. The experimentally determined voltages that produced an ion beam on the sample when the wein filter was turned off were used as the input voltages for the trajectory calculations. N_2^+ ion trajectories were initiated at the anode aperture and propagated through the optics to the exit aperture. These trajectories were saved as an input file for subsequent use as the wein filter parameters were varied. A constant 100 volt potential, - 50 volts top electrode/50 volts bottom electrode, was applied to the wein filters electrodes. This caused the N_2^+ beam to be deflected up to impact the top electrode (Figure 12).
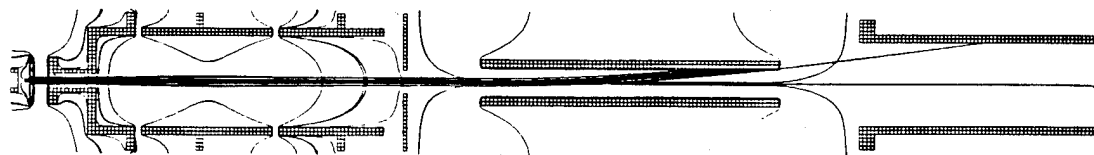
Figure 12. N_2^+ without magnetic field and 100 volt E field.

A homogeneous magnetic field is manually inserted into the area occupied by the wein filter electrodes. By varying the strength of this field,

the ion trajectories can be modified until the ions once more pass through the exit aperture (Figure 13).
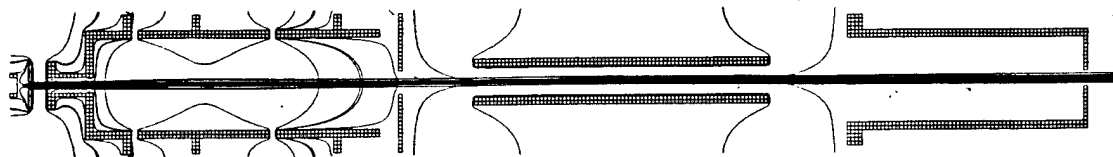
Figure 13. N_2^+ with 210 gauss magnetic field and 100 volt E field.

Changing the mass of the ion beam causes the beam to once again be deflected toward the wein filter electrodes (Figure 14).
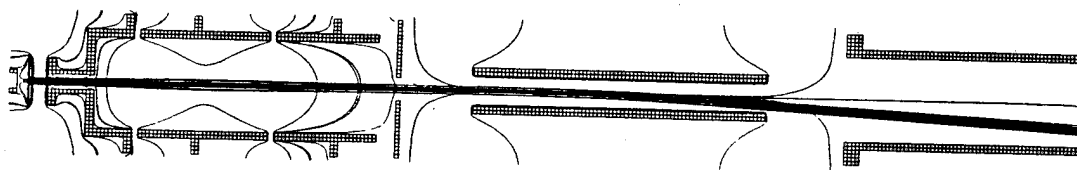
Figure 14. N^+ with 210 gauss magnetic field and 100 volt E field.

The magnetic field is then manually varied until the new mass of interest passes through the exit aperture. The resulting values were compared to experimentally-determined values to verify correct operation of the wein filter. The values for N^+ and N_2^+ are collected in Table 3.

Table 3 : Wein Filter Magnet and E field parameters

Ion	Simion(B)	Exp.(B)	E-Field
N^+	145	195	100
N_2^+	210	297	100
N^+/N_2^+	.69	.67	

The absolute differences between the predicted and measured fields are not important. The ratio of the fields is of more interest since it represents the separation of the two masses. The data in Table 3 reveal that these values are in good agreement.

Ion Beam Focusing

Finally, the second stage ion optics were used to determine the correct aperture dimensions needed to separate the UHV sample chamber from the HV ion chamber. There are two conflicting issues influencing the choice of an aperture size. First, the aperture acts as a controlled leak from the HV to UHV chamber. The rate of sample contamination by background gas species is directly proportional to the pressure inside the UHV chamber. To avoid

contamination of the sample during the time frame of a sputtering experiment the aperture needs to be as small as possible. Second, the ion beam needs to be focused through this aperture. If the aperture is too small, the ion beam focuses at the aperture resulting in spreading out of the ion beam at the sample surface (Figure 15). This will cause a marked degradation in measurable current on the sample. An aperture size was chosen which achieved minimum expansion of the ion beam and at the same time kept the background pressure under 4×10^{-9} torr during the experiments(Figure 16).

Figure 15. Small aperture spreading of ion beam.
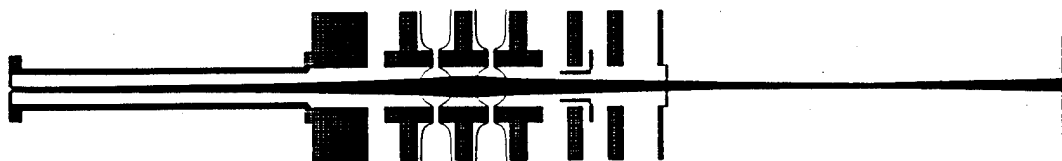
Figure 16. Correct aperture and focusing on sample.

LIF System - Hardware specifications

The LIF detection system includes the laser systems, fiber optic transmission cables, rotatable detector assembly, and the photo multiplier detector (Figure 17). The laser systems consist of a Coherent 699-29 Autoscan ring dye laser pumped by a Coherent Inova 100 Argon Ion laser. The 514.5 nm line of the Argon ion laser is used to saturate the absorption of the Rhodamine 6G dye used in the ring dye laser for the series of experiments covered in this paper. The 699-29 Autoscan laser is a mode locked, continuously scanning , frequency stabilized traveling wave ring dye laser with built in wavemeter. The laser beam is steered using two x,y adjustable mirrors to a 1 cm focal lens and fiber optic holder mounted on two x, y, z translation stages. The fiber optic cable is a fiberguide industries Superguide

G UV-Visible fiber with an inner core diameter of 50 μ and a wavelength range of 180-1100 nm. The fiber optic cable is threaded through Teflon tubing using techniques described in Scoles.[41] The laser is transmitted through 14 meters of fiber optic cable where it enters the UHV sample chamber through a pilot hole drilled in a 3.25 inch conflat flange. The fiber is connected and the corresponding hole sealed using low vapor pressure, high vacuum sealant on both the inside and outside of the flange. The fiber optic cable terminates in a second fiber optic holder mounted into a detector assembly that rotates about the sample in the polar plane.
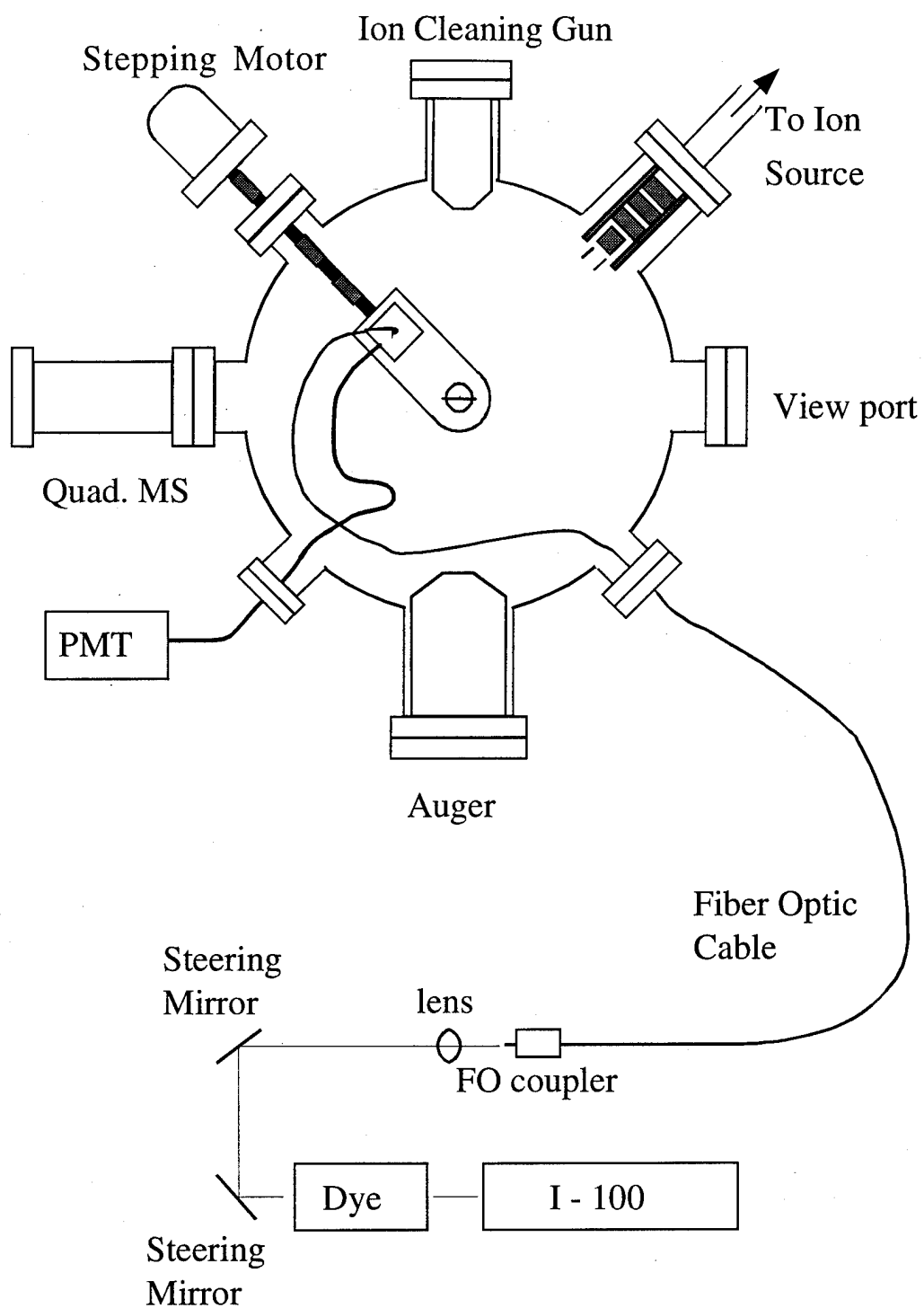
Figure 17. UHV Chamber and Laser Systems.

Optical fiber

Focusing Lens

Collimating Apertures

Spherical Mirrors

Fiber Bundle

Interaction Region

Light Trap
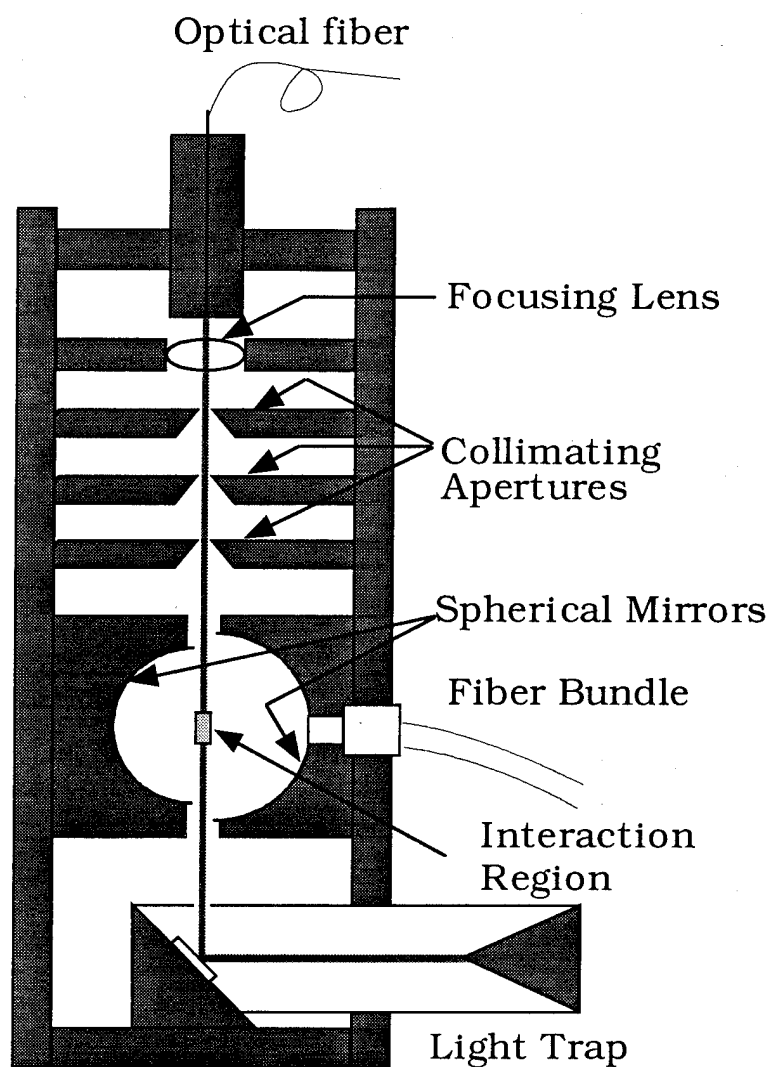
Figure 18. LIF Detector.

The detector assembly is similar in construction to detectors developed for molecular beam work by Hefter and Bergman and Shimizu and Shimizu (Figure 18). The laser beam enters the detector assembly from the top and is collimated by several apertures before passing through the interaction region and exiting into a beam dump. The sputtered Zr atoms pass through the

detector perpendicular to the laser beam (that is, into the plane of the paper in Figure 18), where they are excited. On the left and right of the crossing point of the two beams are two silvered high reflection hemispherical mirrors that collect the fluorescence into a 6 mm fiber optic bundle centered in the right hand mirror. The focal length of the two hemispheres has been chosen to maximize the focus of the fluorescence into this bundle. The fluorescence is carried out of the UHV chamber by the fiber optic bundle into a cooled red-sensitive RCA 31034-2 photon multiplier. The photomultiplier is run in photon counting mode where each photon causes a pulse of electrons that are amplified, discriminated, and then counted.

The current pulse from the photomultiplier is fed into a EG&G Ortec model 9301 fast preamplifier for a factor of ten voltage gain and then into a Ortec 9302 amplifier-discriminator. A multiturn control adjusts the discriminator level between 50 mV and 1V. This voltage pulse is then fed into a Standard Engineering Model TS 201 timer scaler. An Ar^+ sputtering experiment was used to set the correct discriminator level. A 45° incident Ar^+ ion beam produced 470 nA of current as measured on the sample. Using the ground state frequency of 16786.98 cm^{-1} for zirconium and measuring the sputtering yield at 45° for 5 sets of 5 second integrations produces the following curve (Figure 19):

Discriminator level
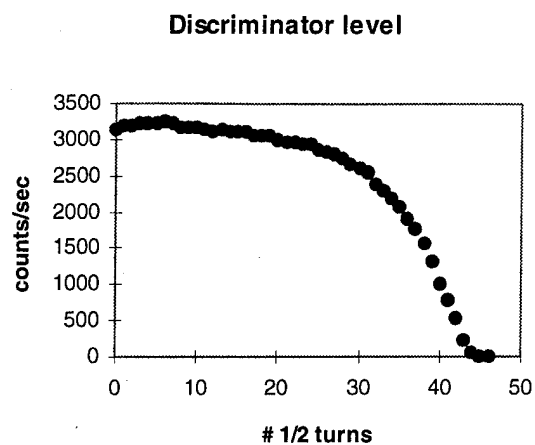
Figure 19. Signal vs. Discriminator level.

Differentiating this curve produces the graph shown in Figure 20.

Discriminator level
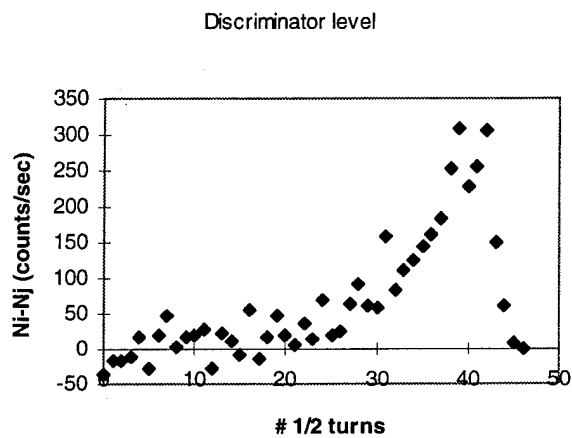
Figure 20. Discriminator differential curve.

Examining Figure 20, one can determine a discrimination level that retains the majority of the signal while removing background counts caused by after pulsing in the photomultiplier. The discriminator level was set at 15 turns.

The wavelength of the laser is controlled by the internal wavemeter of the Autoscan laser (precision of .0017 cm^{-1}). The excitation wavelength for the Zr ground state and first excited state used in the experiments in this thesis occur at 595.5 nm and 579.8 nm. The fluorescence wavelengths cover a wide range, starting from the excitation wavelength. A filter pack that allows only photons in the range 650 - 850 nm is inserted between the fiber optic and the PMT, which filters out both photons from the incoming laser light and long wavelength photons characteristic of thermal sources. Thus, the filter pack greatly reduces the background due to scattered laser light (although scattered laser light still remains the largest contributor to background counts).

The detector is mounted on a computer-controlled rotation stage. The rotation of the detector is limited by the Auger spectrometer in one direction and the ion beam in the other direction. This effectively limits the range of angles that can be scanned to 103°. The origin of the detector is set using limit switches on the rotation motor, which produces a reproducible home position for measurements. The ion beam enters the chamber at 135° compared to the home position.

Detection Efficiency - Zr spectroscopy

An examination of the detection efficiencies of the ground state and first excited state is necessary before any comparison between the sputtering

yields can be accomplished. Figure 21 is a partial schematic of the energy levels of Zr showing some of the excitation and decay pathways of interest in this thesis.
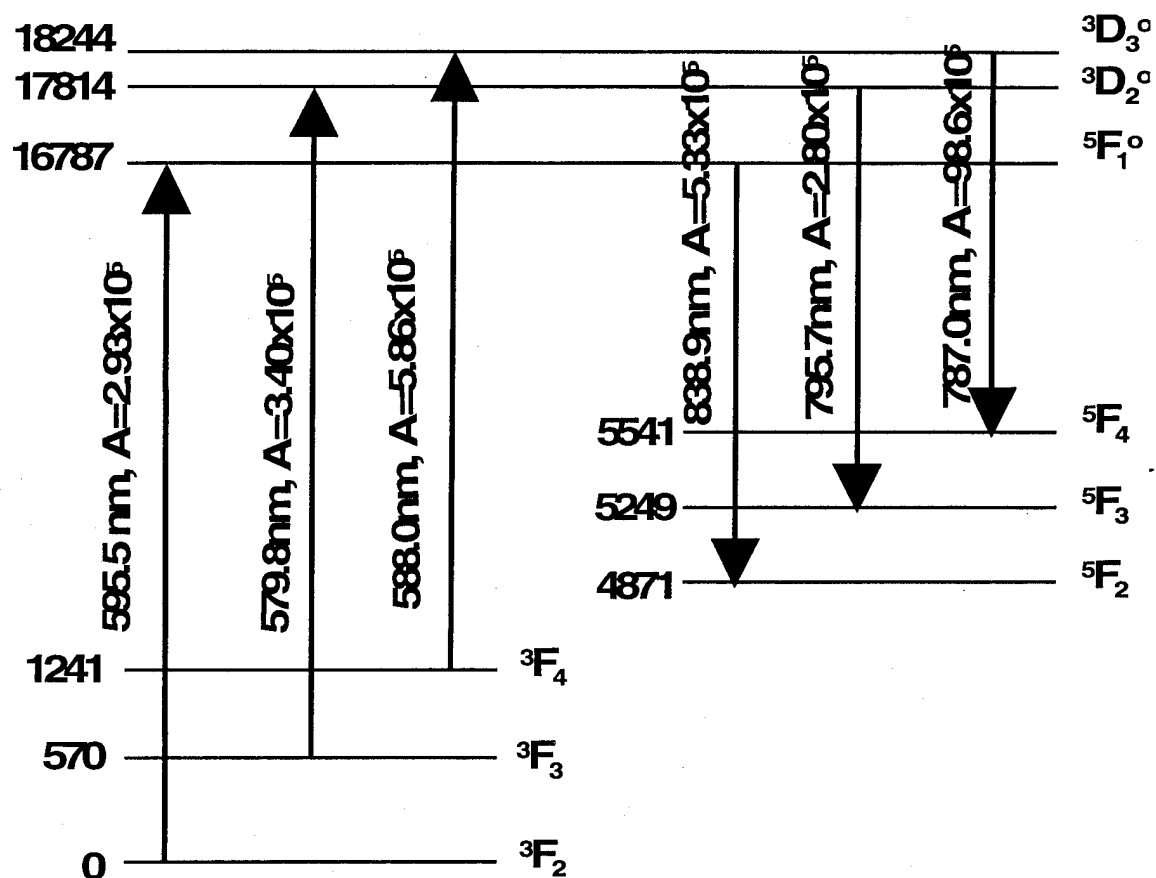
Figure 21. Energy level diagram of zirconium.

Table 4 lists all the major decay pathways, listed by Corliss and Bozmann,[42] of the two upper states that must be considered in this thesis. The observed signal is proportional to the number of atoms in the required state, the fraction of those atoms that are excited, the fraction of the atoms that

fluoresce in the detection region, and the fraction of atoms that are detected by the photomultiplier. In other words, we are interested in calculating the various quantities in the following equation:

$$N_{observed} \propto N_{atoms} \, F_{excited} \, F_{fluorescence} \, F_{detected} \tag{51}$$

Table 4. Major decay pathways of two upper states of Zr (used in thesis).

Lower State E(cm^{-1})	ΔE (cm^{-1})	λ (Å)	gAx10^8 (sec^{-1})
Transitions from $^5F_1^{\,\circ}$ at 16787 cm^{-1}			
0	16787	5955	.0088
4197	12590	7940	.0026
4376	12411	8055	.0026
4871	11916	8389	.0160
5023	11764	8498	.0056
Transitions from $^3D_2^{\,\circ}$ at 17814 cm^{-1}			
0	17814	5612	.0013
570	17244	5797	.017
4186	13628	7336	.0083
4376	13438	7440	.0190
4871	12943	7724	.0016
5023	12791	7816	.0021
5249	12565	7957	.0140

Fraction of atoms excited by laser

The sputtering experiment is operated above the saturation point in the power curve. At the saturation point the fraction of atoms that is excited by the laser is proportional to the ratio of the statistical weight, $g = 2J+1$, of the two states. Thus 60% of the $^3F_2$ atoms would be excited to the $^5F_1°$ state and 71% of the $^3F_3$ state would be excited to the $^3D_2°$ state.

Fraction of atoms that fluoresce in detection volume

The fraction of atoms that would fluoresce in the detection volume of our detector is dependent on both the velocity of the atoms and the lifetime of the upper state. Pellin and Wright[43] measured the velocity distributions of the ground and two lowest excited states of Zirconium and found them to be indistinguishable. At 1-3 kev, the energy distributions follow Thompson's equation (see Chapter 1, equation 43) fairly accurately with $U = 6.305$ eV and $m = 0$ corresponding to the hard sphere model. Plotting this equation, the most probable velocity is 2.6×10^{-5} cm/sec (Figure 22).
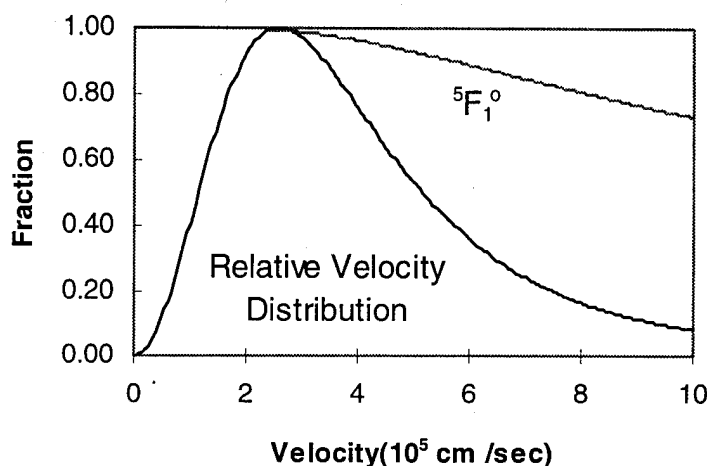
Figure 22. Fraction of $^5F_1^o$ atoms that will radiatively decay before leaving the detection volume.

Measurements by Whitaker[6] and Li[7] indicate the average distance from the point where a Zirconium atom is excited by the laser to a point where it leaves the detection volume is 0.3 cm. The velocity of a Zr atom with energy ε is $1.46 \times 10^5 \sqrt{\varepsilon}$ cm-s^{-1} with ε measured in eV. The time it takes a Zr atom to transverse the 0.3 cm detection volume is $2 \times 10^{-6} \varepsilon^{-1/2}$ seconds. The fraction of atoms of an energy ε decaying in the detection volume can be calculated from

$$f_{flourescet}(\varepsilon) = 1 - e^{\left\{\frac{-2 \cdot 10^{-6}}{\tau \sqrt{\varepsilon}}\right\}} \tag{52}$$

where τ is the lifetime of the upper state. The lifetime of the Zr $^5F_1^o$ state is 228 ± 10 ns.[44] Figure 22 plots the fraction of atoms decaying in the detection

volume as a function of velocity as well as the relative velocity distribution of these particles. We are interested in the total fraction for all velocities. To get this number, Thompson's equation is normalized to determine the fraction of atoms in the energy range ε_0 to $\varepsilon_0 + \Delta\varepsilon$

$$f_{\varepsilon_0 \rightarrow \varepsilon_0 + \Delta\varepsilon} = \frac{2U\varepsilon_0\Delta\varepsilon}{(U + \varepsilon_0)^3} \tag{53}$$

Equation 53 is then multiplied by equation 52 and integrated over all energies to determine the total fraction of excited atoms that decay in the detection volume.

$$f_{flouresce} = 2U\int_0^\infty \frac{\left(1 - e^{\left(\frac{-2x10^{-6}}{\tau\sqrt{\varepsilon}}\right)}\right)d\varepsilon}{(U + \varepsilon)^3} \tag{54}$$

Using U=6.305 eV and $\tau = 228$ ns, 82% of the $^5F_1^{\,o}$ atoms fluoresce before leaving the detection volume.

Hannaford and Lowe did not measure the lifetime of the $^3D_2^{\,o}$ state but did measure the lifetime of the $^3D_3^{\,o}$ state as 267 ± 10 ns.[6] Using equation 53, this would predict 79% of the $^3D_3^{\,o}$ atoms will fluoresce before leaving the detection volume. This lifetime was measured using laser induced fluorescence techniques and is most likely correct. Using the data from the tables of Corliss and Bozman,[2] we would calculate a lifetime of 843 ns, 790 ns, and 776 ns for the $^5F_1^{\,o}$, $^3D_2^{\,o}$, and $^2D_3^{\,o}$ states respectively (using $\tau = \left(\sum A_i\right)^{-1}$).

The discrepancy is addressed by Hannaford and Lowe and is explained by the experimental methods used by Corliss and Bozman to construct their table. Corliss and Bozman's data were based on discharge data to create the excited states. Many higher states are created simultaneously and these higher states can decay into the upper state of interest resulting in an unusually large lifetime. The Corliss and Bozman data must be used with caution, but can be used to determine the relative lifetimes of the three states of interest to this thesis. The data clearly shows the lifetime of the $^3D_2{}^o$ state falling between the $^5F_1{}^o$ and the $^2D_3{}^o$ state. Thus between 82% and 79% of the $^3D_2{}^o$ state will fluoresce in the detection volume. These values are similar enough that they will be assumed to be equivalent for the work in this thesis.

Fraction of atoms detected by photomultiplier

The fraction of fluorescence emitted in the detection volume that is passed by the filters (650 nm to 850 nm) in the photomultiplier can be estimated from the branching ratios of the various states. Using the gA values from Table 4 we can calculate this fraction using the following equation.

$$f_{det\,ected} = \frac{\sum gA_{det\,ected}}{\sum gA_{all}} \tag{55}$$

Assuming only 50% of the 8498 Å decay from the $^5F_1{}^o$ state is detectable, 67% of the atoms in the $^5F_1{}^o$ state will decay with wavelengths detectable by our

system. 71% of the $^3D_2^{\,o}$ system will decay with detectable wavelengths. To normalize the data for differences between excitation efficiency and detector efficiency, the data needs to be divided by the product of f_{detect} and f_{excite}. The $^5F_1^{\,o}$ data needs to be divided by 0.40 and the $^3D_2^{\,o}$ data by 0.50.

UHV Sample Chamber and Surface Characterization

The UHV chamber shown schematically in Figure 17 is made of stainless steel and is pumped out by a Balzers turbo pump backed by a 4" Acatel diffusion pump and rotary stage pump. All flanges incorporate the use of metal seals to insure a base pressure of 5×10^{-10} torr. A titanium sublimation pump is used to preferentially remove O_2 from the residual gas atmosphere measured in the chamber.

The sample specimen is held by a VG 3-way sample manipulator mounted on the top of the chamber. The sample manipulator provides x, y, z translation as well as polar and azimuthal rotation of the sample. Once the sample has been mounted on the sample holder, it is aligned so that the surface and center of the sample coincide with the center axis of the chamber. This is accomplished using an alignment cone constructed specifically for this purpose.

An E beam heater attached to the back of the sample holder is used to flash heat the sample after long periods of sputtering or sputter cleaning. The flash heating, up to 800° C, removes absorbed contaminants from the sample surface. The surface contaminants are monitored using an Auger spectrophotometer both before and after an experiment. A new sample

usually has a large fraction of absorbed C, N, and O on the metal surface (Figure 23). Because heating alone rarely removes these contaminants, sputter cleaning is usually necessary. An electron gun is mounted in the chamber for this purpose. The chamber is back filled with a pressure of 1×10^{-5} torr of argon gas and the resulting ion beam rastered across the sample surface. Figure 24 shows the clean surface that occurs after sputter cleaning and subsequent annealing.
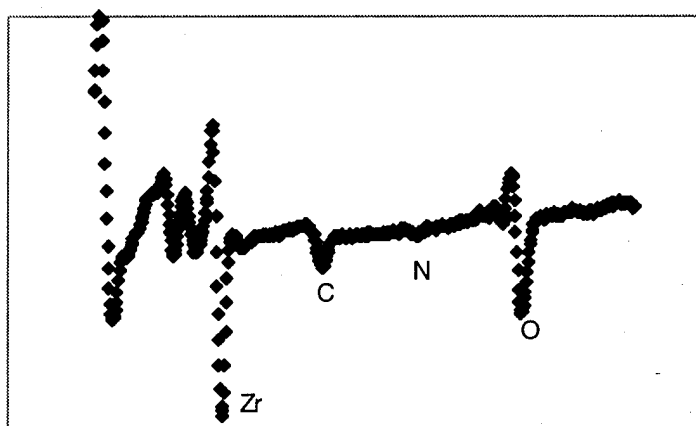
Figure 23. Auger scan of a dirty surface.
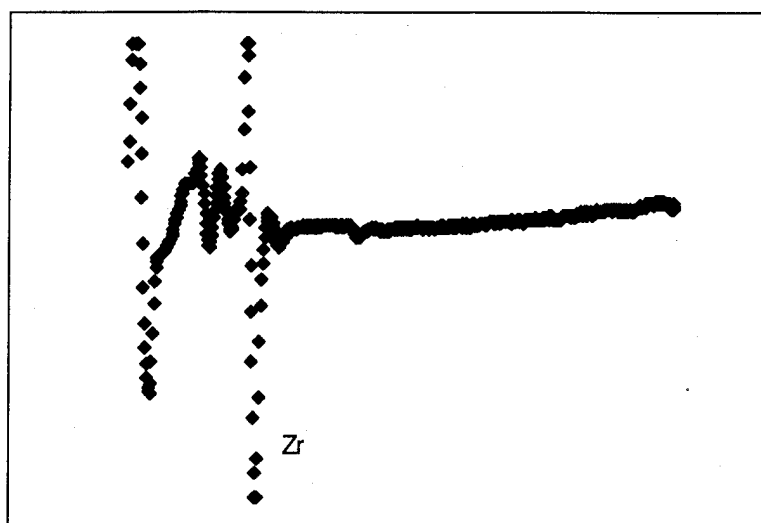
Figure 24. Auger Scan of Clean Surface.

Control Computers and Programs - Hardware

The control system for the experiment consists of an IBM PC AT, CAMAC crate, and Apple 2E computer. The Apple computer controls the laser system and is slaved to the IBM master computer through a serial interface. The CAMAC crate holds the amplifier discriminator for photon detection, the timer scalar for photon counting, and a DACA board to control the shutter on the ion and laser beam. Several programs have been written for the IBM PC to optimize the interfacing of the various components.

Programs - Master Menu

The master control program is a Turbo Pascal menu program (pmesprog.exe) that allows the user to select the type of experiment they would like to run. The menu consists of 8 selections as follows:

1. Auger
2. Mass
3. Freq
4. Angular
5. DAC
6. BASIC
7. DOS
8. init CAMAC

The selections for the most part are self explanatory. Auger allows one to electronically record or plot Auger data. Mass allowed one to electronically record or plot Mass Spectrometry data - this selection was disabled when the mass spectrometer was removed. Freq and Angular allows the user to run frequency or angular scans. DAC is used to test the D/A converter and to open and close the ion beam shutter. BASIC puts one in the BASIC language directory to allow the user to run various characterization programs and DOS transfers the user to the DOS operating system. The Init CAMAC selection rezeros all the CAMAC crate components and is always initiated at the start of a run.

Auger program

The Auger program is written in Pascal and interfaces the Auger Spectrometer, the Timer Scaler, and the A/D converter to the IBM PC to produce an electronic copy of the auger scan. The program starts with a sub

menu that allows the user to choose between recording an auger spectrum, plotting a previously recorded auger scan to the screen, or obtaining a listing of previously recorded spectra saved on the hard disk. The auger scan program prompts the user for the input data required to run the experiment. The input data starts with an output file name which the program uses to save in a DOS directory named c:/sputter/augdat. The convention has been to name this file beginning with an as and then including the date and run number. For example as082703.dat would be the third auger scan taken on august 27. The year is not recorded but can be obtained by examining the date when the file was generated using any regular DOS file program. The operating parameters set to run the Auger are then entered including the lower limit of the voltage scan, usually set at zero; the upper voltage scan limit, set at -600 volts; the scan speed, set at 3 eV/sec; the voltage scale, set at 100 volts/div; the sample speed, set at the (scan speed)/(# of data points at each voltage); the number of data points averaged at each scan point; and finally a comment line to record any pertinent observations. All of these parameters are set by the user on the Auger control unit. The program then calculates the number of points in the interval specified by the lower and upper limits and sets up the arrays to store the data. Once the Auger scan is started, the program uses the A/D converter to input the x, y voltage data from the spectrophotometer. The data is written to the screen as well as the output file at the end of the scan. The auger plot program reads the data from this output file and replots it to the screen.

Mass spectrometer program

The mass spectrometer program operates similarly to the auger program. When the mass spectrometer was removed from the system, this program was disabled so shall not be explained in any detail.

Frequency Scan Program

The frequency program is a Pascal menu that allows the user to select between recording a frequency scan, plotting a previously recorded scan, or listing previously recorded scans saved to the hard disk. The frequency scan program is a BASIC program used to record a frequency scan on the sample keeping the scan angle fixed. The program interfaces the IBM PC to the Apple 2E, the compumotor, the ion beam shutter, the laser shutter, and the photomultiplier tube. The first input parameter requested is the name of the output file to store the data which defaults to the c:\sputter\freqdat\ directory. The convention is to start the file with a "fs" followed by the date of the scan. The next input parameter requested is the integration time. The integration time is the time at each step of the scan the timer scaler will pause to collect data. The data written to the output file will be in units of counts/second so the data will be divided by this integration time if it exceeds 1 second. The standard time used in this thesis was 5 seconds.

The next input parameters are the number of scans, the detector position, the starting energy of the laser in cm^{-1}, the frequency range to scan, and the increments in megahertz(MHz). Up to 100 scans can be recorded to disk although the standard number used was 5 scans. These scans are then

averaged to give the data presented in this thesis. The detector position is usually set between 45° and 60° as measured from surface normal of the sample. Once the user inputs the starting energy, the interval , and the increment to move the laser, the program calculates the number of steps necessary and starts the scan. The sample is rotated to the user defined angle, all shutters closed, and dark background counts collected for 5 sec. The laser wavemeter is queried for the current laser frequency and changed to the starting frequency as needed. The data collection proceeds in the following manner: Photon counts are measured with everything on, the laser on, and then the ion beam on. These counts along with the difference using the following formula $N_{signal} = N_{all\ on} - N_{laser} - N_{ion} + N_{dark}$ are recorded to screen and file. The laser is then scanned forward in frequency and the next data points taken. The laser frequency is checked every 10 points against the built in wavemeter and frequency corrections are made at this time.

Angular Scan program

The angular program is very similar to the frequency program. It is a Pascal menu program that presents the user with the choices of taking an angular scan, plotting a previous scan, or displaying a list of previous scans present on the hard drive. The angular scan program is written in BASIC and its first input parameter is the standard output file name. Following the standard convention, the file name starts with "an" followed by the date. The program then queries the user for the integration time, # of spectra, starting angle, ending angle, and angle increment. These parameters are used to set up the number of steps in the scan and the corresponding data arrays.

Several characterization data are then entered including the ion sputter angle, sample metal, sputter ion, photomultiplier voltage, ion beam current, and ion beam voltage. These data are entered in the lab book and in the heading of the electronic record for redundancy. The laser frequency in cm^{-1} is then entered along with the number of steps to scan before pausing. The program then starts and runs like the frequency scan program with the difference that the frequency is now held constant and the detector angle changed for each point. The laser frequency is also verified at every point and corrected as necessary and the program will pause after a specified number of steps so that the sample can be flash heated to remove any absorbed contaminants. The number of steps is usually set to correspond with one complete scan so that a fresh surface is prepared for each scan.

DAC Test Program

The DAC program is a Pascal program that allows the user to test the D/A board and open the ion beam shutter. It is set up to prompt the user for the DAC channel they want to activate and then allows the user to set the channel to 0 - 5 volts. This is used commonly to insure that the ion beam shutter is receiving 5 volts and is open at the start of the day.

BASIC Program

The BASIC section shunts the user to the BASIC directory and allows the user to directly run any of the various BASIC characterization or alignment programs. The two most common alignment programs used to

align the laser into the fiber optic are ccount.bas and detpos.bas. The ccount.bas program is a looping program that is used to optimize the laser intensity into the fiber optic cable. The user sets the iteration time and the program interfaces the timer scaler and the photomultiplier tube to provide continuous counting of the photons hitting the photomultiplier. The user translates the laser into the fiber optic cable and maximizes the signal displayed on the computer screen. To end the program the user hits the standard BASIC escape keys of control C. The detpos.bas is a subroutine that controls the compumotor used to rotate the detector around the sample. The user specifies the detector position and the program moves the detector to that position. This is used to verify operation of the compumotor before running. Any other subprograms used in the other master programs can be tested here also, making the debugging of the programs easier.

All the programs used to collect data for this thesis are collected in Appendix A. These programs are useful only with an experimental setup similar to the Watts lab, but are included so that the latest versions and changes may be collected in one location.

Procedures

To run an angular scan sputtering experiment, the following procedure needs to be followed. The Argon Ion laser must be turned on for 2 - 3 hours before the laser reaches optimum stability The electronic control unit for the Dye laser must also be turned on at this time. The control unit, which supplies power to the reference leg heater on the Dye laser, takes at least 30 minutes to stabilize.

While the laser system is warming up, the sample surface can be checked for cleanliness and the ion beam started. The sample is centered in front on the Auger Spectrometer using the VG sample manipulator. The center position has been recorded on a 3x5 card that has been taped to the VG manipulator. The recorded position is used as the standard when taking Auger scans. The first Auger scan is recorded and examined to determine surface cleanliness. Zr, C, N, and O peaks are usually observed. If a new sample is not used, flash heating the sample to 800 °C is usually sufficient to clean the sample. After a four minute sample cooling period, a second Auger scan is recorded to verify the presence of a clean surface. The four minute waiting period is necessary to reduce the number of background infrared photons that would swamp the photomultiplier signal. If peaks other than Zr are present, a combination of flash heating and/or sputter cleaning is necessary to clean the surface.

Once a clean surface is present as determined by Auger, the Auger is turned off and the sample is positioned to the center of the chamber. The sample is rotated to determine the angle of incidence of the ion beam impacting the surface. The ion beam enters the chamber at 135° as measured from the zero point of the detector. Setting the VG sample manipulator at 269° will produce a 45° incident beam on the sample.

After positioning the sample, the DAC program is run and channel 2 set to 5 volts to open the ion beam shutter. The ion beam is started and the separation valve between the ion beam chamber and the UHV chamber is opened. The ion beam current is measured on the sample using an electrometer and the various ion optics elements adjusted to maximize the current on the sample. The ion beam is allowed to bombard the sample surface as the lasers finish warming up. The ion beam needs approximately

30 minutes of continuous operation to stabilize so it will not migrate during an experiment.

Once the lasers and ion beam have achieved stable operation, the laser is tuned to the frequency of interest. The laser is blocked, ion chamber valve closed, and ccount BASIC program is run to measure the background counts in the chamber. These should be between 5 - 8 counts/sec. The laser is then unblocked and coupled into the fiber optic cable using two steering mirrors and two three-way translation stages that hold the fiber optic and a 1 cm focal length lens. The scattered laser light is monitored in the chamber and the mirrors and translation stages adjusted to maximize this signal. This occurs when the maximum light is being transmitted by the fiber optic cable.

The laser is placed in remote operation, the ion chamber valve opened, and the ion current measured on the sample is maximized. The angular scan program is then run entering the appropriate parameters for the experiment. The sample is flashed to 800 °C to clean the surface, cooled for four minutes, and the scan started. The program pauses at a user defined time and the sample is again flashed to 800 °C and cooled for four minutes. After completing all scans, the sample is raised and centered in front of the Auger and another scan taken. The sample is flashed to 800 °C and cooled for four minutes and the final auger scan is taken to verify a clean surface.

Chapter 3. Results and Discussion

Laser Power Saturation Determination

The initial set of experiments consisted of characterization experiments to verify that the experimental apparatus was operating correctly. The first series of experiments conducted were frequency scans to determine the optimum laser power needed to saturate a transition. An Argon ion beam at 45° incidence was used as the sputter source to provide 200 nA of current on the zirconium foil sample. The laser was set to the ground state frequency of zirconium, 16786.978 cm^{-1}, and a variable thickness neutral density filter pack was inserted to modulate the laser power input into the fiber optic cable. The LIF detector was rotated to 45° and five scans were measured. These five scans were averaged to give the curves in Figure 25.
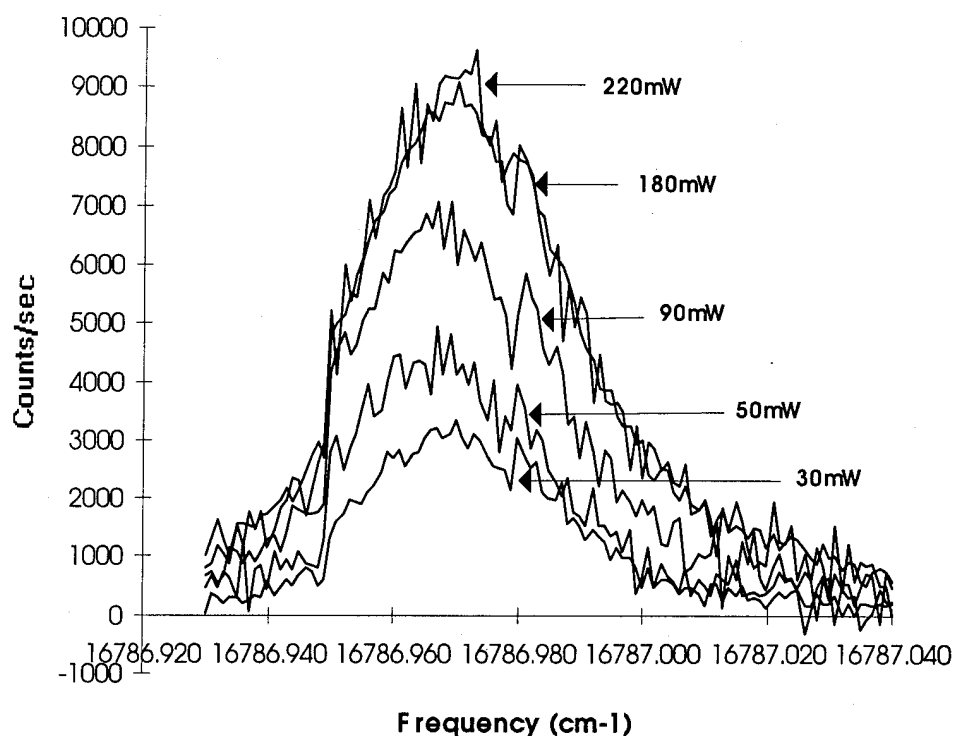
Figure 25. Ar$^+$ ground state frequency curves

The peak of the scan was located and a five point average made to determine the maximum signal point. Several experiments were completed varying the laser power by changing the thickness of the neutral density filter between experiments. The maximum signal vs. laser power is plotted in Figure 26. It can be seen that the signal reaches a plateau around 150 mw in power; thus all experiments were conducted using laser powers in excess of this 150 mw.
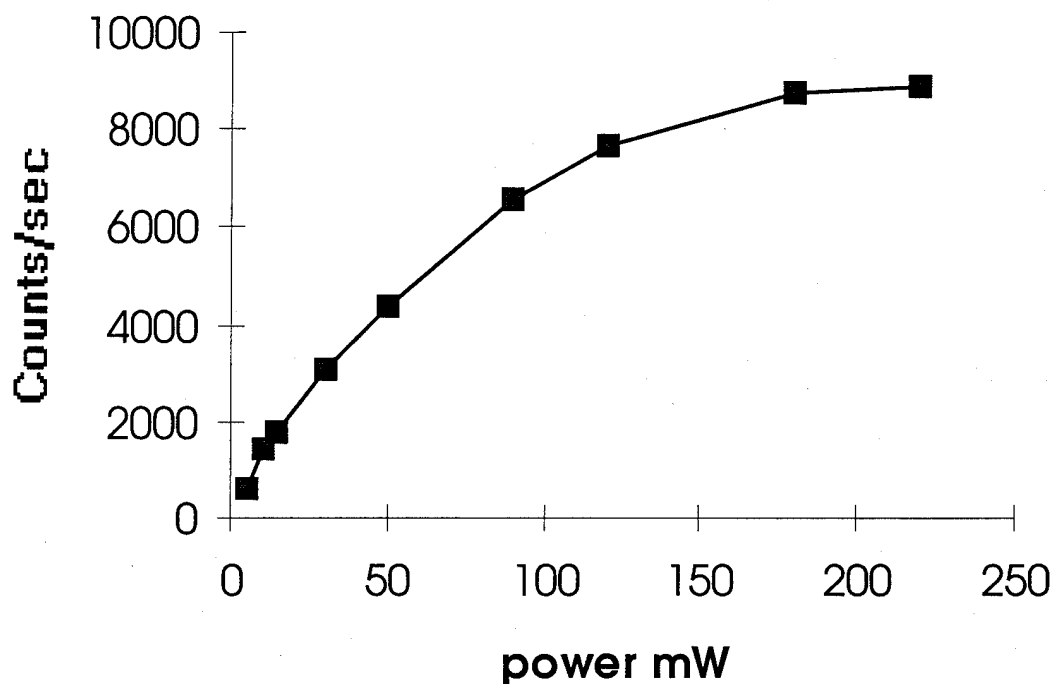
Figure 26. Power saturation curve.

Ar$^+$ Ground State Sputtering

Several changes were made to the instrumentation since Whitaker and Li completed their series of experiments on noble gas ions (see Experimental). Ar$^+$ ion sputtering experiments were reaccomplished to insure that the results would correspond to previous measurements. A 1.9 keV ion beam provided 500 nA of current as measured on the sample. 160 mW at 16786.978 cm^{-1} was

used to probe the $^3F_2 \rightarrow {}^5F_1{}^o$ transition. A one second integration time and a one degree angular resolution was chosen. Five scans were recorded at 15°, 30°, 45°, 60°, 65°, 70°, and 75° as measured from surface normal. These five scans were averaged and plotted together in Figure 27.
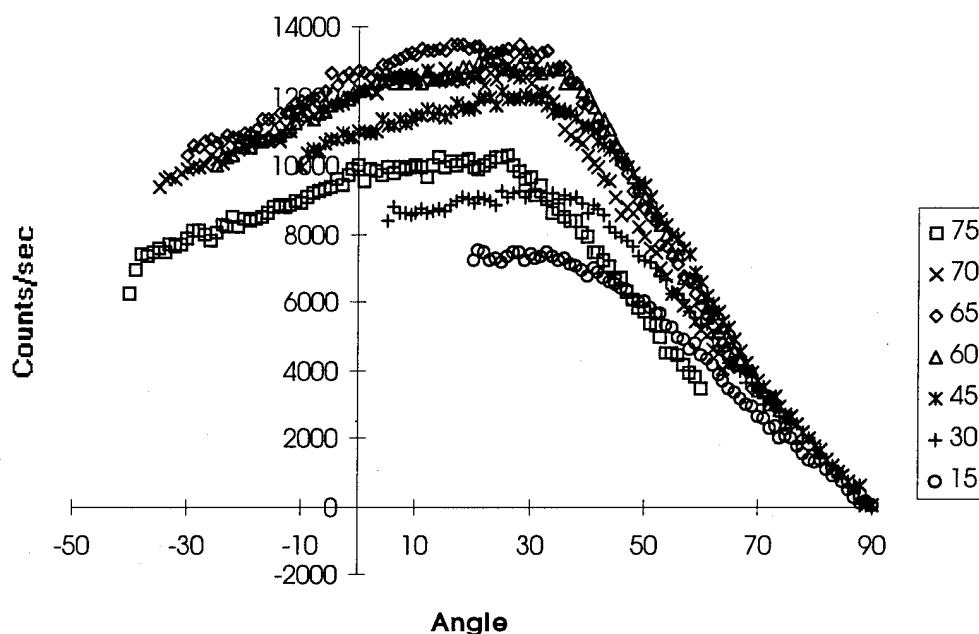
Figure 27. Ar$^+$ sputtering on Zr (raw data)

These curves were normalized and compared to normalized Ar$^+$ data taken by Whitaker and Li. The shape of the curves was consistent between the two experiments, indicating that no new experimental artifacts had been introduced with the changes to the apparatus.

Three basic parameters in the data can be evaluated: 1) the sputtering intensity at various incident ion angles, 2) the peak position of the sputtering distribution, and 3) the shape of the distribution curve. The latter two parameters can be further compared to predictions using Roosandaal Sanders Theory. From the curves in Figure 27, it can be seen that the sputtering yield using an Ar^+ beam reaches a maximum for a 65° incident ion beam. This result was obtained using similar input conditions and with all data recorded in one day, thus eliminating day to day variability in laser power and incident ion beam conditions.

Normalized Shape vs incident ion angle

Normalization of the sputtering intensity data was necessary before the data could be used to analyze the peak position of the distribution and the shape of the yield curve. All the curves in Figure 27 show evidence of a broad plateau occurring at their maxima. The normalization factor was determined by finding the center of the plateau in each curve and calculating a five point average at this point. The data in each curve were then divided by this normalization factor to form the curves in Figure 28.
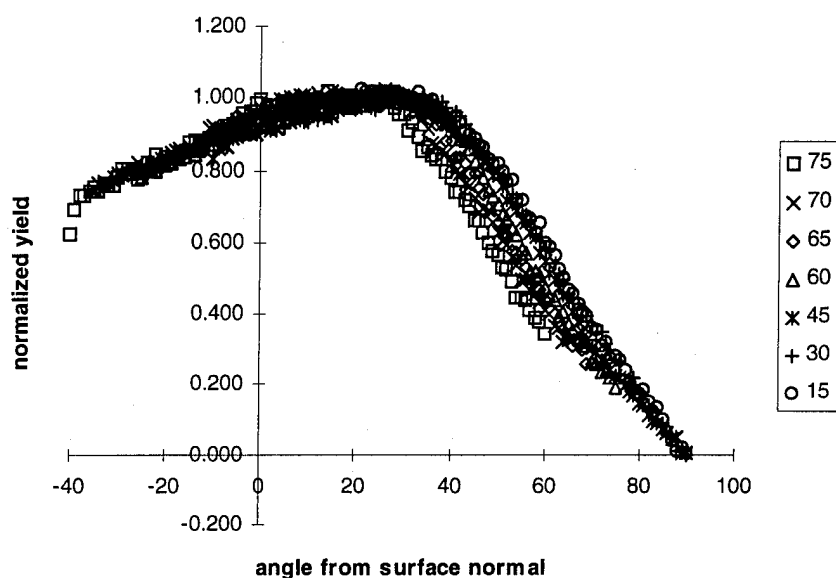
Figure 28. Normalized Ar⁺ sputtering curves.

Examining the normalized curves in Figure 28, two observations can be made: 1) all the curves show approximately the same distribution pattern from the peak position to the ion beam side of the graph (negative angle), and 2) the distribution becomes narrower as the incident ion beam moves from near normal directions, 15°, to glazing incidences, 75°. In other words, all curves show the same back sputtering behavior but less forward sputtering as the incident ion beam moves toward the surface.

Roosandaal Sanders fit to raw data

Following the approach of Whitaker and Li, the curves in Figure 27 were modeled using a nonlinear least squares fit to the modified Roosandaal

Sanders equation using B(E), n, and U as fitting parameters. Table 5 lists the fitting parameters and a χ^2 term indicating the goodness of the fit to the RS equation. As for most applications, a smaller χ^2 term is indicative of a better fit.

Table 5. Fitting parameters using Roosandaal-Sanders equation.

Angle	B(E)	n	U	(χ^2)/N-m
15	16,595	1.55	9.10	0.6330
30	17,032	1.49	9.42	1.4015
45	18,061	1.52	8.10	2.0116
60	15,954	1.53	5.02	3.2590
65	15,525	1.49	2.94	7.8948
70	13,889	1.47	1.78	5.4242
75	10,374	1.42	0.45	2.6606

The B(E) term is an amplitude correction factor and will not be examined in detail. The exponent, n, to the cosine term is approximately 1.5 in all fits. This is consistent with the work of other researchers[45,46,47,48] as well as the data collected by Whitaker and Li. The fitting parameter, U, decreases continuously (neglecting the 15° curve) as the incident ion beam moves toward the surface. The 15° curve is neglected because it is not apparent that there is enough data to fully determine that the peak has been reached. In the original Roosandaal Sanders equation, U corresponded to the surface binding energy, which is usually approximated using the sublimation energy

of the target (6.3 eV for zirconium). In this approach, using U as a fitting parameter clearly removes any relationship to the sublimation energy of the target.

The χ^2 term in Table 5 indicates that the modified RS equation is inadequate at fitting the raw data as the incident ion moves toward the surface. Curves comparing the raw data to the modified Roosandaal Sanders equation are presented in Figure 29-35, along with a compilation of the RS curves in Figure 36.
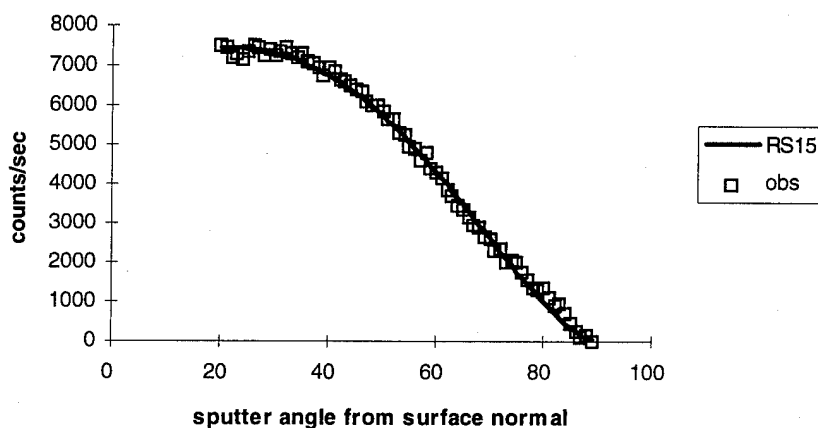
Figure 29. Roosandaal Sanders fit, 1.9 kev Ar$^+$ at 15° angle of incidence.
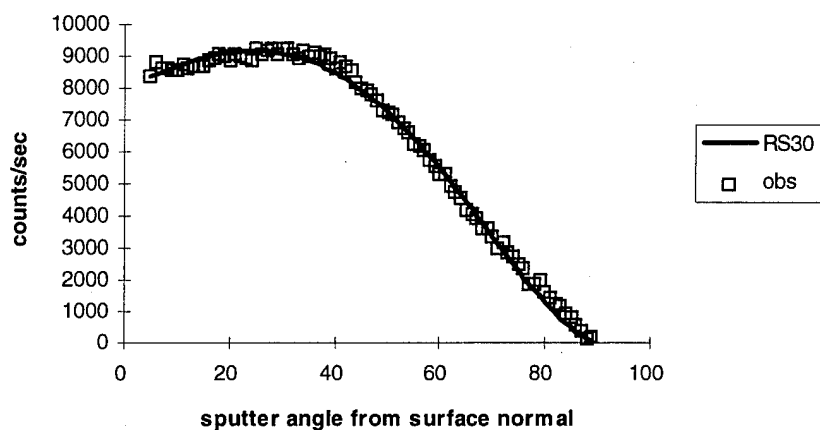
Figure 30. Roosandaal Sanders fit, 1.9 kev Ar$^+$ at 30° angle of incidence.
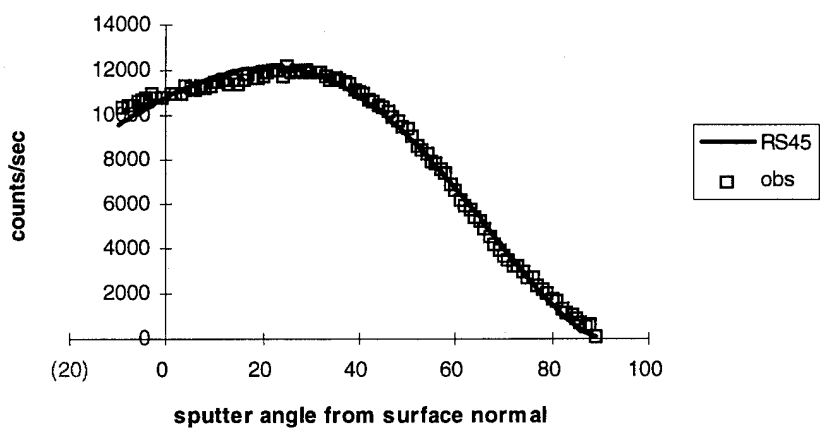
Figure 31. Roosandaal Sanders fit, 1.9 kev Ar$^+$ at 45° angle of incidence.
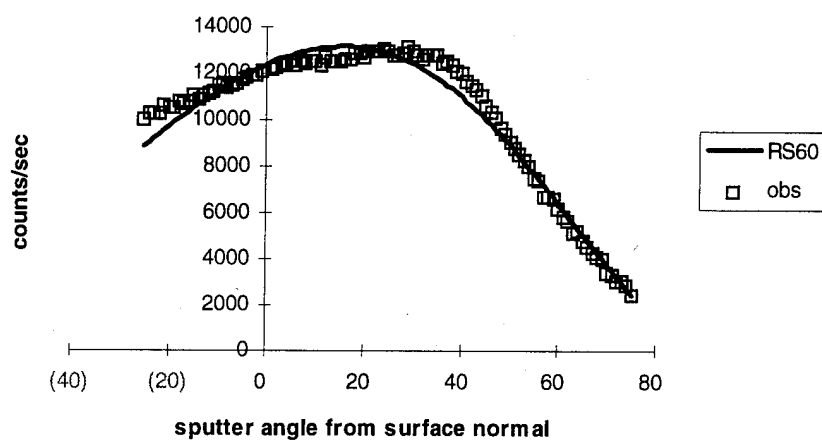
Figure 32. Roosandaal Sanders fit, 1.9 kev Ar$^+$ at 60° angle of incidence.
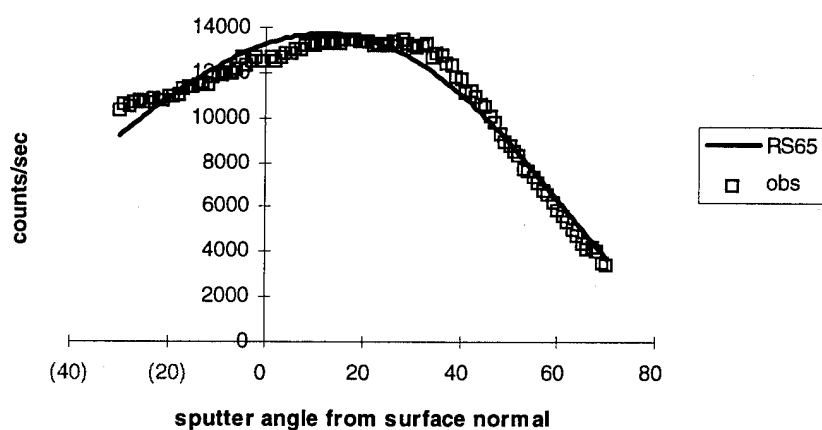
Figure 33. Roosandaal Sanders fit 1.9 kev Ar$^+$ at 65° angle of incidence.
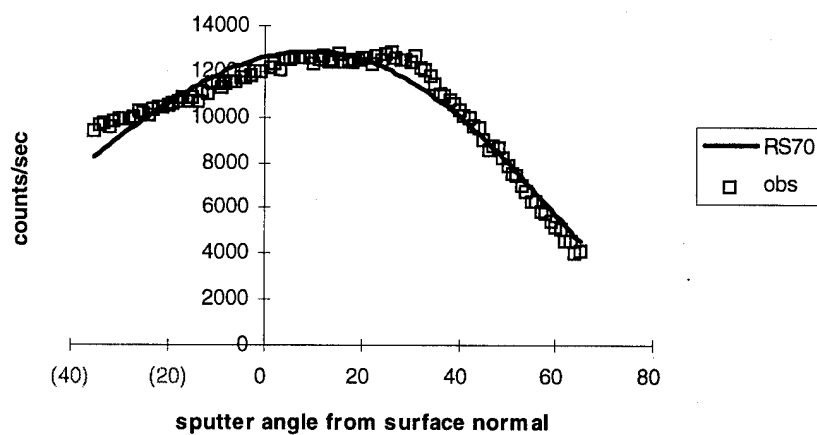
Figure 34. Roosandaal Sanders fit 1.9 kev Ar$^+$ at 70° angle of incidence.
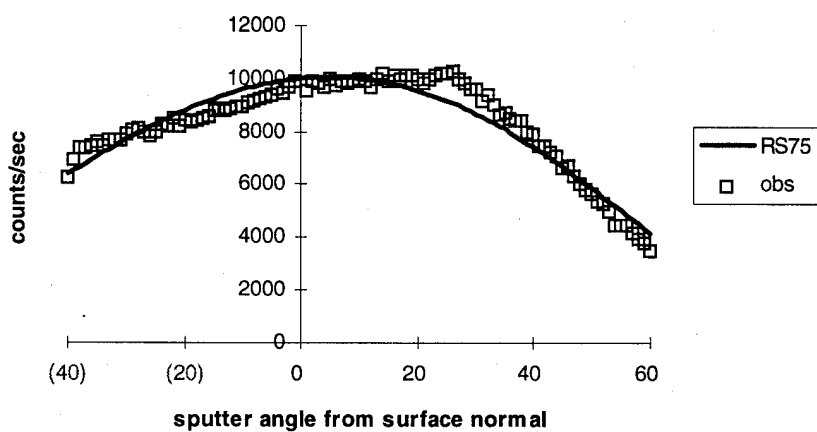
Figure 35. Roosandaal Sanders fit 1.9 kev Ar$^+$ at 75° angle of incidence.
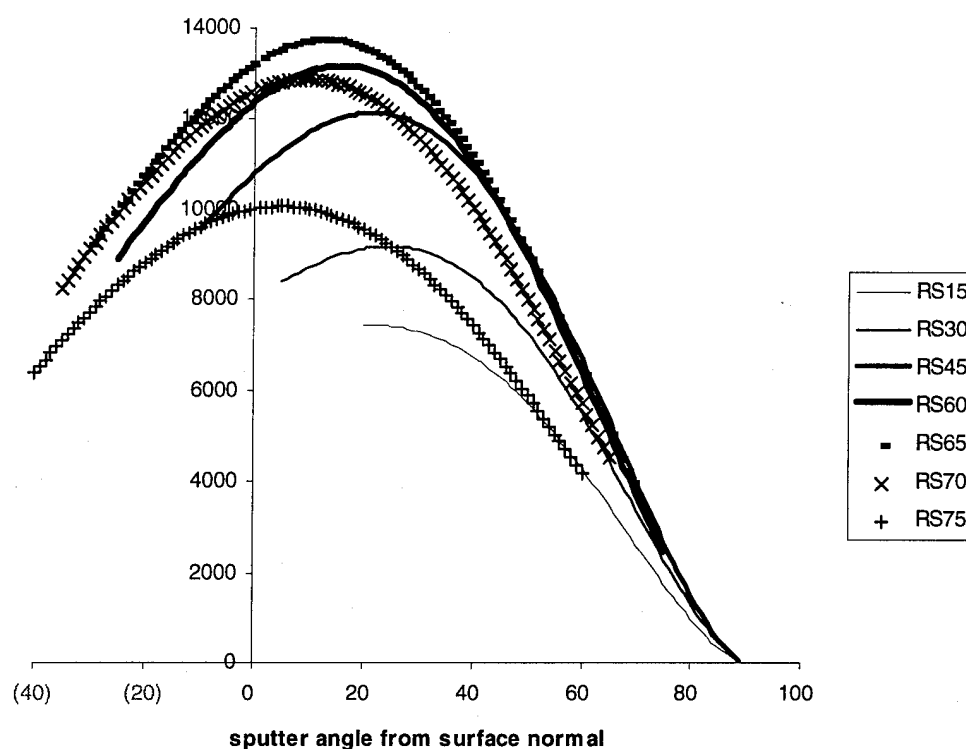
Figure 36. Roosandaal Sanders fits, 15 - 75° compilation.

Examining the curves in Figure 29-35, it is apparent that the modified Roosandaal-Sanders equation does not fit the data as the incident ion beam moves toward the surface. The modified Roosandaal Sanders equation was reexamined to determine the cause of the discrepancy in the fits.

Roosandaal Sanders Discrepancies - Peak Position

The first parameter investigated was the peak position of the distributions, which was determined: 1) by the center of the plateau indicated using the normalization procedure to produce the curves in Figure 28, and 2) by fitting the raw data in Figure 27 to the modified Roosandaal Sanders Equation. The results of this analysis are collected in Table 6.

Table 6. Peak position of Zr sputtered by Ar⁺.

Angle	Peak Position	
	Normal	RS
15	27	21
30	28	24
45	27	21
60	26	16
65	24	12
70	21	9
75	18	5

The data in Table 6 reveal that both procedures show a trend in the peak position of the distribution moving toward surface normal as the incident ion beam moves away from surface normal. The raw experimental data reveal a much slower regression to surface normal of the peak position than the RS data indicate; thus, it is not evident that as the ion beam moves toward the surface that the peak position will ever migrate toward surface

normal as drastically as indicated by the RS data. The behavior indicated by the RS fits is clearly not supported by the experimental data.

Roosandaal Sanders variation of U and incident ion angle

The input parameters to the modified Roosandaal Sanders equation were varied and plotted to determine what affect varying U and/or the incident ion beam angle would have on the peak position of the distribution. Using B(E) = 10,000, n = 1.5, incident ion beam = 30°, and varying U from 10 to .1 produces the curves in Figure 37. Keeping U constant and varying the incident ion beam angle produces the curves in Figure 38.
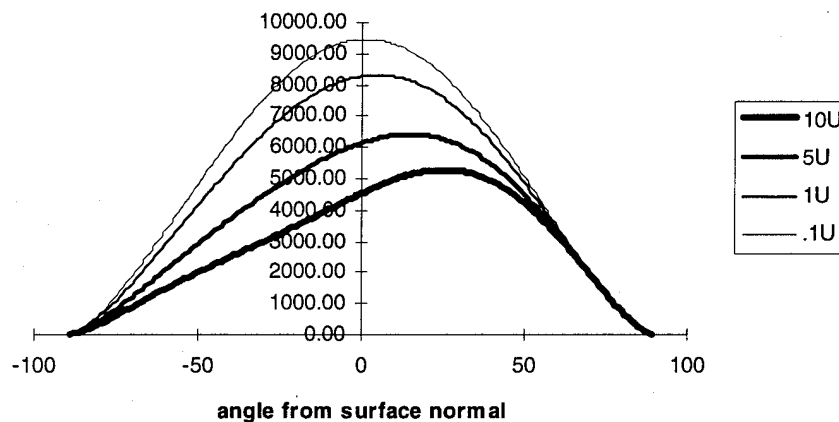
Figure 37. Roosandaal Sanders curves, varying U.
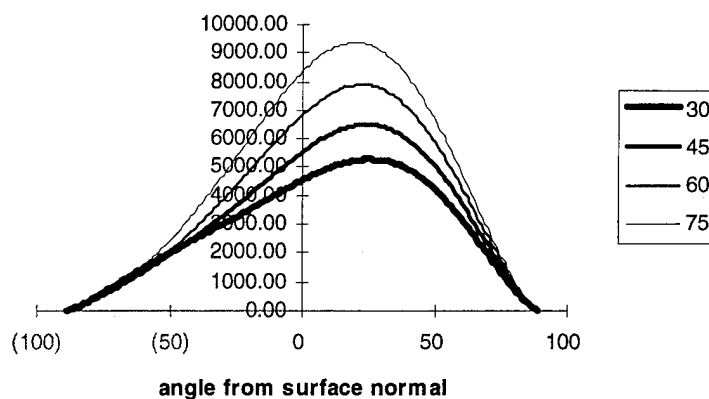
Figure 38. Roosandaal Sanders curves, varying incident ion angle.

Figure 37 provides visual confirmation that variation of the fitting parameter U causes a drastic shift in the peak position of the sputtering distribution. This shift is less dramatic in Figure 38, but varying the incidence angle also causes a slight shift in the peak position. The peak position versus the various fitting parameters is collected in Table 7.

Table 7. Peak position versus U and incident ion angle.

Parameter	10U	5U	1U	.1U	30°	45°	60°	75°
Peak Position	25	15	5	1	25	24	22	20

The data in Table 7 indicate that the large shift toward surface normal produced by the Roosandaal Sanders fit is an artifact of the Roosandaal Sanders equation. The large variation in the input parameter U causes an

artificially large shift in the peak distribution. Clearly, variation of U through two orders of magnitude is unrealistic and results in behavior not evident in experiments.

Roosandaal Sanders fits to forward sputtering

The raw data was reexamined to determine if another fitting routine could provide reasonable fits using physically meaningful parameters. Returning to the observations of the normalized curves in Figure 28, the data suggest two different sputtering behaviors. The back sputtering behavior appears to be independent of incidence angle whereas the forward sputtering shows a slight dependence on incident angle. There is no reason that the same sputtering mechanism is responsible for both behaviors, thus the modified Roosandaal Sanders equation was used to fit each part of the curve independently to see what trends would become evident. The peak position was chosen by the normalization procedure, then both back-sputtered and forward sputtered portions of the curve were fit using the modified Roosandaal Sanders equation. The parameters B(E), n, and U were allowed to vary independently in each portion of the curve. The fitting parameters and the χ^2 term are collected in Table 8. The curves produced using these parameters are plotted in Figure 39-45.

Table 8. Roosandaal Sanders fits to back and forward sputtering.

Angle	Forward			Backward			χ^2
	B(E)	n	U	B(E)	n	U	
15	19,239	1.67	12.22				0.8685
30	18,257	1.56	11.79	12,886	0.68	4.11	0.9158
45	19,492	1.64	13.29	14,855	0.67	3.57	1.0090
60	17,494	1.80	13.95	14,452	0.57	2.61	0.7458
65	17,199	1.97	17.49	14,539	0.54	2.44	0.8768
70	14,941	2.07	18.77	13,280	0.60	1.89	0.6224
75	11,285	2.31	19.18	10,376	0.42	2.33	0.1981
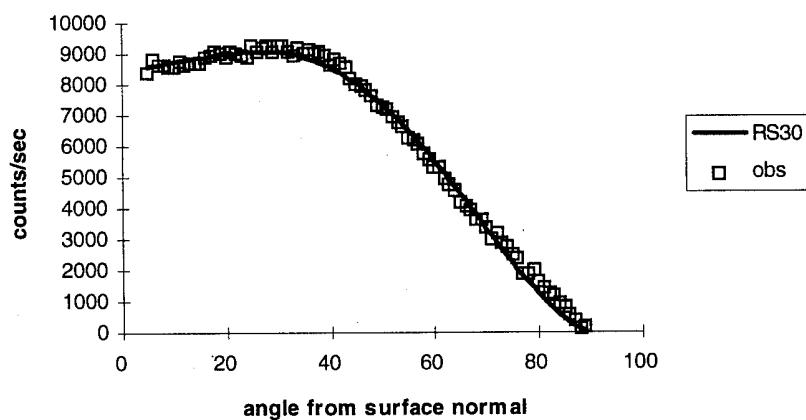
Figure 39. Modified RS fit, Ar^+ at $30°$ angle of incidence.
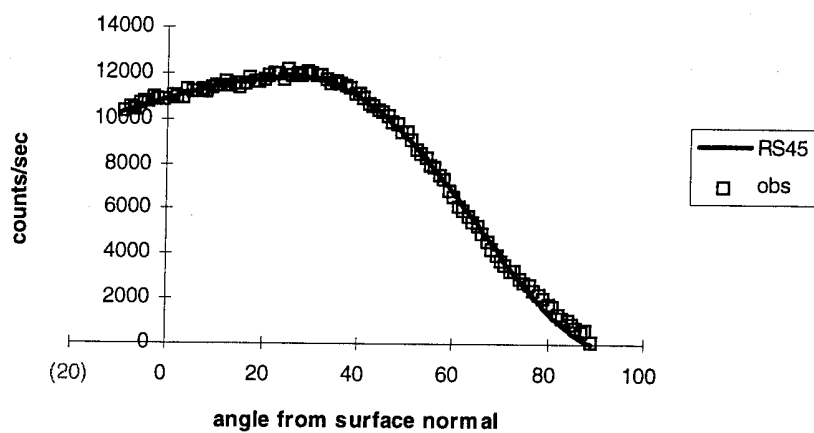
Figure 40. Modified RS fit, Ar^{+} at 45° angle of incidence.
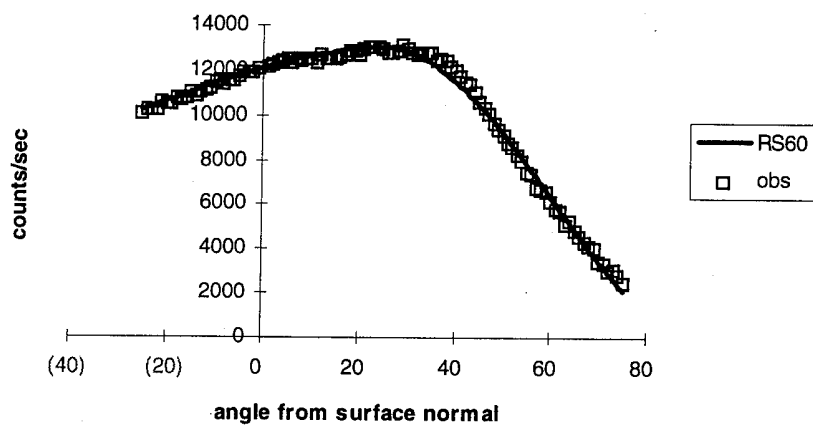
Figure 41. Modified RS fit, Ar^{+} at 60° angle of incidence.
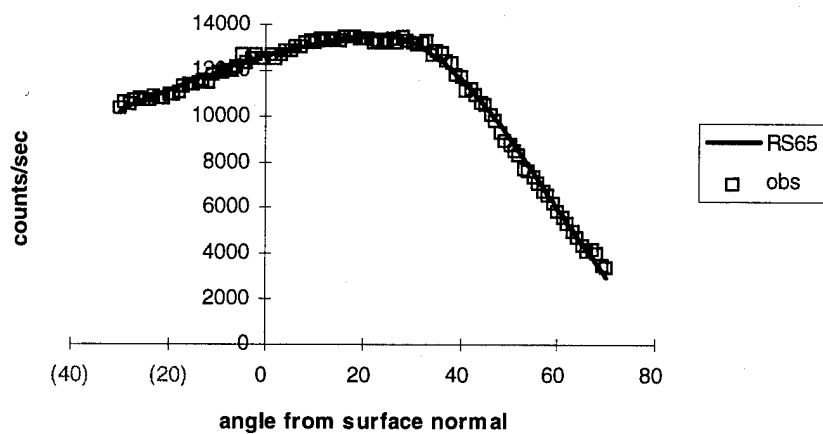
Figure 42. Modified RS fit, Ar$^+$ at 65° angle of incidence.
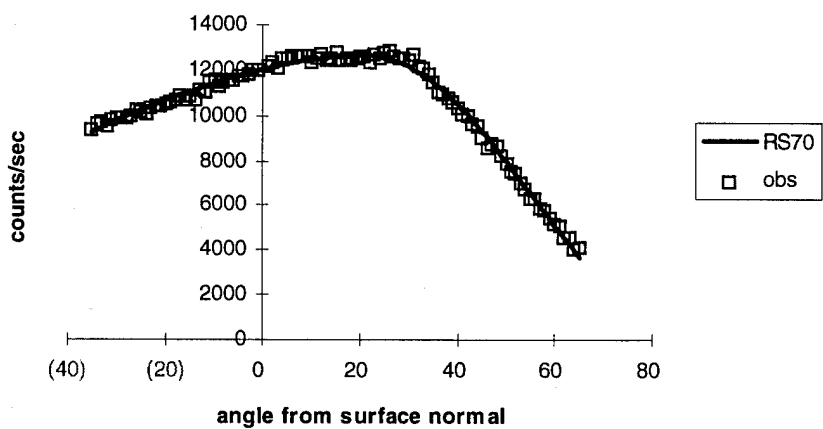
Figure 43. Modified RS fit, Ar$^+$ at 70° angle of incidence.
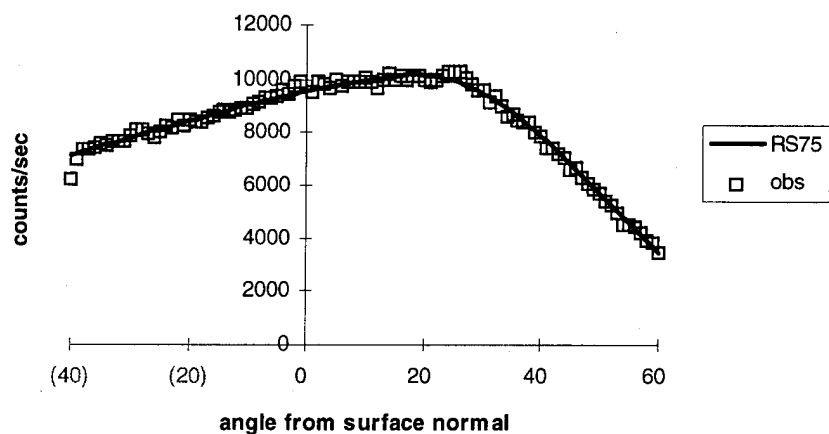
Figure 44. Modified RS fit, Ar$^+$ at 75° angle of incidence.
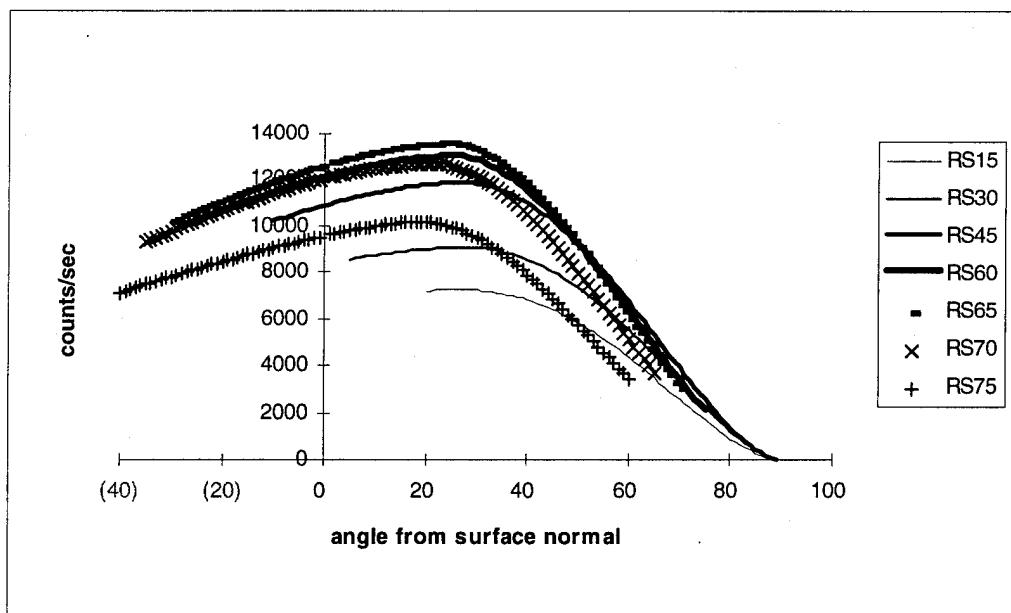
Figure 45. Modified RS Compilation, 15 - 75°.

The relatively small χ^2 terms in Table 8 and the curves in Figure 39-45 attest to the success of this fitting procedure. Whether the fitting parameters are physically meaningful will be addressed in the paragraphs below.

Analyzing the forward and backward sputtering independently shows some interesting trends. In the forward sputtering case, the parameter U increases steadily as the incident ion moves from near surface normal toward the surface. U varies from approximately 2 to 3 times the sublimation energy of zirconium. Garrison[49] et. al. addressed the question of how much energy would be required to remove an atom from a surface. Jackson[50,51] found that the energy loss using a pairwise additive potential approximation is 30% to 40% larger than the heat of sublimation. Using Jackson's results and experimental evidence, Garrison[52] et. al. argue that the energy requirement is much greater than the heat of sublimation. They use a model of a diatomic molecule with bond strength, D_e. The binding energy of each atom is then $D_e/2$, and if one atom of this molecule is clamped tightly, twice the binding energy is required to remove the other atom. This model breaks down upon closer examination as zero energy would then be required to remove the second atom. Regardless, Garrison et. al. argue that U should fall between one and two times the binding energy.

Whitaker fit Ar^+ data at $30°$, $45°$, and $60°$ to the modified Roosandaal Sanders equation and reported a corresponding 300% change in the value of U, similar to our results in Table 5. This large variation could not be explained using Garrison's model. Whitaker explains the discrepancy by invoking expansion of the lattice spacing of the top layer of surface atoms. He argues that the atoms near the surface are set into motion by the

collisional cascade and that the average lattice spacing can change significantly before an atom escapes the attractive forces of the surface. Therefore, the average lattice spacing at the time of ejection will be dependent on the bombardment conditions. A larger lattice spacing will decrease the interaction between an atom leaving the surface at a large angle to surface normal and its nearest neighbor, increasing the probability of ejection in these high angle directions. This behavior is essentially the same seen by increasing the surface binding energy, U. Additionally, a larger lattice spacing increases the probability of deeper atoms being sputtered. In computer simulations,[7] the sputtering distributions of second and third layer atoms has been shown to be highly over cosine due the increased number of collisions at high angles of incidence.

Whitaker's model is consistent with the forward sputtering behavior observed in the Ar^+ data from 30° to 75°. As the incident ion angle moves toward the surface, the interaction with the surface increases. This causes a steady increase in the lattice spacing of the top layer, simulating an increase in the surface binding energy. This increase in the lattice spacing also increases the contribution of second and third layer atoms that can be sputtered. As this contribution increases, the over cosine behavior should increase. This is supported by the increase in the exponent n in Table 8.

Roosandaal Sanders fitting to backward sputtering

The backward sputtering behavior cannot be explained using the same model used to describe forward sputtering behavior. The value of U decreases as the incident ion moves toward the surface and the value of n is

definitely less than 1, showing evidence of undercosine behavior. There are two possible mechanisms for the back sputtered behavior. If the incident ion penetrates deeply into the target, the momentum distribution should be completely randomized and the collision cascades produced should form an isotropic cosine distribution peaking at surface normal. This is clearly not the behavior seen in our experiments.

The second possible mechanism is caused by back scattering of the incident ion by the target atoms. In this scenario, the incident ion heads toward the surface, initiating collision cascades on its way to exiting the surface. The ion will preferentially exit in the surface normal position similar to the behavior of the forward sputtered target atoms. As the ion exits, it can interact with other target atoms, consequently ejecting them at high angles of incidence. This phenomenon would produce a sputtering distribution that would be distinctly undercosine in behavior. This is exactly the behavior observed in our experiments. The data in Table 8 show undercosine behavior in fits at all of the angles. The trend showing a decrease in the parameter U as the incident angle increases can also be explained with this model. As the incident angle increases, the depth of penetration into the target decreases. The back scattered ion would then have a smaller distance to travel before exiting the surface, that is, its interaction time with the surface layers would be less. Thus, as the interaction with the surface decreases, the parameter U, which is the only parameter incorporating surface effects, also decreases.

Variation in n and U allowed by fitting routine

The values of n and U should not be considered absolutes. The parameters reported in Table 8 are for the best fit conditions, but several combinations would give reasonable fits to the data as evidenced Figure 46.
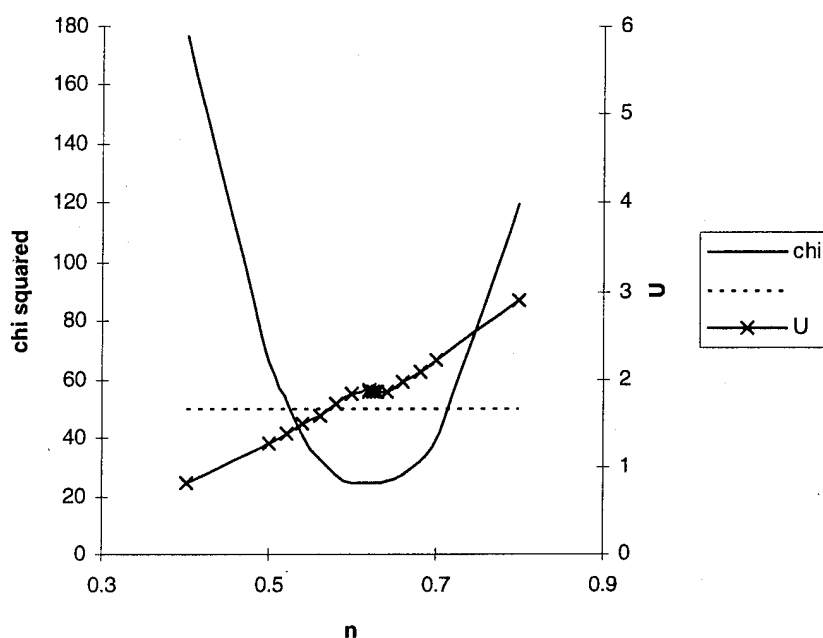
Figure 46. Chi vs. n for 70° Ar^{+} on Zr.

Choosing an arbitrary value of 50 for a maximum chi squared value allows a variety of n's. Since the n and U parameters are linked, they cannot be examined independently. They can vary between $0.52 <n< 0.70$, $1.38 <U< 2.22$ and still give reasonable fits to the modified RS equation. The n parameter consistently shows under cosine behavior at all angles and the U parameter shows a decreasing range as the ion incident angle increases, so

the variability in the n and U parameters does not affect the conclusions drawn from the fitting routine.

The above fitting procedure of treating independently the forward and backward sputtering was applied to Ne^+ and Xe^+ ground and first excited state data available from Whitaker. The results are presented in Table 9. The Ne^+ ground state results are consistent with the Ar^+ data described above. The first excited state results indicate a much sharper peaking (i.e., larger value of n) than seen in the ground state Ne^+ results. This sharper peaking occurs in both the forward and backward sputtering cases, and can be explained by the relaxation of the excited state species upon interaction with the surface along angles close to the surface. This interaction would reduce the near surface population of sputtered excited state atoms, resulting in a more pronounced peaking toward surface normal in the data. The values of U also are larger in the first excited state case than in the ground state case. As we have postulated U to be a surface interaction term, the larger values of U are consistent with the picture of an increased interaction with the surface. However, the larger values of n and U could be an artifact of the fitting routine because their is no mechanism in the RS model for the removal of sputtered atoms from the distribution due to the relaxation of the excited state species by the surface. The smaller yield of excited state species along trajectories close to the surface is compensated(wrongly) in the model as an increase in the n and U parameters.

Table 9. Modified RS fit to Ne$^+$ and Xe$^+$.

Ne,gnd Angle	Forward B(E)	Forward n	Forward U	Backward B(E)	Backward n	Backward U	χ^2
30	1042	1.76	6.23				0.296
45	816	1.57	10.39	749	-0.20	2.24	0.192
60	1212	2.05	15.32	938	0.80	1.61	0.227
75	890	2.21	16.77	821	0.21	4.86	0.134
Ne, 1x	Forward			Backward			
30	1169	1.94	7.63				0.369
45	868	1.94	23.62	860	1.36	4.71	0.209
60	1675	2.52	25.37	1287	0.99	4.23	0.312
75	1119	2.97	45.97	1040	0.91	3.98	0.486
Xe, gnd	Forward			Backward			
30	3376	1.76	16.57				0.470
45	2620	1.82	49.58	2254	-1.36	11.81	0.533
60	4900	2.01	31.81	4112	0.27	15.50	0.202
75	1876	1.69	15.27	1912	-1.68	29.18	0.268
Xe, 1x	Forward			Backward			
30	2771	1.97	21.16				0.266
45	1919	2.09	119.29	1773	-0.16	26.73	0.358
60	3590	2.74	76.79	2520	0.24	23.31	0.277
75	1977	1.89	5.24	1708	-1.30	37.27	0.347

The Xe^+ ground state behavior shows a much sharper peaking and much larger U terms than in either the Ar^+ or Ne^+ data. This result can be explained by the larger mass and size of the Xe^+ ions, that is, by the fact that they will not penetrate as far into the target. Thus, there will be a greater interaction with the surface, which is represented by the increase in the U terms. The larger values of U are representative of a larger lattice spacing, so the n terms should also increase as the sputtered atoms preferentially eject in the surface normal direction. The forward sputtering behavior is consistent with this picture. The back sputtered behavior shows extreme undercosine behavior. The back scattered incident ion will be very near the surface due to the limited penetration of the ion. The top lattice spacing will have expanded considerably due to the ion surface interaction as the ion penetrated the target. On its exit, the ion will have a much more open lattice to penetrate and will exit preferentially in the normal direction. The predominant collisions with surface Zr atoms will occur at glancing angles with the Zr atoms ejecting close to the surface. This behavior will produce the severe undercosine behavior evident in the data. The excited state data show the same increase in the peaking of the distribution toward surface normal as in the Ne^+ data. The relaxation of the near surface excited state atoms by interaction with the surface is consistent with these data. The larger U terms are consistent with the increased size/mass of the incident ion resulting in greater surface-ion interaction.

Chapter 4. Nitrogen Sputtering on Zirconium

Nitrogen absorbates on Zirconium

When using noble gas ion sources, the sputtering intensity does not modulate over time as long as a clean surface is maintained. The noble gas ions do not appear to stick to the surface and/or modulate the sputtering signal measured. Over many hours of experiments, the signal intensity does decrease, but this behavior is attributed to the contamination of the surface by O_2, CO_2, and N_2 from the residual gas atmosphere of the sputtering chamber. These observations are verified by Auger experiments indicating an increase in the C, O, and N peaks after long hours of experimentation. As noted earlier, this effect was minimized by periodically flash heating the sample to 800° C to eliminate the surface contaminants.

Using nitrogen as the sputter source caused some concern about signal degradation due to adsorbed nitrogen on the surface. This issue was addressed by conducting coverage experiments on zirconium. Nitrogen sputtering on zirconium was accomplished holding the sputtering angle constant and measuring the sputtered zirconium intensity at a fixed angle over a long period of time. A graph of the results obtained is shown in Figure 47.
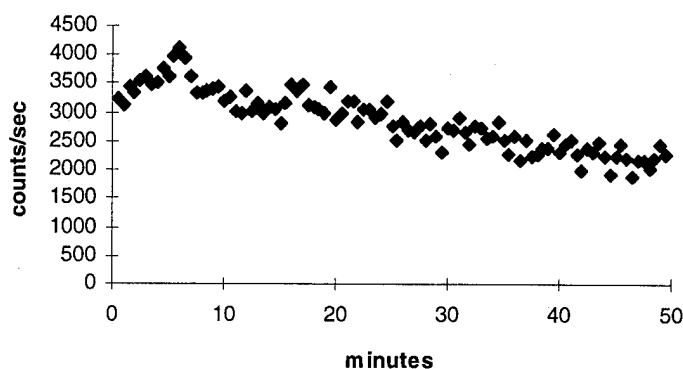
Figure 47. N_2^+ **sputtering on Zirconium at** 60° **incidence.**

The sputtered intensity holds relatively constant for approximately 20 minutes and then steadily decreases. Thus all scans were timed and any that exceeded 20 minutes in duration were eliminated from analysis. The sample was flash heated between scans and scans taken in both forward and backward directions to further minimize any effect absorbed species could have on the sputtering distribution.

N_2^+ ground state sputtering on Zirconium

Ground state scans were accomplished at 30°, 45°, 60°, and 75° using a 2 degree resolution to insure all scans were completed in under 20 minutes. A minimum of 15 scans were recorded and averaged at each angle. These scans were recorded over several days. The laser input power and ion current measured on the sample were recorded to insure similar sputter conditions

were maintained. The raw data are plotted in Figure 48.   The sputtering intensity reaches a maximum for a 60° degree incident ion beam. The curves in Figure 48 were normalized by finding the middle of each plateau and calculating a 3 pt (5 deg) average at this point to use as a normalization factor. The plateau widths and peak positions are present in Table 10 and the normalized curves are presented in Figure 49.
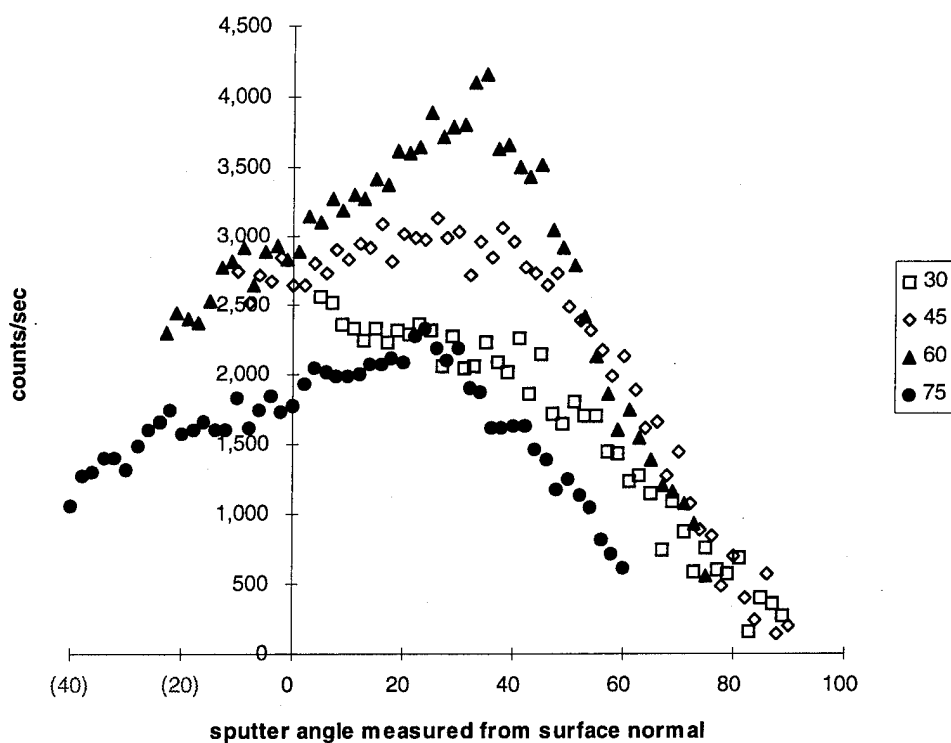
Figure 48. N_2^+ ground state sputtering on Zr (raw data).

Table 10. Plateau width and peak position for N_2^+ on Zr.

Angle	30	45	60	75
Plateau Width	20	32	10	12
Peak	15	24	30	24
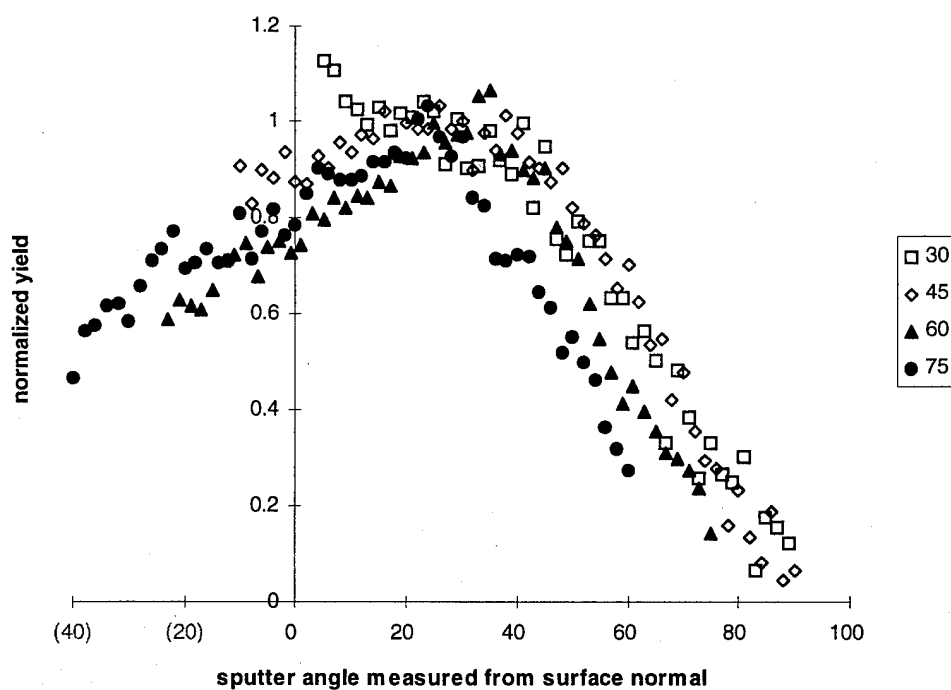
Figure 49. Normalized N_2^+ sputtering curves.

Examining the data in Table 10 and Figure 49, several observations can be made. The peak plateau is broadest at incident angles nearest surface normal, then narrows significantly as the incident angle moves toward the surface. The peak position moves away from surface normal as the incident angle increases to $60°$ and then returns at $75°$. Additionally, a slight narrowing of the overall shape of the distribution is observed as the incident angle moves toward the surface.

The peaks in Figure 48 were fit using the modified Roosandaal Sanders equation using the same method as for Ar^+. The curves are shown in Figure 50-53 and the fitting parameters are collected in Table 11.
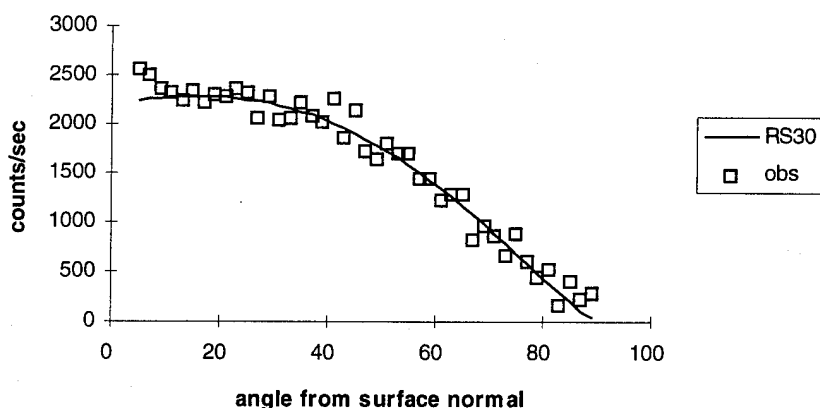
Figure 50. N_2^+ sputtering on Zr at $30°$ incident.
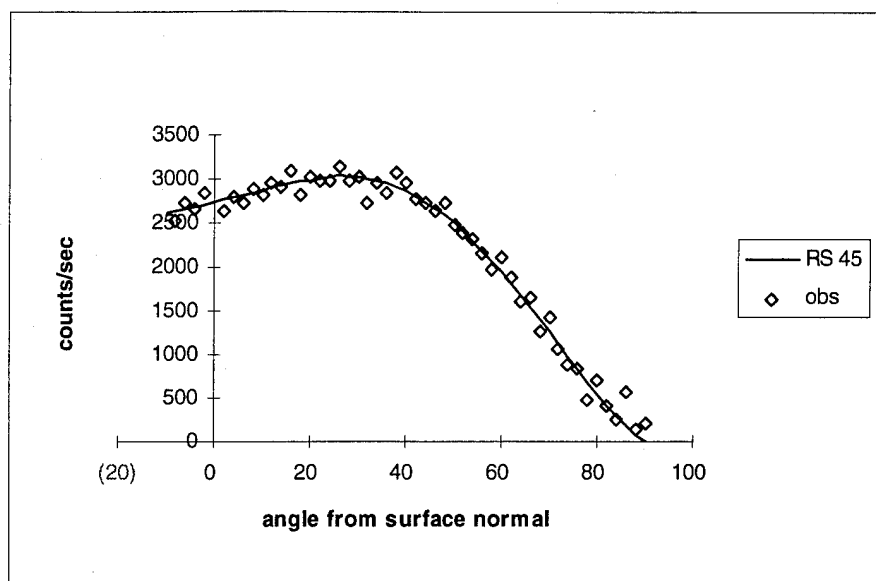
Figure 51. N_2^+ sputtering on Zr at 45° incident.
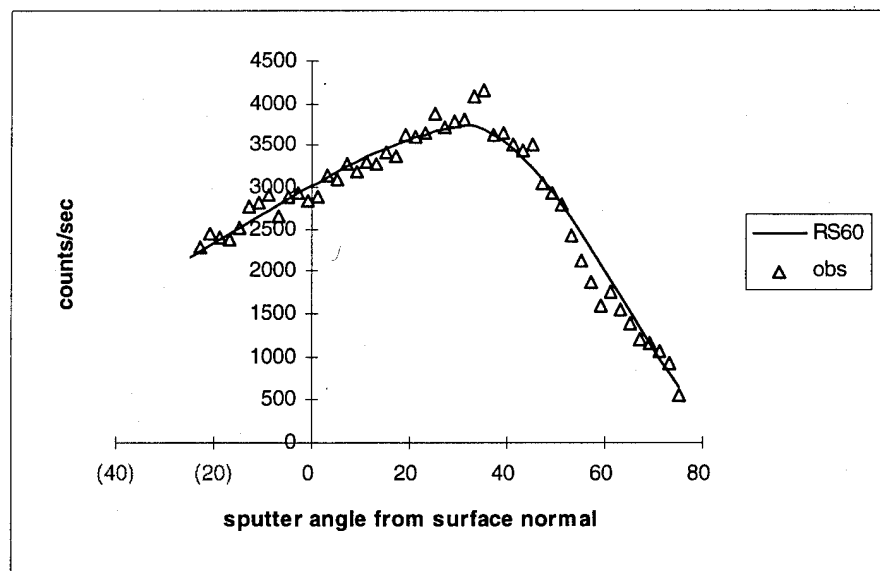
Figure 52. N_2^+ sputtering on Zr at 60° incident.

RS fit 2kev N2+ at 75 gnd
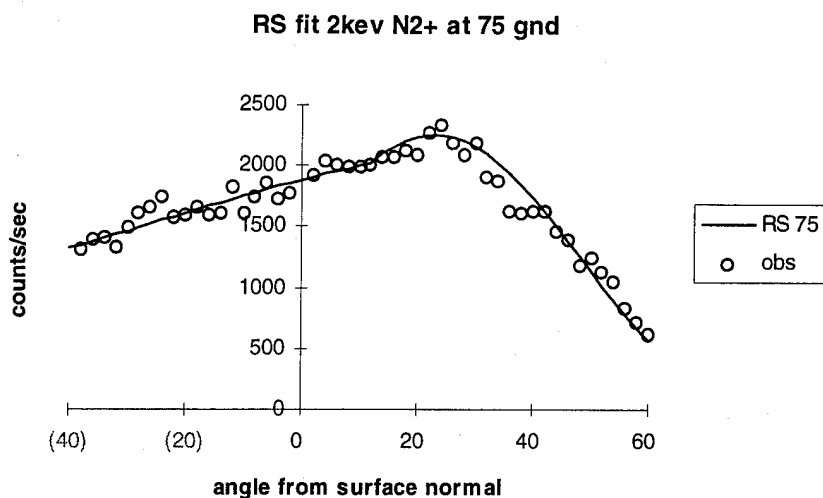
Figure 53. N_2^+ sputtering on Zr at 75° incident.

Table 11. Roosandaal Sanders best fit parameters for N_2^+.

Angle	forward			Backward			
	B(E)	n	U	B(E)	n	U	Chi
30	3325	1.17	3.47				1.21
45	4566	1.31	9.02	3603	0.21	2.58	1.16
60	5359	1.92	32.04	4142	0.36	7.06	1.44
75	2524	3.21	76.88	2103	0.36	3.71	1.99

Examining the data in Figure 50-53 and the data in Table 11, similar patterns to the Ne^+ and Ar^+ data emerge. The forward sputtering behavior shows an increasing interaction (larger U) with the surface as well as a sharper peaking

(increasing n) as the incident ion moves toward the surface. The back sputtering behavior shows consistent undercosine behavior and smaller surface interaction terms. The chi terms are larger than the Ar^+ data due to greater scatter in the data. The increased scatter in the data is caused by the lower sputtering intensity due to the smaller mass of N_2^+. The U terms for N_2^+ are generally larger in magnitude than for Ne^+ at comparable angles. This may result from the N_2^+ ion splitting into two N ions upon impact with the surface. The two N ions distribute the energy to the target more efficiently than a single Ne^+ atom causing a corresponding larger value of U.

The data in Figure 49 show a narrowing of the sputtering distribution of the 60° incident angle. This behavior is also evident in the increase in the n term in the back sputtering fit to the modified Roosandaal Sanders curves. A second series of experiments were conducted at 55°, 60°, and 65° to determine if this was a reproducible phenomena. The data were collected under similar conditions as the data in Figure 49 except using an ion current of 200 nA as measured on the sample as opposed to the 380 nA used previously. The raw data are plotted in Figure 54. The normalized data using our standard procedure is plotted in Figure 55 along with the normalized data from the first series of runs at 45° and 60°.
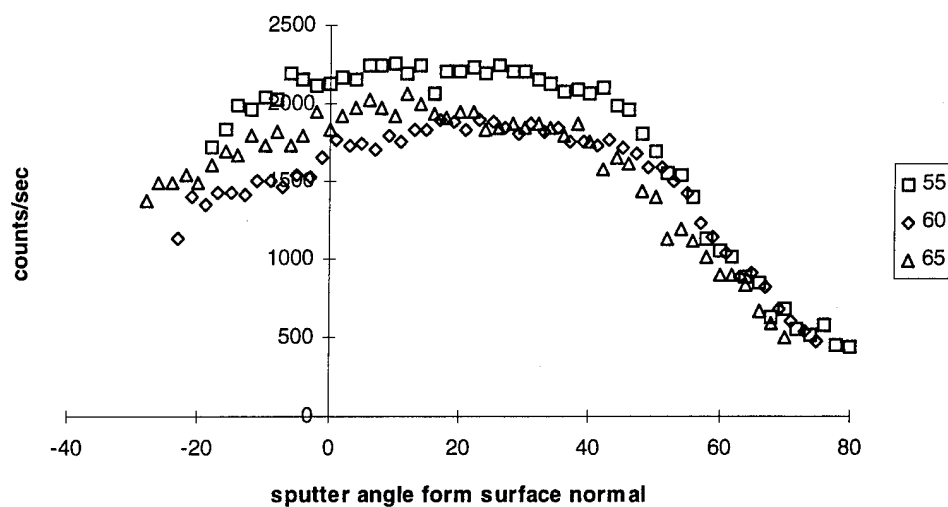
Figure 54. N_2^+ **sputtering on Zr (raw data).**
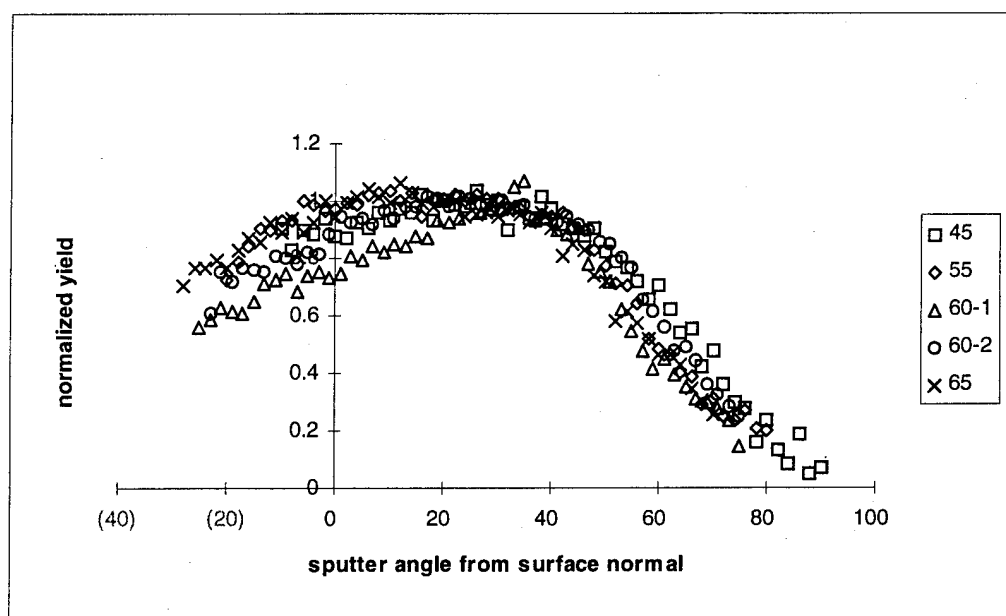
Figure 55. N_2^+ **sputtering on Zr (normalized data).**

As can be seen in Figure 55, there is a less dramatic, but still noticeable narrowing of the 60° curve obtained from the second series of scans. The narrowing of the 60° peak is within the resolution of the scans and is reproducible in all the scans. At this time, we cannot suggest a consistent mechanism to explain this preferential narrowing of the back sputtered behavior of the 60° sputtering peak. The best fit parameters to the modified Roosandaal Sanders equation are shown in Table 12.

Table 12. Roosandaal Sanders fits to curves in Figure 54.

Angle	forward			Backward			
	B(E)	n	U	B(E)	n	U	Chi
55	2981	1.69	16.19	2510	0.83	2.01	1.39
60	2504	1.57	22.11	2235	0.92	6.05	0.72
65	2626	1.89	19.88	2230	0.93	2.50	0.53

The trends in the values of U and n for the second series of scans are consistent with those seen for the first series of scans. The backward sputtering is more peaked than seen in the earlier data. The data in Table 12 must not be taken as absolute but indicative of trends. At all three angles, reasonable fits to the backward sputtering could be obtained with $0.7 < n < 1.2$, $1.5 < U < 7.0$.

First excited state scans of N_2^+ were also accomplished at 30°, 45°, 60°, and 75° using a 2 degree resolution to insure all scans were completed in

under 20 minutes. A minimum of 15 scans were recorded and averaged at each angle. These scans were recorded over several days. The laser input power and ion current measured on the sample were recorded to insure similar sputter conditions were maintained. The raw data are plotted in Figure 56. The sputtering intensity reaches a maximum for a 60° degree incident ion beam. The curves in Figure 56 were normalized by finding the middle of each plateau and calculating a 3 pt (5 deg) average at this point to use as a normalization factor. The plateau widths and peak positions are presented in Table 13 and the normalized curves are presented in Figure 57.
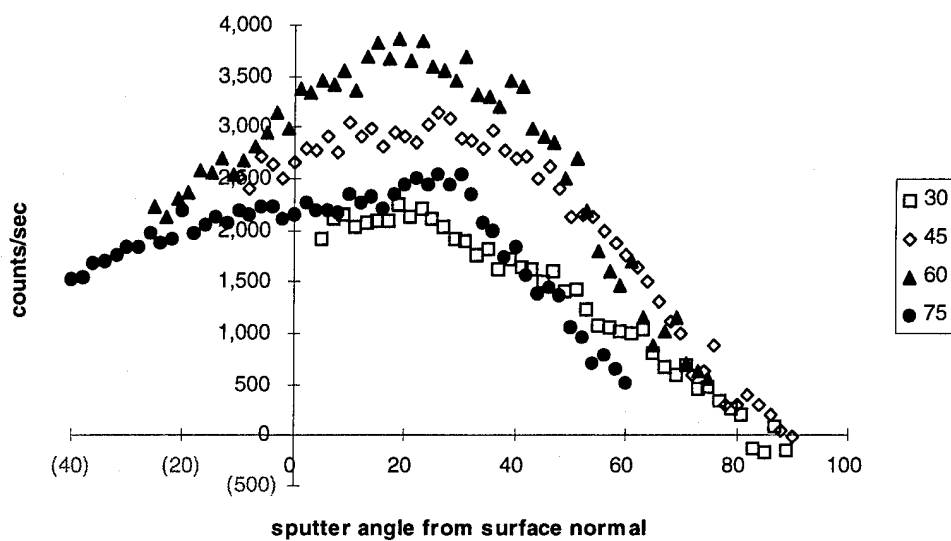
Figure 56. N_2^+ 1x sputtering on Zr.

Table 13. Peak width and Position for N_2^+ 1x on Zr.

Angle	30	45	60	75
Plateau Width	20	30	22	22
Peak	17	21	20	21
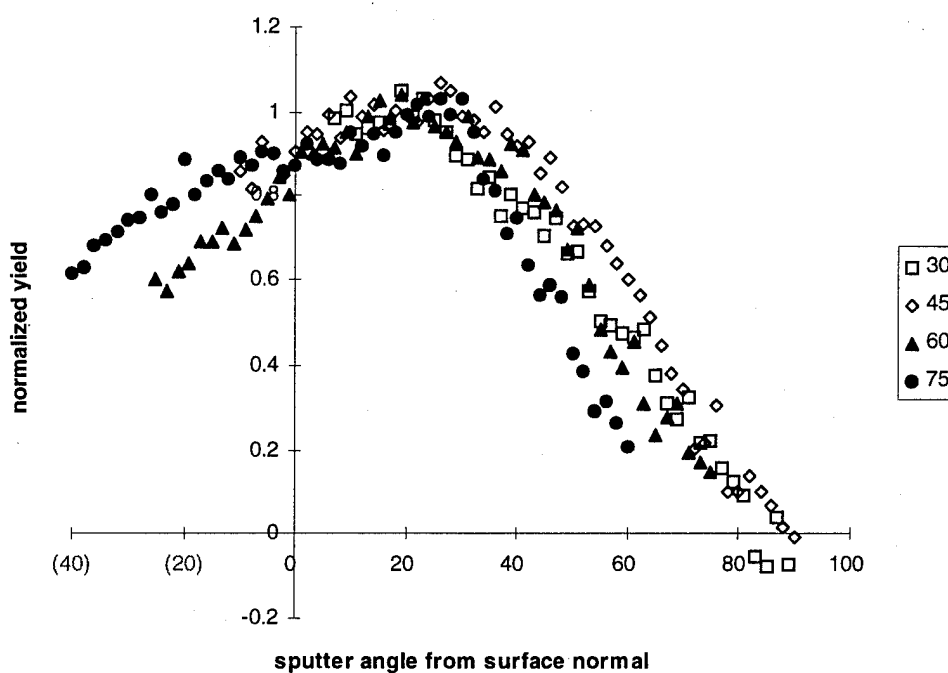
Figure 57. N_2^+ 1x normalized yield.

The data in Figure 56 and Figure 57 show similar behavior to the ground state data on N_2^+. The peak widths are broader at the higher incident angles than in the ground state and the peak position stays relatively constant. The peaks in Figure 56 were fit with the modified RS equation and

are presented in Figure 58-61. The fitting parameters are collected in Table 14.
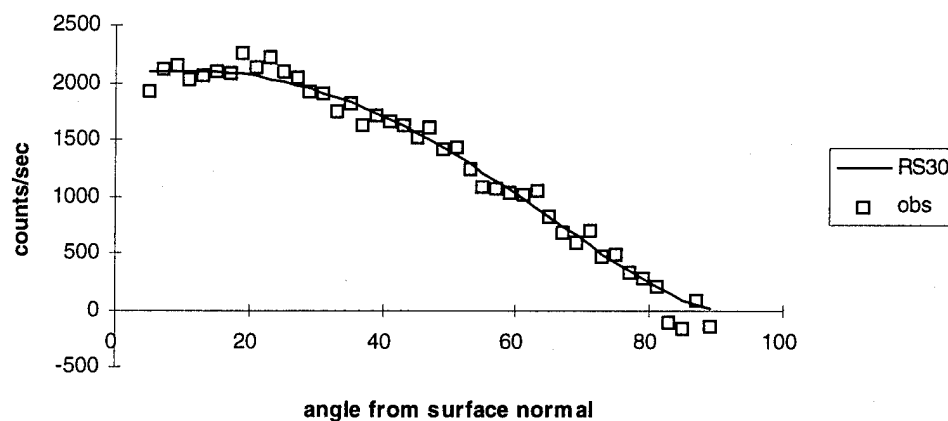
Figure 58. N_2^+ 1x modified RS fit at 30° incidence.
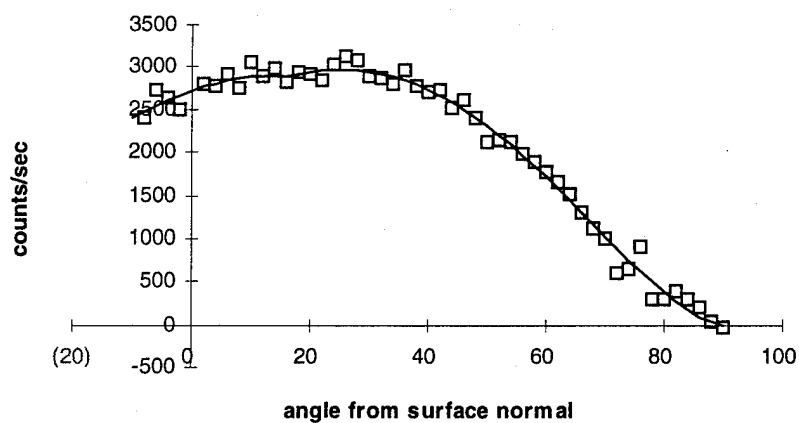
Figure 59. N_2^+ 1x modified RS fit at 45° incidence.
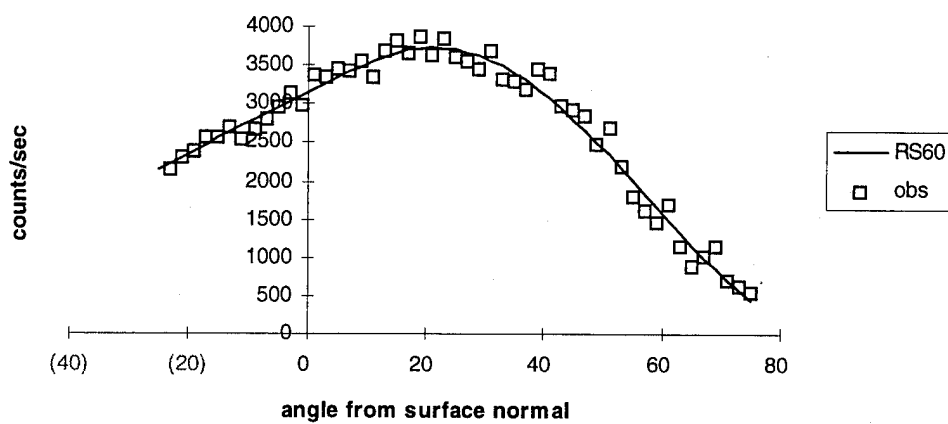
Figure 60. N_2^+ 1x modified RS fit at 60° incidence.
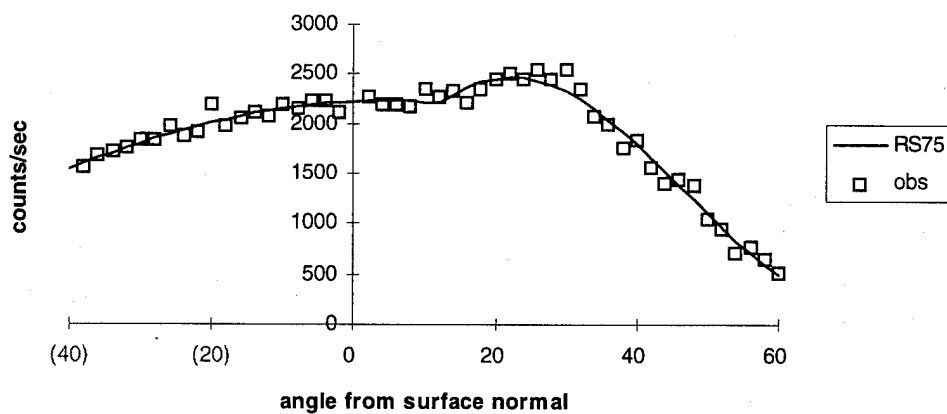
Figure 61. N_2^+ 1x modified RS fit at 75° incidence.

Table 14. Modified RS best fit parameters for N_2^+ 1x.

Angle	Forward			Backward			
	B(E)	n	U	B(E)	n	U	Chi
30	2934	1.45	2.62				0.82
45	4673	1.54	9.97	4335	2.26	6.22	0.89
60	5059	2.02	13.23	4596	0.88	8.76	1.06
75	2819	3.61	88.14	2281	1.18	0.21	1.34

The data in Table 13 and Table 14 support the mechanism of preferential sputtering in the normal direction due to quenching of the near surface excited state atoms by interactions with the surface. The back sputtering behavior at $45°$ shows a much larger n term than expected. This term is calculated with the fewest data points for the backward sputtering cases and thus can have large errors. The terms for the $60°$ and $75°$ angles use significantly more data and are considered more representative of the behavior occurring. Two nearly distinct curves can be seen in the $75°$ fit. The N_2 molecule breaks in two upon impacting the surface and each resulting N atom carries a portion of the incident ion energy. These atoms cannot transfer this energy to the surface atoms as efficiently due to their low mass. Thus the majority of sputtered atoms will exit with lower energy than in the case of Ne, Ar, or Xe. The sputtered atoms traveling close to the surface will have more time to interact with the surface and be quenched. This phenomenon could cause the unexpectedly pronounced peaking of the distribution.

N$^+$ sputtering on Zirconium

Ground state scans using N$^+$ were accomplished at 30°, 45°, 60°, and
75° using a 2 degree resolution to insure all scans were completed in under 20
minutes. A minimum of 30 scans were recorded and averaged at each angle.
These scans were recorded over several days and the laser input power and
ion current measured on the sample were recorded to insure similar sputter
conditions were maintained. The results are plotted in Figure 62. As for N$_2^+$,
the sputtering intensity reaches a maximum for a 60° degree incident ion
beam. The curves in Figure 62 were normalized by finding the middle of
each plateau and calculating a 3 pt (5 deg) average at this point to use as a
normalization factor. The peak positions are present in Table 15 and the
normalized curves are presented in Figure 63.
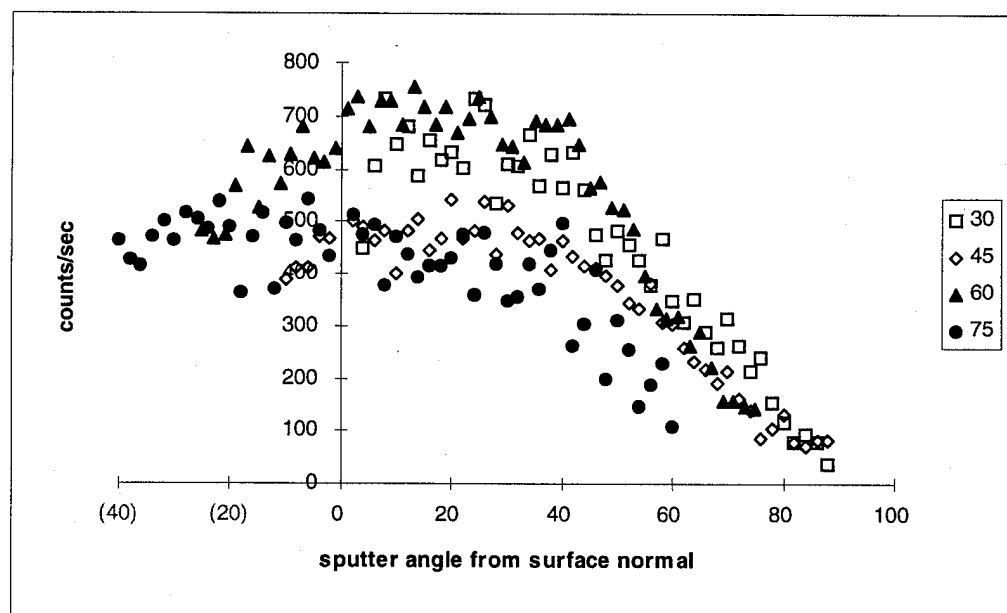
Figure 62. N$^+$ ground state sputtering of Zr (raw data).
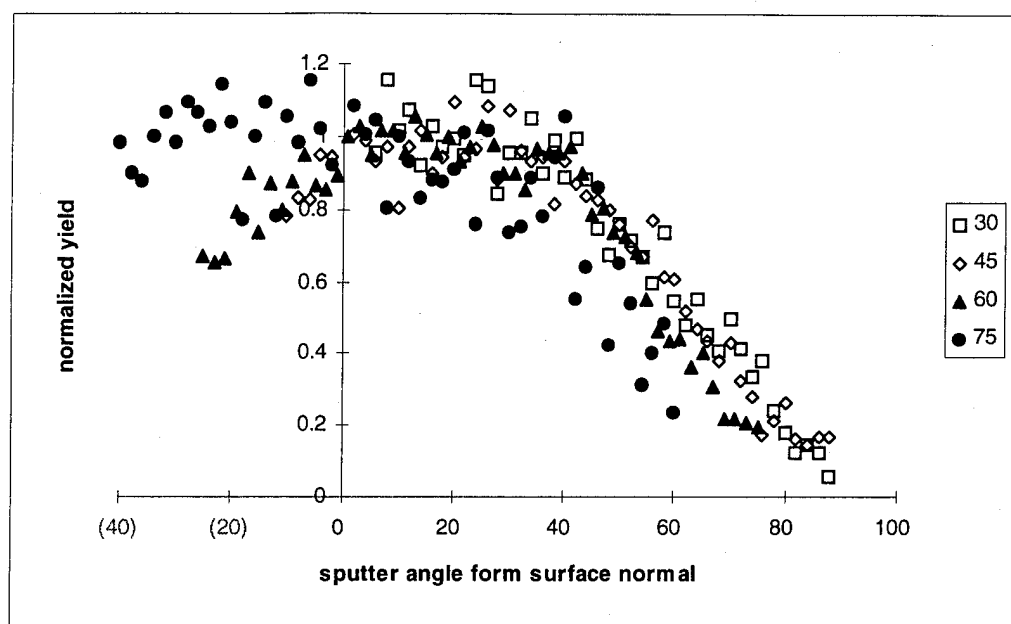
Figure 63. N$^+$ normalized yield.

Table 15. Peak positions of N$^+$.

Angle	30	45	60	75
Peak	24	26	13	-6

The data in Figure 63 show a narrowing in the forward sputtering yield as the incident ion moves toward the surface. A broad plateau is also evident on the 75° incident ion where the back sputtering behavior appears to reach a constant. The 75° curve has the greatest interaction with the surface and the greatest scatter in the data in our experiments. The curves in Figure 62 were fit using the modified Roosandaal Sanders equation. The results are presented in Figure 64 - 67 and Table 16.
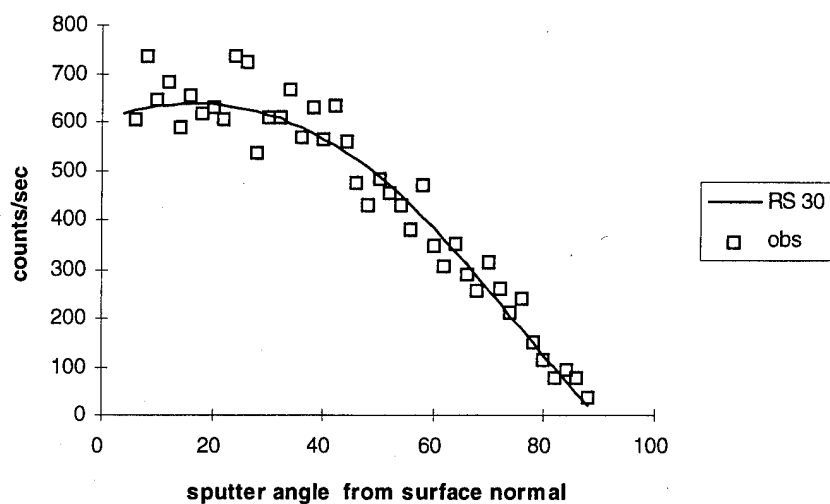
Figure 64. Modified RS fit to N$^+$ on Zr at 30° incidence.
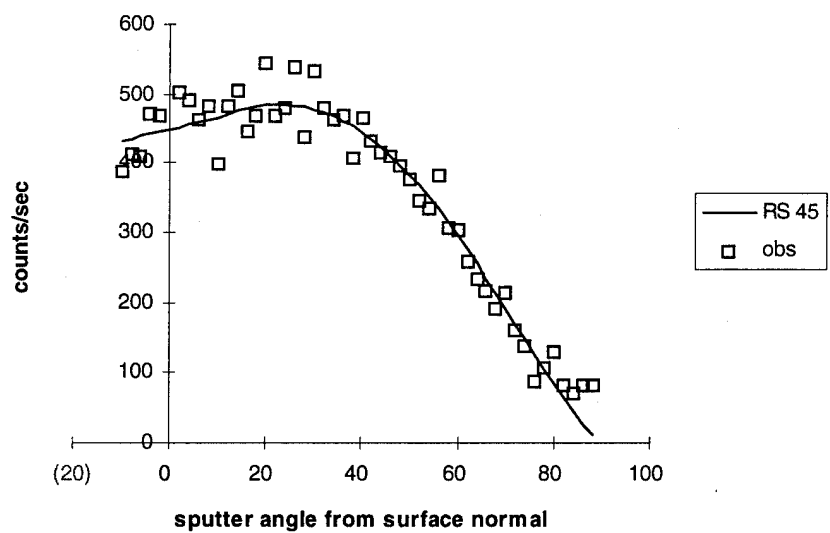
Figure 65. Modified RS fit to N$^+$ on Zr at 45° incidence.
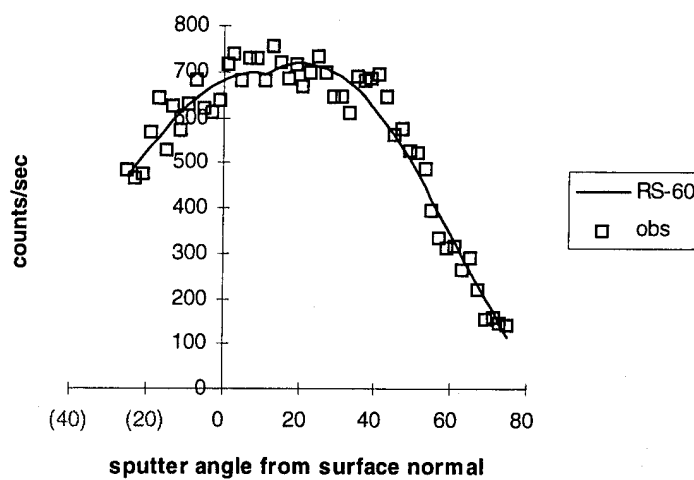
Figure 66. Modified RS fit to N$^+$ on Zr at 60° incidence.
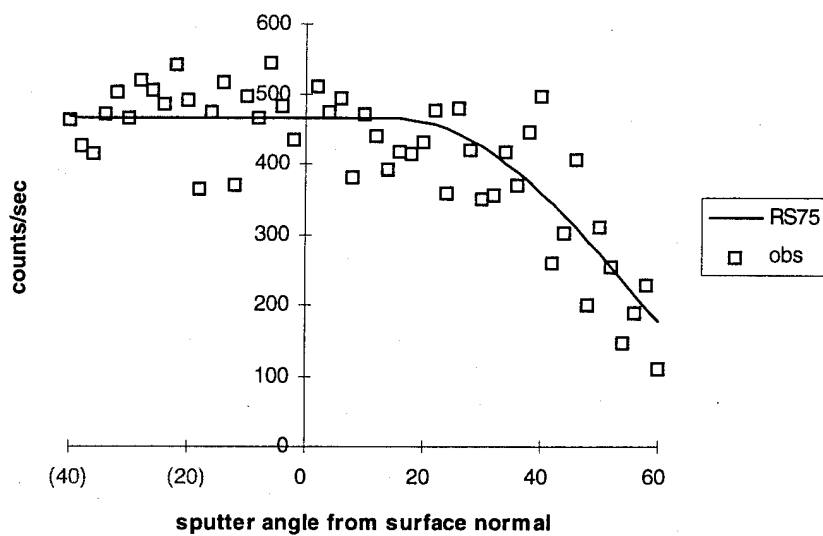
Figure 67. Modified RS fit to N$^+$ on Zr at 75° incidence.

Table 16. Modified RS best fit parameters to N$^+$.

Angle	forward			Backward			
	B(E)	n	U	B(E)	n	U	Chi
30	739	1.12	1.88				0.66
45	595	1.27	2.26	512	.1	.439	1.22
60	937	1.76	3.23	834	2.43	0.950	1.28
75	579	1.94	3.40	463	-0.023	.001	1.70

The forward scattering behavior trend is similar to the forward sputtering behavior of the other incident ions we have studied. The progression in n as the incident ion moves toward the surface is consistent. The progression in U is also consistent although it is very small. N$^+$ is a very small ion and is only 15% of the mass of zirconium. The N$^+$ ion can penetrate deeper into the solid before impacting with the target atoms and as such the surface interaction term should be smaller.

The backward sputtering behavior has the most scatter and is hard to make any determinations. There seems to be the same undercosine behavior seen with other species and the surface interaction term is extremely small. A possible mechanism for this behavior is that the back scattered incident ion is small enough to exit the lattice without interacting substantially with the target, that is, without producing any more collisions. As the incident ion moves toward the surface, the back scattered ion's trajectory also moves toward the surface. The back scattered ion travels subsurface parallel to the surface initiating sputtering from the second and third layers. These atoms

will preferentially eject in the surface normal direction and cause the extreme flattening of the curve at 75°.

Chapter 5. Summary and conclusions

An ion optics system utilizing a wein filter velocity selector has been modeled and characterized for use as a ion source for an instrument to measure high resolution angular distributions of sputtered neutral atoms. The ion source provides the capability to use mixed species as sputter sources and still obtain a well resolved single component ion beam for sputtering. The ion system was tested using N_2^+ and sputtering behavior of diatomic N_2^+ as well as monatomic N^+ has been measured. The capability to explore sputtering behavior using ions other than noble gas ions is now available.

Extensive modifications to the instrument were made including increasing the sample size, modeling and optimizing the ion source conditions and ion flux on the sample, rewriting the control programs to incorporate annealing cycles, and changing the sample annealing conditions. Ar^+ data was reaccomplished to verify consistency with previous data taken in the Watts' lab. No new experimental artifacts or discrepancies were apparent.

A modified Roosandaal Sanders model was used to analyze the sputter data. The model breaks down when high angle of incidence ion beams are used, reproducing neither the peak nor shape of the sputtering distribution seen in experiment. Using the surface binding energy U and the cosine exponent n as fitting parameters removes any relation to physical reality.

Using n and U as fitting parameters, extensive analysis using the modified Roosandaal Sanders equation was accomplished on N_2^+, Ar^+, Ne^+, and Xe^+. Assuming the back sputtered and forward sputtered atoms occur from the top layers of the target and assuming the back sputtered and forward sputtered atoms are caused by different mechanisms; the back and forward sputtering atoms were fit independently using the peak position in the distribution as an arbitrary dividing point. Following this procedure and allowing n and U to vary independently in both fits reveals some interesting trends. Treating U as a measure of the surface interaction energy, U increases as the incident ion angle increases or the mass increases for forward sputtering. The value of n remains between 1 and 3, consistent with the work of other researchers. The value of n also increases for the forward sputtering case as the incident ion angle moves toward the surface. The surface binding energy can be related to the average lattice spacing. If the top layer lattice spacing increases, the number of atoms preferentially sputtered in the surface normal direction increases. Thus, the increase in both n and U are consistent with the picture of an expanding surface lattice.

Figure 68 and Figure 69 are plots of U and n versus the ion beam incidence angle for the forward sputtering case using the gases examined in this thesis. The general trend, of increasing values of U and n as the ion beam angle of incidence increases, is evident for all gases except Xe^+. Xe^+ peaks in U and n at 45°. This can be explained by decreasing penetration of the ion as the angle of incidence increases. The Xe^+ ion is being scattered from the top layers of the target. The N_2^+ ion shows much larger increases in both U and n at the 75° angle. If the N_2^+ ion breaks up into two N^+ ions in the region of the

surface, twice the energy can be localized in those top layers. This would explain the larger values that are seen. The small values of U and n seen at all angles for the N^+ ion can be attributed to the small mass/size of the ion. The ion penetrates deeply into the solid and is not efficient in transferring its energy to the target atoms. This causes a corresponding small sputter yield as well as small values of U and n.
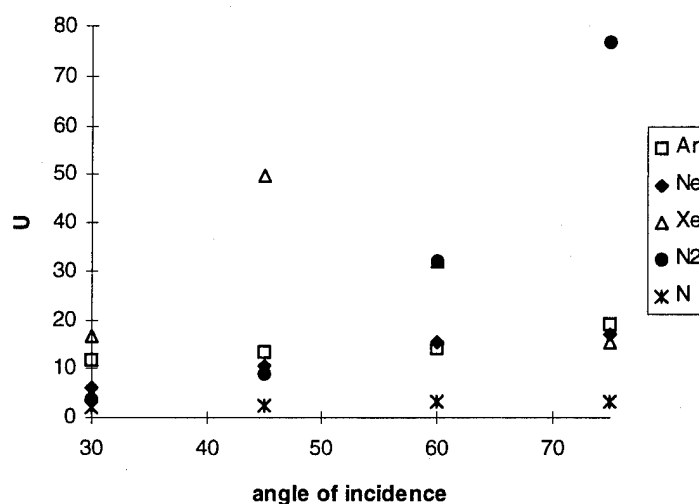
Figure 68. U vs ion beam angle of incidence, forward sputtering case.
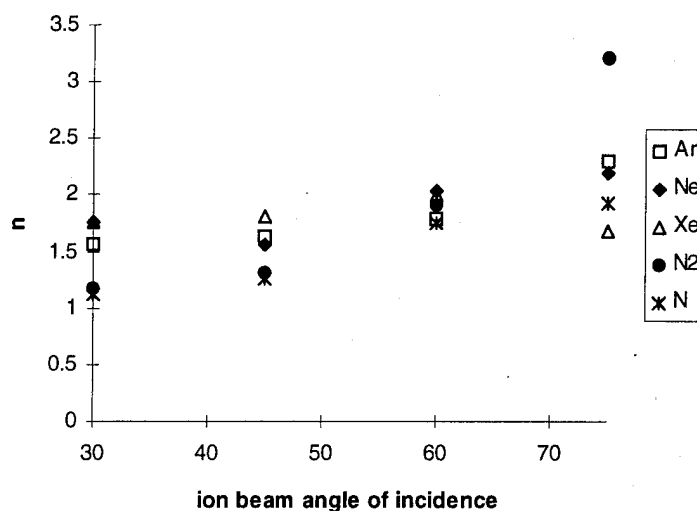
Figure 69. n vs ion beam angle of incidence, forward sputtering case.

For the backward sputtering case, n is consistently less than 1 for the ground state and approximately 1 for the first excited state. This is consistent with a model in which the incident ion is back reflected toward the surface, preferentially exiting in the surface normal direction due to interactions with the target atoms. The target atoms will thus exit at angles close to the surface, resulting in a broad distribution with a distinctly under cosine appearance. If the sputtered atoms are in an excited state and exit at large angles with respect to the surface normal, they can be quenched by interaction with the surface and the distribution of the surviving atoms will become more peaked. This explains why the excited state distributions are consistently more peaked (larger n) than the ground state distributions.

The value of U is consistently smaller for the back sputtered curves than the forward sputtering curves. If the predominant mechanism for the back sputtering behavior is caused by the back reflections of the incident ion,

there could be very little interaction with the surface. The ion will penetrate very shallowly before impacting with a target atom and returning to the surface. The ion can possibly sputter some surface atoms on its way out, but will still retain a large amount of its initial energy upon ejection. The surface interaction will be limited and so the U terms are consistently small. Figure 70 and Figure 71 are plots of U and n versus the ion beam incidence angle for back sputtered atoms obtained for the gases examined in this thesis. There is more scatter in the data than in the forward sputtering case. Due to machine limitations, fewer data points can be recorded in the backward direction causing more variability in the fits. The behavior described above is essentially the same for all the gases used in this thesis.
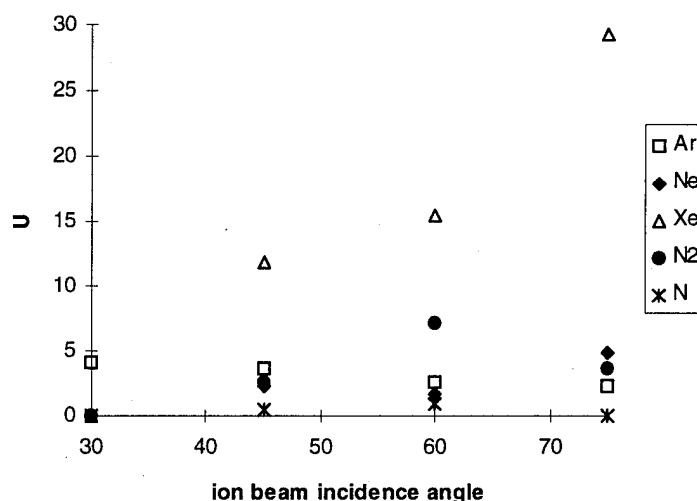
Figure 70. U versus ion beam incidence angle, backward sputtering case.
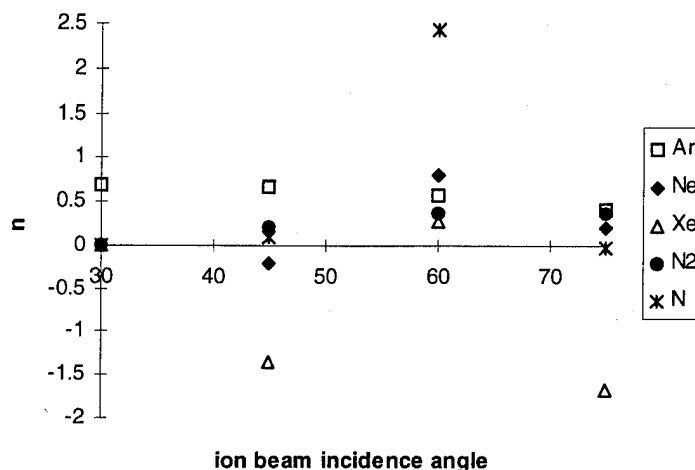
Figure 71. n vs ion beam incidence angle, backward sputtering case.

Aijun Li[7] conducted molecular dynamic simulations of a Zr crystal. He found that the majority of the sputtering events occur in the top surface layers with over 96% of the sputtered atoms originating in the two top surface layers. Sputtering is truly a surface phenomena. The Roosandaal Sanders theory arbitrarily introduces a surface binding energy using a planer binding potential. This treats the surface interaction as a constant for all masses and ion incident angles. This treatment clearly underestimates the effect of the surface and does not reproduce experimental observations. Treating the surface binding energy as an adjustable parameter, and hypothesizing its function as a surface interaction term produces consistent behavior. The modified Roosandaal Sanders equation can be used in this framework to provide insight into the sputtering mechanisms occurring. The experimental data in this thesis can be explained in this manner and provide experimental verification of sputtering as a surface phenomena. Unfortunately, this method does not provide predictive capabilities. Improvements to the

analytical theory of sputtering should concentrate on how to improve the model of the surface as well as the ion-surface interaction.

Future Work

Experiments to probe the ion surface interaction can be accomplished using the current apparatus with some modifications. The instrument can currently record angular resolved sputtering data. Energy and angular resolved spectra are desired to determine if there is an energy difference between the spectrum of forward and backward sputtered atoms. A Doppler shift detector could be added to the detector assembly to provide the energy measurement. This modification would require a frequency scan to be accomplished at each angle, resulting in a prohibitively long experiment using the current single photon counting technique. The predominant source of background counts comes from scattered laser light. The dark counts and the background ion counts are relatively constant and an order of magnitude smaller. Changing the detection system to the current signal off the photomultiplier in conjunction with a lock in technique could provide a more reasonable experiment duration. The ion beam would be chopped and the chopping frequency used as the reference for the lock in detector. Changing to a lock-in technique would also provide the capability of elevated temperature sputtering as the thermal photons created will not affect this technique.

The Auger spectrometer should also be replaced with a retractable LEED Auger system. The angular range of the instrument is currently limited by the Auger in one direction and by the ion beam in the other direction.

Using a retractable Auger system removes one of these limitations and the LEED capability will allow the examination of the surface as the sputtering process evolves.

The above changes would provide expanded capability to explore the ion surface interaction occurring during sputtering. This is the weak link in the current analytical models of sputtering and more work should concentrate in this area.

List of References

[1] Behrisch, R. (ed.); *Sputtering by Particle Bombardment I*, Topics in Applied Physics, Vol. **47**, Chapter 2, (Springer, Berlin, Heidelberg, New York 1981).

[2] Sigmund, P.; *Sputtering by Particle Bombardment I*, Topics in Applied Physics, Vol. **47**, p 11, , (Springer, Berlin, Heidelberg, New York 1981).

[3] Behrisch, R. (ed.); *Sputtering by Particle Bombardment II*, Topics in Applied Physics, Vol. **52**, Chapter 7, (Springer, Berlin, Heidelberg, New York 1983).

[4] Espy, S. L., et al; *Prc. SPIE - Int. Soc. Opt. Eng.*, **1761**, 130 (1992).

[5] Terwagne, G.; Lucas, S.; and Bodart, F.; *Nucl. Instrum. Methods Phys. Res.*, Sect B, B59 (1991).

[6] Whitaker, T; Ph.D. Thesis, University of Washington, 1992.

[7] Li, A; Ph.D. Thesis, University of Washington, 1993.

[8] Grove, W. R.; *Philos. Mag.*, **5**, 203(1853).

[9] Goldstein, E.; *Verh. Dtsch. Phys. Ges.*, **4**, 228,237(1902).

[10] Stark, J.; *Die Elektrizitat in Gasen*, (Barth, Leipzeg 1902).

[11] Stark, J.; *Z. Elektrochem.*, **14,** 752(1908); **15,** 509 (1909).

[12] Kingdon, K. H.; Langmuir, I.; *Phys. Rev.*, **20,** 107(1922); **21,** 210(1923); **22,** 148(1923).

[13] Seeliger, R.; Sommermeyer, K.; *Z. Phys.*, **93,** 692 (1935); *Ann. Phys.* (Leipzig) **25,** 481(1936); *Z. Phys.*, **119,** 482(1942).

[14] Wehner, G. K.; *Phys. Rev.*, **102,** 690(1956).

[15] Lamar, E. S.; Compton, K. T.; *Science,* **80,** 541(1934).

[16] Sigmund, P.; *Sputtering by Particle Bombardment I* (Behrisch, Ed.), Topics in Applied Physics, Vol. **47,** Chapter 2, (Springer, Berlin, Heidelberg, New York 1981).

[17] Goldstein, H.; *Classical Mechanics*, (Addison-Wesley Publishing Company, 1980).

[18] Child, M. S.; *Molecular Collision Theory*, (Academic Press, 1974).

[19] Mcdaniel, E.; *Atomic Collisions*, p. 3; (Wiley Interscience, 1989).

[20] Pauley, H.; *Atom-Molecule Collision Theory* (Bernstein, ed.), Chap. 4; (Plenum Press 1979).

[21] Torrens, I. M.; *Interatomic Potentials*, Academic Press (New York 1972).

[22] McQuarrie, D.; *Statistical Mechanics*, Chap. 10, (Harper and Row 1976).

[23] Torrens, I. M.; *Interatomic Potentials*, Academic Press (New York 1972).

[24] Sigmund, P.; *Rev. Roum. Phys.*, **17**(7), 823(1972).

[25] Gradshetyn, I. S.; *Table of Integrals, Series, and Products*, Academic Press (1980).

[26] Marion, J. B.; *Classical Dynamics of Particles and Systems*, Chap. 9, Academic Press (1970).

[27] Sigmund, P.; *Sputtering by Particle Bombardment I* (Behrisch, Ed.), Topics in Applied Physics, Vol. **47**, Chapter 2, (Springer, Berlin, Heidelberg, New York 1981).

[28] Almen, O. and Bruce; *Nucl. Instr. Meth.*, **11**, 257 (1961).

[29] Thompson, M. W.; *Philos. Mag.*, **18**, 377 (1968).

[30] Thompson, M. W.; *Phys. Reports*, **69(4)**, 336 (1981).

[31] Sigmund, P.; *Phys. Rev.*, **184**, 383 (1969).

[32] Anderson, H. H.; Stenum, B.; Sorenson, T.; and Whitlow, H. J.; *Nucl. Instr. Meth.*, **B6**, 459 (1985).

[33] Roosendaal, H. E. and Sanders, J. B.; *Radat. Eff.*, **52**, 137 (1980).

[34] Sanders, J. B. and Roosendaal, H. E.; *Radat. Eff.*, **24**, 161 (1975).

[35] Hofer, W. O.; *Sputtering by article Bombardment III*, Topics in Applied Physics, Vol. **64**, (Behrisch ed.), (Springer, Berlin, Heidelberg, New York 1991).

[36] Anderson, H. H.; Stenum, B.; Sorenson, T.; and Whitlow, H. J.; *Nucl. Instr. Meth.*, **B6**, 459 (1985).

[37] Brauer, G.; Hasselkamp, D.; Kruger, W.; and Scharman, A.; *Nucl. Instr. Meth.*, **B12**, 458 (1985).

[38] Schwebel, C.; Pellet, C.; and Gautherin, G.; *Nucl. Instr. Meth.*, **B18**, 525 (1987).

[39] Hofer, W. O.; *Sputtering by article Bombardment III*, Topics in Applied Physics, Vol. **64**, (Behrisch ed.), (Springer, Berlin, Heidelberg, New York 1991).

[40] Dahl, D. A.; Delmore, J. E.; *The SIMION PC/PS2 Users Manual Version 4.0*; Prepared for U.S. Department of Energy under DOE contract No. DE-AC07-761D01570: EG&G Idaho: Idaho Falls, ID, 1988.

[41] Hefter, U. and Bergmann, K.; In *Atomic and Molecular Beam Methods*; Scoles, Giancinto, Ed.; Oxford University Press: New York, 1988; Chapter 9.

[42] Corliss, C. H. and W.R. Bozman, *Experimental Transition Probabilities for Spectral Lines of Seventy Elements*, **NBS Monograph 53**, US Department of Commerce (1962).

[43] Pellin, M. J.; Wright, R. B.; and Gruen, D. M.; *J. Chem. Phys.* **74**(11) 6448 (1981).

[44] Hannaford, P. and Lowe, R. M.; *Opt. Eng.* **22**(5) 532(1983).

[45] Anderson, H.H.; Stenum, B.; Sorenson, T.; and Whitlow, H. J.; *Nucl. Instr. Meth.*, **B6**, 459 (1985).

[46] Brauer, G.; Hasselkamp, D.; Kruger, W; and Scharmn, A.; *Nucl. Instr. Meth.,* **B12,** 458 (1985).

[47] Schwebel, C.; Pellet, C.; and Gautherin, G.; *Nucl. Instr. Meth.,* **B18,** 525 (1987).

[48] Hofer, W. O.; Chapter 2 in *Sputtering by Particle Bombardment III*, Eds.: Behrisch, R and Wittmaack, K.; Topics in Appl. Phys., **64** (Springer-Verlag Berlin Heidelberg, 1991).

[49] Garrison, B. J.; Winograd, N.; Lo, D.; Tombrello, T. A.; Shapiro, M. H.; and Harrison, D. E., *Surf. Sci.* **180** L129 (1987).

[50] Jackson, D. P., *Rad. Eff.* **18** 185(1973)

[51] Jackson, D. P., *Can. J. Phys.* **53** 1513 (1975).

[52] Garrison, B. J.; Winograd, N.; Lo, D.; Tombrello, T. A.; Shapiro, M. H.; and Harrison, D. E., *Surf. Sci.* **180** L129 (1987).

Appendix A - Computer Programs

BASIC Programs

ANGSCANP.BAS

```
10   REM   ANGSCANP V1.0, (ROT&CNT) modified 8/13/90 to increase DELAY(3)   (TW)
20   REM   Graphic boundaries and flag offset are set at 910-920   (TW)
30   REM   modified 08/20/92: solenoid operation, waitdelay   (WM)
40   REM        08/27/92: program structure   (WM)
50   REM    10/07/93: program structure (PRS)
60   REM   files needed    : hbasic.exe, basica.com, code.bas, camio.bas
70   '
80   ' *** Find data segment above BASICA ***
90   '
100  CLEAR: CLS                    ' Set all variables to "0"
110  DEF SEG=0                     ' Go to low memory to find BASICA loc.
120  BDATSEG=PEEK(&H510)+256*PEEK(&H511) ' BASICA data segment is in &H510-511
130  CAMSEG=BDATSEG+&H2000            ' Find next data segment after BASICA
140  DEF SEG=CAMSEG
150  '
160  ' *** Open COM channel to Apple Comp. ***
170  '
180  OPEN "COM1:9600,N,8,1" AS #2
190  MSG$="READ"+CHR$(13):GOSUB 4120
200  '
210  ' *** Dimension of variables ***
220  '
230  DIM OVFTBL%(32),MULTBL%(32),DAC0(3),DAC2(3),TDEL(3),SIG(3)
240  DIM WVMTR#(3),D24%(2),LABEL$(3)
250  DATA 4,9,8,9,7,9,8,9,6,9,8,9,7,9,8,9,5,9,8,9,7,9,8,9,6,9,8,9,7,9,8,9
260  DATA 0,16,8,17,4,18,9,19,2,20,10,21,5,22,11,23,1,24,12,25,6,26,13,27
270  DATA 3,28,14,29,7,30,15,31
280  FOR I=1 TO 32: READ OVFTBL%(I): NEXT    'Load in tables for conversion of
290  FOR I=1 TO 32: READ MULTBL%(I): NEXT    'time.
300  '
310  ' *** Setting parameters ***
320  '
330  PRINT " "
340  PRINT "y minimum value for screen plot: ";:INPUT PMIN
350  PRINT "y maximum value for screen plot: ";:INPUT PMAX
360  PFAC=(PMAX-PMIN)
370  XM=600: YM=400: YM2=YM/2
380  '
```

```
390  ' *** Program header ***
400  '
410  CLS:
420  '
430  PRINT "A N G S C A N P (1.0)": PRINT ""
440  PRINT "This program physically SCANS the DETECTOR using user-specified"
450  PRINT "steps and counts for a specified time at each position with laser on,"
460  PRINT "ion beam on, then counts for an equal time with all on,"
470  PRINT "and stores all three counts plus the darkcount onto disk."
480  PRINT ""
490  BEEP: PRINT "": PRINT "Hit any key when ready ..."
500  A$=INKEY$: IF A$="" THEN 500
510  '
520  ' *** Poke in machine language routinge ***
530  '
540  PRINT "": PRINT "Loading PC21 & CAMAC I/O drivers ..."
550  '
560  OPEN "C:\BASICDIR\NEW\CODE.BAS" FOR INPUT AS #1    ' Access machine code data file
570  FOR X = 0! TO 127!
580    INPUT #1, J                        ' Install machine code
590    POKE X,J
600  NEXT
610  CLOSE #1
620  '
630  ' *** Set all program variables ***
640  '
650  ADDRESS% = 768          ' PC21 base address
660  CONTROL = 96            ' Normal state of PC21 Control Byte
670  CRASH = 4              ' Mask for Control Bit 2 (BMA time-out)
680  FAULT = 32             ' Mask for C.B. 5 (restart BMA)
690  PC21WRITE = 0!          ' Address of PC21WRITE subroutine
700  PC21READ = 49!          ' Address of PC21READ subroutine
710  '
720  DAC# = 12              ' DAC module in CAMAC slot #12
730  '              ch#: 0:shutter, 1:not used, 2:ion beam shutter
740  '                   3:not used
750  TIM# = 10             ' TIMER/SCALER module in CAMAC slot #10
760  DELAY1# = 1            ' Delaytime for solenoid shutter in sec.
770  DELAY2# = 2            ' Delaytime for ion beam response in sec.
780  '
790  DATDIR$ = "C:\SPUTTER\ANGDAT\"       ' Directory for data files
```

```
800  '
810  CMSG$="                              "     ' Clear String
820  '
830  FLGOFF = 2.5          ' Offset for 0 degree detector position
840  MAXANG = 110             ' Maximum detector position due to FIBER !!!
850  '
860  ' *** PC21 RESET ***
870  '
880  OUT ADDRESS%+1, ( CONTROL OR CRASH )         ' Control Bit 2 high
890  OUT ADDRESS%+1, ( CONTROL AND NOT CRASH )  ' Control Bit 2 low
900  FOR Y=1 TO 500: NEXT              ' Wait for BMA
910  OUT ADDRESS%+1, ( CONTROL AND NOT FAULT ) ' Control Bit 5 low
920  OUT ADDRESS%+1, ( CONTROL OR FAULT )       ' Control Bit 5 high
930  '
940  ' *** Load CAMAC drivers and initialize crate ***
950  '
960  BLOAD "C:\BASICDIR\NEW\CAMIO",128      ' Load drivers into data segment
970  CAMO=&H80:CAMI=&H86:CAML=&H8C:CAMCL=&H92 ' Driver entry point addresses
980  CAMO24=&HB0:CAMI24=&HB6:CRATE=&HAA      ' Driver entry point addresses
990  CC%=1: CALL CRATE(CC%)           ' Activate controller in J1 slot
1000 OUT &H240,0                ' Clear high write-only data register
1010 I%=64: CALL CAMCL(I%)          ' Reset crate
1020 I%=1: CALL CAMCL(I%)           ' Initialize crate
1030 N%=TIM#:F%=17:A%=13:D%=1:GOSUB 3260  ' Write timer/scaler LAM mask-generate
1040 '                       LAM when channel 1 finishes counting
1050 GOSUB 3370
1060 '
1070 ' *** Initialization ***
1080 '
1090 A1%=0:D1%=0: GOSUB 3910    ' Laser beam shutter should be open!
1100 PRINT "Laser beam shutter:  OPEN!"
1110 N%=DAC#:F%=16:A%=2:D%=32700: GOSUB 3260 ' Ion beam shutter should be open!
1120 PRINT "Ion beam shutter :  OPEN!"
1130 '
1140 PRINT "Detector      :  Moving to 0 degree!"
1150 COMMAND$="FSB1 FSC1 MN A1 V.1": GOSUB 2910  ' Moving detector to 0 degree!
1160 POS0=0: POS1=2: GOSUB 3110          ' Make sure not on an endpoint
1170 POS0=0: POS1=-10: GOSUB 3110
1180 IF ENDFLG=0 THEN 1170
1190 POS0=0: POS1=FLGOFF: GOSUB 3110: POS0=0   ' Compensate for offset
1200 '
```

```
1210 ' *** Define data file name ***

1220 '

1230 PRINT ""

1240 BEEP: PRINT "Data file name: ";: INPUT FILNAM$

1250 ON ERROR GOTO 1340

1260 OPEN DATDIR$+FILNAM$ FOR INPUT AS #1   ' Test to see if file already exists.

1270 '                         File exists.  Ask user what to do!!

1280 BEEP: BEEP

1290 PRINT "File exists! Continue [y/n] : ? "

1300 V$=INKEY$: IF V$="" THEN GOTO 1300

1310 IF V$="Y" OR V$="y" THEN GOTO 1350  ' Overwrite existing file.

1320 IF V$="N" OR V$="n" THEN GOTO 1240  ' Get a new name for file.

1330 GOTO 1300

1340 RESUME 1350                       ' File nonexistent.  Proceed.

1350 ON ERROR GOTO 0

1360 CLOSE #1                   ' Close temporary input file.

1370 OPEN DATDIR$+FILNAM$ FOR OUTPUT AS #1  ' Open output file.

1380 '

1390 ' *** Print file header ***

1400 '

1410 PRINT #1,"Data stored by ANGSCAN1.0 on ";DATE$;" at ";TIME$

1420 PRINT "File header: ";: INPUT HDR$

1430 PRINT #1, "File header: ";HDR$

1440 IF NOT EOF(2) THEN ASMSG$=INPUT$(1,#2): GOTO 1430   'switch to settle.

1450 '

1460 ' *** Enter parameter for measurement ***

1470 '

1480 BEEP: PRINT "": PRINT "Integration time (> 1 sec): ";: INPUT ITIM

1490 IF ITIM<1 THEN ITIM=1

1500 IF ITIM>10000! THEN BEEP: BEEP: PRINT "Out of Range": GOTO 1480: PRINT ""

1510 T=ITIM: GOSUB 3660: TSAVE%=TIMEC%

1520 MSG$=" sec integration.": PRINT #1,STR$(T);MSG$

1530 PRINT "Enter # spectra  (1-100) : ";: INPUT NSPEC%

1540 IF NSPEC%<1 OR NSPEC%>100 THEN 1530

1550 PRINT #1,NSPEC%;" spectra."

1560 PRINT "Starting angle         : ";: INPUT ANG0

1570 IF ANG0 > MAXANG THEN PRINT "Position out of limit!!!": GOTO 1560

1580 PRINT "Ending angle           : ";: INPUT ANGEND

1590 IF ANGEND > MAXANG THEN PRINT "Position out of limit!!!": GOTO 1580

1600 PRINT "Degrees per step       : ";: INPUT DELANG

1610 NINC=INT((ANGEND-ANG0)/DELANG+.5): ANGEND=ANG0+NINC*DELANG
```

```
1620 IF NINC<0 THEN NINC=-NINC: DELANG=-ABS(DELANG)
1630 PRINT "Scanning from ";ANG0;" degrees to ";ANGEND;" degrees in ";NINC;" steps."
1640 PRINT #1, "Scanning from";ANG0;" degrees to ";ANGEND;" degrees in ";NINC;" steps."
1650 PRINT "Ion sputter angle: ";:INPUT A1
1660 PRINT #1, "Ion sputter angle: ";A1
1670 PRINT "Sample Metal : ";:INPUT B1$
1680 PRINT #1, "Sample Metal : ";B1$
1690 PRINT "Sputter Ion : ";:INPUT B2$
1700 PRINT #1, "Sputter Ion : ";B2$
1710 PRINT "photomultiplier voltage :";:INPUT A2
1720 PRINT #1, "photomultiplier voltage :";A2
1730 PRINT "Ion beam current :";: INPUT A3
1740 PRINT #1, "Ion beam current :"; A3
1750 PRINT "Ion beam voltage :";: INPUT A4
1760 PRINT #1, "Ion beam voltage :"; A4
1770 PRINT "Moving detector to ";ANG0;" degree"
1780 POS0=0: POS1=ANG0: GOSUB 3110
1790 IF ENDFLG<>0 THEN BEEP: BEEP: BEEP: PRINT "ERROR!  Limit switch hit!": STOP
1800 BEEP: PRINT "Laser frequency (cm^-1)  : ";: INPUT NRG#
1810 PRINT #1, "Laser frequency (cm^-1) :";NRG#
1820 PRINT "laser power(milliwatts) :";:INPUT A5
1830 PRINT #1, "laser power (milliwatts) :"; A5
1840 PRINT "number of steps before pause:";:INPUT NSTEPS
1850 PRINT #1, "number of steps before pause:"; NSTEPS
1860 PRINT #1, "|all on| - |laser on| - |ion on| + |dark count| = |Total count|"
1870 NRG$=STR$(NRG#): MSG$="GOK"+NRG$+CHR$(13): GOSUB 4120  ' Tell AUTOSCAN to go there.
1880 MSG$="PEAK BRF"+CHR$(13):GOSUB 4120
1890 MSG$="PEAK ETA"+CHR$(13):GOSUB 4120
1900 '
1910 ' *** Initialize plotting on the screen ***
1920 '
1930 CLS: SCREEN 2: DRAW "BM 0,0 R=XM;D=YM2;L=XM;U=YM2;"
1940 LOCATE 15,1: PRINT ANG0: LOCATE 16,1: PRINT NRG#
1950 LOCATE 2,50: PRINT FILNAM$
1960 DELX=XM/NINC                    'DELX used for plotting
1970 '
1980 ' *** Starting the data collection  ***
1990 '
2000 DAC0(1)=6540:DAC0(2)=0:DAC0(3)=0          ' Settings for DAC
2010 DAC2(1)=32700:DAC2(2)=32700:DAC2(3)=0
2020 TDEL(1)=DELAY2#:TDEL(2)=DELAY1#:TDEL(3)=DELAY1#  ' Setting delay after solenoid and ion beam
```

```
2030 LABEL$(1)="Ion beam ON":LABEL$(2)="All ON":LABEL$(3)="Laser on"
2040 '
2050 ' *** NSP data collection loop ***
2060 '
2070 FOR NSP=1 TO NSPEC%              ' Start scans
2080 LOCATE 18,30: PRINT"collecting dark counts for 10 sec!"
2090 N%=DAC#:F%=16:A%=2:D%=0:GOSUB 3260  'ion beam shutter closed
2100 A1%=0:D1%=6540:GOSUB 3910    'laser shutter closed
2110 T=10:GOSUB 3660:GOSUB 3510   'wait delay
2120 T=10:GOSUB 3660:GOSUB 3420   '10 sec dark count collection
2130 SIGNAL# = D24%(1) AND 32767:IF D24%(1)<0 THEN SIGNAL#=SIGNAL#+32768#
2140 FOR I=1 TO D24%(2):SIGNAL#=SIGNAL#+65536#:NEXT I
2150 DARKCNT=CINT(SIGNAL#/10):GOSUB 3590
2160 '
2170   CLS: SCREEN 2: DRAW "BM 0,0 R=XM;D=YM2;L=XM;U=YM2;"
2180   LOCATE 15,1: PRINT ANG0: LOCATE 16,1: PRINT NRG#
2190   LOCATE 2,50: PRINT FILNAM$
2200   ' LOCATE 2,3: PRINT "spec#";NSP
2210   X0%=0                       ' Initialize x-axis of plot
2220   POS1=ANG0: GOSUB 3110       ' then must reset detector,
2230   NNCOUNT=0
2240   FOR INCCNT=0 TO NINC        ' Begin loop incrementing angle.
2250     LOCATE 2,3: PRINT "spec#";NSP
2260     IF INCCNT=0 THEN GOTO 2300    ' Don't need angle increment 1st time.
2270     POS1=DELANG*INCCNT+ANG0: GOSUB 3110 ' Move there.
2280     TIM=TIMER
2290     IF TIMER-TIM<.5 THEN 2290
2300     LOCATE 18,1: PRINT "              "
2310     LOCATE 18,1: PRINT USING "Angle = #######.###";POS0
2320     '
2330     MSG$="READ"+CHR$(13):GOSUB 3960
2340     '****
2350     WVMTR#= VAL(MID$(RMSG$,2,10))
2360     DELNRG= WVMTR#-NRG#
2370     '****
2380     IF ABS(DELNRG)>.006 THEN BEEP: BEEP:
2385     MSG$="PEAK BRF"+CHR$(13): GOSUB 3960
2390     MSG$="PEAK ETA"+CHR$(13):GOSUB 3960
2400     MSG$="GOK"+STR$(NRG#)+CHR$(13):GOSUB 3960
2410     AVDEL= 0
2420       FOR II=1 TO 3
```

```
2430   MSG$="READ"+CHR$(13): GOSUB 3960   ' and verify wavelength.
2440   WVMTR#(II)=VAL(MID$(RMSG$,2,10))
2450   DELNRG=WVMTR#(II)-NRG#
2460   IF ABS(DELNRG)>.006 THEN GOTO 2385
2470   IF ABS(DELNRG)<=.0017 THEN GOTO 2530
2480   IF ABS(DELNRG)>.0017 THEN AVDEL=AVDEL+DELNRG
2490      NEXT II
2500   AVDEL=AVDEL/3!
2510   MSG$="DELK"+STR$(-AVDEL)+CHR$(13): GOSUB 3960
2520   GOTO 2410
2530   FOR DACSW=1 TO 3
2540     LOCATE 18,30: PRINT "                    "
2550     LOCATE 18,30: PRINT LABEL$(DACSW)
2560     N%=DAC#:F%=16:A%=2:D%=DAC2(DACSW): GOSUB 3260        ' Set DAC on/off depending on
2570     A1%=0:D1%=DAC0(DACSW): GOSUB 3910 ' Dacsw
2580     T=TDEL(DACSW): GOSUB 3660: GOSUB 3510      ' Wait for time delay!
2590     TIMEC%=TSAVE%: GOSUB 3420                ' now Count for TSAVE%
2600     SIGNAL#=D24%(1) AND 32767: IF D24%(1)<0 THEN SIGNAL#=SIGNAL#+32768#
2610     FOR I=1 TO D24%(2): SIGNAL#=SIGNAL#+65536#: NEXT I
2620     SIG(DACSW)=SIGNAL#/ITIM: GOSUB 3590
2630   NEXT DACSW
2640   '
2650   DELSIG=SIG(2)-SIG(3)-SIG(1)+DARKCNT
2660   PRINT #1,SIG(2),SIG(3),SIG(1),DARKCNT,DELSIG
2670   LOCATE 19,1: PRINT "                              "
2680   LOCATE 19,1: PRINT INCCNT;": ";SIG(2);"-";SIG(3);"-";SIG(1);"+";DARKCNT;"=";DELSIG
2690   YP%=(YM/PFAC)*(PMAX-DELSIG): XP%=INCCNT*DELX: IF YP%>YM2 THEN YP%=YM2
2700   DRAW "BM=X0%;,=Y0%;": DRAW "M=XP%;,=YP%;": Y0%=YP%: X0%=XP%
2710   SIG(1)=0:SIG(2)=0:SIG(3)=0 'Reset accumulated signals for next frequency
2720   NNCOUNT=NNCOUNT+1
2730      IF NNCOUNT=NSTEPS THEN GOSUB 4210
2740   NEXT INCCNT
2750   '
2760   BEEP
2770 PRINT #1,"xxxxxxxxxxx"
2780 NEXT NSP
2790 '
2800 CLOSE
2810 LOCATE 23,1: BEEP: BEEP: PRINT ">>>(EXIT Angscan1.0 with 'E')"
2820 A$=INKEY$: IF (A$="E") OR (A$="e") THEN CLS ELSE 2820
2830 END
```

```
2840 '
2850 '
2860 ' *** SUBROUTINES ARE LOCATED HERE !!! ***
2870 '
2880 ' *** PC21 WRITE ***
2890 '
2900 BFLAG%=0
2910 IF INSTR(COMMAND$,"W1") OR INSTR(COMMAND$,"w1") THEN BFLAG%=1
2920 IF INSTR(COMMAND$,"PB") OR INSTR(COMMAND$,"pb") THEN BFLAG%=1
2930 IF INSTR(COMMAND$,"X1B") OR INSTR(COMMAND$,"x1b") THEN BFLAG%=1
2940 COMMAND$ = COMMAND$ + CHR$(13)     ' Add carriage return to command
2950 CALL PC21WRITE(COMMAND$, ADDRESS%)  ' Execute machine language write
2960 RETURN
2970 '
2980 ' *** PC21 READ ***
2990 '
3000 ANSWER$="            "+""    ' Reserve string space for response
3010 CALL PC21READ( ANSWER$, ADDRESS%, BFLAG% )  ' Execute read
3020 IF BFLAG%=0 THEN RETURN
3030 NUM#=0: FOR X=1 TO 4
3040 DIGIT%=ASC(MID$(ANSWER$,X,1))
3050 NUM#=NUM#+DIGIT%*256^(4-X): NEXT
3060 ANSWER$=STR$(NUM#)
3070 RETURN                       ' BFLAG% identifies binary report commands
3080 '
3090 ' *** Move Detector POS1-POS0 degrees (check for end switch) ***
3100 '
3110 DEG=ABS(POS1-POS0): DIR$="+": IF POS1-POS0<0 THEN DIR$="-"
3120 AVTIM=DEG*.15: ENDFLG=0: IF AVTIM<.5 THEN AVTIM=.5
3130 STEPS=INT(DEG*4000/180+.5)      ' Convert degrees to motor steps
3140 STEPS$=MID$(STR$(STEPS),2)
3150 DEGAC=STEPS*180/4000: IF DIR$="-" THEN DEGAC=-DEGAC
3160 COMMAND$="D"+DIR$+STEPS$+" G CR P": GOSUB 2910 ' Move & signal when done
3170 T0=TIMER
3180 GOSUB 3000: IF LEFT$(ANSWER$,1)=CHR$(13) THEN 3210  ' Keep going till end
3190 T1=TIMER: IF T1-T0<AVTIM THEN 3180 ' If not done after AVTIM,
3200 ENDFLG=1: GOTO 3220           ' End switch must have been hit.
3210 POS0=POS0+DEGAC
3220 RETURN
3230 '
3240 ' *** Main WRITE/READ Subroutines to CAMAC ***
```

```
3250 '
3260 CALL CAMO(N%,F%,A%,D%,Q%,X%)
3270 RETURN
3280 '
3290 CALL CAMI24(N%,F%,A%,D24%(1),Q%,X%)
3300 RETURN
3310 '
3320 CALL CAMI(N%,F%,A%,D%,Q%,X%)
3330 RETURN
3340 '
3350 ' *** initialize DAC, put all channels (0...3) to OV! ***
3360 '
3370 FOR I=0 TO 3: N%=DAC#:F%=16:A%=I:D%=0: GOSUB 3260: NEXT
3380 RETURN
3390 '
3400 ' *** Set timer/scaler ***
3410 '
3420 N%=TIM#:F%=17:A%=0:D%=TIMEC%: GOSUB 3260    ' D% must contain time.
3430 N%=TIM#:F%=17:A%=4:D%=1: GOSUB 3260         ' Start counting
3440 CALL CAML(L%):IF L%=0 THEN 3440             ' Wait for count to end
3450 N%=TIM#:F%=23:A%=12:D%=1: GOSUB 3260        ' Clear timer LAM
3460 N%=TIM#:F%=0:A%=1: GOSUB 3290: GOTO 3590    ' Read scaler
3470 RETURN
3480 '
3490 ' *** Wait delay ***
3500 '
3510 N%=TIM#:F%=17:A%=0:D%=TIMEC%: GOSUB 3260    ' D% must contain time!
3520 N%=TIM#:F%=17:A%=4:D%=1: GOSUB 3260         ' Start wait delay
3530 CALL CAML(L%): IF L%=0 THEN 3530            ' Wait for time to end
3540 N%=TIM#:F%=23:A%=12:D%=1: GOSUB 3260: GOTO 3590  ' Clear timer LAM
3550 RETURN
3560 '
3570 ' *** Clear timer/scaler ***
3580 '
3590 N%=TIM#:F%=9:A%=0:D%=0: GOSUB 3260
3600 N%=TIM#:F%=9:A%=1:D%=0: GOSUB 3260
3610 RETURN
3620 '
3630 ' *** Subroutine for converting time (T) to a special
3640 '     formatted integer (TIMEC%) used by CAMAC timer. ***
3650 '
```

```
3660 TS=T: TB%=INT((LOG(T/1000))/2.30258): ERFLG%=0

3670 TBAS=10^TB%: T=T/TBAS: N=0

3680 IF T<2 GOTO 3700

3690 N=N+1: T=T/2: GOTO 3680

3700 J=32*(T-1)+1

3710 IOVF%=N-OVFTBL%(J)

3720 IF IOVF%<0 THEN TB%=TB%-1: GOTO 3670

3730 IF IOVF%>15 THEN TB%=TB%+1: GOTO 3670

3740 TB%=TB%+6: IF TB%<0 OR TB%>7 THEN PRINT "Time requested outside range": ERFLG%=1: GOTO 3780

3750 MULT%=MULTBL%(J)

3760 TA=TBAS*(2*MULT%+1)*2^(IOVF%+4)

3770 TIMEC%=(128!*MULT%)+(8!*IOVF%)+TB%

3780 T=TS

3790 RETURN

3800 '

3810 ' *** Initialize the solenoid shutter (result: shutter is CLOSED!) ***

3820 '

3830 N%=DAC#:F%=16:A%=1:D%=32700: GOSUB 3260

3840 N%=DAC#:F%=16:A%=0:D%=32700: GOSUB 3260

3850 N%=DAC#:F%=16:A%=0:D%=0: GOSUB 3260

3860 N%=DAC#:F%=16:A%=0:D%=32700: GOSUB 3260

3870 RETURN

3880 '

3890 ' *** Open/close solenoid shutter ***

3900 '

3910 N%=DAC#:F%=16:A%=A1%:D%=D1%: GOSUB 3260

3920 RETURN

3930 '

3940 ' *** Send Message to Apple via RS232 ***

3950 '

3960 LOCATE 20,1: PRINT ">>> Remote Apple control: "

3970 LOCATE 21,1: PRINT CMSG$: LOCATE 21,1

3980 FOR IMSG=1 TO LEN(MSG$): V$=MID$(MSG$,IMSG,1): PRINT #2,V$;: NEXT

3990 PRINT MSG$: RMSG$=""

4000 IF EOF(2) THEN 4000

4010 WHILE NOT EOF(2)

4020 ASMSG$=INPUT$(1,#2): ASMSG$=CHR$(ASC(ASMSG$)-128)

4030 IF ASMSG$=CHR$(13) GOTO 4080

4040 RMSG$=RMSG$+ASMSG$: WEND

4050 IF ASC(ASMSG$)<>41 THEN GOTO 4000

4060 RETURN
```

```
4070 '
4080 LOCATE 21,1: PRINT CMSG$;: LOCATE 22,1: PRINT CMSG$;
4090 PRMSG$=RMSG$: RMSG$="": LOCATE 21,1: PRINT PRMSG$;
4100 GOTO 4040
4110 '
4120 FOR IMSG=1 TO LEN(MSG$): V$=MID$(MSG$,IMSG,1): PRINT #2,V$;: NEXT
4130 PRINT RMSG$=""
4140 IF EOF(2) THEN 4140
4150 WHILE NOT EOF(2)
4160 ASMSG$=INPUT$(1,#2): ASMSG$=CHR$(ASC(ASMSG$)-128)
4170 IF ASMSG$=CHR$(13) THEN RMSG$=""
4180 PRINT ASMSG$;: RMSG$=RMSG$+ASMSG$: WEND
4190 IF ASC(ASMSG$)<>41 THEN GOTO 4140
4200 RETURN
4210 BEEP:PRINT"":PRINT"flash sample and hit any key to continue"
4220 A$=INKEY$:IF A$="" THEN 4220
4230 NNCOUNT=0
4240 RETURN
```

CCOUNT.BAS

```
1    MAX=60000#

10   DIM OVFTBL%(32),MULTBL%(32)

20   DATA 4,9,8,9,7,9,8,9,6,9,8,9,7,9,8,9,5,9,8,9,7,9,8,9,6,9,8,9,7,9,8,9

30   DATA 0,16,8,17,4,18,9,19,2,20,10,21,5,22,11,23,1,24,12,25,6,26,13,27,3,28,14,29,7,30,15,31

40   CLS

50   ISET=0:T=1

60   FOR I=1 TO 32:READ OVFTBL%(I):NEXT

70   FOR I=1 TO 32:READ MULTBL%(I):NEXT

80   PRINT "CCOUNT: This program continously counts pulses from the photomultiplier"

90   PRINT "     for a user-specified time."

100 '

120 PRINT: PRINT "Loading PC21 and CAMAC I/O drivers."

130 GOTO 520                      ' Load machine language I/O routine

140 GOSUB 300                     ' Set all program variables and reset the PC21

150 GOTO 740                      ' Initialize crate and run main program

160 TS=T:TB%=INT((LOG(T/1000))/2.30258):ERFLG%=0

170 TBAS = 10^TB%:T=T/TBAS:N=0

180 IF T<2 GOTO 200

190 N=N+1:T=T/2:GOTO 180

200 J=32*(T-1)+1

210 IOVF%=N-OVFTBL%(J)

220 IF IOVF%<0 THEN TB%=TB%-1:GOTO 170

230 IF IOVF%>15 THEN TB%=TB%+1:GOTO 170

240 TB%=TB%+6:IF TB%<0 OR TB%>7 THEN PRINT "Time outside range!":ERFLG%=1:GOTO 280

250 MULT% = MULTBL%(J)

260 TA=TBAS*(2*MULT%+1)*2^(IOVF%+4)

270 D%=(128!*MULT%)+(8!*IOVF%)+TB%:DSAVE%=D%

280 T=TS: RETURN

290 '   ****** Set all program variables ******

300 ADDRESS% = 768                ' PC21 base address

310 CONTROL = 96                  ' Normal state of PC21 Control Byte

320 CRASH = 4                     ' Mask for Control Bit 2 (BMA time-out)

330 FAULT = 32                    ' Mask for C.B. 5 (restart BMA)

340 PC21WRITE = 0!                ' Address of PC21WRITE subroutine

350 PC21READ = 49!                ' Address of PC21READ subroutine

360 '   ****** PC21 RESET ******

370 OUT ADDRESS%+1, ( CONTROL OR CRASH )          'Control Bit 2 high

380 OUT ADDRESS%+1, ( CONTROL AND NOT CRASH )   'Control Bit 2 low
```

```
390 FOR Y=1 TO 500:NEXT              'wait for BMA
400 OUT ADDRESS%+1, ( CONTROL AND NOT FAULT )   'Control Bit 5 low
410 OUT ADDRESS%+1, ( CONTROL OR FAULT )        'Control Bit 5 high
420 RETURN
430 '  ****** PC21 WRITE ******
440 BFLAG%=0
450 IF INSTR(COMMAND$,"W1") OR INSTR(COMMAND$,"w1") THEN BFLAG%=1
460 IF INSTR(COMMAND$,"PB") OR INSTR(COMMAND$,"pb") THEN BFLAG%=1
470 IF INSTR(COMMAND$,"X1B") OR INSTR(COMMAND$,"x1b") THEN BFLAG%=1
480 COMMAND$ = COMMAND$ + CHR$(13)          ' Add carriage return to command
490 CALL PC21WRITE(COMMAND$, ADDRESS%)       ' Execute machine language write
500 RETURN
510 '  *** Clear memory and poke in machine language routines ***
520 GOSUB 690                     ' Find data segment above BASICA
530 OPEN "CODE.BAS" FOR INPUT AS #1        ' Access machine code data file
540 FOR X = 0! TO 127!             '
550     INPUT #1, J               ' Install machine code
560     POKE X,J                  '
570 NEXT:CLOSE            '
580 GOTO 140
590 '  ****** PC21 READ ******
600 ANSWER$="         "+""         ' Reserve string space for response
610 CALL PC21READ( ANSWER$, ADDRESS%, BFLAG% )   ' Execute read
620 IF BFLAG%=0 THEN RETURN
630 NUM#=0:FOR X=1 TO 4
640 DIGIT%=ASC(MID$(ANSWER$,X,1))
650 NUM#=NUM#+DIGIT%*256^(4-X):NEXT
660 ANSWER$=STR$(NUM#)
670 RETURN                          ' BFLAG% identifies binary report commands
680 '  ****** Find data segment above BASICA ******
690 DEF SEG=0                 ' Go to low memory to find BASICA loc.
700 BDATSEG=PEEK(&H510)+256*PEEK(&H511)      ' BASICA data segment is in &H510-511
710 CAMSEG=BDATSEG+&H2000               ' Find next data segment after BASICA
720 DEF SEG=CAMSEG:RETURN
730 '  ****** Load CAMAC drivers and initialize crate ******
740 BLOAD "CAMIO",128              ' Load drivers into data segment
750 CAMO=&H80:CAMI=&H86:CAML=&H8C:CAMCL=&H92     ' Driver entry point addresses
760 CAMO24=&HB0:CAMI24=&HB6:CRATE=&HAA        ' Driver entry point addresses
770 CC%=1:CALL CRATE(CC%)            ' Activate controller in J1 slot
780 OUT &H240,0                ' Clear high write-only data register
790 I%=64:CALL CAMCL(I%)             ' Reset crate
```

```
800  I%=1:CALL CAMCL(I%)                    ' Initialize crate
810  N%=10:F%=17:A%=13:D%=1:GOSUB 1050           ' Write timer/scaler LAM mask-generate
820                              ' LAM when channel 1 finishes counting
830   PRINT
840   PRINT "Enter time (in seconds) to count (0 to exit program): "
850   INPUT T$
860 IF T$ = "0" GOTO 1070
870   PRINT
880   ISET = ISET +1: IF T$="" AND ISET >1 THEN D%=DSAVE%:GOTO 920
890   IF ISET=1 AND T$="" THEN T$="1"
900   T=VAL(T$):IF T<.000001 OR T>1000000! THEN PRINT "Out of Range":GOTO 840
910   GOSUB 160:IF ERFLG%<>0 THEN PRINT "Out of Range": GOTO 840
920   F%=17:A%=0:GOSUB 1050                    ' Set preset register
930   A%=4:D%=1:GOSUB 1050                     ' Start counting
940   CALL CAML(L%):IF L%=0 THEN 940              ' Wait for count to end
960   F%=23:A%=12:D%=1:GOSUB 1050              ' Clear timer LAM
970   F%=0:A%=1:CALL CAMI24(N%,F%,A%,D24%(1),Q%,X%)  ' Read scaler
980   SIGNAL#=D24%(1) AND 32767:IF D24%(1)<0 THEN SIGNAL#=SIGNAL#+32768#
985   '***SIGNAL#=D24%(1): IF D24%(1)<0 THEN SIGNAL#=SIGNAL#+32768#
990   FOR I=1 TO D24%(2):SIGNAL#=SIGNAL#+65536#:NEXT I
991   SIG=(SIGNAL#/TA)
1000   PRINT SIG;" counts in ";TA;" seconds."
1010 F%=16:A%=0:N%=12:D%=(32765.2/MAX)*SIGNAL#/TA
1011 IF D% > 32767 THEN D% = 32767
1012 CALL CAMO(N%,F%,A%,D%,Q%,X%)
1013 N%=10
1020   F%=9:A%=0:D%=0:GOSUB 1050                 ' Clear timer
1030   F%=9:A%=1:D%=0:GOSUB 1050                 ' Clear scaler
1040 D%=DSAVE%:GOTO 920                      ' Loop back for next scan
1050 CALL CAMO(N%,F%,A%,D%,Q%,X%)
1060 RETURN
1070 END
```

COVER.BAS

```
10  REM   Cover V1.0, (coverage decrease) modified 8/13/90 to increase DELAY(3)  (TW)
20  REM   Graphic boundaries and flag offset are set at 910-920  (TW)
30  REM   modified 08/20/92: solenoid operation, waitdelay  (WM)
40  REM        08/27/92: program structure  (WM)
50  REM     09/30/93: program structure (PRS)
60  REM   files needed    : hbasic.exe, basica.com, code.bas, camio.bas
70  '
80  ' *** Find data segment above BASICA ***
90  '
100  CLEAR: CLS                ' Set all variables to "0"
110  DEF SEG=0                 ' Go to low memory to find BASICA loc.
120  BDATSEG=PEEK(&H510)+256*PEEK(&H511) ' BASICA data segment is in &H510-511
130  CAMSEG=BDATSEG+&H2000          ' Find next data segment after BASICA
140  DEF SEG=CAMSEG
150  '
160  ' *** Open COM channel to Apple Comp. ***
170  '
180  OPEN "COM1:9600,N,8,1" AS #2
190  MSG$="READ"+CHR$(13):GOSUB 3830
200  '
210  ' *** Dimension of variables ***
220  '
230  DIM OVFTBL%(32),MULTBL%(32),DAC0(4),DAC2(4),TDEL(4),SIG(4)
240  DIM WVMTR#(3),D24%(2),LABEL$(4)
250  DATA 4,9,8,9,7,9,8,9,6,9,8,9,7,9,8,9,5,9,8,9,7,9,8,9,6,9,8,9,7,9,8,9
260  DATA 0,16,8,17,4,18,9,19,2,20,10,21,5,22,11,23,1,24,12,25,6,26,13,27
270  DATA 3,28,14,29,7,30,15,31
280  FOR I=1 TO 32: READ OVFTBL%(I): NEXT    'Load in tables for conversion of
290  FOR I=1 TO 32: READ MULTBL%(I): NEXT    'time.
300  '
310  ' *** Setting parameters ***
320  '
330  PRINT "y minimum value for screen plot : ";:INPUT PMIN
340  PRINT "y maximum value for screen plot : ";: INPUT PMAX
350  PFAC=(PMAX-PMIN)
360  XM=600: YM=400: YM2=YM/2
370  '
380  ' *** Program header ***
```

```
390 '

400 CLS

410 '

420 PRINT "Cover(1.0)": PRINT ""

430 PRINT "This program physically moves the DETECTOR to a user-specified"

440 PRINT "position and counts for a specified time with laser"

450 PRINT "on ion beam on, then counts for an equal time with laser on ion"

460 PRINT "beam off, then counts another period with both off and finally "

470 PRINT "counts with ion beam on and laser off and stores all"

480 PRINT "four counts on to disk."

490 BEEP: PRINT "": PRINT "Hit any key when ready ..."

500 A$=INKEY$: IF A$="" THEN 500

510 '

520 ' *** Poke in machine language routinge ***

530 '

540 PRINT "": PRINT "Loading PC21 & CAMAC I/O drivers ..."

550 '

560 OPEN "C:\BASICDIR\NEW\CODE.BAS" FOR INPUT AS #1     ' Access machine code data file

570 FOR X = 0! TO 127!

580   INPUT #1, J                        ' Install machine code

590   POKE X,J

600 NEXT

610 CLOSE #1

620 '

630 ' *** Set all program variables ***

640 '

650 ADDRESS% = 768          ' PC21 base address

660 CONTROL = 96             ' Normal state of PC21 Control Byte

670 CRASH = 4               ' Mask for Control Bit 2 (BMA time-out)

680 FAULT = 32              ' Mask for C.B. 5 (restart BMA)

690 PC21WRITE = 0!           ' Address of PC21WRITE subroutine

700 PC21READ = 49!           ' Address of PC21READ subroutine

710 '

720 DAC# = 12              ' DAC module in CAMAC slot #12

730 '             ch#: 0:shutter, 1:shutter, 2:ion beam shutter

740 '                  3:not used

750 TIM# = 10             ' TIMER/SCALER module in CAMAC slot #10

760 DELAY1# = 3           ' Delaytime for solenoid shutter in sec.

770 DELAY2# = 3           ' Delaytime for ion beam response in sec.

780 '

790 DATDIR$ = "C:\SPUTTER\COVER\"     ' Directory for data files
```

```
800  '
810  CMSG$="                                "   ' Clear String
820  '
830  FLGOFF = 2.5           ' Offset for 0 degree detector position
840  MAXANG = 110            ' Maximum detector position due to FIBER !!!
850  '
860  ' *** PC21 RESET ***
870  '
880  OUT ADDRESS%+1, ( CONTROL OR CRASH )         ' Control Bit 2 high
890  OUT ADDRESS%+1, ( CONTROL AND NOT CRASH )   ' Control Bit 2 low
900  FOR Y=1 TO 500: NEXT            ' Wait for BMA
910  OUT ADDRESS%+1, ( CONTROL AND NOT FAULT )   ' Control Bit 5 low
920  OUT ADDRESS%+1, ( CONTROL OR FAULT )         ' Control Bit 5 high
930  '
940  ' *** Load CAMAC drivers and initialize crate ***
950  '
960  BLOAD "C:\BASICDIR\NEW\CAMIO",128      ' Load drivers into data segment
970  CAMO=&H80:CAMI=&H86:CAML=&H8C:CAMCL=&H92 ' Driver entry point addresses
980  CAMO24=&HB0:CAMI24=&HB6:CRATE=&HAA      ' Driver entry point addresses
990  CC%=1: CALL CRATE(CC%)           ' Activate controller in J1 slot
1000 OUT &H240,0                 ' Clear high write-only data register
1010 I%=64: CALL CAMCL(I%)           ' Reset crate
1020 I%=1: CALL CAMCL(I%)            ' Initialize crate
1030 N%=TIM#:F%=17:A%=13:D%=1:GOSUB 2960  ' Write timer/scaler LAM mask-generate
1040 '                  LAM when channel 1 finishes counting
1050 GOSUB 3070: GOSUB 3530
1060 '
1070 ' *** Initialization ***
1080 '
1090 A1%=0:D1%=0: GOSUB 3610    ' Laser beam shutter should be open!
1100 PRINT "Laser beam shutter:  OPEN!"
1110 N%=DAC#:F%=16:A%=2:D%=32700: GOSUB 2960 ' Ion beam shutter should be open!
1120 PRINT "Ion beam shutter :  OPEN!"
1130 '
1140 PRINT "Detector     :  Moving to 0 degree!"
1150 COMMAND$="FSB1 FSC1 MN A1 V.1": GOSUB 2610   ' Moving detector to 0 degree!
1160 POS0=0: POS1=2: GOSUB 2810           ' Make sure not on an endpoint
1170 POS0=0: POS1=-10: GOSUB 2810
1180 IF ENDFLG=0 THEN 1170
1190 POS0=0: POS1=FLGOFF: GOSUB 2810: POS0=0   ' Compensate for offset
1200 '
```

```
1210 ' *** Define data file name ***

1220 '

1230 PRINT ""

1240 BEEP: PRINT "Data file name: ";: INPUT FILNAM$

1250 ON ERROR GOTO 1340

1260 OPEN DATDIR$+FILNAM$ FOR INPUT AS #1  ' Test to see if file already exists.

1270 '                          File exists.  Ask user what to do!!

1280 BEEP: BEEP

1290 PRINT "File exists! Continue [y/n] : ? "

1300 V$=INKEY$: IF V$="" THEN GOTO 1300

1310 IF V$="Y" OR V$="y" THEN GOTO 1350  ' Overwrite existing file.

1320 IF V$="N" OR V$="n" THEN GOTO 1240   ' Get a new name for file.

1330 GOTO 1300

1340 RESUME 1350                ' File nonexistent.  Proceed.

1350 ON ERROR GOTO 0

1360 CLOSE #1               ' Close temporary input file.

1370 OPEN DATDIR$+FILNAM$ FOR OUTPUT AS #1  ' Open output file.

1380 '

1390 ' *** Print file header ***

1400 '

1410 PRINT #1,"Data stored by COVER1.0 on ";DATE$;" at ";TIME$

1420 PRINT "File header: ";: INPUT HDR$

1430 PRINT #1,"File Header = ";HDR$

1440 IF NOT EOF(2) THEN ASMSG$=INPUT$(1,#2): GOTO 1430   'switch to settle.

1450 '

1460 ' *** Enter parameter for measurement ***

1470 '

1480 BEEP: PRINT "": PRINT "Integration time (> 1 sec): ";: INPUT ITIM

1490 IF ITIM<1 THEN ITIM=1

1500 IF ITIM>10000! THEN BEEP: BEEP: PRINT "Out of Range": GOTO 1480: PRINT ""

1510 T=ITIM: GOSUB 3360: TSAVE%=TIMEC%

1520 MSG$=" sec integration.": PRINT #1,STR$(T);MSG$

1530 PRINT "Detector angle        : ";: INPUT ANG0

1540 IF ANG0 > MAXANG THEN PRINT "Position out of limit!!!": GOTO 1530

1550 PRINT #1, "Detector angle : ";ANG0

1560 PRINT "Ion sputter angle : ";: INPUT A1

1570 PRINT #1, " Ion sputter angle : ";A1

1580 PRINT "Sample Metal : ";: INPUT B1$

1590 PRINT #1, "Sample Metal : ";B1$

1600 PRINT "Sputter Ion : ";:InPUT B2$

1610 PRINT #1, "Sputter Ion : ";B2$
```

```
1620 PRINT "photomultiplier voltage :";: INPUT A2
1630 PRINT #1, " photomultiplier voltage :  ";A2
1640 PRINT "ion beam current :";:INPUT A3
1650 PRINT #1, "ion beam current: ";A3
1660 PRINT " Ion beam voltage : ";:INPUT A5
1670 PRINT #1, " Ion beam voltage : ";A5
1680 PRINT "NUMBER OF STEPS(EACH STEP=10+(4*integration time): ";: INPUT NINC
1690 PRINT "COUNTING AT ";ANG0;" degrees FOR ";CINT(NINC*(10+4*T)/60);" minutes."
1700 PRINT "Moving detector to ";ANG0;" degrees"
1710 POS0=0: POS1=ANG0: GOSUB 2810
1720 IF ENDFLG<>0 THEN BEEP: BEEP: BEEP: PRINT "ERROR!  Limit switch hit!": STOP
1730 BEEP: PRINT "Laser frequency (cm^-1)   : ";: INPUT NRG#
1740 PRINT #1, "Laser frequency (cm^-1)   : "; NRG#
1750 PRINT "laser power (milliwatts) :";: INPUT A4
1760 PRINT #1, "Laser power (milliwatts) :"; A4
1770 PRINT #1, " I All on I - I Laser on I - I ion on I + I dark count I = I Total count I"
1780 NRG$=STR$(NRG#): MSG$="GOK"+NRG$+CHR$(13): GOSUB 3830  ' Tell AUTOSCAN to go there.
1790 '
1800 ' *** Initialize plotting on the screen ***
1810 '
1820 CLS: SCREEN 2: DRAW "BM 0,0 R=XM;D=YM2;L=XM;U=YM2;"
1830 LOCATE 15,1: PRINT ANG0: LOCATE 16,1: PRINT NRG#
1840 LOCATE 2,50: PRINT FILNAM$
1850 DELX=XM/NINC                    'DELX used for plotting
1860 '
1870 ' *** Starting the data collection ***
1880 '
1890 DAC0(1)=6540:DAC0(2)=0:DAC0(3)=6540:DAC0(4)=0  ' Settings for DAC
1900 DAC2(1)=0:DAC2(2)=0:DAC2(3)=32700:DAC2(4)=32700
1910 TDEL(1)=DELAY1#:TDEL(2)=DELAY1#:TDEL(3)=DELAY1#:TDEL(4)=DELAY1#  ' Setting delay after solenoid
and ion beam
1920 LABEL$(1)="All off":LABEL$(2)="Laser ON":LABEL$(3)="Ion beam ON":LABEL$(4)="All on"
1930 '
1940 ' *** NSP data collection loop ***
1950 '
1960   CLS: SCREEN 2: DRAW "BM 0,0 R=XM;D=YM2;L=XM;U=YM2;"
1970   LOCATE 15,1: PRINT ANG0: LOCATE 16,1: PRINT NRG#
1980   LOCATE 2,50: PRINT FILNAM$
1990   X0%=0                    ' Initialize x-axis of plot
2000 FOR NSP=1 TO NINC
2010   MSG$="READ"+CHR$(13): GOSUB 3670  ' and verify wavelength.
```

```
2020  '****
2030  WVMTR#= VAL(MID$(RMSG$,2,10))
2040  DELNRG= WVMTR#-NRG#
2050  '****
2060  IF ABS(DELNRG)>.006 THEN BEEP: BEEP: MSG$="GOK"+STR$(NRG#)+CHR$(13): GOSUB 3670
2070  AVDEL= 0
2080  FOR II=1 TO 3
2090  MSG$="READ"+CHR$(13): GOSUB 3670   ' and verify wavelength.
2100  WVMTR#(II)=VAL(MID$(RMSG$,2,10))
2110  DELNRG=WVMTR#(II)-NRG#
2120  IF ABS(DELNRG)<=.001 THEN GOTO 2190
2130  IF ABS(DELNRG)>.001 THEN AVDEL=AVDEL+DELNRG
2140  NEXT II
2150  AVDEL=AVDEL/3!
2160  MSG$="DELK"+STR$(-AVDEL)+CHR$(13): GOSUB 3670
2170  GOTO 2070
2180  '
2190    TIM=TIMER
2200    IF TIMER-TIM<.5 THEN 2200
2210    LOCATE 18,1: PRINT "            "
2220    LOCATE 18,1: PRINT USING "Angle = #######.###";POS0
2230    '
2240    FOR DACSW=1 TO 4
2250      LOCATE 18,30: PRINT "                "
2260      LOCATE 18,30: PRINT LABEL$(DACSW)
2270      N%=DAC#:F%=16:A%=2:D%=DAC2(DACSW): GOSUB 2960        ' Set DAC on/off depending on
2280      A1%=0:D1%=DAC0(DACSW): GOSUB 3610 ' Dacsw
2290      T=TDEL(DACSW): GOSUB 3360: GOSUB 3210      ' Wait for time delay!
2300      TIMEC%=TSAVE%: GOSUB 3120              ' now Count for TSAVE%
2310      SIGNAL#=D24%(1) AND 32767: IF D24%(1)<0 THEN SIGNAL#=SIGNAL#+32768#
2320      FOR I=1 TO D24%(2): SIGNAL#=SIGNAL#+65536#: NEXT I
2330      SIG(DACSW)=SIGNAL#/ITIM: GOSUB 3290
2340      NEXT DACSW
2350      '
2360      '***LASERON=LASERON+SIG(1): IONON=IONON+SIG(2): ALLON=ALLON+SIG(3)
2370      ' NEXT NC  ' Save signal
2380      '***DELSIG=ALLON-LASERON-IONON+DRKCNT: PRINT #1,ALLON,LASERON,IONON
2390      DELSIG=SIG(4)-SIG(2)-SIG(3)+SIG(1)
2400      PRINT #1,SIG(4),SIG(2),SIG(3),SIG(1),CINT(DELSIG)
2410      LOCATE 19,1: PRINT "                         "
```

```
2420    '***LOCATE 19,1: PRINT ALLON;"-";LASERON;"-";IONON;"+";DRKCNT;"=";DELSIG;':LPRINT
POS0,DELSIG

2430    LOCATE 19,1: PRINT NSP;": ";SIG(4);"-";SIG(2);"-";SIG(3);"+";SIG(1);"=";CINT(DELSIG)

2440    YP%=(YM/PFAC)*(pmax-delsig): XP%=(NSP-1)*DELX: IF YP%>YM2 THEN YP%=YM2

2450    DRAW "BM=X0%;,=Y0%;": DRAW "M=XP%;,=YP%;": Y0%=YP%: X0%=XP%

2460    SIG(1)=0:SIG(2)=0:SIG(3)=0:SIG(4)=0  'Reset accumulated signals for next frequency

2470    '

2480 NEXT NSP

2490 '

2500 CLOSE

2510 LOCATE 23,1: BEEP: BEEP: PRINT ">>>(EXIT Angscan1.0 with 'E')"

2520 A$=INKEY$: IF (A$="E") OR (A$="e") THEN CLS ELSE 2520

2530 END

2540 '

2550 '

2560 ' *** SUBROUTINES ARE LOCATED HERE !!! ***

2570 '

2580 ' *** PC21 WRITE ***

2590 '

2600 BFLAG%=0

2610 IF INSTR(COMMAND$,"W1") OR INSTR(COMMAND$,"w1") THEN BFLAG%=1

2620 IF INSTR(COMMAND$,"PB") OR INSTR(COMMAND$,"pb") THEN BFLAG%=1

2630 IF INSTR(COMMAND$,"X1B") OR INSTR(COMMAND$,"x1b") THEN BFLAG%=1

2640 COMMAND$ = COMMAND$ + CHR$(13)     ' Add carriage return to command

2650 CALL PC21WRITE(COMMAND$, ADDRESS%)  ' Execute machine language write

2660 RETURN

2670 '

2680 ' *** PC21 READ ***

2690 '

2700 ANSWER$="             "+""    ' Reserve string space for response

2710 CALL PC21READ( ANSWER$, ADDRESS%, BFLAG% )  ' Execute read

2720 IF BFLAG%=0 THEN RETURN

2730 NUM#=0: FOR X=1 TO 4

2740 DIGIT%=ASC(MID$(ANSWER$,X,1))

2750 NUM#=NUM#+DIGIT%*256^(4-X): NEXT

2760 ANSWER$=STR$(NUM#)

2770 RETURN                          ' BFLAG% identifies binary report commands

2780 '

2790 ' *** Move Detector POS1-POS0 degrees (check for end switch) ***

2800 '

2810 DEG=ABS(POS1-POS0): DIR$="+": IF POS1-POS0<0 THEN DIR$="-"
```

```
2820 AVTIM=DEG*.15: ENDFLG=0: IF AVTIM<.5 THEN AVTIM=.5

2830 STEPS=INT(DEG*4000/180+.5)        ' Convert degrees to motor steps

2840 STEPS$=MID$(STR$(STEPS),2)

2850 DEGAC=STEPS*180/4000: IF DIR$="-" THEN DEGAC=-DEGAC

2860 COMMAND$="D"+DIR$+STEPS$+" G CR P": GOSUB 2610 ' Move & signal when done

2870 T0=TIMER

2880 GOSUB 2700: IF LEFT$(ANSWER$,1)=CHR$(13) THEN 2910  ' Keep going till end

2890 T1=TIMER: IF T1-T0<AVTIM THEN 2880  ' If not done after AVTIM,

2900 ENDFLG=1: GOTO 2920              ' End switch must have been hit.

2910 POS0=POS0+DEGAC

2920 RETURN

2930 '

2940 ' *** Main WRITE/READ Subroutines to CAMAC ***

2950 '

2960 CALL CAMO(N%,F%,A%,D%,Q%,X%)

2970 RETURN

2980 '

2990 CALL CAMI24(N%,F%,A%,D24%(1),Q%,X%)

3000 RETURN

3010 '

3020 CALL CAMI(N%,F%,A%,D%,Q%,X%)

3030 RETURN

3040 '

3050 ' *** initialize DAC, put all channels (0...3) to OV! ***

3060 '

3070 FOR I=0 TO 3: N%=DAC#:F%=16:A%=I:D%=0: GOSUB 2960: NEXT

3080 RETURN

3090 '

3100 ' *** Set timer/scaler ***

3110 '

3120 N%=TIM#:F%=17:A%=0:D%=TIMEC%: GOSUB 2960   ' D% must contain time.

3130 N%=TIM#:F%=17:A%=4:D%=1: GOSUB 2960          ' Start counting

3140 CALL CAML(L%):IF L%=0 THEN 3140              ' Wait for count to end

3150 N%=TIM#:F%=23:A%=12:D%=1: GOSUB 2960         ' Clear timer LAM

3160 N%=TIM#:F%=0:A%=1: GOSUB 2990: GOTO 3290    ' Read scaler

3170 RETURN

3180 '

3190 ' *** Wait delay ***

3200 '

3210 N%=TIM#:F%=17:A%=0:D%=TIMEC%: GOSUB 2960    ' D% must contain time!

3220 N%=TIM#:F%=17:A%=4:D%=1: GOSUB 2960          ' Start wait delay
```

```
3230 CALL CAML(L%): IF L%=0 THEN 3230          ' Wait for time to end
3240 N%=TIM#:F%=23:A%=12:D%=1: GOSUB 2960: GOTO 3290 ' Clear timer LAM
3250 RETURN
3260 '
3270 ' *** Clear timer/scaler ***
3280 '
3290 N%=TIM#:F%=9:A%=0:D%=0: GOSUB 2960
3300 N%=TIM#:F%=9:A%=1:D%=0: GOSUB 2960
3310 RETURN
3320 '
3330 ' *** Subroutine for converting time (T) to a special
3340 '     formatted integer (TIMEC%) used by CAMAC timer. ***
3350 '
3360 TS=T: TB%=INT((LOG(T/1000))/2.30258): ERFLG%=0
3370 TBAS=10^TB%: T=T/TBAS: N=0
3380 IF T<2 GOTO 3400
3390 N=N+1: T=T/2: GOTO 3380
3400 J=32*(T-1)+1
3410 IOVF%=N-OVFTBL%(J)
3420 IF IOVF%<0 THEN TB%=TB%-1: GOTO 3370
3430 IF IOVF%>15 THEN TB%=TB%+1: GOTO 3370
3440 TB%=TB%+6: IF TB%<0 OR TB%>7 THEN PRINT "Time requested outside range": ERFLG%=1: GOTO 3480
3450 MULT%=MULTBL%(J)
3460 TA=TBAS*(2*MULT%+1)*2^(IOVF%+4)
3470 TIMEC%=(128!*MULT%)+(8!*IOVF%)+TB%
3480 T=TS
3490 RETURN
3500 '
3510 ' *** Initialize the solenoid shutter (result: shutter is CLOSED!) ***
3520 '
3530 N%=DAC#:F%=16:A%=1:D%=32700: GOSUB 2960
3540 N%=DAC#:F%=16:A%=0:D%=32700: GOSUB 2960
3550 N%=DAC#:F%=16:A%=0:D%=0: GOSUB 2960
3560 N%=DAC#:F%=16:A%=0:D%=32700: GOSUB 2960
3570 RETURN
3580 '
3590 ' *** Open/close solenoid shutter ***
3600 '
3610 N%=DAC#:F%=16:A%=A1%:D%=D1%: GOSUB 2960
3630 RETURN
3640 '
```

```
3650 ' *** Send Message to Apple via RS232 ***

3660 '

3670 LOCATE 20,1: PRINT ">>> Remote Apple control: "

3680 LOCATE 21,1: PRINT CMSG$: LOCATE 21,1

3690 FOR IMSG=1 TO LEN(MSG$): V$=MID$(MSG$,IMSG,1): PRINT #2,V$;: NEXT

3700 PRINT MSG$: RMSG$=""

3710 IF EOF(2) THEN 3710

3720 WHILE NOT EOF(2)

3730 ASMSG$=INPUT$(1,#2): ASMSG$=CHR$(ASC(ASMSG$)-128)

3740 IF ASMSG$=CHR$(13) GOTO 3790

3750 RMSG$=RMSG$+ASMSG$: WEND

3760 IF ASC(ASMSG$)<>41 THEN GOTO 3710

3770 RETURN

3780 '

3790 LOCATE 21,1: PRINT CMSG$;: LOCATE 22,1: PRINT CMSG$;

3800 PRMSG$=RMSG$: RMSG$="": LOCATE 21,1: PRINT PRMSG$;

3810 GOTO 3750

3820 '

3830 FOR IMSG=1 TO LEN(MSG$): V$=MID$(MSG$,IMSG,1): PRINT #2,V$;: NEXT

3840 PRINT RMSG$=""

3850 IF EOF(2) THEN 3850

3860 WHILE NOT EOF(2)

3870 ASMSG$=INPUT$(1,#2): ASMSG$=CHR$(ASC(ASMSG$)-128)

3880 IF ASMSG$=CHR$(13) THEN RMSG$=""

3890 PRINT ASMSG$;: RMSG$=RMSG$+ASMSG$: WEND

3900 IF ASC(ASMSG$)<>41 THEN GOTO 3850

3910 RETURN
```

DACTEST.BAS

```
10 CLS:PRINT "This program sets the 4 channel DAC to a specified voltage"
12 PRINT "Paul/Wolfram: 08/17/92": PRINT ""
20 PRINT "Now loading PC21 and CAMAC I/O drivers..."
30 GOTO 290   'Load machine language I/O routine
40 GOSUB 70   ' Set all program variables and reset the PC21
50 GOTO 510   ' Initialize crate and run main program
60 '   ****** Set all program variables ******
70 ADDRESS% = 768           ' PC21 base address
80 CONTROL = 96             ' Normal state of PC21 Control Byte
90 CRASH = 4             ' Mask for Control Bit 2 (BMA time-out)
100 FAULT = 32             ' Mask for C.B. 5 (restart BMA)
110 PC21WRITE = 0!          ' Address of PC21WRITE subroutine
120 PC21READ = 49!           ' Address of PC21READ subroutine
130 '   ****** PC21 RESET ******
140 OUT ADDRESS%+1, ( CONTROL OR CRASH )         'Control Bit 2 high
150 OUT ADDRESS%+1, ( CONTROL AND NOT CRASH )   'Control Bit 2 low
160 FOR Y=1 TO 500:NEXT           'wait for BMA
170 OUT ADDRESS%+1, ( CONTROL AND NOT FAULT )  'Control Bit 5 low
180 OUT ADDRESS%+1, ( CONTROL OR FAULT )         'Control Bit 5 high
190 RETURN
200 '   ****** PC21 WRITE ******
210 BFLAG%=0
220 IF INSTR(COMMANDX$,"W1") OR INSTR(COMMANDX$,"w1") THEN BFLAG%=1
230 IF INSTR(COMMANDX$,"PB") OR INSTR(COMMANDX$,"pb") THEN BFLAG%=1
240 IF INSTR(COMMANDX$,"X1B") OR INSTR(COMMANDX$,"x1b") THEN BFLAG%=1
250 COMMANDX$ = COMMANDX$ + CHR$(13)     ' Add carriage return to command
260 CALL PC21WRITE(COMMANDX$, ADDRESS%)' Execute machine language write
270 RETURN
280 '   *** Clear memory and poke in machine language routines ***
290 GOSUB 460             ' Find data segment above BASICA
300 OPEN "CODE.BAS" FOR INPUT AS #1    ' Access machine code data file
310 FOR X = 0! TO 127!          '
320      INPUT #1, J             ' Install machine code
330      POKE X,J              '
340 NEXT:CLOSE          '
350 GOTO 40
360 '   ****** PC21 READ ******
370 ANSWER$="          "+""  ' Reserve string space for response
```

```
380 CALL PC21READ( ANSWER$, ADDRESS%, BFLAG% ) ' Execute read
390 IF BFLAG%=0 THEN RETURN
400 NUM#=0:FOR X=1 TO 4
410 DIGIT%=ASC(MID$(ANSWER$,X,1))
420 NUM#=NUM#+DIGIT%*256^(4-X):NEXT
430 ANSWER$=STR$(NUM#)
440 RETURN                          ' BFLAG% identifies binary report commands
450 ' ****** Find data segment above BASICA ******
460 DEF SEG=0                 ' Go to low memory to find BASICA loc.
470 BDATSEG=PEEK(&H510)+256*PEEK(&H511) ' BASICA data segment is in &H510-511
480 CAMSEG=BDATSEG+&H2000            ' Find next data segment after BASICA
490 DEF SEG=CAMSEG:RETURN
500 ' ****** Load CAMAC drivers and initialize crate ******
510 BLOAD "CAMIO",128            ' Load drivers into data segment
520 CAMO=&H80:CAMI=&H86:CAML=&H8C:CAMCL=&H92 ' Driver entry point addresses
530 CAMO24=&HB0:CAMI24=&HB6:CRATE=&HAA        ' Driver entry point addresses
540 CC%=1:CALL CRATE(CC%)         ' Activate controller in J1 slot
550 OUT &H240,0                 ' Clear high write-only data register
560 I%=64:CALL CAMCL(I%)          ' Reset crate
570 I%=1:CALL CAMCL(I%)           ' Initialize crate
580 ' N%=10:F%=17:A%=13:D%=1:GOSUB 630 ' Write timer/scaler LAM mask-generate
590                      ' LAM when channel 1 finishes counting
591 '
592 '
598 ' set all DAC channels to 0V !
599 FOR I=0 TO 3: N%=12:F%=16:A%=I:D%=0: GOSUB 630: NEXT I
609 PRINT "EXIT program by CTRL BREAK": PRINT ""
610 PRINT "DAC channel # : ";: INPUT A%: IF (A%>3 OR A%<0) THEN PRINT "": GOTO 610
611 PRINT "DAC voltage   : ";: INPUT DAC: IF (DAC>5 OR DAC<-5) THEN PRINT "Input out of range": GOTO
611
612 DAC=DAC*1.000: DAC%=INT(DAC*32767/5)
620 PRINT "#: ";A%;"  DAC: ";: PRINT USING "#.###";DAC: PRINT "": D%=DAC%: GOSUB 630
621 GOTO 610
630 CALL CAMO(N%,F%,A%,D%,Q%,X%): RETURN
640 END
```

DETPOS.BAS

```
10   REM   DETPOS
20   REM   Wolfram: 08/20/92  moves detector to a specified position!
21   '
30   GOTO 890                    ' Jump over subroutines
31   '
32   '
33   '
40   ' *******   Subroutines go here    ************
41   '
42   '
50   ' ****** PC21 WRITE ******
51   '
60   BFLAG%=0
70   IF INSTR(COMMANDX$,"W1") OR INSTR(COMMANDX$,"w1") THEN BFLAG%=1
80   IF INSTR(COMMANDX$,"PB") OR INSTR(COMMANDX$,"pb") THEN BFLAG%=1
90   IF INSTR(COMMANDX$,"X1B") OR INSTR(COMMANDX$,"x1b") THEN BFLAG%=1
100  COMMANDX$=COMMANDX$+CHR$(13)     ' Add carriage return to command
110  CALL PC21WRITE(COMMANDX$, ADDRESS%) ' Execute machine language write
120  RETURN
121  '
122  '
130  ' ****** PC21 READ ******
131  '
140  ANSWER$="              "+""   ' Reserve string space for response
150  CALL PC21READ( ANSWER$, ADDRESS%, BFLAG% ) ' Execute read
160  IF BFLAG%=0 THEN RETURN
170  NUM#=0: FOR X=1 TO 4
180  DIGIT%=ASC(MID$(ANSWER$,X,1))
190  NUM#=NUM#+DIGIT%*256^(4-X): NEXT
200  ANSWER$=STR$(NUM#)
210  RETURN                          ' BFLAG% identifies binary report commands
211  '
212  '
220  '******* Move Detector POS1-POS0 degrees (check for end switch) ******
221  '
230  DEG=ABS(POS1-POS0): DIR$="+": IF POS1-POS0<0 THEN DIR$="-"
240  AVTIM=DEG*.15: ENDFLG=0: IF AVTIM<.5 THEN AVTIM=.5
250  STEPS=INT(DEG*4000/180+.5)       ' Convert degrees to motor steps
```

```
260  STEPS$=MID$(STR$(STEPS),2)

270  DEGAC=STEPS*180/4000: IF DIR$="-" THEN DEGAC=-DEGAC

280  COMMANDX$="D"+DIR$+STEPS$+" G CR P": GOSUB 60 ' Move & signal when done

290  T0=TIMER

300  GOSUB 140: IF LEFT$(ANSWER$,1)=CHR$(13) THEN 330  ' Keep going till end

310  T1=TIMER: IF T1-T0<AVTIM THEN 300  'If not done after AVTIM,

320  ENDFLG=1: GOTO 340              'End switch must have been hit.

330  POS0=POS0+DEGAC

340  RETURN

341  '

342  '

870  '    ********* End Subroutines  **********

871  '

872  '

880  ' ****** Find data segment above BASICA and do initial settings ******

881  '

890  CLEAR: CLS               ' Set all variables to "0"

891  '

900  DEF SEG=0                ' Go to low memory to find BASICA loc.

910  BDATSEG=PEEK(&H510)+256*PEEK(&H511) ' BASICA data segment is in &H510-511

920  CAMSEG=BDATSEG+&H2000            ' Find next data segment after BASICA

930  DEF SEG=CAMSEG

931  '

951  '

1120 PRINT: PRINT "DETPOS: Moving the detector to a specified position"

1130 PRINT: PRINT "Now loading PC21 I/O drivers ..."

1131 '

1140 '  *** Poke in machine language routinge ***

1150 OPEN "CODE.BAS" FOR INPUT AS #1    ' Access machine code data file

1160 FOR X = 0! TO 127!            '

1170   INPUT #1, J            ' Install machine code

1180   POKE X,J              '

1190 NEXT

1200 CLOSE #1

1201 '

1202 '

1210 ' ****** Set all program variables ******

1211 '

1220 ADDRESS% = 768              ' PC21 base address

1230 CONTROL = 96               ' Normal state of PC21 Control Byte

1240 CRASH = 4              ' Mask for Control Bit 2 (BMA time-out)
```

```
1250 FAULT = 32              ' Mask for C.B. 5 (restart BMA)
1260 PC21WRITE = 0!          ' Address of PC21WRITE subroutine
1270 PC21READ = 49!          ' Address of PC21READ subroutine
1271 '
1275 FLGOFF = 2.5            ' Offset for 0 degree detector position!
1276 MaxAng = 110            ' Maximum detector position due to FIBER !!!
1277 '
1300 '  ****** PC21 RESET ******
1301 '
1310 OUT ADDRESS%+1, ( CONTROL OR CRASH )        'Control Bit 2 high
1320 OUT ADDRESS%+1, ( CONTROL AND NOT CRASH )   'Control Bit 2 low
1330 FOR Y=1 TO 500: NEXT             'wait for BMA
1340 OUT ADDRESS%+1, ( CONTROL AND NOT FAULT )   'Control Bit 5 low
1350 OUT ADDRESS%+1, ( CONTROL OR FAULT )        'Control Bit 5 high
1351 '
1450 '  ****** Program Begins Here ******
1451 '
1460 '  >>> Initialization <<<
1461 '
1490 '  ****** Send Detector to HOME position  ******
1491 '
1500 PRINT "Moving detector to HOME position!"
1510 COMMANDX$="FSB1 FSC1 MN A1 V.1": GOSUB 60    'Moving detector to 0 degree!
1520 POS0=0: POS1=2: GOSUB 230            'Make sure not on an endpoint
1530 POS0=0: POS1=-10: GOSUB 230
1540 IF ENDFLG=0 THEN 1530
1550 POS0=0: POS1=FLGOFF: GOSUB 230: POS0=0     'compensate for offset
1551 '
1781 '
1790 '  >>> Enter parameter for measurement <<<
1791 '
1870 ANG0=0
1880 PRINT "": PRINT "Desired detector position : ";: INPUT ANG1
1890 IF ANG1 > MaxAng THEN PRINT "Position out of limit!!!": GOTO 1880
1930 PRINT "Moving detector to ";ANG1;" degree"
1940 POS0=ANG0: POS1=ANG1: GOSUB 230
1950 IF ENDFLG<>0 THEN PRINT "ERROR!  Limit switch hit!": STOP
1970 ANG0=ANG1: GOTO 1880
1980 POS0=ANG0: POS1=ANG1: GOSUB 230
2612 '
2620 CLOSE
```

2630 END

FREQSCAN.BAS

```
10   REM   FREQSCAN V1.0, (MOD2CNT) modified 8/13/90 to increase DELAY(3)   (TW)

20   REM   Graphic boundaries and flag offset are set at 910-920   (TW)

30   REM   modified 08/20/92: solenoid operation, waitdelay   (WM)

40   REM        08/27/92: program structure   (WM)

50   REM        04/30/93: changed check for frequency   (WM)

60   REM   files needed    : hbasic.exe, basica.com, code.bas, camio.bas

70   '

80   ' *** Find data segment above BASICA ***

90   '

100  CLEAR: CLS                    ' Set all variables to "0"

110  DEF SEG=0                     ' Go to low memory to find BASICA loc.

120  BDATSEG=PEEK(&H510)+256*PEEK(&H511) ' BASICA data segment is in &H510-511

130  CAMSEG=BDATSEG+&H2000          ' Find next data segment after BASICA

140  DEF SEG=CAMSEG

150  '

160  ' *** Open COM channel to Apple Comp. ***

170  '

180  OPEN "COM1:9600,N,8,1,CS3000,DS3000" AS #2

190  '

200  ' *** Dimension of variables ***

210  '

220  DIM OVFTBL%(32),MULTBL%(32),DAC0(3),DAC1(3),DAC2(3),DELAY(3),SIG(3)

230  DIM WVMTR#(3),D24%(2),LABEL$(3)

240  DATA 4,9,8,9,7,9,8,9,6,9,8,9,7,9,8,9,5,9,8,9,7,9,8,9,6,9,8,9,7,9,8,9

250  DATA 0,16,8,17,4,18,9,19,2,20,10,21,5,22,11,23,1,24,12,25,6,26,13,27

260  DATA 3,28,14,29,7,30,15,31

270  FOR I=1 TO 32: READ OVFTBL%(I): NEXT    'Load in tables for conversion of

280  FOR I=1 TO 32: READ MULTBL%(I): NEXT    'time.

290  '

300  ' *** Setting parameters ***

310  '

320  PMIN=-50: PMAX=5000

330  PFAC=(PMAX-PMIN)*2

340  XM=600: YM=400: YM2=YM/2     'XM=719: YM=347: YM2=YM/2

350  '

360  ' *** Program header ***

370  '

380  CLS:
```

```basic
390 '
400 PRINT "F R E Q S C A N (1.0)": PRINT ""
410 PRINT "This program counts number of photons for a user-specified time as"
420 PRINT "a function of the LASER FREQUENCY with the ion beam on hitting the"
430 PRINT "target and the laser beam off, then counts for an equal time with"
440 PRINT "the ion beam off and the laser beam on and then again with ion beam"
450 PRINT "and laser beam on and stores all three counts on to disk."
460 BEEP: PRINT "": PRINT "Hit any key when ready ..."
470 A$=INKEY$: IF A$="" THEN 470
480 '
490 ' *** Poke in machine language routinge ***
500 '
510 PRINT "": PRINT "Loading PC21 & CAMAC I/O drivers ..."
520 '
530 OPEN "C:\BASICDIR\NEW\CODE.BAS" FOR INPUT AS #1    ' Access machine code data file
540 FOR X = 0! TO 127!
550   INPUT #1, J                          ' Install machine code
560   POKE X,J
570 NEXT
580 CLOSE #1
590 '
600 ' *** Set all program variables ***
610 '
620 ADDRESS% = 768          ' PC21 base address
630 CONTROL = 96            ' Normal state of PC21 Control Byte
640 CRASH = 4              ' Mask for Control Bit 2 (BMA time-out)
650 FAULT = 32             ' Mask for C.B. 5 (restart BMA)
660 PC21WRITE = 0!          ' Address of PC21WRITE subroutine
670 PC21READ = 49!          ' Address of PC21READ subroutine
680 '
690 DAC# = 12             ' DAC module in CAMAC slot #12
700 '                  ch#: 0:shutter, 1:shutter, 2:ion beam shutter
710 '                       3:not used
720 TIM# = 10             ' TIMER/SCALER module in CAMAC slot #10
730 DELAY1# = 1            ' Delaytime for solenoid shutter in sec.
740 DELAY2# = 1            ' Delaytime for ion beam response in sec.
750 '
760 DATDIR$ = "C:\SPUTTER\FREQDAT\"    ' Directory for data files
770 '
780 CMSG$="                           "    ' Clear String
790 '
```

```
800  FLGOFF = 2.5              ' Offset for 0 degree position
810  MAXANG = 110                ' Maximum position of detector due to FIBER!!!
820  '
830  ' *** PC21 RESET ***
840  '
850  OUT ADDRESS%+1, ( CONTROL OR CRASH )          ' Control Bit 2 high
860  OUT ADDRESS%+1, ( CONTROL AND NOT CRASH ) ' Control Bit 2 low
870  FOR Y=1 TO 500: NEXT             ' Wait for BMA
880  OUT ADDRESS%+1, ( CONTROL AND NOT FAULT ) ' Control Bit 5 low
890  OUT ADDRESS%+1, ( CONTROL OR FAULT )          ' Control Bit 5 high
900  '
910  ' *** Load CAMAC drivers and initialize crate ***
920  '
930  BLOAD "C:\BASICDIR\NEW\CAMIO",128        ' Load drivers into data segment
940  CAMO=&H80:CAMI=&H86:CAML=&H8C:CAMCL=&H92 ' Driver entry point addresses
950  CAMO24=&HB0:CAMI24=&HB6:CRATE=&HAA        ' Driver entry point addresses
960  CC%=1: CALL CRATE(CC%)            ' Activate controller in J1 slot
970  OUT &H240,0                 ' Clear high write-only data register
980  I%=64: CALL CAMCL(I%)           ' Reset crate
990  I%=1: CALL CAMCL(I%)            ' Initialize crate
1000 N%=TIM#:F%=17:A%=13:D%=1:GOSUB 3300  ' Write timer/scaler LAM mask-generate
1010 '                    LAM when channel 1 finishes counting
1020 GOSUB 3410
1030 '
1040 ' *** Initialization ***
1050 '
1060 A1%=0:D1%=6540: GOSUB 3870    ' Laser beam shutter should be closed!
1070 PRINT "Laser beam shutter:   CLOSED!"
1080 N%=DAC#:F%=16:A%=2:D%=32700: GOSUB 3300 ' Ion beam shutter should be open!
1090 PRINT "Ion beam shutter :   OPEN!"
1100 '
1110 PRINT "Detector      :  Moving to 0 degree!"
1120 COMMAND$="FSB1 FSC1 MN A1 V.1": GOSUB 2950   ' Moving detector to 0 degree!
1130 POS0=0: POS1=2: GOSUB 3150             ' Make sure not on an endpoint
1140 POS0=0: POS1=-10: GOSUB 3150
1150 IF ENDFLG=0 THEN 1140
1160 POS0=0: POS1=FLGOFF: GOSUB 3150: POS0=0     ' Compensate for offset
1170 '
1180 ' *** Define data file name ***
1190 '
1200 PRINT ""
```

```
1210  BEEP: PRINT "Data file name: ";: INPUT FILNAM$

1220  ON ERROR GOTO 1310

1230  OPEN DATDIR$+FILNAM$ FOR INPUT AS #1  ' Test to see if file already exists.

1240  '                         File exists.  Ask user what to do!!

1250  BEEP: BEEP

1260  PRINT "File exists! Continue [y/n] : ? "

1270  V$=INKEY$: IF V$="" THEN GOTO 1270

1280  IF V$="Y" OR V$="y" THEN GOTO 1320  ' Overwrite existing file.

1290  IF V$="N" OR V$="n" THEN GOTO 1210  ' Get a new name for file.

1300  GOTO 1270

1310 RESUME 1320                     ' File nonexistent.  Proceed.

1320 ON ERROR GOTO 0

1330 CLOSE #1                    ' Close temporary input file.

1340 OPEN DATDIR$+FILNAM$ FOR OUTPUT AS #1  ' Open output file.

1350 '

1360 ' *** Print file header ***

1370 '

1380 PRINT #1,"Data stored by FREQSCAN1.0 on ";DATE$;" at ";TIME$;            ". Format is"

1390 PRINT #1," I Header$ I

1400 PRINT #1," I #Sec/point-#Spec-E(start)-E(width)-#Steps-E/step-DetPos-DarkCnt I "

1410 PRINT #1,"Data -- #Spec * #Steps *  I All on-Laser on-Ion beam on I ": PRINT #1,

1430 PRINT "File header: ";: INPUT HDR$

1440 PRINT #1,HDR$

1450 '

1460 ' *** Enter parameter for measurement ***

1470 '

1480 BEEP: PRINT "": PRINT "Integration time (> 1 sec): ";: INPUT ITIM

1490 IF ITIM<1 THEN ITIM=1

1500 IF ITIM>10000! THEN BEEP: BEEP: PRINT "Out of Range": GOTO 1480: PRINT ""

1510 T=ITIM: GOSUB 3700: TSAVE%=TIMEC%

1520 MSG$=" sec integration.": PRINT #1,STR$(T);

1530 PRINT "Enter # spectra  (1-100)  : ";: INPUT NSPEC%

1540 IF NSPEC%<1 OR NSPEC%>100 THEN 1530

1550 PRINT #1,NSPEC%;

1560 PRINT "Detector position       : ";: INPUT ANG

1570 IF ANG > MAXANG THEN PRINT "Position out of limit!!!": GOTO 1560

1580 PRINT "Starting energy (cm^-1)  : ";: INPUT ESTART$

1590 NRG#=VAL(ESTART$) ' NRG# is the laser energy

1600 MSG$="GOK"+ESTART$+CHR$(13): GOSUB 4080: PRINT""  ' Tell AUTOSCAN to go there

1610 PRINT "Enter # of cm^-1 to scan  : ";: INPUT SCAN

1620 SCAN$=STR$(SCAN)
```

```
1630 PRINT "Enter increments in MHz   : ";: INPUT INC
1640 INC=INC/30000: INC$=STR$(INC)          ' Convert INC into cm^-1
1650 NINC=INT((SCAN/INC)+.5)               ' NINC is total # of incremental steps to scan
1660 PRINT #1,ESTART$;SCAN$;NINC+1;INC,ANG;
1670 PRINT "Moving detector to ";ANG;" degree."
1680 POS0=0: POS1=ANG: GOSUB 3150
1690 IF ENDFLG<>0 THEN BEEP: BEEP: BEEP: PRINT "ERROR!  Limit switch hit!": STOP
1700 '
1710 ' *** Initialize plotting on the screen ***
1720 '
1730 CLS: SCREEN 2: DRAW "BM 0,0 R=XM;D=YM2;L=XM;U=YM2;"
1740 LOCATE 15,1: PRINT ESTART$: LOCATE 16,1: PRINT ANG
1750 LOCATE 2,50: PRINT FILNAM$
1760 DELX=XM/NINC                    'DELX used for plotting
1770 '
1780 ' *** Starting the data collection ***
1790 '
1800 DAC0(1)=0:DAC0(2)=6540:DAC0(3)=0        ' Settings for DAC
1810 DAC2(1)=0:DAC2(2)=32700:DAC2(3)=32700
1820 TDEL(1)=DELAY2#:TDEL(2)=DELAY2#:TDEL(3)=DELAY1#  ' Setting delay after solenoid and ion beam
1830 LABEL$(1)="Laser beam ON":LABEL$(2)="Ion beam ON":LABEL$(3)="All ON"
1840 '
1850 LOCATE 18,30: PRINT "Collecting dark counts for 5 sec!"
1860 N%=DAC#:F%=16:A%=2:D%=0: GOSUB 3300      ' Ion beam shutter CLOSED!
1870 A1%=0:D1%=6540: GOSUB 3870  ' Solenoid shutter CLOSED!
1880 T=DELAY2#: GOSUB 3700: GOSUB 3550        ' Time delay after solenoid operation!
1890 T=5: GOSUB 3700: GOSUB 3460              ' 5 sec dark count collection!
1900 SIGNAL#=D24%(1) AND 32767: IF D24%(1)<0 THEN SIGNAL# = SIGNAL# +32768#
1910 FOR I=1 TO D24%(2): SIGNAL#=SIGNAL# +65536#: NEXT I
1920 DRKCNT=cint(SIGNAL#/5): GOSUB 3630: PRINT #1,DRKCNT
1930 LOCATE 17,1: PRINT DRKCNT
1940 LOCATE 18,30: PRINT "                         "
1950 '
1960 ' *** NSP data collection loop ***
1970 '
1980 NLASINC=0
1990 FOR NSP=1 TO NSPEC%              ' Start scans
2000   CLS: SCREEN 2: DRAW "BM 0,0 R=XM;D=YM2;L=XM;U=YM2;"
2010   LOCATE 15,1: PRINT ESTART$: LOCATE 16,1: PRINT ANG
2020   LOCATE 2,50: PRINT FILNAM$
2030   ' LOCATE 2,3: PRINT "spec#";NSP
```

```
2040   '****
2050   X0%= 0                    ' Initialize x-axis of plot
2060   IF NSP= 1 GOTO 2270          ' If not first scan
2070   MSG$= "DELK-"+SCAN$+CHR$(13): GOSUB 3920  ' then must reset laser,
2080   NRG#= VAL(ESTART$)             ' reset energy variable,
2090   MSG$= "READ"+CHR$(13): GOSUB 3920       ' and verify wavelength.
2100   WVMTR#= VAL(MID$(RMSG$,2,10))
2110   DELNRG= WVMTR#-NRG#
2120   IF ABS(DELNRG)>.006 THEN BEEP: MSG$= "GOK"+STR$(NRG#)+CHR$(13): GOSUB 3920
2130   AVDEL= 0
2140   FOR II=1 TO 3
2150   MSG$= "READ"+CHR$(13): GOSUB 3920
2160   WVMTR#= VAL(MID$(RMSG$,2,10)): PRINT WVMTR#
2170   DELNRG= WVMTR#-NRG#: PRINT DELNRG
2180   IF ABS(DELNRG)<=.001 THEN GOTO 2260
2190   IF ABS(DELNRG)>.001 THEN AVDEL= AVDEL+DELNRG
2200   NEXT II
2210   AVDEL= AVDEL/3!: PRINT AVDEL
2220   MSG$="DELK"+STR$(-AVDEL)+CHR$(13): GOSUB 3920
2230   GOTO 2140
2240   '****
2250   '
2260   SUM=0: CNT=0
2270   FOR INCCNT=0 TO NINC            ' Begin loop incrementing angle.
2280     LOCATE 2,3: PRINT "spec#";NSP
2290     IF INCCNT=0 THEN GOTO 2490        ' Don't need increment 1st time.
2300     MSG$="DELK"+INC$+CHR$(13): GOSUB 3920: NRG#=NRG#+INC  ' Tell AUTOSCAN
2310     '                     to increment frequency, update NRG
2320     NLASINC=NLASINC+1: IF NLASINC<10 THEN GOTO 2540  ' After 10 increments
2330     '****
2340     NLASINC=0: MSG$="READ"+CHR$(13): GOSUB 3920    ' check wavemeter to
2350     WVMTR#=VAL(MID$(RMSG$,2,10))
2360     DELNRG=WVMTR#-NRG#
2370     IF ABS(DELNRG)>.006 THEN BEEP: MSG$="GOK"+STR$(NRG#)+CHR$(13): GOSUB 3920
2380     AVDEL= 0
2390     FOR II=1 TO 3
2400     MSG$= "READ"+CHR$(13): GOSUB 3920
2410     WVMTR#=VAL(MID$(RMSG$,2,10))
2420     DELNRG= WVMTR#-NRG#
2430     IF ABS(DELNRG)<=.001 THEN GOTO 2500
2440     IF ABS(DELNRG)>.001 THEN AVDEL=AVDEL+DELNRG
```

```
2450    NEXT II

2460    AVDEL=AVDEL/3!: PRINT AVDEL

2470    MSG$="DELK"+STR$(-AVDEL)+CHR$(13): GOSUB 3920

2480    GOTO 2380

2490    '****

2500    LOCATE 18,1: PRINT "              "

2510    LOCATE 18,1: PRINT USING "Energy = #######.###";NRG#

2520    '

2530    CNT=0: SUM=0

2540    FOR DACSW=1 TO 3

2550      LOCATE 18,30: PRINT "                    "

2560      LOCATE 18,30: PRINT LABEL$(DACSW)

2570      N%=DAC#:F%=16:A%=2:D%=DAC2(DACSW): GOSUB 3300        ' Set DAC on/off depending on

2580      A1%=0:D1%=DAC0(DACSW): GOSUB 3870 ' Dacsw

2590      LOCATE 18,1: PRINT "              "

2600      LOCATE 18,1: PRINT USING "Energy = #######.###";NRG#

2610      T=TDEL(DACSW): GOSUB 3700: GOSUB 3550       ' Wait for time delay!

2620      TIMEC%=TSAVE%: GOSUB 3460                ' now Count for TSAVE%

2630      SIGNAL#=D24%(1) AND 32767: IF D24%(1)<0 THEN SIGNAL#=SIGNAL#+32768#

2640      FOR I=1 TO D24%(2): SIGNAL#=SIGNAL#+65536#: NEXT I

2650      SIG(DACSW)=SIGNAL#/ITIM  : GOSUB 3630

2660    NEXT DACSW

2670    '

2680    '***LASERON=LASERON+SIG(1): IONON=IONON+SIG(2): ALLON=ALLON+SIG(3)

2690    ' NEXT NC  ' Save signal

2700    '***DELSIG=ALLON-LASERON-IONON+DRKCNT: PRINT #1,ALLON,LASERON,IONON

2710    DELSIG=SIG(3)-SIG(1)-SIG(2)+DRKCNT: PRINT #1, SIG(3),SIG(1),SIG(2),DELSIG

2720    LOCATE 19,1: PRINT "                        "

2730    '***LOCATE 19,1: PRINT ALLON;"-";LASERON;"-";IONON;"+";DRKCNT;"=";DELSIG;':LPRINT
POS0,DELSIG

2740    LOCATE 19,1: PRINT INCCNT;":  ";SIG(3);"-";SIG(1);"-";SIG(2);"+";DRKCNT;"=";DELSIG;':LPRINT
POS0,DELSIG

2750    YP%=(YM/PFAC)*(PMAX-(DELSIG)/ITIM): XP%=INCCNT*DELX: IF YP%>YM2 THEN YP%=YM2

2760    DRAW "BM=X0%;,=Y0%;": DRAW "M=XP%;,=YP%;": Y0%=YP%: X0%=XP%

2770    SIG(1)=0: SIG(2)=0: SIG(3)=0  'Reset accumulated signals for next frequency

2780    NEXT INCCNT

2790    '

2800    BEEP

2810    PRINT #1,"xxxx"

2820 NEXT NSP

2830 '
```

```
2840 CLOSE

2850 LOCATE 23,1: BEEP: BEEP: PRINT ">>>(EXIT Freqscan1.0 with 'E')"

2860 A$=INKEY$: IF (A$="E") OR (A$="e") THEN CLS ELSE 2860

2870 END

2880 '

2890 '

2900 ' *** SUBROUTINES ARE LOCATED HERE !!! ***

2910 '

2920 ' *** PC21 WRITE ***

2930 '

2940 BFLAG%=0

2950 IF INSTR(COMMAND$,"W1") OR INSTR(COMMAND$,"w1") THEN BFLAG%=1

2960 IF INSTR(COMMAND$,"PB") OR INSTR(COMMAND$,"pb") THEN BFLAG%=1

2970 IF INSTR(COMMAND$,"X1B") OR INSTR(COMMAND$,"x1b") THEN BFLAG%=1

2980 COMMAND$ = COMMAND$ + CHR$(13)     ' Add carriage return to command

2990 CALL PC21WRITE(COMMAND$, ADDRESS%)  ' Execute machine language write

3000 RETURN

3010 '

3020 ' *** PC21 READ ***

3030 '

3040 ANSWER$="            "+""   ' Reserve string space for response

3050 CALL PC21READ( ANSWER$, ADDRESS%, BFLAG% )  ' Execute read

3060 IF BFLAG%=0 THEN RETURN

3070 NUM#=0: FOR X=1 TO 4

3080 DIGIT%=ASC(MID$(ANSWER$,X,1))

3090 NUM#=NUM#+DIGIT%*256^(4-X): NEXT

3100 ANSWER$=STR$(NUM#)

3110 RETURN                          ' BFLAG% identifies binary report commands

3120 '

3130 ' *** Move Detector POS1-POS0 degrees (check for end switch) ***

3140 '

3150 DEG=ABS(POS1-POS0): DIR$="+": IF POS1-POS0<0 THEN DIR$="-"

3160 AVTIM=DEG*.15: ENDFLG=0: IF AVTIM<.5 THEN AVTIM=.5

3170 STEPS=INT(DEG*4000/180+.5)      ' Convert degrees to motor steps

3180 STEPS$=MID$(STR$(STEPS),2)

3190 DEGAC=STEPS*180/4000: IF DIR$="-" THEN DEGAC=-DEGAC

3200 COMMAND$="D"+DIR$+STEPS$+" G CR P": GOSUB 2950 ' Move & signal when done

3210 T0=TIMER

3220 GOSUB 3040: IF LEFT$(ANSWER$,1)=CHR$(13) THEN 3250  ' Keep going till end

3230 T1=TIMER: IF T1-T0<AVTIM THEN 3220  ' If not done after AVTIM,

3240 ENDFLG=1: GOTO 3260              ' End switch must have been hit.
```

```
3250 POS0=POS0+DEGAC
3260 RETURN
3270 '
3280 ' *** Main WRITE/READ Subroutines to CAMAC ***
3290 '
3300 CALL CAMO(N%,F%,A%,D%,Q%,X%)
3310 RETURN
3320 '
3330 CALL CAMI24(N%,F%,A%,D24%(1),Q%,X%)
3340 RETURN
3350 '
3360 CALL CAMI(N%,F%,A%,D%,Q%,X%)
3370 RETURN
3380 '
3390 ' *** initialize DAC, put all channels (0...3) to OV! ***
3400 '
3410 FOR I=0 TO 3: N%=DAC#:F%=16:A%=I:D%=0: GOSUB 3300: NEXT
3420 RETURN
3430 '
3440 ' *** Set timer/scaler ***
3450 '
3460 N%=TIM#:F%=17:A%=0:D%=TIMEC%: GOSUB 3300    ' D% must contain time.
3470 N%=TIM#:F%=17:A%=4:D%=1: GOSUB 3300         ' Start counting
3480 CALL CAML(L%):IF L%=0 THEN 3480             ' Wait for count to end
3490 N%=TIM#:F%=23:A%=12:D%=1: GOSUB 3300        ' Clear timer LAM
3500 N%=TIM#:F%=0:A%=1: GOSUB 3330: GOTO 3630    ' Read scaler
3510 RETURN
3520 '
3530 ' *** Wait delay ***
3540 '
3550 N%=TIM#:F%=17:A%=0:D%=TIMEC%: GOSUB 3300    ' D% must contain time!
3560 N%=TIM#:F%=17:A%=4:D%=1: GOSUB 3300         ' Start wait delay
3570 CALL CAML(L%): IF L%=0 THEN 3570            ' Wait for time to end
3580 N%=TIM#:F%=23:A%=12:D%=1: GOSUB 3300: GOTO 3630 ' Clear timer LAM
3590 RETURN
3600 '
3610 ' *** Clear timer/scaler ***
3620 '
3630 N%=TIM#:F%=9:A%=0:D%=0: GOSUB 3300
3640 N%=TIM#:F%=9:A%=1:D%=0: GOSUB 3300
3650 RETURN
```

```
3660 '

3670 ' *** Subroutine for converting time (T) to a special

3680 '    formatted integer (TIMEC%) used by CAMAC timer. ***

3690 '

3700 TS=T: TB%=INT((LOG(T/1000))/2.30258): ERFLG%=0

3710 TBAS=10^TB%: T=T/TBAS: N=0

3720 IF T<2 GOTO 3740

3730 N=N+1: T=T/2: GOTO 3720

3740 J=32*(T-1)+1

3750 IOVF%=N-OVFTBL%(J)

3760 IF IOVF%<0 THEN TB%=TB%-1: GOTO 3710

3770 IF IOVF%>15 THEN TB%=TB%+1: GOTO 3710

3780 TB%=TB%+6: IF TB%<0 OR TB%>7 THEN PRINT "Time requested outside range": ERFLG%=1: GOTO 3820

3790 MULT%=MULTBL%(J)

3800 TA=TBAS*(2*MULT%+1)*2^(IOVF%+4)

3810 TIMEC%=(128!*MULT%)+(8!*IOVF%)+TB%

3820 T=TS

3830 RETURN

3840 '

3850 ' *** Open/close solenoid shutter ***

3860 '

3870 N%=DAC#:F%=16:A%=A1%:D%=D1%: GOSUB 3300

3880 RETURN

3890 '

3900 ' *** Send Message to Apple via RS232  ***

3910 '

3920 LOCATE 20,1: PRINT ">>> Remote Apple control: "

3930 LOCATE 21,1: PRINT CMSG$: LOCATE 21,1

3940 FOR IMSG=1 TO LEN(MSG$): V$=MID$(MSG$,IMSG,1): PRINT #2,V$;: NEXT

3950 PRINT MSG$: RMSG$=""

3960 IF EOF(2) THEN 3960

3970 WHILE NOT EOF(2)

3980 ASMSG$=INPUT$(1,#2): ASMSG$=CHR$(ASC(ASMSG$)-128)

3990 IF ASMSG$=CHR$(13) GOTO 4040

4000 RMSG$=RMSG$+ASMSG$: WEND

4010 IF ASC(ASMSG$)<>41 THEN GOTO 3960

4020 RETURN

4030 '

4040 LOCATE 22,1: PRINT CMSG$;

4050 PRMSG$=RMSG$: RMSG$="": LOCATE 22,1: PRINT PRMSG$;

4060 GOTO 4000
```

```
4070 '
4080 FOR IMSG=1 TO LEN(MSG$): V$=MID$(MSG$,IMSG,1): PRINT #2,V$;: NEXT
4090 PRINT RMSG$=""
4100 IF EOF(2) THEN 4100
4110 WHILE NOT EOF(2)
4120 ASMSG$=INPUT$(1,#2): ASMSG$=CHR$(ASC(ASMSG$)-128)
4130 IF ASMSG$=CHR$(13) THEN RMSG$=""
4140 PRINT ASMSG$;: RMSG$=RMSG$+ASMSG$: WEND
4150 IF ASC(ASMSG$)<>41 THEN GOTO 4100
4160 RETURN
```

MASPLOT.BAS

```
10   REM   MASPLOT: plotting of mass spectra
11   REM   V1.0: 02/04/93
13   REM   files needed    : hbasic.exe, basica.com, code.bas, camio.bas
14   '
15   REM   Program structure from MASSPEC.BAS !!!
16   '
100  '  ****** Find data segment above BASICA ******
101  '
110  CLEAR: CLS                  ' Set all variables to "0"
120  DEF SEG=0                   ' Go to low memory to find BASICA loc.
130  BDATSEG=PEEK(&H510)+256*PEEK(&H511) ' BASICA data segment is in &H510-511
140  CAMSEG=BDATSEG+&H2000          ' Find next data segment after BASICA
150  DEF SEG=CAMSEG
151  '
200  ' *** Dimension of variables ***
201  '
210  DIM XVal(1000),YVal(1000)
211  '
280  ' *** Setting parameters ***
281  '
290  XPix=600: YPix=250: YNeg=-0.1: YPos=3.5: YScale=1.0
300  YFac=YPix/(1.0*(-YNeg+YPos))           ' YFac for y-axis
301  '
320  ' *** Program header ***
321  '
340  PRINT "M A S P L O T  (1.0)": PRINT ""
350  PRINT "This program is plotting mass spec"
351  '
400  ' *** Poke in machine language routinge ***
401  '
410  OPEN "C:\BASICDIR\NEW\CODE.BAS" FOR INPUT AS #1     ' Access machine code data file
420  FOR X = 0! TO 127!
430    INPUT #1, J                     ' Install machine code
440    POKE X,J
450  NEXT
460  CLOSE #1
461  '
500  ' *** Set all program variables ***
```

```
501 '
510 ADDRESS% = 768          ' PC21 base address
520 CONTROL = 96            ' Normal state of PC21 Control Byte
530 CRASH = 4               ' Mask for Control Bit 2 (BMA time-out)
540 FAULT = 32              ' Mask for C.B. 5 (restart BMA)
550 PC21WRITE = 0!          ' Address of PC21WRITE subroutine
560 PC21READ = 49!          ' Address of PC21READ subroutine
561 '
570 ADC#= 9                 ' ADC module in CAMAC slot #9
580 DAC# = 12               ' DAC module in CAMAC slot #12
581 '                  ch#0,1,2: +/- 5V
582 '                       3: + 10V (scan mass spec)
590 TIM# = 10               ' TIMER/SCALER module in CAMAC slot #10
600 DELAY# = 0.025          ' Delaytime for DAC respnse in sec.
601 '
610 DATDIR$ = "C:\MASSDAT\"   ' Directory for data files
615 '
620 UMassFac = 0.0333       ' Conversion factor mass --> DAC voltage
630 NADC = 2                ' # of measurements done by ADC
631 '
1000 ' *** Open data file ***
1001 '
1020 PRINT ""
1030 BEEP: PRINT "Data file name: ";: INPUT FILNAM$
1040 ON ERROR GOTO 1070
1050 OPEN DATDIR$+FILNAM$ FOR INPUT AS #1  ' Test to see if file already exists.
1051 '                    File exists.  Ask user what to do!!
1055 GOTO 1140
1060 BEEP: BEEP
1070 PRINT "File does'nt exists!": GOTO 1030
1080 '***V$=INKEY$: IF V$="" THEN GOTO 1080
1090 '***IF V$="Y" OR V$="y" THEN GOTO 1130     ' Overwrite existing file.
1100 '***IF V$="N" OR V$="n" THEN GOTO 1030     ' Get a new name for file.
1110 '***GOTO 1080
1120 RESUME 1130                ' File nonexistent.  Proceed.
1130 ON ERROR GOTO 0
1140 '***CLOSE #1               ' Close temporary input file.
1150 '***OPEN DATDIR$+FILNAM$ FOR INPUT AS #1  ' Open output file.
1151 '
1200 ' *** Enter parameter for measurement ***
1201 '
```

```
1210 BEEP: PRINT "": PRINT "YScale: ";: INPUT YScale
1271 '
1300 ' *** Read data ***
1301 '
1310 FOR i=1 TO 3 : LINE INPUT #1, A$: NEXT i        ' Skip first 3 lines!
1320 INPUT #1, IntNum
1330 FOR i=1 TO IntNum:
1340   INPUT #1, XVal(i),YVal(i)
1345 '***  PRINT XVal(i); YVal(i)
1350 NEXT i
1351 '
1400 ' *** Initialize plotting ***
1401 '
1410 CLS: SCREEN 2: DRAW "BM0,0 R=XPix;D=YPix;L=XPix;U=YPix;"
1420 '*** LOCATE 20,1: PRINT LowMass: LOCATE 21,1: PRINT HighMass
1430 LOCATE 2,50: PRINT FILNAM$
1434 DifMass=XVal(IntNum)-XVal(1)
1435 XFac=XPix/(1.0*DifMass)              ' XFac for x-axis
1440 X0%=0.0
1445 '
1700 ' *** Plotting data ***
1701 '
1705 X0%=0.0
1706 '
1710 FOR ii=1 TO IntNum
1711 '
1720   DRAW "BM=X0%;,=Y0%;"
1730   YP%=YPix-(YVal(ii)-YNeg)*YFac*YScale: XP%=(XVal(ii)-XVal(1))*XFac
1740   IF YVal(ii)>YPos THEN YP%=0
1750   IF YVal(ii)<YNeg THEN YP%=YPix
1760   Y0%=YP%
1770   DRAW "M=X0%;,=Y0%;": DRAW "M=X0%;,=YP%;": DRAW "M=XP%;,=YP%;": X0%=XP%
1771 '
1780 NEXT ii
1781 '
1790 LOCATE 22,1: PRINT "                                    "
1800 PRINT "Rescale Plot: ";: INPUT YScale
1810 IF YScale > 1.0 THEN GOTO 1705
1820 LOCATE 23,1: BEEP: BEEP: PRINT ">>>(EXIT MasPlot1.0 with 'E')"
1830 A$=INKEY$: IF (A$="E") OR (A$="e") THEN CLS ELSE 1830
1831 '
```

```
1900 CLOSE          ' Close OutputFile
1901 '
1910 END
1911 '
1912 '
3000 ' *** SUBROUTINES ARE LOCATED HERE !!! ***
3001 '
3010 ' *** PC21 WRITE ***
3011 '
3020 BFLAG%=0
3030 IF INSTR(COMMAND$,"W1") OR INSTR(COMMAND$,"w1") THEN BFLAG%=1
3040 IF INSTR(COMMAND$,"PB") OR INSTR(COMMAND$,"pb") THEN BFLAG%=1
3050 IF INSTR(COMMAND$,"X1B") OR INSTR(COMMAND$,"x1b") THEN BFLAG%=1
3060 COMMAND$ = COMMAND$ + CHR$(13)      ' Add carriage return to command
3070 CALL PC21WRITE(COMMAND$, ADDRESS%)  ' Execute machine language write
3080 RETURN
3081 '
3100 ' *** PC21 READ ***
3101 '
3110 ANSWER$="              "+""    ' Reserve string space for response
3120 CALL PC21READ( ANSWER$, ADDRESS%, BFLAG% )  ' Execute read
3130 IF BFLAG%=0 THEN RETURN
3140 NUM#=0: FOR X=1 TO 4
3150 DIGIT%=ASC(MID$(ANSWER$,X,1))
3160 NUM#=NUM#+DIGIT%*256^(4-X): NEXT
3170 ANSWER$=STR$(NUM#)
3180 RETURN                            ' BFLAG% identifies binary report commands
3181 '
```

MASSPEC.BAS

```basic
10  REM  MASSPEC: recording of mass spectra
11  REM  V1.0: 02/04/93
13  REM  files needed    : hbasic.exe, basica.com, code.bas, camio.bas
14  '
15  REM  Program structure from ANGSCAN.BAS !!!
16  '
100 '  ****** Find data segment above BASICA ******
101 '
110 CLEAR: CLS                 ' Set all variables to "0"
120 DEF SEG=0                  ' Go to low memory to find BASICA loc.
130 BDATSEG=PEEK(&H510)+256*PEEK(&H511) ' BASICA data segment is in &H510-511
140 CAMSEG=BDATSEG+&H2000          ' Find next data segment after BASICA
150 DEF SEG=CAMSEG
151 '
200 ' *** Dimension of variables ***
201 '
210 DIM YVal(1000)
211 '
280 ' *** Setting parameters ***
281 '
290 XPix=600: YPix=250: YNeg=-0.1: YPos=3.5: YScale=1.0
300 YFac=YPix/(1.0*(-YNeg+YPos))          ' YFac for y-axis
301 '
320 ' *** Program header ***
321 '
340 PRINT "M A S S P E C  (1.0)": PRINT ""
350 PRINT "This program is taking mass spec"
351 '
400 ' *** Poke in machine language routinge ***
401 '
410 OPEN "C:\BASICDIR\NEW\CODE.BAS" FOR INPUT AS #1    ' Access machine code data file
420 FOR X = 0! TO 127!
430   INPUT #1, J                       ' Install machine code
440   POKE X,J
450 NEXT
460 CLOSE #1
461 '
500 ' *** Set all program variables ***
```

```
501 '
510 ADDRESS% = 768          ' PC21 base address
520 CONTROL = 96            ' Normal state of PC21 Control Byte
530 CRASH = 4               ' Mask for Control Bit 2 (BMA time-out)
540 FAULT = 32              ' Mask for C.B. 5 (restart BMA)
550 PC21WRITE = 0!          ' Address of PC21WRITE subroutine
560 PC21READ = 49!          ' Address of PC21READ subroutine
561 '
570 ADC#= 9                 ' ADC module in CAMAC slot #9
580 DAC# = 12               ' DAC module in CAMAC slot #12
581 '                ch#0,1,2: +/- 5V
582 '                     3: + 10V (scan mass spec)
590 TIM# = 10               ' TIMER/SCALER module in CAMAC slot #10
600 DELAY# = 0.025          ' Delaytime for DAC respnse in sec.
601 '
610 DATDIR$ = "C:\MASSDAT\"   ' Directory for data files
615 '
620 UMassFac = 0.0333       ' Conversion factor mass --> DAC voltage
630 NADC = 2                ' # of measurements done by ADC
631 '
700 ' *** PC21 RESET ***
701 '
710 OUT ADDRESS%+1, ( CONTROL OR CRASH )          ' Control Bit 2 high
720 OUT ADDRESS%+1, ( CONTROL AND NOT CRASH )  ' Control Bit 2 low
730 FOR Y=1 TO 500: NEXT              ' Wait for BMA
740 OUT ADDRESS%+1, ( CONTROL AND NOT FAULT ) ' Control Bit 5 low
750 OUT ADDRESS%+1, ( CONTROL OR FAULT )         ' Control Bit 5 high
751 '
800 ' *** Load CAMAC drivers and initialize crate ***
801 '
810 BLOAD "C:\BASICDIR\NEW\CAMIO",128       ' Load drivers into data segment
820 CAMO=&H80:CAMI=&H86:CAML=&H8C:CAMCL=&H92 ' Driver entry point addresses
830 CAMO24=&HB0:CAMI24=&HB6:CRATE=&HAA       ' Driver entry point addresses
840 CC%=1: CALL CRATE(CC%)            ' Activate controller in J1 slot
850 OUT &H240,0                  ' Clear high write-only data register
860 I%=64: CALL CAMCL(I%)           ' Reset crate
870 I%=1: CALL CAMCL(I%)            ' Initialize crate
871 '
1000 ' *** Initialization ***
1001 '
1010 N%=DAC#:F%=16:A%=3:D%=-32767: GOSUB 3410 ' Mass spec set to 0 amu
```

```
1011 '
1100 ' *** Define data file name ***
1101 '
1020 PRINT ""
1030 BEEP: PRINT "Data file name: ";: INPUT FILNAM$
1040 ON ERROR GOTO 1120
1050 OPEN DATDIR$+FILNAM$ FOR INPUT AS #1   ' Test to see if file already exists.
1051 '                        File exists.  Ask user what to do!!
1060 BEEP: BEEP
1070 PRINT "File exists! Continue [y/n] : ? "
1080 V$=INKEY$: IF V$="" THEN GOTO 1080
1090 IF V$="Y" OR V$="y" THEN GOTO 1130     ' Overwrite existing file.
1100  IF V$="N" OR V$="n" THEN GOTO 1030    ' Get a new name for file.
1110 GOTO 1080
1120 RESUME 1130                ' File nonexistent.  Proceed.
1130 ON ERROR GOTO 0
1140 CLOSE #1                   ' Close temporary input file.
1150 OPEN DATDIR$+FILNAM$ FOR OUTPUT AS #1  ' Open output file.
1151 '
1200 ' *** Enter parameter for measurement ***
1201 '
1210 BEEP: PRINT "": PRINT "Low Mass: ";: INPUT LowMass
1220 PRINT "High Mass: ";: INPUT HighMass
1230 DifMass=HighMass-LowMass
1240 PRINT "# of Intervals: ";: INPUT IntNum
1250 DelMass=DifMass/(1.0*IntNum)
1260 PRINT "# of Scans per Mass: ";: INPUT NumScan
1270 PRINT "Comment: ";: INPUT HDR$
1271 '
1300 ' *** Print file header ***
1301 '
1310 PRINT #1, "Data stored by MasSpec1.0 on ";DATE$;" at ";TIME$
1320 PRINT #1, HDR$
1330 PRINT #1, " I Mass Signal I "
1335 PRINT #1, NumInt
1331 '
1400 ' *** Initialize plotting/DAC ***
1401 '
1410 CLS: SCREEN 2: DRAW "BM0,0 R=XPix;D=YPix;L=XPix;U=YPix;"
1420 LOCATE 20,1: PRINT LowMass: LOCATE 21,1: PRINT HighMass
1430 LOCATE 2,50: PRINT FILNAM$
```

```
1435 XFac=XPix/(1.0*DifMass)            ' XFac for x-axis
1440 X0%=0.0
1445 '
1550 ' *** Data collection loop ***
1551 '
1570 FOR NInt=0 TO IntNum              ' Start scans
1571 '
1580   Mass = LowMass + NInt*DelMass
1590   URamp = Mass * UMassFac: GOSUB 3540   ' GoSub SetDAC
1610   TAverage= 0.0
1611 '
1620   FOR NScan=1 TO NumScan              ' ADC loop
1621 '
1630     GOSUB 4110: Sig= ADC           ' GoSub ADCloop: get Signal from ADC
1640     TAverage = TAverage + Sig
1650     Signal = TAverage/(1.0*NumScan)
1660     YVal(NInt) = Signal
1661 '
1670   NEXT NScan                   ' FOR NScan ...
1671 '
1680 '***  PRINT #1, USING "##.###"; Mass; Signal
1681 '
1690 NEXT NInt                   ' FOR NInt ...
1691 '
1695 FOR ii=0 TO IntNum: PRINT #1, USING "##.###"; Mass; Signal: NEXT ii
1696 '
1700 ' *** Plotting data ***
1701 '
1705 X0%=0.0
1706 '
1710 FOR ii=1 TO IntNum
1711 '
1720   DRAW "BM=X0%;,=Y0%;"
1730   YP%=YPix-(YVal(ii)-YNeg)*YFac*YScale: XP%=ii*DelMass*XFac
1740   IF YVal(ii)>YPos THEN YP%=0
1750   IF YVal(ii)<YNeg THEN YP%=YPix
1760   Y0%=YP%
1770   DRAW "M=X0%;,=Y0%;": DRAW "M=X0%;,=YP%;": DRAW "M=XP%;,=YP%;": X0%=XP%
1771 '
1780 NEXT ii
1781 '
```

```
1790 LOCATE 22,1: PRINT "                          "
1800 PRINT "Rescale Plot: ";: INPUT YScale
1810 IF YScale > 1.0 THEN GOTO 1705
1820 LOCATE 23,1: BEEP: BEEP: PRINT ">>>(EXIT MasSpec1.0 with 'E')"
1830 A$=INKEY$: IF (A$="E") OR (A$="e") THEN CLS ELSE 1830
1831 '
1900 CLOSE          ' Close OutputFile
1901 '
1910 END
1911 '
1912 '
3000 ' *** SUBROUTINES ARE LOCATED HERE !!! ***
3001 '
3010 ' *** PC21 WRITE ***
3011 '
3020 BFLAG%=0
3030 IF INSTR(COMMAND$,"W1") OR INSTR(COMMAND$,"w1") THEN BFLAG%=1
3040 IF INSTR(COMMAND$,"PB") OR INSTR(COMMAND$,"pb") THEN BFLAG%=1
3050 IF INSTR(COMMAND$,"X1B") OR INSTR(COMMAND$,"x1b") THEN BFLAG%=1
3060 COMMAND$ = COMMAND$ + CHR$(13)     ' Add carriage return to command
3070 CALL PC21WRITE(COMMAND$, ADDRESS%)  ' Execute machine language write
3080 RETURN
3081 '
3100 ' *** PC21 READ ***
3101 '
3110 ANSWER$="              "+""    ' Reserve string space for response
3120 CALL PC21READ( ANSWER$, ADDRESS%, BFLAG% )  ' Execute read
3130 IF BFLAG%=0 THEN RETURN
3140 NUM#=0: FOR X=1 TO 4
3150 DIGIT%=ASC(MID$(ANSWER$,X,1))
3160 NUM#=NUM#+DIGIT%*256^(4-X): NEXT
3170 ANSWER$=STR$(NUM#)
3180 RETURN                        ' BFLAG% identifies binary report commands
3181 '
3400 ' *** Main WRITE/READ Subroutines to CAMAC ***
3401 '
3410 CALL CAMO(N%,F%,A%,D%,Q%,X%)
3420 RETURN
3421 '
3430 CALL CAMI24(N%,F%,A%,D24%(1),Q%,X%)
3450 RETURN
```

```
3451 '
3460 CALL CAMI(N%,F%,A%,D%,Q%,X%)
3470 RETURN
3471 '
3530 ' *** Set DAC ***
3531 '
3540 URamp#=INT(URamp*32767/5)-32767
3545 N%=DAC#:F%=16:A%=3:D%=URamp#: GOSUB 3410
3550 RETURN
3551 '
4100 '   ****** Get ADC data
4101 '
4110 FOR I=1 TO NADC: N%=ADC#:F%=2:A%=0: GOSUB 3460: NEXT I  'GoSub CAMO
4120 ADC=D%: ADC=-1*ADC*10/4096
4130 RETURN
```

PASCAL Programs

IO_CHECK.PAS

```
{
   file      :  IO_CHECK.PAS
   function  :  Check routines for I/O operations
   author    :  W.Maring
   changes   :  12.02.93
            14.05.93  (for GRAFICS mode)
}


UNIT IO_Check;
{$IFDEF CPU87}
{$N+}
{$ELSE}
{$N-}
{$ENDIF}


INTERFACE

USES
  Crt,
  Dos,
  Graph;


TYPE
{$IFDEF CPU87}
  Real      = Single;
{$ENDIF}
  TStr1     = String[1];
  TStr2     = String[2];
  TStr4     = String[4];
  TStr40    = String[40];
  TStr80    = String[80];


CONST
  Esc       = Char(27);



PROCEDURE GetDateTime(VAR DatTimStr: TStr40);
PROCEDURE PrintFileHeader(VAR FileFil: Text; FileName: TStr40);
```

```
PROCEDURE PClearMenuXY(MenuX,MenuY,Color,BkColor: Integer; MenuStr: TStr80);
PROCEDURE PMenuXY(MenuX,MenuY,Color,BkColor: Integer; MenuStr: TStr80);
PROCEDURE InpChar(VAR InputString: TStr80);
PROCEDURE PInpChar(MenuX,MenuY,Color,BkColor: Integer; VAR MenuStr: TStr80;
          VAR InputString: TStr80);
PROCEDURE InputFile(FilePrompt: TStr40; VAR FileFil: Text; FilePath: TStr40;
          VAR FileName: TStr40);
PROCEDURE PInputFile(FilePrompt: TStr80; VAR FileFil: Text; FilePath: TStr40;
          VAR FileName: TStr40; MenuX,MenuY,Color,BkColor: Integer;
          VAR MenuStr: TStr80);
PROCEDURE OutputFile(FilePrompt: TStr40; VAR FileFil: Text; FilePath: TStr40;
          VAR FileName: TStr40);
PROCEDURE POutputFile(FilePrompt: TStr80; VAR FileFil: Text; FilePath: TStr40;
          VAR FileName: TStr40; MenuX,MenuY,Color,BkColor: Integer;
          VAR MenuStr: TStr80);
PROCEDURE ReadReal(InpString: TStr40; DoLimit,UpLimit: Real;
          VAR RealInp: Real);
PROCEDURE PReadReal(InpString: TStr80; DoLimit,UpLimit: Real;
          VAR RealInp: Real; MenuX,MenuY,Color,BkColor: Integer;
          VAR MenuStr: TStr80);
PROCEDURE ReadInt(InpString: TStr40; DoLimit,UpLimit: Integer;
          VAR IntInp: Integer);
PROCEDURE PReadInt(InpString: TStr80; DoLimit,UpLimit: Real;
          VAR IntInp: Integer; MenuX,MenuY,Color,BkColor: Integer;
          VAR MenuStr: TStr80);
PROCEDURE ReadStr(InpString: TStr40; VAR StrInp: TStr80);
PROCEDURE PReadStr(InpString: TStr80; DoLimit,UpLimit: Real;
          VAR StrInp: TStr80; MenuX,MenuY,Color,BkColor: Integer;
          VAR MenuStr: TStr80);
PROCEDURE ReadChar(InpString: TStr40; VAR CharInp: TStr1);
PROCEDURE PReadChar(InpString: TStr80; DoLimit,UpLimit: Real;
          VAR CharInp: TStr1; MenuX,MenuY,Color,BkColor: Integer;
          VAR MenuStr: TStr80);
```

IMPLEMENTATION

```
{----------------------------------------------------------------------
PROCEDURE GetDateTime:  String containing date/time
```

```
-----------------------------------------------------------------}
  PROCEDURE GetDateTime(VAR DatTimStr: TStr40);
  VAR
   YearStr        : TStr4;
   MonthStr,
   DayStr,
   HourStr,
   MinStr,
   SecStr        : TStr2;
   ActYear,
   ActMonth,
   ActDay,
   ActDayOfWeek,
   ActHour,
   ActMin,
   ActSec,
   ActSec100     : Word;
  BEGIN
   GetDate(ActYear,ActMonth,ActDay,ActDayOfWeek);
   GetTime(ActHour,ActMin,ActSec,ActSec100);

   Str(ActYear:4,YearStr);
   Str(ActMonth:2,MonthStr);
   Str(ActDay:2,DayStr);
   Str(ActHour:2,HourStr);
   Str(ActMin:2,MinStr);
   Str(ActSec:2,SecStr);
   DatTimStr:= MonthStr+'/'+DayStr+'/'+YearStr
              +' '+HourStr+':'+MinStr+':'+SecStr;
  END;


{-------------------------------------------------------------------

  PROCEDURE PrintFileHeader:   prints file header


-----------------------------------------------------------------}
  PROCEDURE PrintFileHeader(VAR FileFil: Text; FileName: TStr40);
  VAR
   ActYear,
   ActMonth,
   ActDay,
   ActDayOfWeek,
```

```
      ActHour,
      ActMin,
      ActSec,
      ActSec100          : Word;
    BEGIN
     GetDate(ActYear,ActMonth,ActDay,ActDayOfWeek);
     GetTime(ActHour,ActMin,ActSec,ActSec100);

     Writeln(FileFil,FileName,'   ',ActMonth:1,'/',ActDay:1,'/',ActYear
                                ,'   ',ActHour:1,':',ActMin:1,':',ActSec:1);
    END;


{------------------------------------------------------------------
PROCEDURE PClearMenuXY:   clears the MenuStr on the screen
        (needs to be in GRAFICS mode !!!)
    ------------------------------------------------------------------------}
PROCEDURE PClearMenuXY(MenuX,MenuY,Color,BkColor: Integer; MenuStr: TStr80);
BEGIN
 SetColor(BkColor);
 OutTextXY(MenuX,MenuY,MenuStr);
 SetColor(Color);
END;


{------------------------------------------------------------------
PROCEDURE PMenuStr:   plots the MenuStr on the screen
        (needs to be in GRAFICS mode !!!)
    --------------------------------------------------------------------}
PROCEDURE PMenuXY(MenuX,MenuY,Color,BkColor: Integer; MenuStr: TStr80);
BEGIN
 OutTextXY(MenuX,MenuY,MenuStr);
END;

{------------------------------------------------------------------
 PROCEDURE InpChar:  uses ReadKey command to read string var


    --------------------------------------------------------------------}
PROCEDURE InpChar(VAR InputString: TStr80);
VAR
 InpChar    : Char;
 InpStr     : String;
 XCPos,
```

```
                XCPosSav,
                YCPos,
                YCPosSav   : Integer;
                InpOkay    : Boolean;
        BEGIN
          InpStr:= '';
          XCPos:= WhereX;
          XCPosSav:= XCPos;
          YCPos:= WhereY;
          YCPosSav:= YCPos;
          InpOkay:= False;
        REPEAT
          InpChar:= ReadKey;
          IF InpChar=Char(27) THEN
          BEGIN
            InputString:= Char(27);
            Exit;
          END
          ELSE
          BEGIN
            IF InpChar=Char(8) THEN
            BEGIN
                    IF XCPos>=XCPosSav+1 THEN
                    BEGIN
                      GotoXY(XCPos-1,YCPosSav);
                      Write(' ');
                      GotoXY(XCPos-1,YCPosSav);
                      XCPos:= XCPos-1;
                      InpStr:= Copy(InpStr,1,XCPos-XCPosSav);
                    END;
          END
          ELSE
          BEGIN
                    Write(InpChar);
                    XCPos:= XCPos+1;
                    IF NOT (InpChar=Char(13)) THEN InpStr:= InpStr+InpChar;
                    InpOkay:= True;
          END;
          END;
        UNTIL InpChar=Char(13);
        Writeln;
```

```
  IF InpOkay THEN
  BEGIN
   InputString:= InpStr;
  END
  ELSE
  BEGIN
   GotoXY(1,YCPosSav);
   ClrEOL;
  END;
 END;


{------------------------------------------------------------------
 PROCEDURE PInpChar:  uses ReadKey command to read string var
              (needs to be in GRAFICS mode !!!)
 ------------------------------------------------------------------}
 PROCEDURE PInpChar(MenuX,MenuY,Color,BkColor: Integer; VAR MenuStr: TStr80;
             VAR InputString: TStr80);
 VAR
  StrLength:  Integer;
  Ch     :  Char;
  InputStr :  String;
 BEGIN
  InputStr:= '';
  StrLength:= 0;
  REPEAT
   Ch:= ReadKey;
   IF Ch=Char(27) THEN
   BEGIN
    InputString:= Char(27);
    Exit;
   END
   ELSE
   BEGIN
    IF (Ch=Chr(8)) AND (StrLength<Length(InputStr)+1)
    AND (StrLength>0) THEN
    BEGIN
     PClearMenuXY(MenuX,MenuY,Color,BkColor,MenuStr);
     StrLength:= StrLength-1;
     InputStr:= Copy(InputStr,1,(Length(InputStr)-1));
     MenuStr:= Copy(MenuStr,1,(Length(MenuStr)-1));
```

```
    END;
    IF NOT (Ch=Chr(13)) AND (Ch IN ['!'..'~']) THEN
    BEGIN
      StrLength:= StrLength+1;
      InputStr:= InputStr+Ch;
      MenuStr:= MenuStr+Ch;
    END;
  END;
  InputString:= InputStr;
  PMenuXY(MenuX,MenuY,Color,BkColor,MenuStr);
 UNTIL Ch IN [Chr(13)];
END;


{------------------------------------------------------------------

PROCEDURE InputFile:  checks that input file exists and opens input file

          FileName=Char(27) if ESC

              =NonEx if not existent

------------------------------------------------------------------------}

PROCEDURE InputFile(FilePrompt: TStr40; VAR FileFil: Text; FilePath: TStr40;
          VAR FileName: TStr40);
VAR
 Ch       : Char;
 IOokay    : Boolean;
BEGIN
 IOokay:=False;
 REPEAT
  Write(FilePrompt);
  InpChar(FileName);
  IF FileName=Char(27) THEN Exit;
  IF FilePath='LocalDir' THEN
  BEGIN
   Assign(FileFil,FileName);
  END
  ELSE
  BEGIN
   Assign(FileFil,FilePath+FileName);
  END;
  {$I-}
  Reset(FileFil);
  Close(FileFil);
  {$I+}
```

```
        IF IOResult<>0 THEN
        BEGIN
          FileName:= 'NonEx';
          Writeln('*** file not found!! ***');
          IOokay:=False;
        END
        ELSE
        BEGIN
          Reset(FileFil);
          IOokay:=True;
        END;
      UNTIL IOokay;
    END;


{------------------------------------------------------------------------
  PROCEDURE PInputFile:  checks that input file exists and opens input file
                  (needs to be in GRAFICS mode !!!)
                  FileName=Char(27) if ESC
                          =NonEx if not existent
  ------------------------------------------------------------------------}
  PROCEDURE PInputFile(FilePrompt: TStr80; VAR FileFil: Text; FilePath: TStr40;
              VAR FileName: TStr40; MenuX,MenuY,Color,BkColor: Integer;
              VAR MenuStr: TStr80);
  VAR
    Ch      : Char;
    IOokay    : Boolean;
  BEGIN
    IOokay:=False;
    REPEAT
      MenuStr:= FilePrompt;
      PMenuXY(MenuX,MenuY,Color,BkColor,MenuStr);
      PInpChar(MenuX,MenuY,Color,BkColor,MenuStr,FileName);
      IF FileName=Char(27) THEN
      BEGIN
        Exit;
      END;
      IF FilePath='LocalDir' THEN
      BEGIN
        Assign(FileFil,FileName);
      END
      ELSE
```

```
  BEGIN
    Assign(FileFil,FilePath+FileName);
  END;
  {$I-}
  Reset(FileFil);
  Close(FileFil);
  {$I+}
  IF IOResult<>0 THEN
  BEGIN
    FileName:= 'NonEx';
    PClearMenuXY(MenuX,MenuY,Color,BkColor,MenuStr);
    MenuStr:= '*** file NOT found !! ***';
    PMenuXY(MenuX,MenuY,Color,BkColor,MenuStr);
    Delay(500);
    IOokay:=False;
    PClearMenuXY(MenuX,MenuY,Color,BkColor,MenuStr);
  END
  ELSE
  BEGIN
    Reset(FileFil);
    IOokay:=True;
  END;
  UNTIL IOokay;
END;


{-------------------------------------------------------------------
 PROCEDURE OutputFile:  checks that out putfile doesnot already exist and
               opens output file
               FileName=Char(27) if ESC
                    =AlreadyEx if already existent
 -------------------------------------------------------------------}
PROCEDURE OutputFile(FilePrompt: TStr40; VAR FileFil: Text; FilePath: TStr40;
         VAR FileName: TStr40);
VAR
  Ch     : Char;
  IOokay   : Boolean;
BEGIN
  IOokay:=False;
  REPEAT
    Write(FilePrompt);
    InpChar(FileName);
```

```
IF FileName=Char(27) THEN Exit;
IF FilePath='LocalDir' THEN
BEGIN
  Assign(FileFil,FileName);
END
ELSE
BEGIN
  Assign(FileFil,FilePath+FileName);
END;
{$I-}
Reset(FileFil);
Close(FileFil);
{$I+}
IF IOResult=0 THEN
BEGIN
  FileName:= 'AlreadyEx';
  Writeln('*** file already exists!! ***');
  Write('new file name ?  (Y/N) : ');
  REPEAT
        Ch:= ReadKey;
  UNTIL Ch IN ['n','N','y','Y'];
  Writeln(Ch);
  IF (Ch='N') OR (Ch='n') THEN
  BEGIN
    {$I-}
    Rewrite(FileFil);
    {$I+}
    IF IOResult<>0 THEN
    BEGIN
      Writeln('*** write ERROR!! ***');
      IOokay:=False;
    END
    ELSE
    BEGIN
      IOokay:=True;
    END;
  END
  ELSE
    IOokay:=False;
END
ELSE
```

```
      BEGIN
        {$I-}
        Rewrite(FileFil);
        {$I+}
        IF IOResult<>0 THEN
        BEGIN
         Writeln('*** write ERROR!! ***');
         IOokay:=False;
        END
        ELSE
        BEGIN
         IOokay:=True;
        END;
      END;
    UNTIL IOokay;
  END;


{---------------------------------------------------------------------

  PROCEDURE POutputFile:  checks that out putfile doesnot already exist and

                opens output file
                (needs to be in GRAFICS mode !!!)
                FileName=Char(27) if ESC
                    =AlreadyEx if already existent
  ----------------------------------------------------------------------}
  PROCEDURE POutputFile(FilePrompt: TStr80; VAR FileFil: Text; FilePath: TStr40;
              VAR FileName: TStr40; MenuX,MenuY,Color,BkColor: Integer;
              VAR MenuStr: TStr80);
VAR
  Ch       : Char;
  IOokay    : Boolean;
BEGIN
  IOokay:=False;
  REPEAT
    MenuStr:= FilePrompt;
    PMenuXY(MenuX,MenuY,Color,BkColor,MenuStr);
    PInpChar(MenuX,MenuY,Color,BkColor,MenuStr,FileName);
    IF FileName=Char(27) THEN
    BEGIN
     Exit;
    END;
    IF FilePath='LocalDir' THEN
```

```
BEGIN
  Assign(FileFil,FileName);
END
ELSE
BEGIN
  Assign(FileFil,FilePath+FileName);
END;
{$I-}
Reset(FileFil);
Close(FileFil);
{$I+}
IF IOResult=0 THEN
BEGIN
  FileName:= 'AlreadyEx';
  PClearMenuXY(MenuX,MenuY,Color,BkColor,MenuStr);
  MenuStr:= '*** file already EXISTS !! ***';
  PMenuXY(MenuX,MenuY,Color,BkColor,MenuStr);
  Delay(500);
  PClearMenuXY(MenuX,MenuY,Color,BkColor,MenuStr);
  MenuStr:= 'new file name ?  (Y/N) : ';
  PMenuXY(MenuX,MenuY,Color,BkColor,MenuStr);
  REPEAT
       Ch:= ReadKey;
  UNTIL Ch IN ['n','N','y','Y'];
  MenuStr:= MenuStr+Ch;
  PMenuXY(MenuX,MenuY,Color,BkColor,MenuStr);
  PClearMenuXY(MenuX,MenuY,Color,BkColor,MenuStr);
  IF (Ch='N') OR (Ch='n') THEN
  BEGIN
   {$I-}
   Rewrite(FileFil);
   {$I+}
   IF IOResult<>0 THEN
   BEGIN
    PClearMenuXY(MenuX,MenuY,Color,BkColor,MenuStr);
    MenuStr:= '*** write ERROR !! ***';
    PMenuXY(MenuX,MenuY,Color,BkColor,MenuStr);
    Delay(500);
    IOokay:=False;
    PClearMenuXY(MenuX,MenuY,Color,BkColor,MenuStr);
   END
```

```
        ELSE
        BEGIN
         IOokay:=True;
        END;
       END
       ELSE
       IOokay:=False;
      END
      ELSE
      BEGIN
       {$I-}
       Rewrite(FileFil);
       {$I+}
       IF IOResult<>0 THEN
       BEGIN
        PClearMenuXY(MenuX,MenuY,Color,BkColor,MenuStr);
        MenuStr:= '*** write ERROR !! ***';
        PMenuXY(MenuX,MenuY,Color,BkColor,MenuStr);
        Delay(500);
        IOokay:=False;
        PClearMenuXY(MenuX,MenuY,Color,BkColor,MenuStr);
       END
       ELSE
       BEGIN
        IOokay:=True;
       END;
      END;
     UNTIL IOokay;
    END;


 {----------------------------------------------------------------

  PROCEDURE ReadReal:   checks that input is REAL and is in input interval


  ------------------------------------------------------------------}
  PROCEDURE ReadReal(InpString: TStr40; DoLimit,UpLimit: Real; VAR RealInp: Real);
  VAR
   IOokay    : Boolean;
   RealStr   : TStr40;
   Code      : Integer;
  BEGIN
   IOokay:=False;
```

```
    REPEAT
      Write(InpString);
      {$I-}
{    ReadIn(RealInp); }
      InpChar(RealStr);
{    IF RealStr=Esc THEN Exit; }
      Val(RealStr,RealInp,Code);
      IF Code<>0 THEN
      BEGIN
        Writeln('*** input ERROR !! ***');
      END
      ELSE
      BEGIN
        IF (RealInp < DoLimit) OR (RealInp > UpLimit) THEN
        BEGIN
          IOokay:=False;
          Writeln('*** out of Range !! ***');
        END
        ELSE
          IOokay:=True;
      END;
      {$I+}
    UNTIL IOokay;
END;


{----------------------------------------------------------------------
PROCEDURE PReadReal:   checks that input is REAL and is in input interval
        (needs to be in GRAFICS mode!)
-------------------------------------------------------------------------}
PROCEDURE PReadReal(InpString: TStr80; DoLimit,UpLimit: Real;
              VAR RealInp: Real; MenuX,MenuY,Color,BkColor: Integer;
              VAR MenuStr: TStr80);
VAR
  IOokay    : Boolean;
  RealStr   : TStr40;
  Code      : Integer;
BEGIN
  IOokay:=False;
  REPEAT
    MenuStr:= InpString;
    PMenuXY(MenuX,MenuY,Color,BkColor,MenuStr);
```

```pascal
      PInpChar(MenuX,MenuY,Color,BkColor,MenuStr,RealStr);
      {$I-}
      Val(RealStr,RealInp,Code);
      IF Code<>0 THEN
      BEGIN
       PClearMenuXY(MenuX,MenuY,Color,BkColor,MenuStr);
       MenuStr:= '*** input ERROR !! ***';
       PMenuXY(MenuX,MenuY,Color,BkColor,MenuStr);
       Delay(500);
       PClearMenuXY(MenuX,MenuY,Color,BkColor,MenuStr);
      END
      ELSE
      BEGIN
       IF (RealInp < DoLimit) OR (RealInp > UpLimit) THEN
       BEGIN
        IOokay:=False;
        PClearMenuXY(MenuX,MenuY,Color,BkColor,MenuStr);
        MenuStr:= '*** out of RANGE !! ***';
        PMenuXY(MenuX,MenuY,Color,BkColor,MenuStr);
        Delay(500);
        PClearMenuXY(MenuX,MenuY,Color,BkColor,MenuStr);
       END
       ELSE
        IOokay:=True;
      END;
      {$I+}
     UNTIL IOokay;
END;


{-----------------------------------------------------------------------

 PROCEDURE ReadInt:   checks that input is INTEGER and is in input interval


 ------------------------------------------------------------------------}
PROCEDURE ReadInt(InpString: TStr40; DoLimit,UpLimit: Integer;
           VAR IntInp: Integer);
VAR
 IOokay    : Boolean;
 IntStr    : TStr40;
 Code      : Integer;
BEGIN
 IOokay:=False;
```

```pascal
   REPEAT
    Write(InpString);
    {$I-}
    InpChar(IntStr);
    Val(IntStr,IntInp,Code);
    IF Code<>0 THEN
    BEGIN
     Writeln('*** input ERROR !! ***');
    END
    ELSE
    BEGIN
     IF (IntInp < DoLimit) OR (IntInp > UpLimit) THEN
     BEGIN
      IOokay:=False;
      Writeln('*** out of Range !! ***');
     END
     ELSE
      IOokay:=True;
    END;
    {$I+}
   UNTIL IOokay;
  END;


{-------------------------------------------------------------------
 PROCEDURE PReadInt:   checks that input is INTEGER and is in input interval
       (needs to be in GRAFICS mode!)
 -----------------------------------------------------------------------}
 PROCEDURE PReadInt(InpString: TStr80; DoLimit,UpLimit: Real;
          VAR IntInp: Integer; MenuX,MenuY,Color,BkColor: Integer;
          VAR MenuStr: TStr80);
 VAR
  IOokay    : Boolean;
  IntStr   : TStr40;
  Code     : Integer;
 BEGIN
  IOokay:=False;
  REPEAT
   MenuStr:= InpString;
   PMenuXY(MenuX,MenuY,Color,BkColor,MenuStr);
   PInpChar(MenuX,MenuY,Color,BkColor,MenuStr,IntStr);
   {$I-}
```

```
            Val(IntStr,IntInp,Code);
          IF Code<>0 THEN
          BEGIN
            PClearMenuXY(MenuX,MenuY,Color,BkColor,MenuStr);
            MenuStr:= '*** input ERROR !! ***';
            PMenuXY(MenuX,MenuY,Color,BkColor,MenuStr);
            Delay(500);
            PClearMenuXY(MenuX,MenuY,Color,BkColor,MenuStr);
          END
          ELSE
          BEGIN
            IF (IntInp < DoLimit) OR (IntInp > UpLimit) THEN
            BEGIN
              IOokay:=False;
              PClearMenuXY(MenuX,MenuY,Color,BkColor,MenuStr);
              MenuStr:= '*** out of RANGE !! ***';
              PMenuXY(MenuX,MenuY,Color,BkColor,MenuStr);
              Delay(500);
              PClearMenuXY(MenuX,MenuY,Color,BkColor,MenuStr);
            END
            ELSE
              IOokay:=True;
          END;
          {$I+}
        UNTIL IOokay;
      END;


{----------------------------------------------------------------------

  PROCEDURE ReadStr:    reads a STRING


------------------------------------------------------------------------}
PROCEDURE ReadStr(InpString: TStr40; VAR StrInp: TStr80);
VAR
  IOokay    : Boolean;
BEGIN
  IOokay:=False;
  REPEAT
    Write(InpString);
    {$I-}
    InpChar(StrInp);
{   ReadIn(StrInp); }
```

```pascal
     IF IOResult<>0 THEN
     BEGIN
      Writeln('*** input ERROR !! ***');
     END
     ELSE
     BEGIN
      IOokay:=True;
     END;
     {$I+}
   UNTIL IOokay;
 END;


{-----------------------------------------------------------------------

 PROCEDURE PReadStr:   reads a STRING

                 (needs to be in GRAFICS mode!)

 ----------------------------------------------------------------------}
 PROCEDURE PReadStr(InpString: TStr80; DoLimit,UpLimit: Real;
             VAR StrInp: TStr80; MenuX,MenuY,Color,BkColor: Integer;
             VAR MenuStr: TStr80);
 VAR
  IOokay    : Boolean;
 BEGIN
  IOokay:=False;
  REPEAT
   MenuStr:= InpString;
   PMenuXY(MenuX,MenuY,Color,BkColor,MenuStr);
   PInpChar(MenuX,MenuY,Color,BkColor,MenuStr,StrInp);
   {$I-}
   IF IOResult<>0 THEN
   BEGIN
    PClearMenuXY(MenuX,MenuY,Color,BkColor,MenuStr);
    MenuStr:= '*** input ERROR !! ***';
    PMenuXY(MenuX,MenuY,Color,BkColor,MenuStr);
    Delay(500);
    PClearMenuXY(MenuX,MenuY,Color,BkColor,MenuStr);
   END
   ELSE
   BEGIN
    IOokay:=True;
   END;
   {$I+}
```

```
  UNTIL IOokay;
END;


{-------------------------------------------------------------------
 PROCEDURE ReadChar:   reads a CHARACTER


-------------------------------------------------------------------}
PROCEDURE ReadChar(InpString: TStr40; VAR CharInp: TStr1);
VAR
 IOokay    : Boolean;
BEGIN
 IOokay:=False;
 REPEAT
  Write(InpString);
  {$I-}
  InpChar(CharInp);
{   Readln(CharInp); }
  IF IOResult<>0 THEN
  BEGIN
   Writeln('*** input ERROR !! ***');
  END
  ELSE
  BEGIN
   IOokay:=True;
  END;
  {$I+}
 UNTIL IOokay;
END;


{-------------------------------------------------------------------
 PROCEDURE PReadChar:   reads a CHARACTER
              (needs to be in GRAFICS mode!)
-------------------------------------------------------------------}
PROCEDURE PReadChar(InpString: TStr80; DoLimit,UpLimit: Real;
           VAR CharInp: TStr1; MenuX,MenuY,Color,BkColor: Integer;
           VAR MenuStr: TStr80);
VAR
 IOokay    : Boolean;
BEGIN
 IOokay:=False;
 REPEAT
```

```
      MenuStr:= InpString;
      PMenuXY(MenuX,MenuY,Color,BkColor,MenuStr);
      PInpChar(MenuX,MenuY,Color,BkColor,MenuStr,CharInp);
      {$I-}
      IF IOResult<>0 THEN
      BEGIN
       PClearMenuXY(MenuX,MenuY,Color,BkColor,MenuStr);
       MenuStr:= '*** input ERROR !! ***';
       PMenuXY(MenuX,MenuY,Color,BkColor,MenuStr);
       Delay(500);
       PClearMenuXY(MenuX,MenuY,Color,BkColor,MenuStr);
      END
      ELSE
      BEGIN
       IOokay:=True;
      END;
      {$I+}
     UNTIL IOokay;
   END;


   END.          {IMPLEMENTATION of IO_Check }
```

P_ANDATA.PAS

```
{
  file     : P_ANDATA.PAS
  function : data
  author   : W.Maring
  changes  : 05-07-93
}

{$I P_ComOpt}


UNIT P_AnData;
{$IFDEF CPU87}
{$N+}
{$ELSE}
{$N-}
{$ENDIF}


INTERFACE

{$IFDEF AngSpec}
TYPE
{$IFDEF CPU87}
  Real        = Single;
{$ENDIF}
TArray       = Array [0..2000] of Real;
TMessRec = RECORD
                AngStart,
                AngEnd,
                DifAng,
                DelAng,
                Mass,
                URamp,
                SigAverage,
                Signal,
                DarkCnt,
                LaserFreq,
                IntegTime,
                Difx,
                YScale,
```

```
                    XValMax,
                    YValMax     : Real;
                    XVal,
                    YVal,
            ActPlotData : TArray;
                    IntSpec,
                    IntNum,
            IntRange,
                    NumScan,
                    NInt,
                    NScan       : Integer;
                    NormScale,
                    NewFile,
                    EnExit,
                    PltHP       : Boolean;
                    Comment     : String[80];
                    PltDirec,
            FileDirec,
                    ExeDirec,
            PltFileName,
                    FileName    : String[60];
                    PltFil,
                    DatFil      : Text;
                    END;
        {$ENDIF}



        IMPLEMENTATION


        END.•
```

P_CAM2.PAS

```
{
    file     : P_CAM2.PAS
    function : programming CAMAC functions
    author   : B.Ungerer, MPI Stroemungsforschung, Goettingen, FRG
    files    : MES4OBJ.OBJ
    changes  : 04.08.88//09.02.93//18.02.93
}


UNIT P_Cam2;
{$IFDEF CPU87}
{$N+}
{$ELSE}
{$N-}
{$ENDIF}


INTERFACE

USES
  crt,
  dos,
  printer;


TYPE
{$IFDEF CPU87}
  Real       = Single;
{$ENDIF}


{$L Mes4Obj }  { orig.:Cam6001.Asm, without "Real_cvt" }


PROCEDURE Init_Crate;
FUNCTION TimeConv(timeval:Real): Integer;
PROCEDURE Clear_TimSca(subadress,d:Integer);
PROCEDURE Clear_LAM(subadress,d:Integer);
PROCEDURE Write_TimSca(subadress,d:Integer);
PROCEDURE Read_Scaler(subadress:Integer; VAR signal:Integer);
PROCEDURE Wait_Scaler;
PROCEDURE Init_TimSca;
PROCEDURE Reset_TimSca;
```

```
PROCEDURE Start_TimSca(time:Real);
PROCEDURE Set_DAC1(subadress:Integer; volt:Real);
PROCEDURE Set_DAC2(subadress:Integer; volt:Real);
PROCEDURE Get_ADC1(subadress:Integer; VAR signal:Real);
PROCEDURE Get_ADC2(subadress:Integer; VAR signal:Real);


CONST
  N0   : Integer = 0;
  N1   : Integer = 1;
  N2   : Integer = 2;
  N3   : Integer = 3;
  N9   : Integer = 9;
  N10  : Integer = 10;
  N16  : Integer = 16;
  N17  : Integer = 17;
  N23  : Integer = 23;
  N25  : Integer = 25;
  N26  : Integer = 26;
  N27  : Integer = 27;
  N64  : Integer = 64;


{-----------------------------------------------------------------

  modul addresses :

-----------------------------------------------------------------}

  timsca : Integer = 10;   {timer/scaler}
  dac    : Integer = 12;   {12bit DAC}
  adc    : Integer =  9;   {12bit ADC}


{-----------------------------------------------------------------

  sub addresses :

-----------------------------------------------------------------}

  dac_c3 : Integer = 3;   {DAC: Ch #3}
  adc_c1 : Integer = 0;   {ADC: Ch #1}
  adc_c2 : Integer = 1;   {ADC: Ch #2}
  tim_c1 : Integer = 0;   {Timer: Ch #1}
  tim_c2 : Integer = 1;   {Timer: Ch #2}
  sca_c1 : Integer = 1;   {Scaler: Ch #1}
  sca_c2 : Integer = 3;   {Scaler: Ch #2}


{-----------------------------------------------------------------
```

```
----------------------------------------------------------------------}

  d  : integer = 0;
  q  : integer = 0;
  x  : integer = 0;
  ld : LongInt = 0;


{----------------------------------------------------------------------

  data to calculate time value for timer/scaler modul

  ----------------------------------------------------------------------}

  OvfTab : Array [1..32] of Integer = (4,9,8,9,7,9,8,9,6,9,8,9,7,9,8,9,5,
                                       9,8,9,7,9,8,9,6,9,8,9,7,9,8,9);
  MulTab : Array [1..32] of Integer = ( 0,16,8,17,4,18,9,19,2,20,10,21,5,
                                       22,11,23,1,24,12,25,6,26,13,27,3,
                                       28,14,29,7,30,15,31);


  CountInit: LongInt = $00FFFFFF;


  NADC = 3;


IMPLEMENTATION

{ Beginn DSP-Software }
{
  Declarations of CAMAC I/O routines for inclusion in
  a PASCAL program.
}


PROCEDURE Crate_Set (VAR Crate : INTEGER);              EXTERNAL;
PROCEDURE Camo (VAR N,F,A,Data : INTEGER;
                    VAR Q,X : INTEGER);                 EXTERNAL;
PROCEDURE Cami (VAR N,F,A : INTEGER;
                    VAR Data,Q,X : INTEGER);            EXTERNAL;
PROCEDURE Cami24 (VAR N,F,A : INTEGER;
                  VAR Data : LongInt;
                  VAR Q,X : INTEGER);                   EXTERNAL;
PROCEDURE Camo24 (VAR N,F,A : INTEGER;
                  VAR Data : LongInt;
                  VAR Q,X : INTEGER);                   EXTERNAL;
PROCEDURE Caml (VAR Encoded_Lam : INTEGER);             EXTERNAL;
PROCEDURE Camcl (VAR Control_word : INTEGER);           EXTERNAL;
PROCEDURE DMAset (VAR Crate,Nbytes,Qmode,Count : INTEGER);   EXTERNAL;
```

```
PROCEDURE DMAI (VAR N,F,A : INTEGER;

                Data : Longint ;

                VAR Error : INTEGER);              EXTERNAL;

PROCEDURE DMAO (VAR N,F,A : INTEGER;

                Data  : Longint;

                VAR Error : INTEGER);             EXTERNAL;

PROCEDURE CamCyc (VAR ncycles : INTEGER);         EXTERNAL;

PROCEDURE Crate ;                        EXTERNAL;
```

{************ Ende DSP - Software ********************************}

```
{----------------------------------------------------------------

PROCEDURE Init_Crate

         modul   : dsp 6001 CAMAC crate controller

         function : initilize CC #1

-----------------------------------------------------------------}

PROCEDURE Init_Crate;

BEGIN

 Crate_Set(N1);

 Camcl(N64);

 Camcl(N1);

END;


{----------------------------------------------------------------

FUNCTION TimeConv

         function : converting time into timer/scaler format

-----------------------------------------------------------------}

FUNCTION TimeConv(timeval:Real): Integer;

VAR

 TimeBas  : Real;

 NCount   : Integer;

 j,

 TimeB    : LongInt;

 ContLoop : Boolean;

BEGIN

 ContLoop:= True;

 timeval:= timeval/1000.0;

 WHILE ContLoop DO

 BEGIN

  TimeB:= Round(Ln(timeval/2.30258)/Ln(10));

  TimeBas:= Exp(TimeB*Ln(10));
```

```
        timeval:= timeval/TimeBas*1000.0;
      NCount:= 0;
      WHILE (timeval>2.0) DO
      BEGIN
        timeval:= timeval/2.0;
        NCount:= NCount + 1;
      END;
      j:= Round(32*(timeval-1)+1);
      ContLoop:= False;
      IF (NCount-OvfTab[j])<0 THEN
      BEGIN
        TimeB:=TimeB-1;
        ContLoop:= True;
      END;
      IF (NCount-OvfTab[j])>15 THEN
      BEGIN
        TimeB:=TimeB+1;
        ContLoop:= True;
      END;
     END;
    TimeB:=TimeB+6;
    IF (TimeB<0) OR (TimeB>7) THEN Writeln('ERROR: Time outside range!!!');
    TimeConv:=(128*MulTab[j])+(8*(NCount-OvfTab[j])+TimeB);
  END;


{---------------------------------------------------------------

  PROCEDURE Clear_TimSca

              modul   : sec_ts201 timer/scaler

              input   : subadress

-----------------------------------------------------------------}

  PROCEDURE Clear_TimSca(subadress,d:Integer);
  BEGIN
   camo(timsca,N9,subadress,d,q,x);
  END;


{---------------------------------------------------------------

  PROCEDURE Clear_LAM

              modul   : sec_ts201 timer/scaler

              input   : subadress

-----------------------------------------------------------------}

  PROCEDURE Clear_LAM(subadress,d: Integer);
```

```
BEGIN
  camo(timsca,N23,subadress,d,q,x);
END;


{--------------------------------------------------------------
 PROCEDURE Write_TimSca
            modul  : sec_ts201 timer/scaler
            input  : subadress
 --------------------------------------------------------------}
PROCEDURE Write_TimSca(subadress,d: Integer);
BEGIN
  camo(timsca,N17,subadress,d,q,x);
END;


{--------------------------------------------------------------
 PROCEDURE Read_Scaler
            modul  : sec_ts201 timer/scaler
            input  : subadress (1,3)
 --------------------------------------------------------------}
PROCEDURE Read_Scaler(subadress:Integer; VAR signal: Integer);
CONST
  data: LongInt= 0;
BEGIN
  cami24(timsca,N0,subadress,data,q,x);
  signal:= countinit - data;
END;


{--------------------------------------------------------------
 PROCEDURE Wait_Scaler
            modul  : sec_ts201 timer/scaler
            input  : subadress (1,3)
 --------------------------------------------------------------}
PROCEDURE Wait_Scaler;
VAR
  LTest  : Integer;
BEGIN
  LTest:=0;
  WHILE LTest=0 DO
  BEGIN
   CAML(LTest);
  END;
```

```
END;


{-------------------------------------------------------------------

 PROCEDURE Init_TimSca


-------------------------------------------------------------------}

PROCEDURE Init_TimSca;
BEGIN
 Write_TimSca(13,1);
END;


{-------------------------------------------------------------------

 PROCEDURE Reset_TimSca


-------------------------------------------------------------------}

PROCEDURE Reset_TimSca;
BEGIN
 Clear_TimSca(0,0);
 Clear_TimSca(1,0);
END;


{-------------------------------------------------------------------

 PROCEDURE Start_TimSca

         input:   Time value
-------------------------------------------------------------------}

PROCEDURE Start_TimSca(time:Real);
BEGIN
 Write_TimSca(13,1);
 Write_TimSca(0,TimeConv(time));
 Write_TimSca(4,1);
 Wait_Scaler;
 Clear_LAM(12,1);
END;


{-------------------------------------------------------------------

 PROCEDURE Set_DAC1

         modul    : dspE250 DAC, 0...+10V range !!!

         input    : subadress, volt

-------------------------------------------------------------------}

PROCEDURE Set_DAC1(subadress:Integer; volt:Real);
VAR
```

213

```pascal
  digval : Integer;
CONST
 r2_xx : Real = 32767.0;
BEGIN
 digval:=(round (volt*r2_xx/5.0 - r2_xx) + 0);
 Camo(dac,N16,subadress,digval,q,x);
END;


{------------------------------------------------------------------

 PROCEDURE Set_DAC2

             modul  : dspE250 DAC, -5...+5V range !!!

             input  : subadress, volt

 ------------------------------------------------------------------}

PROCEDURE Set_DAC2(subadress:Integer; volt:Real);
VAR
 digval : Integer;
CONST
 r2_xx : Real = 32767.0;
BEGIN
 digval:=(round (volt*r2_xx/5.0) + 0);
 Camo(dac,N16,subadress,digval,q,x);
END;


{------------------------------------------------------------------

 PROCEDURE Get_ADC1

             modul  : sec_adc, -5...+5V

             input  : subadress

             output : sig

 ------------------------------------------------------------------}

PROCEDURE Get_ADC1 (subadress:Integer; VAR signal:Real);
VAR
 i,
 digval : Integer;
CONST
 r2_xx  : Real = 4096.0;
BEGIN
 FOR i:=1 TO NADC DO
 BEGIN
  Cami(adc,N2,subadress,digval,q,x);
 END;
 Signal:= -1.0*digval*10.0/r2_xx;
```

END;

```
{----------------------------------------------------------------
 PROCEDURE Get_ADC2
              modul   : sec_adc, -10...+10V
              input   : subadress
              output  : sig
 ----------------------------------------------------------------}
PROCEDURE Get_ADC2 (subadress:Integer; VAR signal:Real);
VAR
 i,
 digval   : Integer;
CONST
 r2_xx    : Real = 4096.0;
BEGIN
 FOR i:=1 TO NADC DO
 BEGIN
  Cami(adc,N2,subadress,digval,q,x);
 END;
 Signal:= -1.0*digval*20.0/r2_xx;
END;


END.           {IMPLEMENTATION OF PCAM2 }
```

P_FDATA.PAS

```
{
  file     : P_FDATA.PAS
  function : data
  author   : W.Maring
  changes  : 05-07-93
}

{$I P_ComOpt}

UNIT P_FData;
{$IFDEF CPU87}
{$N+}
{$ELSE}
{$N-}
{$ENDIF}

INTERFACE

{$IFDEF FreqSpec}
TYPE
{$IFDEF CPU87}
  Real        = Single;
{$ENDIF}
TArray        = Array [0..2000] of Real;
TMessRec = RECORD
                EStart,
                EWidth,
                DifEn,
                DelEn,
                Mass,
                URamp,
                SigAverage,
                Signal,
                DarkCnt,
            DetPos,
                IntegTime,
                Difx,
                YScale,
```

```
                XValMax,
                YValMax     : Real;
                XVal,
                YVal,
        ActPlotData  : TArray;
                IntSpec,
                IntNum,
        IntRange,
                NumScan,
                NInt,
                NScan      : Integer;
                NormScale,
                NewFile,
                EnExit,
                PltHP      : Boolean;
                Comment    : String[80];
                PltDirec,
        FileDirec,
                ExeDirec,
        PltFileName,
                FileName    : String[60];
                PltFil,
                DatFil     : Text;
                END;
{$ENDIF}


IMPLEMENTATION

END.
```

P_GRAF.PAS

```
{
  file     : P_GRAF.PAS
  function  : plotting of data
  author   : W.Maring
  files    : P_GrafIn.Pas (inividual settings)
  changes  : 09/02/93
           14/05/93  (input in GRAFIC mode)
}

{$I P_ComOpt}

UNIT P_Graf;
{$IFDEF CPU87}
{$N+}
{$ELSE}
{$N-}
{$ENDIF}

INTERFACE

USES
  Crt,
  Dos,
  Graph,
  Printer,
{$IFDEF Masspec}
  P_MData,
{$ENDIF}
{$IFDEF Augspec}
  P_AData,
{$ENDIF}
{$IFDEF Angspec}
  P_AnData,
{$ENDIF}
{$IFDEF Freqspec}
  P_FData,
{$ENDIF}
  IO_Check;
```

```
TYPE
{$IFDEF CPU87}
 Real        = Single;
{$ENDIF}
 TStr2      = String[2];
 TStr6      = String[6];
 TStr10     = String[10];
 TStr30     = String[30];
 TStr80     = String[80];
 TGrafRec   = RECORD
          PlotType  : TStr10;
                    GDirec    : TStr30;
                    Color,
                    BkColor,
                    ScalCount,
                    ScalCountCorr,
                    ScalDigCount,
                    XDig1,
                    XDig2,
          XPix,
          YPix,
          XOffset,
          YOffset,
          GlattNum,
          NoiseLevel,
          MenuX,
          MenuY     : Integer;
                    YNorm,
                    YNormSav,
          XPixFac,
          YPixFac,
                    XScalInt,
          XScalIntLoc,
                    YScalInt,
          XScalStart,
                    YScalStart : Real;
          EnGrafics,
                    EnHrdCpy,
          PltHP     : Boolean;
```

```pascal
                END;

    VAR
     XPltFac,
     YPltFac,
     DifX        : Real;
     ii,
     StrLength,
     Gd,
     Gm,
     ErrCode,
     ValCode     : Integer;
     Ch          : Char;
     InputStr,
     MenuStr     : TStr80;
     GData       : TGrafRec;
     ExitSave    : Pointer;


    CONST
    {$IFDEF Masspec}
     YNeg      = -0.25;
     YPos      = 4.0;
     YTop      = 0.15;
    {$ENDIF}


    {$IFDEF Augspec}
     YNeg      = -5.25;
     YPos      = 5.25;
     YTop      = 0.2;
    {$ENDIF}


    {$IFDEF Angspec}
     YNeg      = -100;
     YPos      = 1000;
     YTop      = 50;
    {$ENDIF}


    {$IFDEF Freqspec}
     YNeg      = -100;
     YPos      = 1000;
     YTop      = 50;
```

```
{$ENDIF}

XPlt          : Integer = 8800;
YPlt          : Integer = 5950;
PltOffsetX    : Integer = 1000;
PltOffSetY    : Integer = 1000;
XPltTickHeight : Integer = 105;
YPltTickHeight : Integer = 105;

XPixTickHeight : Integer = 5;
YPixTickHeight : Integer = 5;

EndTrue     : Boolean = False;
Bpage       : Array [0..1] of Byte=(1,0);    { Fuer Page-Switching }
Buf         : Byte = 1;


PROCEDURE RunTimeError;
PROCEDURE PltSetColor(PltColor: Integer;VAR Fil: Text);
PROCEDURE PltLineType(PltLine: Integer;VAR Fil: Text);
PROCEDURE PltPlotStart(VAR Fil: Text);
PROCEDURE PltPlotStop(VAR Fil: Text);
PROCEDURE HrdCpy (VAR OptChar: Char);
PROCEDURE Hardcopy(VAR PlotData : TMessRec);
PROCEDURE GetXYValMax(VAR PlotData: TMessRec);
PROCEDURE PltLineXY(XP1,YP1,XP2,YP2: Integer;VAR Fil: Text);
PROCEDURE PLineXY(XP1,YP1,XP2,YP2: Integer; GData: TGrafRec);
PROCEDURE PltWriteXY(XP1,YP1: Integer; PlotString: TStr80;VAR Fil: Text);
PROCEDURE PWriteXY(XP1,YP1: Integer; PlotString: String; GData: TGrafRec);
PROCEDURE PltMoveToXY(XP1,YP1: Integer;VAR Fil: Text);
PROCEDURE PMoveToXY(XP1,YP1: Integer; GData: TGrafRec);
PROCEDURE PltCircleXY(XP1,YP1,Radius: Integer;VAR Fil: Text);
PROCEDURE PCircleXY(XP1,YP1,Radius: Integer; GData: TGrafRec);
PROCEDURE XPlotTicks(VAR PlotData: TMessRec; VAR GData: TGrafRec);
PROCEDURE YPlotTicks(VAR PlotData: TMessRec; GData: TGrafRec);
PROCEDURE XScaling(VAR PlotData: TMessRec; VAR GData: TGrafRec);
PROCEDURE YScaling(VAR PlotData: TMessRec; VAR GData: TGrafRec);
PROCEDURE PlotFrame(VAR PlotData: TMessRec; VAR GData: TGrafRec);
PROCEDURE ClearMenuStr(VAR GData: TGrafRec);
PROCEDURE PMenuStr(VAR GData: TGrafRec);
PROCEDURE SmoothData(VAR PlotData: TMessRec; VAR GData: TGrafRec);
```

```pascal
PROCEDURE CutNoise(VAR PlotData: TMessRec; VAR GData: TGrafRec);
PROCEDURE HPGLFile(VAR PlotData: TMessRec; VAR GData: TGrafRec);
PROCEDURE NextFile(VAR PlotData: TMessRec; VAR GData: TGrafRec);
PROCEDURE DataPlot(VAR PlotData: TMessRec; VAR GData: TGrafRec);
PROCEDURE GrafInit(VAR GData: TGrafRec);    {file: P_GrafIn.Pas}
PROCEDURE StartGraphics(VAR GData: TGrafRec);
PROCEDURE PlotLoop(VAR PlotData: TMessRec; VAR GData: TGrafRec);
PROCEDURE StopGraphics(VAR GData: TGrafRec);



IMPLEMENTATION



{-----------------------------------------------------------------

 PROCEDURE RunTimeError


 ----------------------------------------------------------------}
{$F+}
PROCEDURE RunTimeError;
VAR
 Ch    : Char;
BEGIN
 ExitProc:= ExitSave;
 IF GData.EnGrafics THEN
 BEGIN
  PWriteXY(100,100,'>>> RunTimeError, press any key ! ',GData);
  Ch:= Readkey;
  CloseGraph;
 END;
 Halt;
END;
{$F-}


{-----------------------------------------------------------------

 PROCEDURE PltSetColor


 ----------------------------------------------------------------}
PROCEDURE PltSetColor(PltColor: Integer;VAR Fil: Text);
BEGIN
 Write(Fil,'SP',Chr(PltColor+$30),';');
END;
```

```
{-------------------------------------------------------------------
 PROCEDURE PltLineType


-------------------------------------------------------------------}
PROCEDURE PltLineType(PltLine: Integer;VAR Fil: Text);
BEGIN
 IF PltLine < 0 THEN
   Write(Fil,'LT;')
 ELSE
   Write(Fil,'LT',Chr(PltLine+$30),';');
END;


{-------------------------------------------------------------------
 PROCEDURE PltPlotStart


-------------------------------------------------------------------}
PROCEDURE PltPlotStart(VAR Fil: Text);
BEGIN
 Write(Fil,Chr(27),'E',Chr(27),'%0B','IN;');   { Set: HPGL }
 Write(Fil,'RO90;');                  { Set: Portrait }
 PltSetColor(1,Fil);
END;


{-------------------------------------------------------------------
 PROCEDURE PltPlotStop


-------------------------------------------------------------------}
PROCEDURE PltPlotStop(VAR Fil: Text);
BEGIN
 Write(Fil,'IN;SP0;');
 Write(Fil,Chr(27),'%0A',Chr(27),'E');    { Set: PCL }
 Close(Fil);
END;


{-------------------------------------------------------------------
 PROCEDURE HrdCpy


-------------------------------------------------------------------}
PROCEDURE HrdCpy (VAR OptChar: Char);
BEGIN
```

```
inline
  ($06/              {      push    es                  }
  $55/               {      push    bp                  }
  $c4/$be/optchar/{         les     di,optchar[bp]      }
  $26/$8a/$1d/   {          mov     bl,es:[di]  }
  $31/$c0/           {      xor     ax,ax               }
  $8e/$c0/           {      mov     es,ax               }
  $bd/$16/$00/   {          mov     bp,16               }
  $26/$8b/$46/$00/{         mov     ax,es:[bp] }
  $8e/$c0/           {      mov     es,ax               }
  $bd/$30/$01/   {          mov     bp,130              }
  $26/$88/$5e/$00/{         mov     [bp],bl             }
  $cd/$05/           {      int     5                   }
  $5d/               {      pop     bp                  }
  $07);              {      pop     es                  }
END;
```

```
{----------------------------------------------------------------

 PROCEDURE Hardcopy


----------------------------------------------------------------}

PROCEDURE Hardcopy(VAR PlotData : TMessRec);
VAR
  c     : Char;
  Gesamt : Real;
  Steps : Integer;
BEGIN
  WITH PlotData DO
  BEGIN
   Assign(lst,'LPT1');
   Rewrite(lst);
   writeln(lst,chr(27),'3',chr(24));   { SET Graphics Linespace }
   c:=chr(Bpage[Buf]+49);
   hrdcpy(c);
   writeln(lst,chr(27),'2');           { START Text Linespace }
   FOR ii:=1 TO 35 DO
   BEGIN
    writeln(lst,'');
   END;
   Close(lst);
  END;
```

```
END;        {Ausgabe weiterer Infos: PROCEDURE Drucken/Mes4inc1.pas)}



{-----------------------------------------------------------------

 PROCEDURE GetXYValMax


----------------------------------------------------------------}
PROCEDURE GetXYValMax(VAR PlotData: TMessRec);
VAR
 ii      : Integer;
BEGIN
 WITH PlotData DO
 BEGIN
  XValMax:= 0.0;
  YValMax:= 0.0;
  FOR ii:=0 TO IntNum DO
   IF YVal[ii]>YValMax THEN
   BEGIN
         XValMax:= XVal[ii];
         YValMax:= YVal[ii];
   END;
  IF YValMAx=0.0 THEN YValMax:=1.0;
 END;
END;



{-----------------------------------------------------------------

 PROCEDURE PltLineXY:


----------------------------------------------------------------}
PROCEDURE PltLineXY(XP1,YP1,XP2,YP2: Integer;VAR Fil: Text);
VAR
 Xstr,
 Ystr   : TStr6;
BEGIN
 PltMoveToXY(XP1,YP1,Fil);
 Str(1.0*(XP2+PltOffSetX):6:2,Xstr);
 Str(1.0*(YP2+PltOffSetY):6:2,Ystr);
 Write(Fil,'PD',Xstr,',',Ystr,';');
END;
```

```
{----------------------------------------------------------------------
PROCEDURE PLineXY:


----------------------------------------------------------------------}
PROCEDURE PLineXY(XP1,YP1,XP2,YP2: Integer; GData: TGrafRec);
BEGIN
 WITH GData DO
 BEGIN
   Line(XP1+XOffset,YP1+YOffset,XP2+XOffset,YP2+YOffset);
  END;
END;


{----------------------------------------------------------------------
PROCEDURE PltWriteXY:


----------------------------------------------------------------------}
PROCEDURE PltWriteXY(XP1,YP1: Integer; PlotString: TStr80;VAR Fil: Text);
BEGIN
 PltMoveToXY(XP1,YP1,Fil);
 Write(Fil,'LB',PlotString,Chr(3));
END;


{----------------------------------------------------------------------
PROCEDURE PWriteXY:


----------------------------------------------------------------------}
PROCEDURE PWriteXY(XP1,YP1: Integer; PlotString: String; GData: TGrafRec);
BEGIN
 WITH GData DO
 BEGIN
   OutTextXY(XP1+XOffset,YP1+YOffset,PlotString);
  END;
END;


{----------------------------------------------------------------------
PROCEDURE PltMoveToXY


----------------------------------------------------------------------}
PROCEDURE PltMoveToXY(XP1,YP1: Integer;VAR Fil: Text);
VAR
 XStr,
```

```
  YStr: TStr6;
BEGIN
 Str(1.0*(XP1+PltOffSetX):6:2,XStr);
 Str(1.0*(YP1+PltOffSetY):6:2,YStr);
 Write(Fil,'PU',XStr,',',YStr,';');
END;


{-------------------------------------------------------------------
 PROCEDURE PMoveToXY


 -----------------------------------------------------------------}
PROCEDURE PMoveToXY(XP1,YP1: Integer; GData: TGrafRec);
BEGIN
 WITH GData DO
 BEGIN
  MoveTo(XP1+XOffset,YP1+YOffset);
 END;
END;


{-------------------------------------------------------------------
 PROCEDURE PltCircleXY


 -----------------------------------------------------------------}
PROCEDURE PltCircleXY(XP1,YP1,Radius: Integer;VAR Fil: Text);
VAR
 PltStr: TStr2;
 RStr  : TStr6;
BEGIN
 PltMoveToXY(XP1-38,YP1,Fil);
 Str(16.0*(Radius):3:1,RStr);
{ Write(Fil,'CI',RStr,';'); }
 PltStr:= '.';
 Write(Fil,'LB',PltStr,Chr(3));
END;


{-------------------------------------------------------------------
 PROCEDURE PCircleXY


 -----------------------------------------------------------------}
PROCEDURE PCircleXY(XP1,YP1,Radius: Integer; GData: TGrafRec);
BEGIN
```

```
  WITH GData DO
  BEGIN
    Circle(XP1+XOffset,YP1+YOffset,Radius);
  END;
END;


{---------------------------------------------------------------------
 PROCEDURE XPlotTicks
 ---------------------------------------------------------------------}
PROCEDURE XPlotTicks(VAR PlotData: TMessRec;VAR GData: TGrafRec);
VAR
  ii,
  XPixT0,
  XPltT0,
  YPixT0,
  YPltT0,
  YPixT1,
  YPltT1 : Integer;
  LStr   : TStr10;
BEGIN
  WITH PlotData, GData DO
  BEGIN
    FOR ii:=0 TO ScalCount-1+ScalCountCorr DO
    BEGIN
      XPixT0:= Round((XScalStart+ii*XScalIntLoc)*XPixFac);
      YPixT0:= YPix;
      YPixT1:= YPix-YPixTickHeight;
      IF (XPixT0<XPix+1) THEN
      BEGIN
            PMoveToXY(XPixT0,YPixT0,GData);
            PLineXY(XPixT0,YPixT0,XPixT0,YPixT1,GData);
            Str((1.0*ii*XScalIntLoc+XVal[0]
                +XScalStart):(XDig1+ScalDigCount):(XDig2+ScalDigCount),LStr);
            PWriteXY(XPixT0-10,(YPix+10),LStr,GData);
            IF PltHP THEN
            BEGIN
              XPltT0:= Round((XScalStart+ii*XScalIntLoc)*XPltFac);
              YPltT0:= 0;
              YPltT1:= YPltTickHeight;
              PltMoveToXY(XPltT0,YPltT0,PltFil);
              PltLineXY(XPltT0,YPltT0,XPltT0,YPltT1,PltFil);
```

```
                    Str((1.0*ii*XScalIntLoc+XVal[0]
                              +XScalStart):(XDig1+ScalDigCount):(XDig2+ScalDigCount),LStr);
                    PltWriteXY(XPltT0-190,-320,LStr,PltFil);
                END;
        END;
      END;
    END;
  END;


{----------------------------------------------------------------------
 PROCEDURE YPlotTicks


 ----------------------------------------------------------------------}
PROCEDURE YPlotTicks(VAR PlotData: TMessRec; GData: TGrafRec);
VAR
 ii,
 XPixT0,
 XPltT0,
 YPixT0,
 YPltT0,
 XPixT1,
 XPltT1 : Integer;
 LStr   : TStr10;
BEGIN
 WITH PlotData, GData DO
 BEGIN
  FOR ii:=0 TO ScalCount DO
  BEGIN
    YPixT0:= YPix+Round(YNeg*YPixFac)
              -Round((YScalStart+ii*YScalInt)*YPos/100.0*YPixFac);
    XPixT0:= 0;
    XPixT1:= YPixTickHeight;
    PMoveToXY(XPixT0,YPixT0,GData);
    PLineXY(XPixT0,YPixT0,XPixT1,YPixT0,GData);
    Str((1.0*(ii+ScalCountCorr)*YScalInt):3:0,LStr);
    PWriteXY(-30,YPixT0-2,LStr,GData);
    IF PltHP THEN
    BEGIN
          YPltT0:= -Round(YNeg*YPltFac)
                    +Round((YScalStart+ii*YScalInt)*YPos/100.0*YPltFac);
          XPltT0:= 0;
```

```
                XPltT1:= YPltTickHeight;
                PltMoveToXY(XPltT0,YPltT0,PltFil);
                PltLineXY(XPltT0,YPltT0,XPltT1,YPltT0,PltFil);
           Str((1.0*(ii+ScalCountCorr)*YScalInt):3:0,LStr);
                PltWriteXY(-440,YPltT0-50,LStr,PltFil);
      END;
    END;
  END;
END;


{-----------------------------------------------------------------

  PROCEDURE XScaling


  -----------------------------------------------------------------}
PROCEDURE XScaling(VAR PlotData: TMessRec; VAR GData: TGrafRec);
VAR
 ScalMag,
 ScalTest    : Real;
 ScalOK      : Boolean;
BEGIN
 WITH PlotData, GData DO
 BEGIN
  ScalOk:= False;
  ScalDigCount:= 0;
  XScalIntLoc:= XScalInt;
  ScalMag:= 1.0;
  WHILE NOT ScalOK DO
  BEGIN
   ScalCount:= 0;
   ScalCountCorr:= 0;
   ScalTest:= 10;
   WHILE ScalTest>1 DO
   BEGIN
        ScalTest:= Abs(Abs(XVal[0])-ScalCount*XScalIntLoc)/XScalIntLoc;
        ScalCount:= ScalCount+1;
   END;
   IF XVal[0]>0.0 THEN
   BEGIN
        XScalStart:= -XVal[0]+XScalIntLoc*ScalCount;
   END
   ELSE
```

```
     BEGIN
           XScalStart:= -Abs(XVal[0])+XScalIntLoc*ScalCount;
     END;


     IF XScalStart<XScalIntLoc/10 THEN ScalCountCorr:= 1;
     ScalCount:= Round(Int(ScalMag*(XVal[IntNum]-XVal[0]+XScalStart))
              /(ScalMag*XScalIntLoc));


     IF (Abs(XVal[IntNum]-XVal[0])<2.0*XScalIntLoc) THEN  {2.0}
     BEGIN
           XScalIntLoc:= XScalIntLoc/10.0;
       ScalMag:= ScalMag*10.0;
       ScalDigCount:= ScalDigCount+1;
      END
      ELSE
      BEGIN
        ScalOK:= True;
      END;
     END;
     IF XScalIntLoc>0.9 THEN
       ScalDigCount:= ScalDigCount-1
     ELSE
       ScalDigCount:= ScalDigCount-2;
      XPlotTicks(PlotData,GData);
     END;
    END;


    {---------------------------------------------------------------

     PROCEDURE YScaling


     --------------------------------------------------------------}

    PROCEDURE YScaling(VAR PlotData: TMessRec; VAR GData: TGrafRec);
    VAR
     ScalTest   : Real;
    BEGIN
     WITH PlotData, GData DO
     BEGIN
      YScalStart:= 0.0;
      ScalCountCorr:= 0;
      ScalCount:= 5;
      YPlotTicks(PlotData,GData);
```

```
  END;
END;


{------------------------------------------------------------------

 PROCEDURE PlotFrame


 ------------------------------------------------------------------}
PROCEDURE PLotFrame(VAR PlotData: TMessRec; VAR GData: TGrafRec);
VAR
 LStr     : TStr10;
 DatTimStr : TStr30;
BEGIN
 WITH PlotData, GData DO
 BEGIN
  PLineXY(0,0,XPix,0,GData);
  PLineXY(XPix,0,XPix,YPix,GData);
  PLineXY(XPix,YPix,0,YPix,GData);
  PLineXY(0,YPix,0,0,GData);
  PWriteXY(XPix-150,10,FileName,GData);
  IF NormScale THEN
  BEGIN
   Str(YScale*YNorm:4:2,LStr);
  END
  ELSE
  BEGIN
   Str(YScale:4:2,LStr);
  END;
  PWriteXY(XPix-150,20,'x '+LStr,GData);
  Str(YNormSav:5:2,LStr);
  PWriteXY(XPix-150,30,'YNorm: '+LStr,GData);
  IF NormScale THEN
  BEGIN
   Str((YValMax/(100.0*YScale)):10,LStr);
  END
  ELSE
  BEGIN
   Str((YValMax*YNormSav/(100.0*YScale)):10,LStr);
  END;
  PWriteXY(XPix-150,40,'Sig: x'+LStr,GData);
  Str(GlattNum:1,LStr);
  PWriteXY(XPix-150,50,'Smoothed Ch.: '+LStr,GData);
```

```
Str(NoiseLevel:1,LStr);
PWriteXY(XPix-150,60,'Noiselevel: '+LStr,GData);
Str(XVal[0]:4:1,LStr);
{$IFDEF Freqspec}
  Str(XVal[0]:6:3,LStr);
{$ENDIF}
PWriteXY(-10,-10,LStr,GData);
{$IFDEF Freqspec}
  Str(EStart:9:3,LStr);
  PWriteXY(50,-10,'('+LStr+')',GData);
{$ENDIF}
Str(XVal[IntNum]:4:1,LStr);
{$IFDEF Freqspec}
  Str(XVal[IntNum]:6:3,LStr);
{$ENDIF}
PWriteXY(XPix-20,-10,LStr,GData);
GetDateTime(DatTimStr);
PWriteXY(XPix-160,-20,DatTimStr,GData);
PWriteXY(0,-20,Comment,GData);

IF PltHP THEN
BEGIN
  PltLineXY(0,0,XPlt,0,PltFil);
  PltLineXY(XPlt,0,XPlt,YPlt,PltFil);
  PltLineXY(XPlt,YPlt,0,YPlt,PltFil);
  PltLineXY(0,YPlt,0,0,PltFil);
  PltWriteXY(XPlt-2000,YPlt-340,FileName,PltFil);
  Str(YScale:5:1,LStr);
  PltWriteXY(XPlt-2000,YPlt-510,'x'+LStr,PltFil);
  Str(YNormSav:5:2,LStr);
  PltWriteXY(XPlt-2000,YPlt-680,'Sig: x'+LStr,PltFil);
  Str(GlattNum:1,LStr);
  PltWriteXY(XPlt-2000,YPlt-750,'Smoothed Ch.: '+LStr,PltFil);
  Str(NoiseLevel:1,LStr);
  PltWriteXY(XPix-2000,YPlt-820,'Noiselevel: '+LStr,PltFil);
  Str(XVal[0]:6:3,LStr);
  PltWriteXY(-160,YPlt+80,LStr,PltFil);
  {$IFDEF Freqspec}
    Str(EStart:9:3,LStr);
    PltWriteXY(480,YPlt+80,'('+LStr+')',PltFil);
  {$ENDIF}
```

```
      Str(XVal[IntNum]:6:3,LStr);
      PltWriteXY(XPlt-240,YPlt+80,LStr,PltFil);
      PltWriteXY(XPlt-2050,YPlt+250,DatTimStr,PltFil);
      PltWriteXY(0,YPlt+250,Comment,PltFil);
     END;
   END;
  END;


{------------------------------------------------------------------

 PROCEDURE ClearMenuStr


 -----------------------------------------------------------------}

 PROCEDURE ClearMenuStr(VAR GData: TGrafRec);
 BEGIN
  WITH GData DO
  BEGIN
   SetColor(BkColor);
   PWriteXY(0,YPix+25,MenuStr,GData);
   SetColor(Color);
   END;
  END;


{------------------------------------------------------------------

 PROCEDURE PMenuStr


 -----------------------------------------------------------------}

 PROCEDURE PMenuStr(VAR GData: TGrafRec);
 BEGIN
  WITH GData DO
  BEGIN
   PWriteXY(0,YPix+25,MenuStr,GData);
  END;
  END;


{------------------------------------------------------------------

 PROCEDURE SmoothData:        using Savitzky-Golay-algorithmus


 -----------------------------------------------------------------}

 PROCEDURE SmoothData(VAR PlotData: TMessRec; VAR GData: TGrafRec);
 VAR
  ii,
```

```
  s        : Integer;
  SmoothBuf  : Real;
  Glatt,
  Abl_1,
  Abl_2     : Array [-20..20] of Real;
  LStr      : TStr10;
BEGIN
 WITH PlotData, GData DO
 BEGIN
  ClearMenuStr(GData);
  PReadInt('>>> Chan. l&r to be smoothed? : ',-20,20,GlattNum,MenuX,MenuY,Color,BkColor,MenuStr);
  FOR s:=-GlattNum TO GlattNum DO
  BEGIN
   Glatt[s] := 3*(3*sqr(GlattNum)+3*GlattNum-1-5*sqr(s)) /


          ((2*GlattNum+3)*(2*GlattNum+1)*(2*GlattNum-1));
{
   ClearMenuStr(GData);
   Str(Glatt[s]:10:4,LStr);
   Menustr:= LStr;
   PMenuStr(GData);
}
   Abl_1[s]:= 3*s /
                 ((2*GlattNum+1)*(GlattNum+1)*GlattNum);
   Abl_2[s]:= 30*(3*sqr(s)-GlattNum*(GlattNum+1)) /
                 ((2*GlattNum+3)*(2*GlattNum+1)*(2*GlattNum-1)*(GlattNum+1)*GlattNum);
  END;
  SmoothBuf:= 0.0;
  FOR ii:=GlattNum+1 TO IntNum-GlattNum-1 DO
  BEGIN
   FOR s:=-GlattNum TO GlattNum DO
   BEGIN
    SmoothBuf:= SmoothBuf+Glatt[s]*YVal[s+ii];
   END;
   ActPlotData[ii]:= SmoothBuf;
   SmoothBuf:= 0.0;
  END;
  FOR ii:=0 TO GlattNum DO ActPlotData[ii]:= -1.0*NoiseLevel;
  FOR ii:=IntNum-GlattNum TO IntNum DO ActPlotData[ii]:= -1.0*NoiseLevel;
 END;
END;
```

```
{----------------------------------------------------------------------
    PROCEDURE CutNoise:     substracts noiselevel


----------------------------------------------------------------------}
PROCEDURE CutNoise(VAR PlotData: TMessRec; VAR GData: TGrafRec);
VAR
 ii    : Integer;
BEGIN
 WITH PlotData,GData DO
 BEGIN
  ClearMenuStr(GData);
  PReadInt('>>> Noise level : ',-10,10000,NoiseLevel,MenuX,MenuY,Color,BkColor,MenuStr);
  FOR ii:=0 TO IntNum DO
  BEGIN
    YVal[ii]:= YVal[ii]-NoiseLevel;
    ActPlotData[ii]:= YVal[ii];
{    IF SmoothedData[ii]<0.0 THEN SmoothedData[ii]:=0.0; }
  END;
  GlattNum:= 1;    { CutNoise is using the ORIGINAL data in YVal! }
 END;
END;


{----------------------------------------------------------------------
 PROCEDURE HPGLFile


----------------------------------------------------------------------}
PROCEDURE HPGLFile(VAR PlotData: TMessRec; VAR GData: TGrafRec);
BEGIN
 WITH PlotData, GData DO
 BEGIN
  PltHP:= True;
  ClearMenuStr(GData);
  POutputFile('>>> Plt file name: ',PltFil,PltDirec,PltFileName
        ,MenuX,MenuY,Color,BkColor,MenuStr);
  IF PltFileName=Esc THEN
  BEGIN
   EnExit:= True;
   EnGrafics:= False;
   CloseGraph;
   NewFile:= False;
```

```
      Exit;
     END;
    PltPlotStart(PltFil);
    ClearDevice;
   END;
 END;


 {------------------------------------------------------------------

 PROCEDURE YScaleSet


 ------------------------------------------------------------------------}
 PROCEDURE YScaleSet(VAR PlotData: TMessRec; VAR GData: TGrafRec);
 BEGIN
  WITH PlotData, GData DO
  BEGIN
   ClearMenuStr(GData);
   PReadReal('>>> YScale: ',1E-2,1E2,Yscale,MenuX,MenuY,Color,BkColor,MenuStr);
   NormScale:= False;
   YNorm:= 1.0;
  END;
 END;


 {------------------------------------------------------------------

 PROCEDURE NextFile


 ------------------------------------------------------------------------}
 PROCEDURE NextFile(VAR PlotData: TMessRec; VAR GData: TGrafRec);
 BEGIN
  WITH PlotData, GData DO
  BEGIN
   ClearMenuStr(GData);
   PInputFile('>>> New File: ',DatFil,FileDirec,FileName,MenuX,MenuY,Color,BkColor,MenuStr);
   IF FileName=Esc THEN
   BEGIN
    EnExit:= True;
    EnGrafics:= False;
    CloseGraph;
    NewFile:= False;
    Exit;
   END;
   EndTrue:= True;
```

```
     NewFile:= True;
   END;
 END;


{------------------------------------------------------------------

 PROCEDURE DataPlot


 ------------------------------------------------------------------}
PROCEDURE DataPlot(VAR PlotData: TMessRec; VAR GData: TGrafRec);
VAR
 i,
 XPix0,
 YPix0,
 XPixP,
 YPixP,
 XPlt0,
 YPlt0,
 XPltP,
 YPltP : Integer;
 DelX  : Real;
BEGIN
 WITH PlotData, GData DO
 BEGIN
  DifX:= XVal[IntNum]-XVal[0];
  DelX:= (XVal[IntNum]-XVal[0])/IntNum;
  XPixFac:= XPix/(1.0*DifX);
  XPltFac:= XPlt/(1.0*DifX);

  XScaling(PlotData,GData);
  YScaling(PlotData,GData);

  XPix0:= 0;
  YPix0:= YPix-Round(-YNeg*YPixFac*YScale);
  XPlt0:= 0;
  IF (-YNeg*YPltFac*YScale)<YPlt THEN
  BEGIN
   YPlt0:= Round(-YNeg*YPltFac*YScale);
  END
  ELSE
  BEGIN
   YPlt0:= YPlt;
```

```
END;
IF YPix0<0 THEN YPix0:= 0;
IF YPlt0<0 THEN YPlt0:= 0;
PMoveToXY(XPix0,YPix0,GData);
IF  PltHP THEN
BEGIN
 PltMoveToXY(XPlt0,YPlt0,PltFil);
END;
FOR i:=0 TO IntNum DO
BEGIN
 IF NormScale THEN
 BEGIN
  IF (ActPlotData[i]*YScale*YNorm)<2.0*YPos THEN
  BEGIN
   YPixP:= YPix-Round((ActPlotData[i]*YScale*YNorm-YNeg)*YPixFac);
       YPltP:= Round((ActPlotData[i]*YScale*YNorm-YNeg)*YPltFac);
  END;
 END
 ELSE
 BEGIN
       IF (ActPlotData[i]*YScale*YNorm)<2.0*YPos THEN
  BEGIN
   YPixP:= YPix-Round((ActPlotData[i]*YScale-YNeg)*YPixFac);
   YPltP:= Round((ActPlotData[i]*YScale-YNeg)*YPltFac);
  END;
 END;
 IF (ActPlotData[i]*YScale*YNorm)>(YPos+0.1) THEN
 BEGIN
  YPixP:= 0;
  YPltP:= YPlt;
 END;
 IF (ActPlotData[i]*YScale*YNorm)<YNeg THEN
 BEGIN
  YPixP:= YPix;
  YPltP:= 0;
 END;
 IF PlotType='Histo' THEN
 BEGIN
  XPixP:= Round((XVal[i]-XVal[0]+0.5*DelX)*XPixFac);
  XPltP:= Round((XVal[i]-XVal[0]+0.5*DelX)*XPltFac);
  IF XPixP>XPix THEN
```

```
        BEGIN
          XPixP:= XPix;
          XPltP:= XPlt;
        END;
            PLineXY(XPix0,YPix0,XPix0,YPixP,GData);
            PLineXY(XPix0,YPixP,XPixP,YPixP,GData);
            IF PltHP THEN
            BEGIN
              PltLineXY(XPlt0,YPlt0,XPlt0,YPltP,PltFil);
              PltLineXY(XPlt0,YPltP,XPltP,YPltP,PltFil);
            END;
        END;
        IF PlotType='Point' THEN
        BEGIN
          XPixP:= Round((XVal[i]-XVal[0])*XPixFac);
          XPltP:= Round((XVal[i]-XVal[0])*XPltFac);
            PCircleXY(XPixP,YPixP,1,GData);
            IF PltHP THEN
            BEGIN
              PltCircleXY(XPltP,YPltP,1,PltFil);
            END;
        END;
        XPix0:= XPixP;
        YPix0:= YPixP;
        XPlt0:= XPltP;
        YPlt0:= YPltP;
      END;
    END;
END;


{$I P_GrafIn}    {file: P_GrafIn.Pas}


{-----------------------------------------------------------------
  PROCEDURE StartGraphics


 -----------------------------------------------------------------}
PROCEDURE StartGraphics(VAR GData: TGrafRec);
BEGIN
  WITH GData DO
  BEGIN
    Gd:= Detect;
```

```
    GrafInit(GData);
    InitGraph(Gd,Gm,GDirec);
    EnGrafics:= True;
    ErrCode:= GraphResult;
    IF ErrCode <> grOK THEN
    BEGIN
      Writeln('Graphics error: ', GraphErrorMsg(ErrCode));
    END;
  END;
END;


{---------------------------------------------------------------

 PROCEDURE StopGraphics


 ----------------------------------------------------------------}
PROCEDURE StopGraphics(VAR GData: TGrafRec);
BEGIN
 WITH GData DO
 BEGIN
   CloseGraph;
 END;
END;


{---------------------------------------------------------------

 PROCEDURE PlotLoop


 ----------------------------------------------------------------}
PROCEDURE PlotLoop(VAR PlotData: TMessRec; VAR GData: TGrafRec);
VAR
 Ch  : Char;
BEGIN
 WITH PlotData,GData DO
 BEGIN
   ExitSave:= ExitProc;
   ExitProc:= @RunTimeError;
   PltHP:= False;
   YNorm:= 1.0;
   GlattNum:= 1;
   NoiseLevel:= 0;


   {
```

```
            Gd:= Detect;
            GrafInit(GData);
            InitGraph(Gd,Gm,GDirec);
            EnGrafics:= True;
            ErrCode:= GraphResult;
            IF ErrCode = grOK THEN
}
        BEGIN
          WHILE NOT EndTrue DO
          BEGIN
            ClearDevice;
                SetColor(Color);
                SetBkColor(BkColor);
                GetXYValMax(PlotData);
                IF NormScale THEN
                BEGIN
                  YNorm:= YPos/YValMax;
                  YNormSav:= YNorm;
                END;
                YPixFac:= YPix/(1.0*(YTop+YPos-YNeg));
          YPltFac:= YPlt/(1.0*(YTop+YPos-YNeg));
                PlotFrame(PlotData, GData);
                DataPlot(PlotData, GData);

            IF EnHrdCpy THEN
            BEGIN
              MenuStr:= '(S)moothing/(H)ardcopy/(Y)scale/H(P)GLFile/(N)ext'
                    +'/N(o)ise/(E)xit: ';
              PMenuStr(GData);
                  REPEAT
                  Ch:= ReadKey;
                  UNTIL Ch IN ['s','S','h','H','y','Y','e','E','n','N','p','P'
                    ,'o','O'];
            END
            ELSE
            BEGIN
              MenuStr:='(S)moothing/(Y)scale/H(P)GLFile/(N)ext/N(o)ise/(E)xit: ';
              PMenuStr(GData);
                  REPEAT
                  Ch:= ReadKey;
                  UNTIL Ch IN ['s','S','y','Y','e','E','n','N','p','P','o','O'];
```

```
END;
    MenuStr:= MenuStr+Ch;
    PMenuStr(GData);

CASE Ch OF
'h','H': BEGIN
    Hardcopy(PlotData);
    END;
's','S': BEGIN
    SmoothData(PlotData,GData);
    END;
'y','Y': BEGIN
    YScaleSet(PlotData,GData);
    END;
'n','N': BEGIN
    NextFile(PlotData,GData);
    END;
'p','P': BEGIN
    HPGLFile(PlotData,GData);
    END;
'o','O': BEGIN
    CutNoise(PlotData,GData);
    END;
'e','E': BEGIN
        NewFile:= False;
    EndTrue:= True;
        IF PltHP THEN
        BEGIN
        PltPlotStop(PltFil);
        PltHP:= False;
        END;
        END;
END; {CASE Ch OF...}
END;

{

CloseGraph;

}

EndTrue:= False;
ExitProc:= ExitSave;
END;
```

```
{
  ELSE
  BEGIN
    Writeln('Graphics error: ', GraphErrorMsg(ErrCode));
  END;
}
 END;
END;


END.    { IMPLEMENTATION of P_Graf }
```

P_MDATA.PAS

```pascal
{
  file     : P_MDATA.PAS
  function : data
  author   : W.Maring
  changes  : 12.02.93
}

{$I P_ComOpt}

UNIT P_MData;
{$IFDEF CPU87}
{$N+}
{$ELSE}
{$N-}
{$ENDIF}

INTERFACE

{$IFDEF Masspec}
TYPE
{$IFDEF CPU87}
  Real        = Single;
{$ENDIF}
TArray       = Array [0..2000] of Real;
TMessRec = RECORD
             LowMass,
             HighMass,
             DifMass,
             DelMass,
             Mass,
             URamp,
             SigAverage,
             Signal,
             Difx,
             YScale,
             XValMax,
             YValMax     : Real;
             XVal,
```

```
            YVal,
        ActPlotData  : TArray;
            IntNum,
        IntRange,
            NumScan,
            NInt,
            NScan       : Integer;
            NormScale,
            NewFile,
            EnExit,
            PltHP       : Boolean;
            Comment     : String[80];
            PltDirec,
        FileDirec,
            ExeDirec,
        PltFileName,
            FileName    : String[60];
            PltFil,
            DatFil      : Text;
            END;
{$ENDIF}


IMPLEMENTATION

END.
```

PAN_SCAN.PAS

```
{
   file    : PAn_Scan.PAS
   function : recording and plotting of angular spectra
   author  : B.Ungerer (MPI Goe), W.Maring
   files   : P_AnData,
                  IO_Check.PAS
   changes : 22.10.88 (BU)
                  09/02/93 (WM)
}

{$DEFINE AngSpec}

PROGRAM PFr_Scan;
{$IFDEF CPU87}
{$N+}
{$ELSE}
{$N-}
{$ENDIF}

{$M 8192,0,65360}        { Leave memory for child process }

USES
 Crt,
 Dos,
 Printer,
 P_AnData,
 P_Cam2,
 P_Graf,
 IO_Check;

TYPE
{$IFDEF CPU87}
 Real       = Single;
{$ENDIF}

VAR
 MessData   : TMessRec;
 DirListVar : String[20];
```

```
Ch        : String[1];
CurDir,
SavDir    : String;
ChMenu1   : Char;
EnExitProg : Boolean;


CONST
 Esc      = Char(27);



{$I P_AngIn}   {file: P_AngIn.Pas}


{------------------------------------------------------------------}
                { Main }
BEGIN

 WITH MessData, GData DO
 BEGIN
  InitData(MessData);
  REPEAT
   ClrScr;
   Writeln;
   Writeln('P A n _ S c a n ');
   Writeln('_____');
   Writeln;
   Writeln('  1 --> record angular spectra');
   Writeln('  2 --> plot angulur spectra');
   Writeln('  3 --> c:\sputter\angdat listing');
   Writeln;
   Writeln('  E --> End');
   Writeln;
   Write('>>> Enter menu number : ');
   REPEAT
        ChMenu1:= ReadKey;
   UNTIL ChMenu1 IN [Esc,'1'..'3','e','E'];
   IF ChMenu1=Esc THEN Exit;
   Writeln(ChMenu1);
   REPEAT
        CASE ChMenu1 OF
        '1': BEGIN
            GetDir(0,CurDir);
```

```
        SavDir:= CurDir;
        {$I-}
        ChDir('c:\basicdir\new');
        IF IOResult<>0 THEN
        BEGIN
                Writeln('Cannot find c:\basicdir\new');
        END;
        {$I+}
        SwapVectors;
        Exec(GetEnv('COMSPEC'),'/C '+
                'c:\basicdir\new\hbasic angscan.bas');
        SwapVectors;
        IF DosError <> 0 THEN
        BEGIN
                Writeln('DOSError: ',DosError);
        END;
   ChDir(SavDir);
END;
    '2': BEGIN
        SwapVectors;
        Exec(GetEnv('COMSPEC'),'/C '+ExeDirec+'pangplot.exe');
        SwapVectors;
   IF DosError <> 0 THEN
   BEGIN
                Writeln('DOSError: ',DosError);
   END;
        END;
    '3': BEGIN
        Writeln;
        ReadStr('DirListVar: ',DirListVar);
        SwapVectors;
        Exec(GetEnv('COMSPEC'),'/C '+'dir/w/p '+FileDirec+DirListVar);
        SwapVectors;
   IF DosError <> 0 THEN
   BEGIN
                Writeln('DOSError: ',DosError);
   END;
        Writeln;
        Write('>>> Press any key to continue! ');
        Ch:= ReadKey;
        END;
```

```
                END; {CASE ChMenu1 OF...}
        UNTIL ChMenu1 IN ['1'..'3','E','e'];
        UNTIL ChMenu1 IN ['E','e'];


    END;


    END.
```

PANGPLOT.PAS

```
{
  file     : PANGPLOT.PAS
  function : plotting of angular spectra
  author   : W.Maring
  files    : --
  changes  : 05-07-93
}

{$I P_ComOpt}

PROGRAM PAngPlot;
{$IFDEF CPU87}
{$N+}
{$ELSE}
{$N-}
{$ENDIF}

USES
  Crt,
  Dos,
  Graph,
  Printer,
  P_AnData,
  P_AngInp,
  P_Graf,
  IO_Check;

TYPE
{$IFDEF CPU87}
  Real       = Single;
{$ENDIF}

VAR
  PlotData   : TMessRec;
  XValMax,
  YValMax,
  YFac       : Real;
  Gd,
```

```
      Gm,
   ErrCode    : Integer;
   Ch         : String[2];
   EnExitProg : Boolean;


CONST
  Esc       = Char(27);



{$I P_AngIn}   {file: P_AngIn.Pas}


{--------------------------------------------------------------}
              { Main }
BEGIN

   WITH PlotData, GData DO
   BEGIN
    EnExitProg:= False;
    InitData(PlotData);
    ClrScr;
    Writeln;
    Writeln('P A n g P l o t');
    Writeln('_____');
    Writeln;
    InputFile('file name: ',DatFil,FileDirec,FileName);
    IF FileName=Esc THEN
    BEGIN
     EnExitProg:= True;
     Exit;
    END;
    StartGraphics(GData);
    WHILE NewFile DO
    BEGIN
     InitData(PlotData);
     Input(PlotData);
     IF EnExitProg THEN Exit;
     GData.PlotType:= 'Histo';
     PlotLoop(PlotData,GData);
     IF NOT NewFile THEN
     BEGIN
      Close(DatFil);
```

```
    END;
   END;
  StopGraphics(GData);
 END;

END
```

PAUGER.PAS

```pascal
{
  file     : PAUGER.PAS
  function : fast recording of auger spectra
  author   : W.Maring
  files    : PCAM2.PAS, IO_CHECK.PAS
  changes  : 22.10.88 (B.Ungerer)
                 09/02/93 (W.Maring)
}

{$I P_ComOpt}


PROGRAM PAuger;
{$IFDEF CPU87}
{$N+}
{$ELSE}
{$N-}          .
{$ENDIF}


{$M 8192,0,65360}          { Leave memory for child process }


USES
 Crt,
 Dos,
 Printer,
 P_AugInp,
 P_AData,
 P_Cam2,
 P_Graf,
 IO_Check;


TYPE
{$IFDEF CPU87}
 Real       = Single;
{$ENDIF}


VAR
 MessData   : TMessRec;
 LoLim,
```

```
        UpLim,

        SampleSpeed,

        ScanSpeed,

        ScaleRange,

        SigAverageX,

        SigAverageY,

        SignalX,

        SignalY    : Real;

        Ch        : String[1];

        DirListVar  : String[20];

        ChMenu1    : Char;

        EnExitProg  : Boolean;


CONST

    Esc      = Char(27);


{$I P_AugIn}    { file: P_AugIn.Pas }


{---------------------------------------------------------------

PROCEDURE ScanInput


----------------------------------------------------------------}

PROCEDURE ScanInput;

VAR

    Ch1     : Char;

    InputOkay : Boolean;

BEGIN

    WITH MessData DO

    BEGIN

      InputOkay:= False;

      WHILE NOT InputOkay DO

      BEGIN

        OutputFile('file name: ',DatFil,FileDirec,FileName);

        IF FileName=Char(27) THEN

        BEGIN

              EnExitProg:= True;

              Exit;

      END;

      ReadReal('Lower Limit: ',-2000,200,LoLim);

      ReadReal('Upper Limit: ',-2000,LoLim,UpLim);

      ReadReal('Scan speed [sec]: ',0,1000,ScanSpeed);
```

```
ReadReal('Scale eV/Div: ',0,200,ScaleRange);

ReadReal('Sample speed [sec]: ',0,1000,SampleSpeed);

IntNum:= Round(Round(Abs(UpLim-LoLim)/ScanSpeed)/SampleSpeed);

ReadInt('# of Sample Points: ',0,1000,NumScan);

ReadStr('Comment: ',Comment);

Writeln;

Write('>>> Input okay [Y/N]: ');

REPEAT

        Ch1:= ReadKey;

UNTIL Ch1 IN [Esc,'n','N','y','Y'];

IF Ch1=Esc THEN

BEGIN

        EnExitProg:= True;

        Exit;

END;

Writeln(Ch1);

IF (Ch1='y') OR (Ch1='Y') THEN

BEGIN

        InputOkay:= True;

END;

Writeln;

END;

IF EnExitProg THEN Exit;

Writeln;

Writeln;

Write('>>> Press any key to START and then START on Auger Controler ! ');

ReadStr('',Ch);


PrintFileHeader(DatFil,FileName);

Writeln(DatFil,LoLim:5:1,' ',UpLim:5:1,' ',ScanSpeed:5:1,' ',

                    ScaleRange:5:1,' ',SampleSpeed:8:3,' ',NumScan:4);

Writeln(DatFil,Comment);

END;

END;


{----------------------------------------------------------------

PROCEDURE DataLoop


----------------------------------------------------------------}

PROCEDURE DataLoop;

VAR
```

```pascal
      NInt,
      NScan  : Integer;
BEGIN
 WITH MessData DO
 BEGIN
  Init_TimSca;
  Writeln;
  Writeln('DataLoop: START');

  FOR NInt:=0 TO IntNum DO
  BEGIN
   IF NotFirstScan THEN
   BEGIN
        Reset_TimSca;
        Start_TimSca(SampleSpeed);
   END;
   SigAverageX:=0.0;
   SigAverageY:=0.0;
   FOR NScan:=1 TO NumScan DO
   BEGIN
        Get_ADC2(adc_c2,SignalX);
        SigAverageX:= SigAverageX + SignalX;
        Get_ADC1(adc_c1,SignalY);
        SigAverageY:= SigAverageY + SignalY;
   END;      { FOR NScan= ... }
   SignalX:= SigAverageX/(1.0*NumScan);
   XVal[NInt]:= SignalX;
   SignalY:= SigAverageY/(1.0*NumScan);
   YVal[NInt]:= SignalY;
   ActPlotData[NInt]:= YVal[NInt];
{
   Writeln('NInt= ',NInt:4,' X= ',XVal[NInt]:10:2,
             ' Y= ',YVal[NInt]:10:2);
}
   NotFirstScan:= True;
  END;      { FOR NInt ... }

  Writeln('DataLoop: STOP');
 END;
END;
```

```
{-----------------------------------------------------------------
PROCEDURE WriteData


------------------------------------------------------------------}
PROCEDURE WriteData;
VAR
 NInt   : Integer;
BEGIN
 WITH MessData DO
 BEGIN
  Writeln(DatFil,IntNum);
  FOR NInt:=0 TO IntNum DO
  BEGIN
   Writeln(DatFil,XVal[NInt]:10:4,'   ',YVal[NInt]:10:4);
  END;
  Close(DatFil);
 END;
END;



{-----------------------------------------------------------------
PROCEDURE MainLoop


------------------------------------------------------------------}
PROCEDURE MainLoop;
VAR
 Ch1    : Char;
BEGIN
 WITH MessData, GData DO
 BEGIN
  EnExitProg:= False;
  ClrScr;
{***   Init_Crate;}
  WHILE NOT EndTrue DO
  BEGIN
   InitData(MessData);
   ScanInput;
   IF EnExitProg THEN
   BEGIN
        IF NOT (FileName=Esc) THEN
        BEGIN
```

```pascal
              Close(DatFil);
              Writeln;
              Writeln;
              Write('>>> empty ',FileName,' will be deleted [Y/N]: ');
              REPEAT
                Ch1:= ReadKey;
              UNTIL Ch1 IN ['n','N','y','Y'];
              Writeln(Ch1);
              IF (Ch1='y') OR (Ch1='Y') THEN
              BEGIN
                Erase(DatFil);    { file will be deleted if existing by Exit! }
              END;
              Writeln;
            END;
            Exit;
       END;
       DataLoop;
       WriteData;
       StartGraphics(GData);
       GData.PlotType:= 'Point';
       PlotLoop(MessData,GData);
       WHILE NewFile DO
       BEGIN
         Input(MessData);
         PlotLoop(MessData,GData);
       END;
       StopGraphics(GData);
       EndTrue:= True;
      END;
     END;
    END;


{----------------------------------------------------------------}


BEGIN                        { Main }

  WITH MessData, GData DO
  BEGIN
   InitData(MessData);
   REPEAT
    ClrScr;
```

```pascal
Writeln;
Writeln('P A u g e r ');
Writeln('_____');
Writeln;
Writeln('  1 --> record auger spectra');
Writeln('  2 --> plot auger spectra');
Writeln('  3 --> c:\sputter\augdat listing');
Writeln;
Writeln('  E --> End');
Writeln;
Write('>>> Enter menu number : ');
REPEAT
      ChMenu1:= ReadKey;
UNTIL ChMenu1 IN [Esc,'1'..'3','e','E'];
IF ChMenu1=Esc THEN Exit;
Writeln(ChMenu1);
REPEAT
    CASE ChMenu1 OF
    '1': BEGIN
    EndTrue:= False;
        MainLoop;
       END;
    '2': BEGIN
        SwapVectors;
        Exec(GetEnv('COMSPEC'),'/C'+ExeDirec+'paugplot.exe');
        SwapVectors;
    IF DosError <> 0 THEN
    BEGIN
            Writeln('DOSError: ',DosError);
    END;
       END;
    '3': BEGIN
        Writeln;
        ReadStr('DirListVar: ',DirListVar);
        SwapVectors;
        Exec(GetEnv('COMSPEC'),'/C'+'dir/w/p '+FileDirec+DirListVar);
        SwapVectors;
    IF DosError <> 0 THEN
    BEGIN
            Writeln('DOSError: ',DosError);
    END;
```

```
            Writeln;

            Write('>>> Press any key to continue! ');

            Ch:= ReadKey;

         END;

      END; {CASE ChMenu1 OF...}

   UNTIL ChMenu1 IN ['1'..'3','E','e'];

  UNTIL ChMenu1 IN ['E','e'];

 END;


END.
```

PAUGPLOT.PAS

```pascal
{
   file     : PAUGPLOT.PAS
   function : plotting of mass spectra
   author   : W.Maring
   files    : --
   changes  : 09/02/93
}

{$DEFINE Augspec}

PROGRAM PAugPlot;
{$IFDEF CPU87}
{$N+}
{$ELSE}
{$N-}
{$ENDIF}

USES
 Crt,
 Dos,
 Graph,
 Printer,
 P_AData,
 P_AugInp,
 P_Graf,
 IO_Check;

TYPE
{$IFDEF CPU87}
 Real       = Single;
{$ENDIF}

VAR
 PlotData   : TMessRec;
 XValMax,
 YValMax,
 YFac       : Real;
 Gd,
```

```
 Gm,
 ErrCode    : Integer;
 Ch         : String[2];
 EnExitProg : Boolean;


CONST
 Esc       = Chr(27);



{$I P_AugIn}   { file: P_AugIn.Pas }


{-----------------------------------------------------------------}
                    { Main }
BEGIN

 WITH PlotData, GData DO
 BEGIN
  EnExitProg:= False;
  InitData(PlotData);
  ClrScr;
  Writeln;
  Writeln('P A u g P l o t');
  Writeln('_____');
  Writeln;
  InputFile('file name: ',DatFil,FileDirec,FileName);
  IF FileName=Esc THEN
  BEGIN
   EnExitProg:= True;
   Exit;
  END;
  StartGraphics(GData);
  WHILE NewFile DO
  BEGIN
   InitData(PlotData);
   Input(PlotData);
   IF EnExitProg THEN Exit;
   GData.PlotType:= 'Point';
   PlotLoop(PlotData,GData);
   IF NOT NewFile THEN
   BEGIN
    Close(DatFil);
```

```
    END;
   END;
  StopGraphics(GData);
 END;

END.
```

PCAMINIT.PAS

```pascal
{
   file     : PCAMINIT.PAS
   function : initilization of CAMAC crate
   author   : W.Maring
   files    : P_Cam2.PAS,
                    IO_Check.PAS
   changes  : 06-04-93
}

PROGRAM PCamInit;
{$IFDEF CPU87}
{$N+}
{$ELSE}
{$N-}
{$ENDIF}


USES
 Crt,
 Dos,
 Printer,
 P_Cam2,
 IO_Check;


VAR
 Ch1   : Char;


TYPE
{$IFDEF CPU87}
 Real         = Single;
{$ENDIF}



BEGIN
 Write('>>> Reset and init CAMAC crate [Y/N]: ');
 REPEAT
  Ch1:= ReadKey;
 UNTIL Ch1 IN [Esc,'n','N','y','Y'];
 IF Ch1=Esc THEN Exit;
```

```
Writeln(Ch1);
IF (Ch1='y') OR (Ch1='Y') THEN
BEGIN
 Init_Crate;
END;
END.
```

PFR_SCAN.PAS

```pascal
{
  file      : PFR_SCAN.PAS
  function  : recording of frequency spectra
  author    : B.Ungerer (MPI Goe), W.Maring
  files     : P_FData,
                  IO_Check.PAS
  changes   : 22.10.88 (BU)
                  09/02/93 (WM)
}


{$DEFINE FreqSpec}

PROGRAM PFr_Scan;
{$IFDEF CPU87}
{$N+}
{$ELSE}
{$N-}
{$ENDIF}


{$M 8192,0,65360}        { Leave memory for child process }

USES
  Crt,
  Dos,
  Printer,
  P_FData,
  P_Cam2,
  P_Graf,
  IO_Check;

TYPE
{$IFDEF CPU87}
  Real        = Single;
{$ENDIF}

VAR
  MessData    : TMessRec;
  DirListVar  : String[20];
```

```pascal
  Ch        : String[1];
  CurDir,
  SavDir    : String;
  ChMenu1   : Char;
  EnExitProg : Boolean;


CONST
  Esc       = Char(27);



{$I P_FreqIn}    {file: P_FreqIn.Pas}


{----------------------------------------------------------------}
                { Main }
BEGIN

  WITH MessData, GData DO
  BEGIN
    InitData(MessData);
    REPEAT
      ClrScr;
      Writeln;
      Writeln('P F r _ S c a n');
      Writeln('_____');
      Writeln;
      Writeln('  1 --> record frequency spectra');
      Writeln('  2 --> plot frequency spectra');
      Writeln('  3 --> c:\sputter\freqdat listing');
      Writeln;
      Writeln('  E  --> End');
      Writeln;
      Write('>>> Enter menu number : ');
      REPEAT
          ChMenu1:= ReadKey;
      UNTIL ChMenu1 IN [Esc,'1'..'3','e','E'];
      IF ChMenu1=Esc THEN Exit;
      Writeln(ChMenu1);
      REPEAT
          CASE ChMenu1 OF
          '1': BEGIN
              GetDir(0,CurDir);
```

```
            SavDir:= CurDir;
            {$I-}
            ChDir('c:\basicdir\new');
            IF IOResult<>0 THEN
            BEGIN
                    Writeln('Cannot find c:\basicdir\new');
            END;
            {$I+}
            SwapVectors;
            Exec(GetEnv('COMSPEC'),'/C '+
                    'c:\basicdir\new\hbasic freqscan.bas');
            SwapVectors;
            IF DosError <> 0 THEN
            BEGIN
                    Writeln('DOSError: ',DosError);
            END;
        ChDir(SavDir);
    END;
      '2': BEGIN
            SwapVectors;
            Exec(GetEnv('COMSPEC'),'/C '+ExeDirec+'pfreplot.exe');
            SwapVectors;
    IF DosError <> 0 THEN
    BEGIN
                    Writeln('DOSError: ',DosError);
    END;
            END;
      '3': BEGIN
            Writeln;
            ReadStr('DirListVar: ',DirListVar);
            SwapVectors;
            Exec(GetEnv('COMSPEC'),'/C '+'dir/w/p '+FileDirec+DirListVar);
            SwapVectors;
    IF DosError <> 0 THEN
    BEGIN
                    Writeln('DOSError: ',DosError);
    END;
            Writeln;
            Write('>>> Press any key to continue! ');
            Ch:= ReadKey;
            END;
```

```
        END; {CASE ChMenu1 OF...}
   UNTIL ChMenu1 IN ['1'..'3','E','e'];
   UNTIL ChMenu1 IN ['E','e'];

 END;

 END.
```

PFREPLOT.PAS

```
{
   file     : PFREPLOT.PAS
   function : plotting of angular spectra
   author   : W.Maring
   files    : --
   changes  : 05-07-93
}

{$I P_ComOpt}

PROGRAM PFrePlot;
{$IFDEF CPU87}
{$N+}
{$ELSE}
{$N-}
{$ENDIF}  .

USES
  Crt,
  Dos,
  Graph,
  Printer,
  P_FData,
  P_FInput,
  P_Graf,
  IO_Check;

TYPE
{$IFDEF CPU87}
  Real       = Single;
{$ENDIF}

VAR
  PlotData   : TMessRec;
  XValMax,
  YValMax,
  YFac       : Real;
  Gd,
```

```
        Gm,
        ErrCode    : Integer;
        Ch         : String[2];
        EnExitProg : Boolean;


CONST
    Esc        = Char(27);



{$I P_FreqIn}    {file: P_FreqIn.Pas}


{------------------------------------------------------------------}
                    { Main }
BEGIN

    WITH PlotData, GData DO
    BEGIN
      EnExitProg:= False;
      ClrScr;
      InitData(PlotData);
      Writeln;
      Writeln('P F r e P l o t');
      Writeln('_____');
      Writeln;
      InputFile('file name: ',DatFil,FileDirec,FileName);
      StartGraphics(GData);
      WHILE NewFile DO
      BEGIN
        InitData(PlotData);
        Input(PlotData);
        IF EnExitProg THEN Exit;
        GData.PlotType:= 'Histo';
        PlotLoop(PlotData,GData);
        IF NOT NewFile THEN
        BEGIN
          Close(DatFil);
        END;
      END;
      StopGraphics(GData);
    END;
```

END.

PMASPLOT.PAS

```
{
   file     : PMASPLOT.PAS
   function : plotting of mass spectra
   author   : W.Maring
   files    : --
   changes  : 09/02/93
}

{$I P_ComOpt}

PROGRAM PMasPlot;
{$IFDEF CPU87}
{$N+}
{$ELSE}
{$N-}
{$ENDIF} .

USES
  Crt,
  Dos,
  Graph,
  Printer,
  P_MData,
  P_MInput,
  P_Graf,
  IO_Check;

TYPE
{$IFDEF CPU87}
  Real       = Single;
{$ENDIF}

VAR
  PlotData   : TMessRec;
  XValMax,
  YValMax,
  YFac       : Real;
  Gd,
```

```
      Gm,
      ErrCode    : Integer;
      Ch         : String[2];
      EnExitProg : Boolean;


CONST
  Esc      = Char(27);



{$I P_MasIn}    {file: P_MasIn.Pas}


{-------------------------------------------------------------------}
                    { Main }
BEGIN

  WITH PlotData, GData DO
  BEGIN
   EnExitProg:= False;
   InitData(PlotData);
   ClrScr;
   Writeln;
   Writeln('P M a s P l o t');
   Writeln('_____');
   Writeln;
   InputFile('file name: ',DatFil,FileDirec,FileName);
   IF FileName=Esc THEN
   BEGIN
    EnExitProg:= True;
    Exit;
   END;
   StartGraphics(GData);
   WHILE NewFile DO
   BEGIN
    InitData(PlotData);
    Input(PlotData);
    IF EnExitProg THEN Exit;
    GData.PlotType:= 'Histo';
    PlotLoop(PlotData,GData);
    IF NOT NewFile THEN
    BEGIN
     Close(DatFil);
```

```
    END;
   END;
   StopGraphics(GData);
   END;

END.
```

PMASS.PAS

```
{
  file     : PMASS.PAS
  function : fast recording of mass spectra
  author   : B.Ungerer (MPI Goe), W.Maring
  files    : P_Cam2.PAS,
                  P_Graf,
                  P_MData,
                  IO_Check.PAS
  changes  : 22.10.88 (BU)
                  09/02/93 (WM)
}


{$DEFINE Masspec}

PROGRAM PMass;
{$IFDEF CPU87}
{$N+}
{$ELSE}
{$N-}
{$ENDIF}


{$M 8192,0,65360}        { Leave memory for child process }

USES
  Crt,
  Dos,
  Printer,
  P_MInput,
  P_MData,
  P_Cam2,
  P_Graf,
  IO_Check;

TYPE
{$IFDEF CPU87}
  Real       = Single;
{$ENDIF}
```

```pascal
VAR
  MessData    : TMessRec;
  DirListVar  : String[20];
  Ch          : String[1];
  ChMenu1     : Char;
  EnExitProg  : Boolean;


CONST
  UMassFac    : Real = 0.033333;   { Voltage/mass conversion }
  MassCorr    : Real = 1.0;        { Correction for mass value }
  Esc         = Char(27);



{$I P_MasIn}   {file: P_MasIn.Pas}


{--------------------------------------------------------------------

PROCEDURE ScanInput


----------------------------------------------------------------}

PROCEDURE ScanInput;
VAR
  Ch1       : Char;
  InputOkay : Boolean;
BEGIN
  WITH MessData DO
  BEGIN
  InputOkay:= False;
  WHILE NOT InputOkay DO
   BEGIN
    OutputFile('file name: ',DatFil,FileDirec,FileName);
    IF FileName=Esc THEN
    BEGIN
         EnExitProg:= True;
         Exit;
    END;
    ReadReal('LowMass: ',0,300,LowMass);
    ReadReal('HighMass: ',LowMass,300,HighMass);
    DifMass:= HighMass-LowMass;
    ReadInt('# of Intervals: ',0,1000,IntNum);
    DelMass:= DifMass/(1.0*IntNum);
    ReadInt('# of Scans per Mass: ',0,1000,NumScan);
```

```
        ReadInt('range 10^-x: ',5,12,IntRange);
        ReadStr('Comment: ',Comment);
        Writeln;
        Write('>>> Input okay [Y/N]: ');
        REPEAT
              Ch1:= ReadKey;
        UNTIL Ch1 IN [Esc,'n','N','y','Y'];
        IF Ch1=Esc THEN
        BEGIN
              EnExitProg:= True;
              Exit;
        END;
        Writeln(Ch1);
        IF (Ch1='y') OR (Ch1='Y') THEN
        BEGIN
              InputOkay:= True;
        END;
        Writeln;
      END;
      IF EnExitProg THEN Exit;
      Writeln;
      Writeln;
      Write('>>> Press any key to START! ');
      ReadStr('',Ch);

      PrintFileHeader(DatFil,FileName);
      Writeln(DatFil,LowMass:5:1,' ',HighMass:5:1,' ',NumScan:4,' ',
                     '10^-',IntRange:2);
      Writeln(DatFil,Comment);
      Writeln(DatFil,IntNum);
    END;
END;


{--------------------------------------------------------------------
PROCEDURE DataLoop

--------------------------------------------------------------------}
PROCEDURE DataLoop;
BEGIN
  WITH MessData DO
  BEGIN
```

```
    Writeln;
    Writeln('DataLoop: START');
    FOR NInt:=0 TO IntNum DO
    BEGIN
     Mass:= LowMass+NInt*DelMass;
     XVal[NInt]:= Mass;
     URamp:= (Mass-MassCorr)*UMassFac;
     Set_DAC1(dac_c3,URamp);     {** Set_DAC1 **}
     SigAverage:= 0.0;

     FOR NScan:=1 TO NumScan DO
     BEGIN
            Get_ADC2(adc_c2,Signal);
            SigAverage:= SigAverage + Signal;
            Signal:= SigAverage/(1.0*NumScan);
            YVal[NInt]:= Signal;
       ActPlotData[NInt]:= YVal[NInt];
     END;        {FOR NScan= ...}
    END;     {FOR NInt= ...}
    Writeln('DataLoop: STOP');
   END;
END;


{-------------------------------------------------------------------
PROCEDURE WriteData


---------------------------------------------------------------------}

PROCEDURE WriteData;
BEGIN
 WITH MessData DO
 BEGIN
  FOR NInt:=0 TO IntNum DO
  BEGIN
   Writeln(DatFil,XVal[NInt]:10:4,'   ',YVal[NInt]);
  END;
  Close(DatFil);
 END;
END;


{-------------------------------------------------------------------
PROCEDURE MainLoop
```

```
-------------------------------------------------------------------------}
PROCEDURE MainLoop;
VAR
  Ch1   : Char;
BEGIN
  WITH MessData, GData DO
  BEGIN
    EnExitProg:= False;
    ClrScr;
{***    Init_Crate;}
    WHILE NOT EndTrue DO
    BEGIN
      InitData(MessData);
      ScanInput;
      IF EnExitProg THEN
      BEGIN
            IF NOT (FileName=Esc) THEN
            BEGIN
              Close(DatFil);
              Writeln;
              Writeln;
              Write('>>> empty ',FileName,' will be deleted [Y/N]: ');
              REPEAT
                Ch1:= ReadKey;
              UNTIL Ch1 IN ['n','N','y','Y'];
              Writeln(Ch1);
              IF (Ch1='y') OR (Ch1='Y') THEN
              BEGIN
                Erase(DatFil);    { file will be deleted if existing by Exit! }
              END;
              Writeln;
            END;
        Exit;
      END;
      DataLoop;
      WriteData;
      StartGraphics(GData);
      GData.PlotType:= 'Histo';
      PlotLoop(MessData,GData);
      WHILE NewFile DO
```

```
        BEGIN
          Input(MessData);
          PlotLoop(MessData,GData);
        END;
        StopGraphics(GData);
        EndTrue:= True;
      END;
    END;
  END;


{-------------------------------------------------------------------}
                { Main }
BEGIN

  WITH MessData, GData DO
  BEGIN
    InitData(MessData);
    REPEAT
      ClrScr;
      Writeln;
      Writeln('P M a s s ');
      Writeln('_____');
      Writeln;
      Writeln('   1 --> record mass spectra');
      Writeln('   2 --> plot mass spectra');
      Writeln('   3 --> c:\sputter\masdat listing');
      Writeln;
      Writeln('   E --> End');
      Writeln;
      Write('>>> Enter menu number : ');
      REPEAT
            ChMenu1:= ReadKey;
      UNTIL ChMenu1 IN [Esc,'1'..'3','e','E'];
      IF ChMenu1=Esc THEN Exit;
      Writeln(ChMenu1);
      REPEAT
            CASE ChMenu1 OF
            '1': BEGIN
          EndTrue:= False;
                MainLoop;
```

```
                  END;
             '2': BEGIN
                  SwapVectors;
                  Exec(GetEnv('COMSPEC'),'/C '+ExeDirec+'pmasplot.exe');
                  SwapVectors;
           IF DosError <> 0 THEN
           BEGIN
                      Writeln('DOSError: ',DosError);
           END;
                  END;
             '3': BEGIN
                  Writeln;
                  ReadStr('DirListVar: ',DirListVar);
                  SwapVectors;
                  Exec(GetEnv('COMSPEC'),'/C '+'dir/w/p '+FileDirec+DirListVar);
                  SwapVectors;
           IF DosError <> 0 THEN
           BEGIN
                      Writeln('DOSError: ',DosError);
           END;
                  Writeln;
                  Write('>>> Press any key to continue! ');
                  Ch:= ReadKey;
                  END;
                  END; {CASE ChMenu1 OF...}
         UNTIL ChMenu1 IN ['1'..'3','E','e'];
         UNTIL ChMenu1 IN ['E','e'];


     END;


     END.
```

PMESPROG.PAS

```pascal
{
    file     : PMESPROG.PAS
    function : recording mass, auger, angular or frequency spectra
    author   : W.Maring
    files    : PAn_Scan.EXE,
               PAngPlot.EXE,
               PAuger.EXE,
               PAugPlot.EXE,
               PFr_Scan.EXE,
               PFrePlot.EXE,
                         PMass.EXE,
               PMasPlot.EXE
               PCamInit.EXE
    changes  : 21/04/93 (WM)
}


PROGRAM PMesProg;


{$M 8192,0,65360}       { Leave memory for child process }


USES
 Crt,
 Dos,
 Printer,
 IO_Check;


TYPE
 TDatRec = RECORD
         BasicProgFileName,
         DirListVar     : String[20];
         CurDir,
         SavDir,
         ExeDir,
         BasicDir       : String;
         DriveCh        : Byte;
         ChMenu1,
         Ch             : Char;
         ExecBasicProg  : Boolean;
```

```
        END;


VAR
  BasicDirX        : String;
  Data          : TDatRec;


{$I P_MesIn}    {file: PMesIn.Pas}


{----------------------------------------------------------------------}
                { Main }
BEGIN
 WITH DATA DO
 BEGIN
  InitData(Data);
  REPEAT
   ClrScr;
   Writeln;
   Writeln('P M e s P r o g');
   Writeln('_____');
   Writeln;
   Writeln('  1 --> AUGER:      pauger.exe');
   Writeln('  2 --> MASS:       pmass.exe');
   Writeln('  3 --> FREQ:       pfr_scan.exe');
   Writeln('  4 --> ANGULAR:    pan_scan.exe');
   Writeln('  5 --> DAC:        dac_test.exe');
   Writeln('  6 --> BASIC:      run BASIC programs');
   Writeln('  7 --> DOS:        dir/w/p');
   Writeln('  8 --> init CAMAC:  pcaminit.exe');
   Writeln;
   Writeln('  E --> End');
   Writeln;
   Write('>>> Enter menu number : ');
   REPEAT
    ChMenu1:= ReadKey;
   UNTIL ChMenu1 IN ['1'..'8','e','E'];
   Writeln(ChMenu1);
   REPEAT
    CASE ChMenu1 OF
    '1': BEGIN
              SwapVectors;
              Exec(GetEnv('COMSPEC'),'/C '+ExeDir+'pauger.exe');
```

```
        SwapVectors;

        IF DosError <> 0 THEN

        BEGIN

         Writeln('DOSError: ',DosError);

        END;

       END;

  '2': BEGIN

        SwapVectors;

        Exec(GetEnv('COMSPEC'),'/C '+ExeDir+'pmass.exe');

        SwapVectors;

        IF DosError <> 0 THEN

        BEGIN

         Writeln('DOSError: ',DosError);

        END;

       END;

  '3': BEGIN

        SwapVectors;

        Exec(GetEnv('COMSPEC'),'/C '+ExeDir+'pfr_scan.exe');

        SwapVectors;

        IF DosError <> 0 THEN

        BEGIN

         Writeln('DOSError: ',DosError);

        END;

       END;

  '4': BEGIN

        SwapVectors;

        Exec(GetEnv('COMSPEC'),'/C '+ExeDir+'pan_scan.exe');

        SwapVectors;

        IF DosError <> 0 THEN

        BEGIN

         Writeln('DOSError: ',DosError);

        END;

       END;

  '5': BEGIN

        SwapVectors;

        Exec(GetEnv('COMSPEC'),'/C '+ExeDir+'dac_test.exe');

        SwapVectors;

        IF DosError <> 0 THEN

        BEGIN

         Writeln('DOSError: ',DosError);

        END;
```

```
                    END;
'6': BEGIN
            ExecBasicProg:= True;
            ClrScr;
            GetDir(0,CurDir);
            SavDir:= CurDir;
            {$I-}
            BasicDirX:= Copy(BasicDir,1,(Length(BasicDir)-1));
            ChDir(BasicDirX);
            IF IOResult<>0 THEN
            BEGIN
                   Writeln('Cannot find ',BasicDirX);
             Writeln;
             Write('>>> Press any key to continue! ');
             Ch:= ReadKey;
        ChDir(SavDir);
        Exit;
            END;
            {$I+}
            Writeln;
            Writeln('ccount.bas   : counting of the photomultiplier signal');
            Writeln('detpos.pas   : new detector position');
            Writeln('shutter.bas  : test of the laser beam shutter');
            Writeln;
            Writeln;
            Write('BASIC program: ');
            InpChar(BasicProgFileName);
            IF BasicProgFileName=Char(27) THEN
        BEGIN
            ExecBasicProg:= False;
            END;
            IF ExecBasicProg THEN
            BEGIN
             SwapVectors;
                   Exec(GetEnv('COMSPEC'),'/C '+BasicDir+'hbasic '
                       +BasicProgFileName);
             SwapVectors;
             IF DosError <> 0 THEN
             BEGIN
                    Writeln('DOSError: ',DosError);
             END;
```

```
                END;

                  ChDir(SavDir);

                END;

              '7': BEGIN

                  Writeln;

                  GetDir(0,CurDir);

                  Writeln('CurDir: ',CurDir);

                  Writeln;

                  ReadStr('[NewDir\]DirListVar: ',DirListVar);

                  SwapVectors;

                  Exec(GetEnv('COMSPEC'),'/C '+'dir/w/p '+DirListVar);

                  SwapVectors;

              IF DosError <> 0 THEN

              BEGIN

                        Writeln('DOSError: ',DosError);

              END;

                  Writeln;

                  Write('>>> Press any key to continue! ');

                  Ch:= ReadKey;

                END;

          '8': BEGIN

                  SwapVectors;

                  Exec(GetEnv('COMSPEC'),'/C '+ExeDir+'pcaminit.exe');

                  SwapVectors;

                  IF DosError <> 0 THEN

                  BEGIN

                   Writeln('DOSError: ',DosError);

                  END;

                END;

          END; {CASE ChMenu1 OF...}

        UNTIL ChMenu1 IN ['1'..'8','E','e'];

       UNTIL ChMenu1 IN ['E','e'];

      END;


      END.
```

VITA

Paul Robert Schomber was born in Texarkana, Arkansas on May 11, 1961.
He received his B.S. degree in Chemistry from Oklahoma State University in
1983 and was commissioned in the United States Air Force the same year.
The Air Force selected him for advanced graduate training in 1986 and he
received his M.S. degree in Chemistry from the University of Washington in
1988. He was again selected by the Air Force for additional graduate training
in 1991. He returned to the University of Washington in September 1991 and
was awarded a Ph.D. in Physical Chemistry in March 1995.