



Incipient oxidation of Al(111) studied using optical second harmonic generation
by Kejia Wan

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in
Physics

Montana State University

© Copyright by Kejia Wan (1989)

Abstract:

Optical second harmonic generation and complementary surface analytical techniques have been used to study the oxidation of the Al(111) surface. Two aspects of the oxidation properties are investigated. The first is the identification of oxide growth phases. When oxygen adsorbs on the Al(111) surface, oxygen bonds initially in two distinct sites such that both a surface phase and a subsurface phase exist on the surface prior to the formation of Al_2O_3 . Auger electron spectroscopy, contact potential difference work function measurements and second harmonic generation support this conclusion. Additionally, the second harmonic generation and contact potential difference data indicate that two types of bonds exist between the adsorbed oxygen and the substrate aluminum with increasing exposure. At low oxygen coverage, the bond serves to reduce the density of free electrons of the substrate surface. At higher oxygen exposures, charge transfer occurs from the aluminum to the oxygen atoms. This transfer produces a localized, permanent contribution to the surface dipole moment.

INCIPIENT OXIDATION OF AL(111) STUDIED
USING OPTICAL SECOND HARMONIC GENERATION

by
Kejia Wan

A thesis submitted in partial fulfillment
of the requirements for the degree

of
Master of Science

in
Physics

MONTANA STATE UNIVERSITY
Bozeman, Montana

April, 1989

N378
W1812

ii

APPROVAL

of a thesis submitted by

Kejia Wan

This thesis has been read by each member of the thesis committee and has been found to be satisfactory regarding content, English usage, format, citations, bibliographic style, and consistency, and is ready for submission to the College of Graduate Studies.

22 March 90

Date

Wayne Hord

Chairperson, Graduate Committee

Approved for the Major Department

3/22/90

Date

[Signature]

Head, Physics Department

Approved for the College of Graduate Studies

April 4, 1990

Date

Henry L. Parsons

Graduate Dean

STATEMENT OF PERMISSION TO USE

In presenting this thesis in partial fulfillment of the requirements for a master's degree at Montana State University, I agree that the Library shall make it available to borrowers under rules of the Library. Brief quotations from this thesis are allowable without special permission, provided that accurate acknowledgment of source is made.

Permission for extensive quotation from or reproduction of this thesis may be granted by my major professor, or in his/her absence, by the Dean of Libraries when, in the opinion of either, the proposed use of the material is for scholarly purposes. Any copying or use of the material in this thesis for financial gain shall not be allowed without my written permission.

Signature Kesia Wan

Date 3/22/90

TABLE OF CONTENTS

	Page
LIST OF FIGURES	v
ABSTRACT	vii
INTRODUCTION	1
DESCRIPTION OF EXPERIMENTAL TECHNIQUES	3
Apparatus	3
Sample Preparation	12
Auger Electron Spectroscopy	15
Contact Potential Difference Measurement	22
Surface Second Harmonic Generation	31
EXPERIMENTAL RESULTS	38
Auger Electron Spectroscopy	38
Contact Potential Difference Measurements	41
Surface Second Harmonic Generation Measurements	45
DISCUSSION	49
Oxidation of Al (111)	49
Incipient Oxidation Process	56
Summary and Final Considerations	63
REFERENCES CITED	65
APPENDIX	70
Computer Interface	71

LIST OF FIGURES

Figure	Page
1. Schematic description of the ultra high vacuum experimental chamber.....	4
2. Experimental view of the sample holder.....	5
3. Second harmonic generation experimental configuration.....	8
4. Schematic description of beamsizer reducer.....	9
5. Boxcar electronic setup.....	9
6. Auger spectrum of sample before (a) and after (b) cleaning.....	14
7. Schematic of Auger process.....	16
8. Auger cross section as a function of primary beam energy.....	21
9. Effect of the surface on the binding energy of an electron.....	24
10. Schematic of contact potential difference method.....	26
11. Schematic of two capacitor model for CPD measurements.....	28
12. Two capacitor model results for CPD measurements.....	29
13. Measured dependence of CPD with sample-probe distance.....	30
14. Schematic of SHG measurements.....	32
15. O ₂ /Al(111) Auger uptake.....	39
16. Auger oxygen uptake.....	40
17. Kelvin probe result.....	42

LIST OF FIGURES (Continued)

18. Stop oxygen exposure the work function response.	44
19. SHG results p-polarized incident.	46
20. SHG results s-polarized incident.	47
21. SHG results for sputtering surface.	48
22. Finite well model for the surface electrons, work function (a) ΦI and (b) ΦII	62
23. Diagram of computer programs.	73
24. Software listing.	74

ABSTRACT

Optical second harmonic generation and complementary surface analytical techniques have been used to study the oxidation of the Al(111) surface. Two aspects of the oxidation properties are investigated. The first is the identification of oxide growth phases. When oxygen adsorbs on the Al(111) surface, oxygen bonds initially in two distinct sites such that both a surface phase and a subsurface phase exist on the surface prior to the formation of Al_2O_3 . Auger electron spectroscopy, contact potential difference work function measurements and second harmonic generation support this conclusion. Additionally, the second harmonic generation and contact potential difference data indicate that two types of bonds exist between the adsorbed oxygen and the substrate aluminum with increasing exposure. At low oxygen coverage, the bond serves to reduce the density of free electrons of the substrate surface. At higher oxygen exposures, charge transfer occurs from the aluminum to the oxygen atoms. This transfer produces a localized, permanent contribution to the surface dipole moment.

INTRODUCTION

The oxidation of aluminum is of fundamental importance to both basic scientific research and industrial applications. Alumina is the fully oxidized form of aluminum and is widely used as support material in industrially invaluable bimetallic catalysts. Hence, the commercial importance of alumina provides a practical stimulation for basic studies of aluminum oxidation. One goal of basic research in this area is the microscopic description of the formation and chemical activity of the oxidized surface. In this manner, properties of the surface such as structure and composition might be related to specific catalytic properties.¹ Ultimately, it is envisioned that catalysts with desired activities and specificities can be designed and manufactured.

Microscopic studies of the oxidation process have utilized a wide variety of surface sensitive techniques. These include: low energy electron diffraction (LEED), photoemission experiments, contact potential difference work function measurements (CPD), Auger electron spectroscopy (AES) and electron energy loss spectroscopy. Batra and Kleinman have published a comprehensive review of this literature.² Nevertheless, the oxidation of aluminum still presents many controversial issues. The most studied surface is Al(111). Different authors have drawn different conclusions regarding the oxidation mechanism for the (111) surface. Some claim that surface and subsurface phases occur in a distinct sequence, while others have concluded that the two phases always coexist.²

This thesis proposes a new description of the incipient oxidation of Al(111) using the experimental techniques of AES, CPD, and a recently developed technique called surface enhanced second harmonic generation (SHG).³ The SHG technique specifically emphasizes the changes in electronic structure and the concomitant changes of the surface bonding that occur during the oxygen exposure. At very low exposures molecular oxygen dissociates at the clean surface. The oxygen adatoms provide a decrease in the density of states of the electrons at the surface for

energies near the Fermi level but no significant charge transfer occurs. Thus, for coverages up to 0.1 monolayer (ML) the oxygen adatom is in a "metallic" chemisorption bond. At higher coverages charge is localized in the Al-O bonds and charge transfer of electrons into the electronegative oxygen occurs. Ionic bonding begins at 0.1 ML and dominates with increasing coverage. Thus, with increasing oxygen coverage the chemisorption bond converts from metallic to ionic in character.

DESCRIPTION OF EXPERIMENTAL TECHNIQUES

Apparatus

The experimental apparatus used in our experiments is part of the surface science laboratories at the Physics Department of Montana State University. It consisted of two parts: an ultra high vacuum (UHV) surface science chamber and a pulsed nanosecond laser source and optical system.

The stainless-steel UHV chamber was once part of a Physical Electronics PHI-545 Scanning Auger Microprobe. The chamber was pumped by a 220 L/sec ion pump, a Leybold-Heraeus 150 L/sec turbomolecular pump, and a titanium sublimation pump. After a 24-hour, 140 C bake-out the base pressure of the system was routinely 1.5×10^{-10} torr. A schematic diagram of this apparatus is presented in Figure 1.

Argon and oxygen, purchased from Matheson Gas Products Inc. with purity of 99.9995% for argon and 99.99% for oxygen, were leaked into the chamber during the experiments, using separate Varian model 951-5106 leak valves. The gas manifold system, displayed in Figure 1, is designed to store the gases separately and can be pumped out using the turbomolecular pump after first closing the UHV gate valve to the main chamber. The argon gas reservoir was trapped with liquid nitrogen to reduce the water impurity level of the gas. Argon was used to clean the sample by ion bombardment. The sample was exposed to oxygen by backfilling the chamber with flowing oxygen to the desired pressure. Before introducing the gases into the chamber, the ion pump was valved closed for protection from back-contamination from the pump and the chamber pumped using only the turbomolecular pump. During experiments, the pressure was recorded by a bare ion gauge. This gauge was not in proximity nor in line of sight to the sample so as to prevent excited oxygen from hitting the sample. A voltage meter was used to extend the precision of the pressure reading so as to more precisely control the exposure rate.

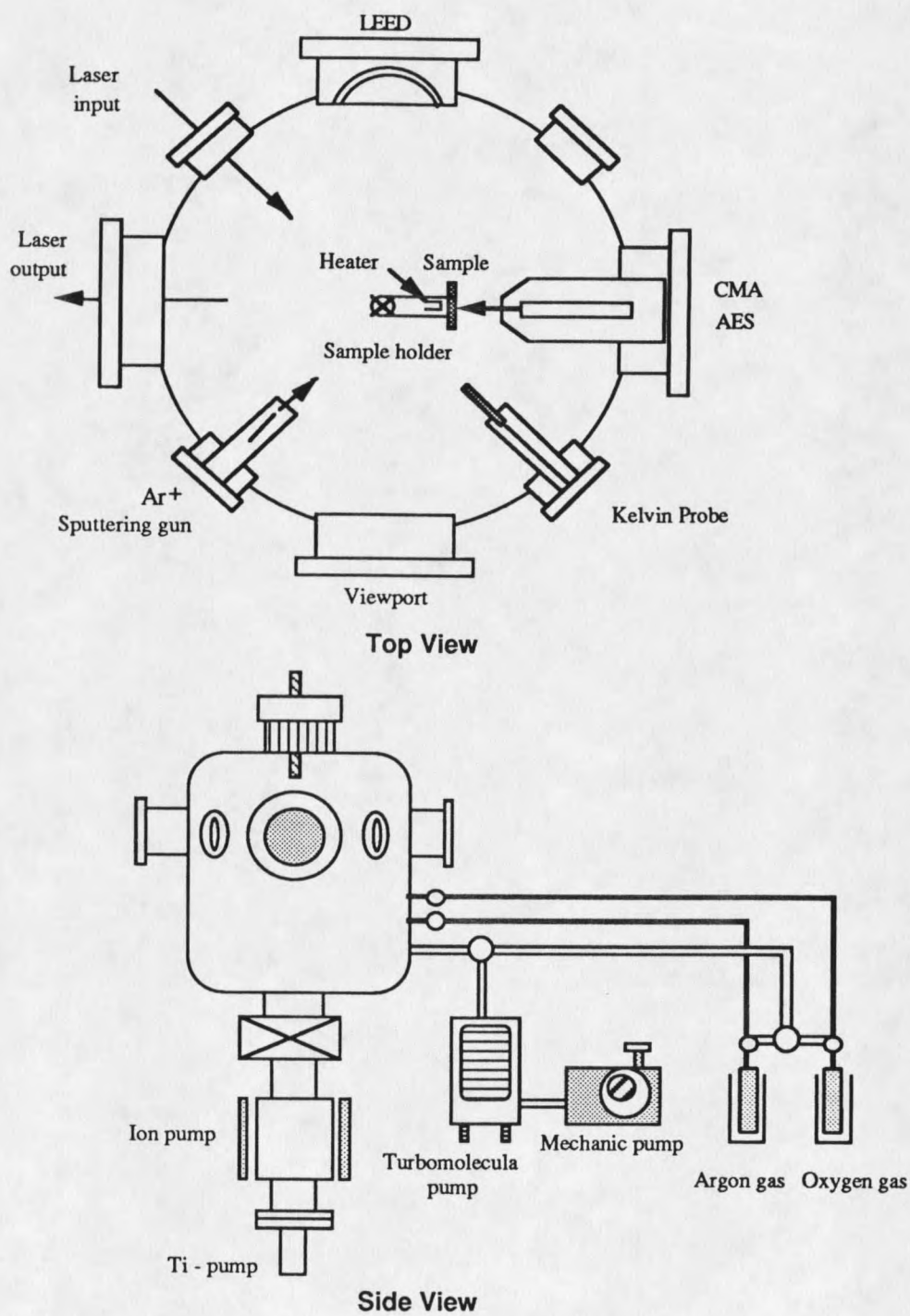


Figure 1. Schematic description of the ultra high vacuum experimental chamber; used in the experimental measure.

The sample was heated during the surface cleaning process using an electron beam heater. This heating scheme is shown in Figure 2. The sample was mounted above a hollow cylinder 1/2 inch in diameter. Inside this cavity a 0.05 inch tantalum filament was resistively heated to provide an electron source. A tantalum foil cylinder encircled and was connected electrically with the filament. The sample was held at ground potential and the filament floated at a high negative potential. The typical operating voltage on the filament was negative 1000 V, which produced a 4-mA emission current. This heating arrangement gave good temperature control and allowed heating from room temperature to 430 C in less than 5 minutes. Between heating cycles the rigidity of this sample holder kept the sample in a stable position, an essential requirement to retain a reproducible optical alignment for the second harmonic generation experiments.

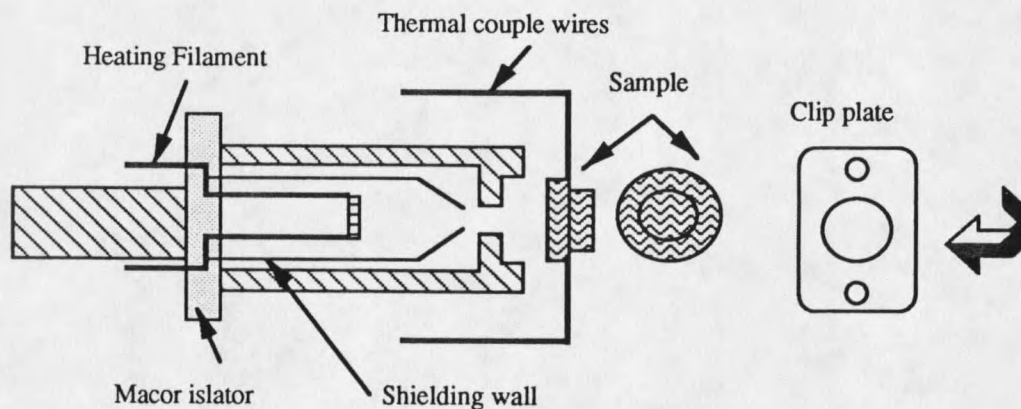


Figure 2. Experimental view of the sample holder.

Auger electron spectroscopy (AES) was used to monitor the sample surface contamination prior to making each oxygen exposure. We used a single pass cylindrical mirror analyzer (CMA) with an integral gun, operated in the low resolution modulated pass energy mode. In this mode, a computer utilized the digital to analog converter in the Stanford Research System Model SR510

lock-in to generate a ramp voltage V_s with a prescribed lower limit, upper limit, and sweep rate. The ramp voltage was modulated by a sinusoidal wave generated by the reference signal output of the SR510 lock-in, typically of 1.6 V peak to peak at a frequency of 4.07 KHz. The voltage V_s was connected to the V_m input of the CMA which is connected internally to the outer cylinder. The inner cylinder was grounded. The pass energy of the analyzer E is proportional to V_m . This relationship was taken to be that given by the operation manual,⁴ $E = 1.76 V_m$, where E is in electron volts and V_m is in volts. The exact value of the analyzer constant (1.76) depends on the structural geometry of the CMA and the work function variation in the system. Subsequent measurements of this system suggest that the proportionality value should be 1.72. However, this error leads only to a small and unimportant general shift of the peak energies. The CMA and associated control electronics derived from the original PHI 545 scanning Auger microprobe.

A Varian LEED unit was also available on the chamber. It consisted of the Varian LEED optics, a coaxial electron gun, model number 981-2145, and LEED control unit, model number 981-2148. LEED was routinely used to monitor the cleaning and annealing of the sample by checking surface orientation. It is also capable of performing AES.

A Delta-Phi Electronic Model 05 Kelvin probe was used to measure the change in work function of the sample during oxygen exposure. The probe used for the measurements was a molybdenum paddle mounted on a piezoelectric crystal. The assembly was driven to oscillate at its resonant frequency near 170 hz by the Kelvin Control Unit. The change of work function was measured in two ways. As purchased, the Kelvin probe control unit contained a lock-in amplifier and oscillation feedback circuit. The work function difference between sample and paddle, or CPD, is available as a separate output of the control unit. This CPD reading was digitized and stored in a microcomputer as a function of exposure time. In second method, an alternative instrumentation setup was used to eliminate a voltage drift associated with Kelvin control unit and to improve the signal-to-noise. In this case the lock-in amplifier of the Kelvin probe control unit

was replaced by the SR510. However, no significant differences in the CPD measurements were found using the two methods.

Measurement of the SHG signal was performed using the experimental arrangement shown in Figure 3. The laser was a pulsed Q-switched mode-locked neodymium-doped yttrium aluminum garnet (Nd:YAG) laser, a Spectra-Physics Quanta-Ray® DCR-3. It provided light pulses at a wavelength of $1.06\ \mu$. Although the pulse full width at half maximum was rated at between 7 and 9 ns by the manufacturer, we operated in a low power mode that provided pulses of about 15 ns at $1.06\ \mu$ with a repetition rate of 10 hz. A second harmonic component was uncontrollably generated within the laser cavity and had to be blocked by a dielectric mirror chosen to allow transmission only of the main $1.06\ \mu$ component.

Variation of the laser intensity was monitored using a reference signal, channel B in Figure 3. This reference signal was generated by reflecting twenty percent of the incident beam through a quartz quarter wave plate as indicated in Figure 3. The reference signal was detected using a photomultiplier tube (PMT), after passing through a BG-18 filter to the block $1.06\ \mu$ beam. Some plastic neutral density filters were used to further reduce the signal intensity.

The remaining eighty percent of original $1.06\ \mu$ beam was directed into the experimental chamber. The SH signal produced in this beam was directed to channel A in the electronics. Dielectric mirrors optimized for reflecting $1.06\ \mu$ light were utilized to direct the beam onto the sample. Before the beam was incident into the chamber, a Fresnel rhomb polarizer was used to select the desired polarization with respect to the incident plane of the sample.

The laser produced a diffraction-limited spot size of about 9 mm diameter. In order to have a smaller, nondivergent beam at the sample, a beam size reducer was built using two thin lenses, a converging and a diverging as illustrated in Figure 4. The ratio of the input and output beam diameters w_1 and w_2 can be determined by simple plane geometry resulting in the relation:

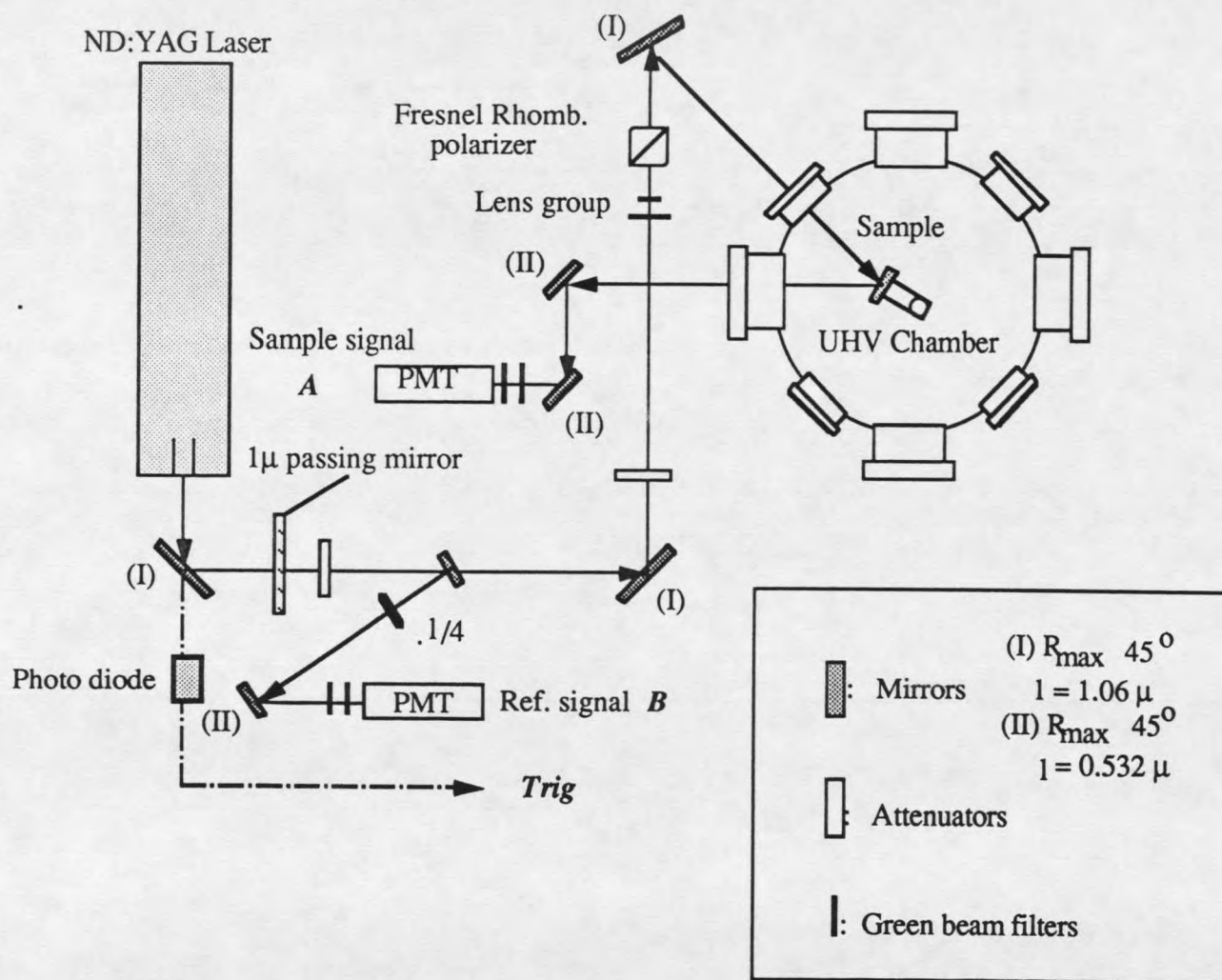


Figure 3: Second harmonic generation experimental configuration

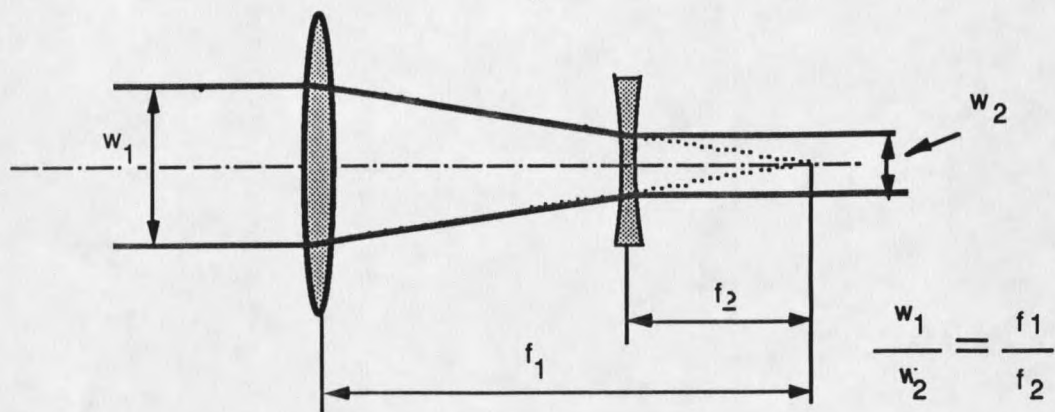


Figure 4. Schematic description of beamsize reducer.

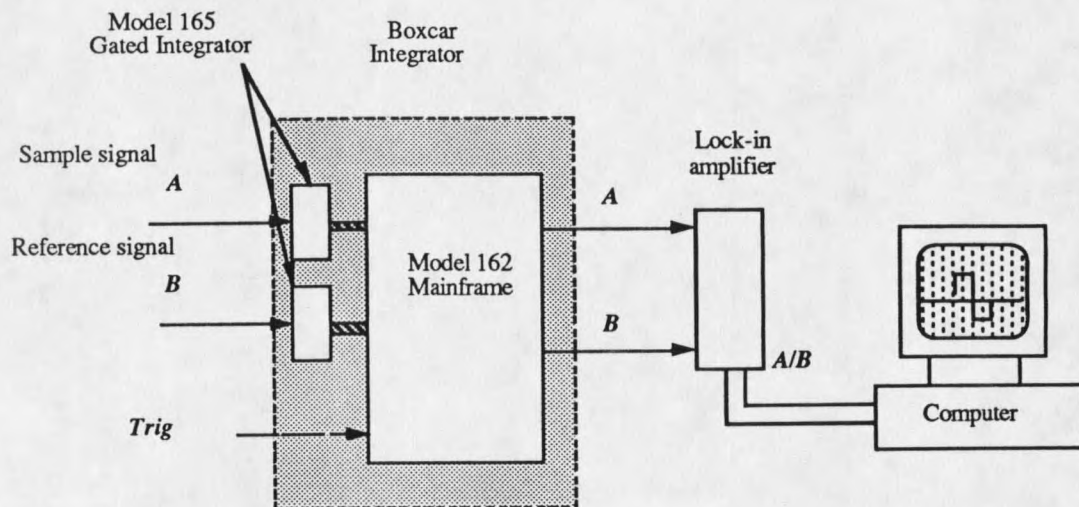


Figure 5. Boxcar electronic setup.

$$\frac{w_1}{w_2} = \frac{f_1}{f_2}. \quad (1)$$

The choice of the beam size was accomplished by appropriate selection of focal lengths. In our experiment, we needed the beam size of about 2.5 mm diameter. Thus, we chose $f_1 = 200$ cm and $f_2 = 50$ cm.

The beam was then directed into the experimental chamber through a zero-length 7056 glass/Kovar viewing window attached to a 2.75" Conflat[®] flange. The light was incident on the sample at an angle of 30°. The reflected 1.06 μ and the surface generated 0.532 μ second harmonic light were collinear from the sample, although some unwanted diffuse scattering occurred, indicating that the sample was not polished optically flat or smooth. The reflected beam exited the chamber through a similar glass window. The 1.06 μ beam was filtered from the signal optical path using two dielectric mirrors which reflected only the 0.536 μ light. Further spectral filtering of this light was performed with a KG-3 filter before it entered the second PMT.

The electronic instrumentation consisted of a pair of photomultiplier tubes (model RFI/S MK 11) from Thorn EMI Gencom Inc., a pair of Model 165 gated integrators and a Model 162 boxcar from EG&G Princeton Applied Research.⁵ The boxcar signal-averaging technique is standardly used when making measurements of a pulsed nature to increase the signal to noise ratio. The electronics of this measurement are outlined in Figure 5. The sample signal was weaker than the reference signal and needed to be amplified by a factor of 10 prior to signal processing. In order to accommodate the time delay internal to the boxcar, both sample and reference signals were delayed by 75 ns before entering the boxcar. The boxcar samples the input signals with a time width aperture that can be fixed independently at any point on each of the input signal waveforms. The boxcar integrates each signal within its time aperture using a preselected, variable time constant. The output of integrator is the average of large number of repetitions of the input signal

over the aperture duration. In our experiment we used an integrating time constant of 100 μ s and an aperture duration of 50 ns, yielding analog signal averaged over about 130 pulses.

The Model 165 Gated Integrator module is a plug-in module for the Model 162 boxcar. Its full-scale signal-sensitivity is adjustable in seven steps, from ± 50 mV to ± 5 V, to accommodate a wide range of input-signal and noise levels. In our experiments, the sensitivity setting was chosen from 100 mV to 250 mV according to the signal magnitude.

The Model 162 mainframe contains the primary timing and control functions, as well as the output signal-processing circuitry. There is provision for supplying the gated-and-averaged signal from either channel to the mainframe output. Alternatively, one can select the difference between the two signals at the mainframe output. We used both channels connected with the model 165 gated integrator. With the addition of optional signal-processing circuit cards, the model 162 can provide an output consisting of the product, ratio, or log of the ratio of the two independent output channels. The mainframe output signal can be smoothed by a low-pass filter. In our experiment, two outputs of the integrators were directly connected to the ancillary analog-to-digital converters of the SR510 Lock-in so that we could digitize each channel in real time. Front-panel delay controls are provided on the boxcar that permit the aperture in each channel to be delayed independently from about 5% to 100% of the delay range, selectable in 1-2-5 sequence from 0.1 μ s to 50 ms. The boxcar output of each channel was normally optimized by adjusting the delay so as to best position the integrating window over the signal pulse. This value was typically 200 ns for our experimental configuration.

The output sample and reference signals channels A and B were digitized using two A/D converters contained in SR510 lock-in amplifier. The digitized signals were read by computer which also plotted the ratio of the sample signal to the reference signal. Performing this division was necessary to eliminate the effect of laser fluctuations. The computer was instructed to accumulate two or three points within the time constant of the boxcar electronics.

All the above measurements were controlled by an ATT 6300+ computer. A Stanford Research System Model SR510 lock-in amplifier, which provides a lock-in amplifier and D/A and A/D functions, was used as an interface between the computer and experimental hardware. The software control system was developed by adapting a general purpose experimental control program written by William Hough, an MSU undergraduate research assistant. When performing AES measurements, the lock-in output was recorded as a function of the electron kinetic energy. This output is proportional to the derivative of the Auger intensity with the CMA operating in the modulated pass energy mode. When performing Kelvin probe work function measurements using the standard configuration, the nulling voltage equal to the negative of the CPD, as described below, was measured as a function of time. In the second method of CPD measurements in which the SR510 is used, the lockin output was first calibrated to a known externally applied nulling voltage and this calibration used to translate the changes in lockin output with required nulling voltage during the measurements. That is, no nulling voltage is applied to the Kelvin probe in this approach. When performing second harmonic generation measurements, the SHG intensity was divided by the reference signal and then was plotted vs oxygen exposure as for the Kelvin probe measurements. The computer program is described in the appendix.

Sample Preparation

In general, the initial oxidation of the Al(111) surface has been shown to be strongly dependent upon surface geometry and surface condition,² the preparation of the sample being an important process. Several factors that might influence the surface preparation are the sample polish, the chemical etching process, the energy of argon ions for bombardment, and the annealing procedure.

The Al(111) single crystal sample with a purity greater than 99.999% was purchased from the Monocrystal Corporation of Cleveland, Ohio. The surface normal was determined to be within

0.5° of the (111) orientation using Laue X-ray diffraction. The shape of the sample was circular, about 1 cm in diameter, and about 3 mm thick. This thickness was selected to help ensure macroscopic flatness during the polishing and annealing steps. The sample was machined by electric discharge to have a raised surface. This geometry allowed for secure mounting and eliminated shadowing effects when the sample was cleaned by sputter ion bombardment.

It is necessary to get a good quality optical surface in order to reflect the laser beam with a minimum of scattered light during second harmonic generation measurements. To accomplish this, the sample was hand polished using successively finer grades of an alumina suspension micropolish, Buehler Model number 40-6353-006, with particle size from 2 μ to 0.05 μ . This step was followed by one where a Syton colloidal suspension of silicon polish purchased from Remet Chemical Corp. 278 Chadwicks, N.Y. 13319 was used. As a last step the crystal was chemically etched in an aluminum etchant solution: distilled water (95 ml), hydrochloric acid (1.5 ml), hydrofluoric acid (1.5 ml) and nitric acid (2.5 ml). With the use of this method, the surface retained its metallic mirror finish throughout the experiment. It also did not exhibit a loss of optical specularity by developing a matte-finish, as sometimes occurs in single crystal aluminum studies.⁶ We attribute this to use of the Syton final polish and low energy sputtering described below.

Following the initial chamber bake-out, the sample was found to be covered by an oxide layer which contained a high concentration of carbon. An AES spectrum of this surface is shown in the top panel of Figure 6. The crystal was initially cleaned by heating to 400 C in UHV for about 1 hour followed by an ion bombardment sputtering. For this sputtering and for subsequent cleaning we used 500 eV Ar⁺ ions, typically for 30 minutes at a beam current of 1-10 μ A with the sample at room temperature. It is believed that higher sputtering ion kinetic energies significantly damage the subsurface region and leads to increased light scattering after repeated cleanings.

The sample was annealed after each sputtering. There is evidence⁷ that the microscopic surface structure is sensitive to the final annealing procedure. This microscopic structure dramatically alters the initial oxidation kinetics of the Al(111) surface. In our preparation process,

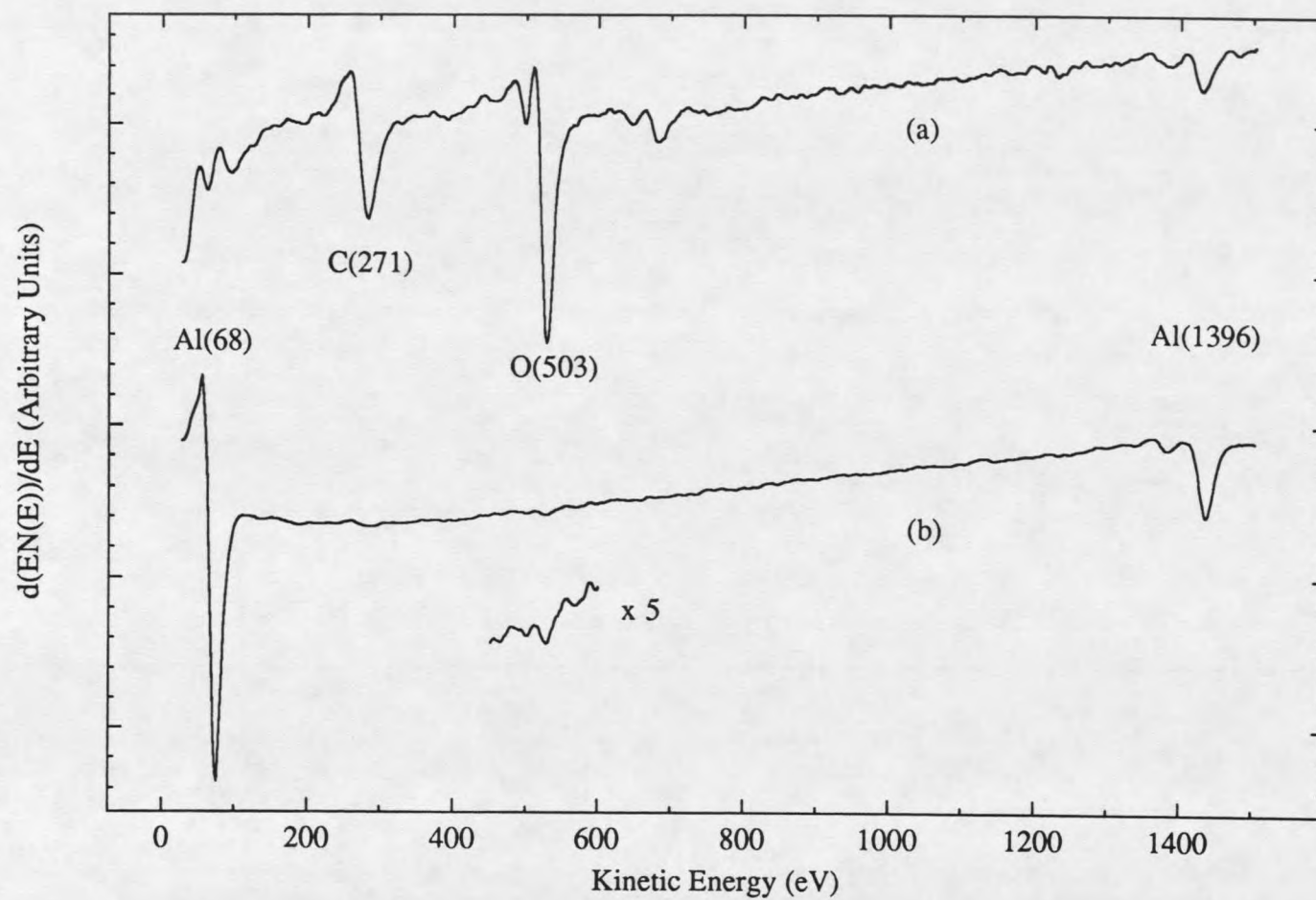


Figure 6. Auger spectrum of sample before (a) and after (b) cleaning.

the sample was annealed for 30 minutes at 425 C taking care to heat and cool the sample slowly, taking more than 20 min to heat to 425 C and 20 min to cool to 300 C. Heating or cooling more rapidly is believed to introduce microscopic surface faceting.⁷ This cleaning procedure was monitored by AES until the ratio of O (KLL) to Al(LMM) Auger signals was reduced to about 0.01. The Auger spectrum after cleaning is shown in panel (b) of Figure 6.

Low energy electron diffraction (LEED) patterns were checked to assure that a high-quality 1×1 pattern could be detected after sputtering and annealing the sample. We also observed the LEED pattern as the sample was exposed to O₂. The clean Al(111) surface displayed a very bright 1×1 structure, characteristic of a well ordered surface. During the initial 0 L to 50 L exposure of oxygen the diffraction spot brightness showed no large changes. In the range of 50 - 96 L the spot brightness showed deterioration. After 96 L exposure all the spots began to lose their brightness and gradually disappeared. These observations are in good agreement with the data of Michel et al.⁶

Auger Electron Spectroscopy

Auger electron spectroscopy (AES) was used to characterize the elemental composition of the sample surface. In this section, the essential ingredients of this technique are described.

A high energy beam of electrons with kinetic energies in the range of 1 to 10 keV is applied to bombard the sample surface. The electron incident to the sample creates a hole in one or more of the substrate atomic core levels. The ionized atom subsequently decays by one of two mechanisms. One is the Auger process. Referring to Figure 7, the hole initially created in the K shell is filled by an electron from the L shell, simultaneously ejecting a second electron from the L shell. This ejected electron is called the Auger electron and has a kinetic energy approximately equal to $E_K - 2E_L$, where E_K and E_L are the binding energies of the K and L shells, respectively. That is, the energy gained by the ion when the outer shell electron makes a downward transition to fill the inner

shell hole is carried away as kinetic energy by the Auger electron. Since the detailed energy level diagram is different for each element, the Auger electron has a kinetic energy that is unique to a specific atomic transition, and thus is sensitive to elemental types and local bonding configuration. However, as can be seen from Figure 7, there is a manifold of possible final states, one corresponding to each of the two-hole final-state wave functions of the L-shell. The same description can be applied to other pairs of shells and final-states involving these shells, leading to a characteristic, albeit complicated secondary electron emission spectrum.

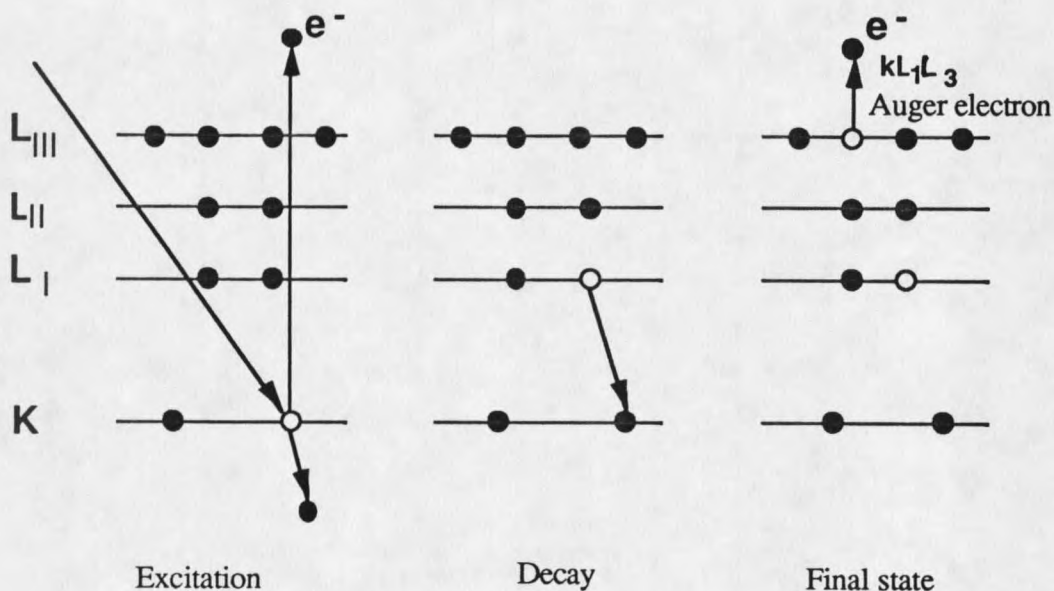


Figure 7. Schematic of Auger process.

A relaxation mechanism that competes with Auger emission is electromagnetic radiation, in which the relaxation energy is carried away by a characteristic x ray. However, the cross-section of x-ray emission in the core level binding energy range is usually negligible compared with that of the AES. Basically, the ratio of the probability of x-ray emission P_X to the probability of Auger electron emission P_A is given empirically⁴ as a function of atomic number Z by $P_X/P_A = (-$

$6.4 \times 10^{-2} + 3.40 \times 10^{-2}Z - 1.03 \times 10^{-6}Z^3$). The probability of Auger emission is greater than that of the x-ray emission at atomic numbers lower than arsenic, $Z = 33$.

Because of the large incident beam flux and the multifarious open relaxation channels in a solid, the characteristic Auger electrons appear as small peaks on top of a slowly varying secondary electron energy distribution $N(E)$. Hence, in practice one usually measures the derivative of the energy distribution function to suppress this large background. In general, the peaks used to identify the elemental composition of the sample occur in the energy range of 50 - 2400 eV and are identified by comparing the analysis spectra with published "standard" spectra.⁸

Auger analysis is sensitive to the elemental composition of the first few atomic layers. The surface sensitivity of this technique is due to the small escape depth of electrons in the 50 - 2400 eV kinetic energy range,⁹ typically about 10 - 30 Angstroms. Hence, in Auger electron spectroscopy the volume of the analyzed region near the surface is proportional to the escape depth and the electron beam spot area.

In our system, a single pass cylindrical mirror analyzer (CMA) was used to measure the kinetic energy of the Auger electrons. An important asset of the CMA is its high transmission and good energy resolution. This combination provides a very sensitive instrument with the resolution desired for Auger electron spectroscopy measurements. The CMA can be operated in several modes. The one chosen for present measurements was the retarding field mode. In this mode, electrons enter the analyzer through a circular entrance aperture in the inner cylinder. Electrons of a particular energy are reflected by the potential applied to the outer cylinder back through the exit aperture in the inner cylinder and are focused on the final exit aperture. The inner cylinder is at zero, or ground potential. The pass energy of the analyzer is the kinetic energy of the electron that can travel the entire analyzer and exit through the exit aperture when a given voltage V_M , the voltage is applied between the inner and outer cylinders.⁴ The relation between the pass energy E and the voltage V_M was taken to be the theoretical value $E = 1.76V_M$ as discussed above. The

electron current $I(E)$ passing through the exit aperture of the CMA is related to the secondary electron energy distribution function $N(E)$, given as a function of pass energy as:⁴

$$I(E) \sim \left(\frac{T}{R}\right) E N(E). \quad (2)$$

Here T is CMA transmission and $R = E/\Delta E$ is the CMA resolving power and ΔE is the energy window. $R=160$ for our CMA. The factor (T/R) reaches a maximum when the electron beam is located at the focal point of the CMA. Thus, T/R summarizes for the electron optical properties of the CMA.

In order to accentuate the small Auger peaks in $N(E)$ one usually monitors the derivative of $N(E)$. To perform this differentiation a small-amplitude sinusoidal voltage is added to the DC voltage applied to the outer cylinder of the CMA, V_M . For a sinusoidal voltage, $k \sin(\omega t)$, applied to V_M , the energy of detected electron would be $E + \Delta E$, where $\Delta E = k \sin(\omega t)$. Since k is a small number, one can expand the $I(E)$ by Taylor expansion of ΔE . From Eq. (2)

$$I(E + \Delta E) \sim (E + \Delta E) [N(E) + N'(E)\Delta E], \quad (3)$$

and one obtains the first harmonic of the current detected at the exit aperture $i_\omega(E)$:

$$i_\omega(E) \sim kR [N(E) + E N'(E)]. \quad (4)$$

The first term in the brackets is small except for low kinetic energies, less than 50 eV. Thus the first harmonic current primarily consists of the derivative of the kinetic energy distribution. The first harmonic current is typically 10^{-12} Amps and must be amplified using a channel electron

multiplier prior to detection. After amplification, the measured first harmonic current for $E > 50$ eV is approximated by the following equation:

$$i_{\omega}(E) = GG_{em}k\left(\frac{T}{R}\right)EN'(E). \quad (5)$$

where G_{em} is the electron multiplier gain, G is the linear gain of the electronic instrumentation, k is amplitude of the modulation voltage (< 12 V), E is the Auger electron pass energy, and N' is derivative of the Auger electron distribution function. and G_{em} is a nonlinear function of the multiplier voltage and the energy of the electron striking the first dynode of the electron multiplier.

Ionization cross sections and backscattering factors are also important for the interpretation of the Auger signal. The ionization cross section is the probability of ionization in an electron-atom scattering event. The backscattering factor is the number of Auger secondary electrons that will undergo additional scattering events prior to being emitted from the sample. Bishop and Riviere¹⁰ have developed the following parameterized formula for the cross section $\sigma(\phi, E_p)$:

$$\sigma(\phi, E_p) = \frac{2\pi e^2}{E_p \phi} b \ln \frac{4E_p}{B}. \quad (\text{cm}^2) \quad (6)$$

Here, E_p is the primary beam energy of the incident electrons, ϕ is the ionization potential, e is the charge on the electron, b is equal 0.35 for the K shell electrons and 0.25 for the L shell electrons, and B is a function of E_p and ϕ . Worthington and Tomlin¹¹ found B to have the form

$$B = [1.65 + 2.35 \exp(1 - \frac{E_p}{\phi})]\phi \quad (7)$$

Combining these results yields, using $\beta = E_p/\phi$:

$$\sigma(\phi, E_p) = 1.3 \times 10^{-13} \frac{b}{\beta \phi^2} \ln \left[\frac{4\beta}{1.65 + 2.35 \exp(1 - \beta)} \right] (\text{cm}^2). \quad (8)$$

They arrived at this empirical form of cross-section so that $\sigma(\phi, E_p)$ would vanish at E_p less than ϕ , would reach its maximum at about $E_p \approx 3\phi$ and so that the shape would be the same as had been shown experimentally. The plot of cross-section as function of β is shown in Figure 8.

This result for $\sigma(\phi, E_p)$ should also be corrected to include the contributions due to multiply scattered and backscattered electrons. This correction term is given by

$$\sigma' = \int_{\phi}^{E_p} P \sigma(\phi) n(\phi) d\phi. \quad (9)$$

where $n(\phi)$ is the number of backscattered electrons at energy ϕ . This correction term can be combined with $\sigma(\phi, E_p)$ to give the total corrected cross section

$$\sigma(\phi, E_p) = \sigma'(\phi, E_p) [1 + r_m(E_p, \phi, \alpha)] \quad (10)$$

where $r_m(E_p, \phi, \alpha)$ is the correction due to the backscattered electrons. The term $r_m(E_p, \phi, \alpha)$ is dependent on both the incident and ionization energies and the matrix, denoted by α , that the atoms are embedded in. Reuter¹² has empirically determined a relationship for the backscattering term. The relationship is

$$r_m(E_p, \phi, \alpha) = 1 + 2.8\eta(1 - 0.9 \frac{\phi}{E_p}) \quad (11)$$

η is a material-dependent parameter and is given by

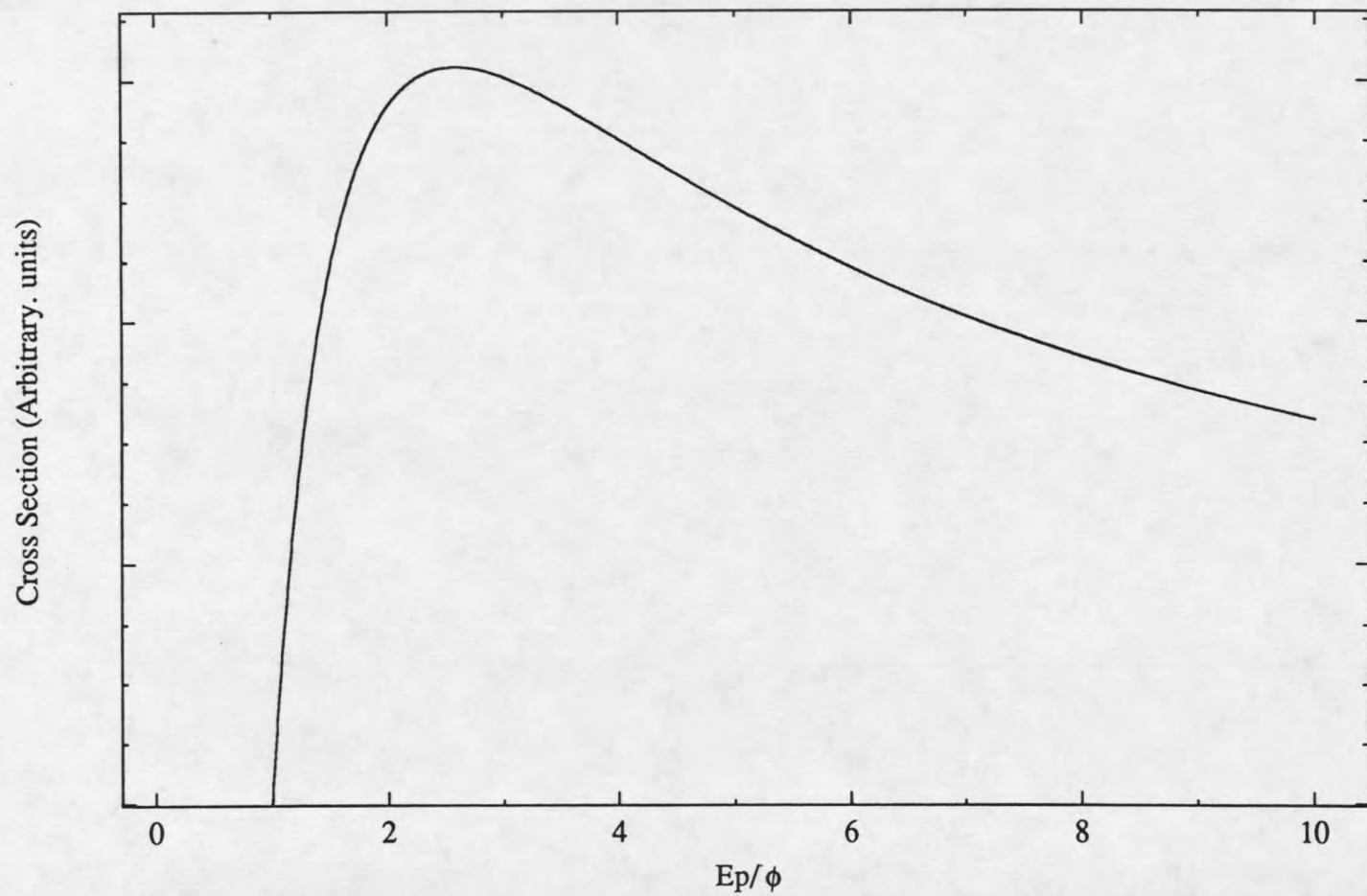


Figure 8. Auger cross section as a function of primary beam energy.

$$\eta = -0.1254 + 0.016Z - 0.000186Z^2 + 8.3 \times 10^{-7}Z^3 \quad (12)$$

Z is the substrate atomic number. In the case of aluminum, where $Z = 13$, the backscattering correction term $r_m(E_p, \phi, a)$ is equal to 1.08 for a choice of $\phi/E_p = 0.5$.

Following the development summarized above, the Auger current is proportional to the ionization cross-section and probability for Auger emission P_A as shown:

$$I \propto \sigma(E_p, \phi) \times P_A \quad (13)$$

One can compute that for a typical current of $1 \mu A$ at 2 keV incident on a sample, a monolayer of oxygen adsorbed on copper will produce approximately 10^{-11} A of Auger electrons from the oxygen K shell. To detect 1% of one monolayer the relative sensitivity of the instrumentation system would have to be at 10^{-4} and its absolute sensitivity be 10^{-13} A. Thus, for the detection of trace atomic coverages on surfaces using AES is problematic.

Contact Potential Difference Measurement

The work function of a metal is defined as the minimum energy required to free an electron from the solid. If the charge distribution on the surface is the same as that in the bulk, the work function Φ is equal to the negative of the Fermi energy, $-\epsilon_F$ relative to the vacuum level. However, this is usually not the case because of the distortion of the positions of the surface ions and a nonvanishing electric dipole moment in cells near the surface. Thus the charge distribution on the surface is not the same as that in the bulk. The difference in the distribution can be modeled as a uniform macroscopic surface dipole density. This surface dipole density is often termed the double layer.¹³ Figure 9 gives a schematic diagram of how the charge density (a) and the crystal potential

(b) vary in the surface region. To obtain a correct expression for the work function Φ the amount of energy required to move an electron through this double layer W_s must be added to the negative of the Fermi energy ϵ_F :

$$\Phi = -\epsilon_F + W_s. \quad (14)$$

The double charge layer is affected by many factors, such as, crystal orientation, adsorbed molecules, and surface condition. This sensitivity can be used to advantage in surface studies. For example, the change of the work function due to adsorbed molecules characterizes the charge rearrangement that occurs upon chemisorption. For metal this rearrangement occurs very near the surface. By measuring the work function change at specific coverages and assuming specific chemical valency, geometric bond models can then be inferred.

The contact potential difference (CPD) relates the work function of two dissimilar materials. The origin of the CPD is as following: If two conductors A and B of work functions Φ_A and Φ_B , respectively, and at the same temperature, are placed in electrical contact, electrons will flow in one direction until an equilibrium state is reached, at which point the Fermi levels in the two conductors are the same. That is, at equilibrium the electrochemical potentials of electrons in the two conductors must be equal. Then the electric potentials V_A and V_B just outside the surface of A and B, respectively, will be

$$V_A = -\frac{\phi_A}{e}, \quad (15)$$

and

$$V_B = -\frac{\phi_B}{e}, \quad (16)$$

Obviously

$$V_A - V_B = \frac{1}{e}(\phi_B - \phi_A) = V_{AB}. \quad (17)$$

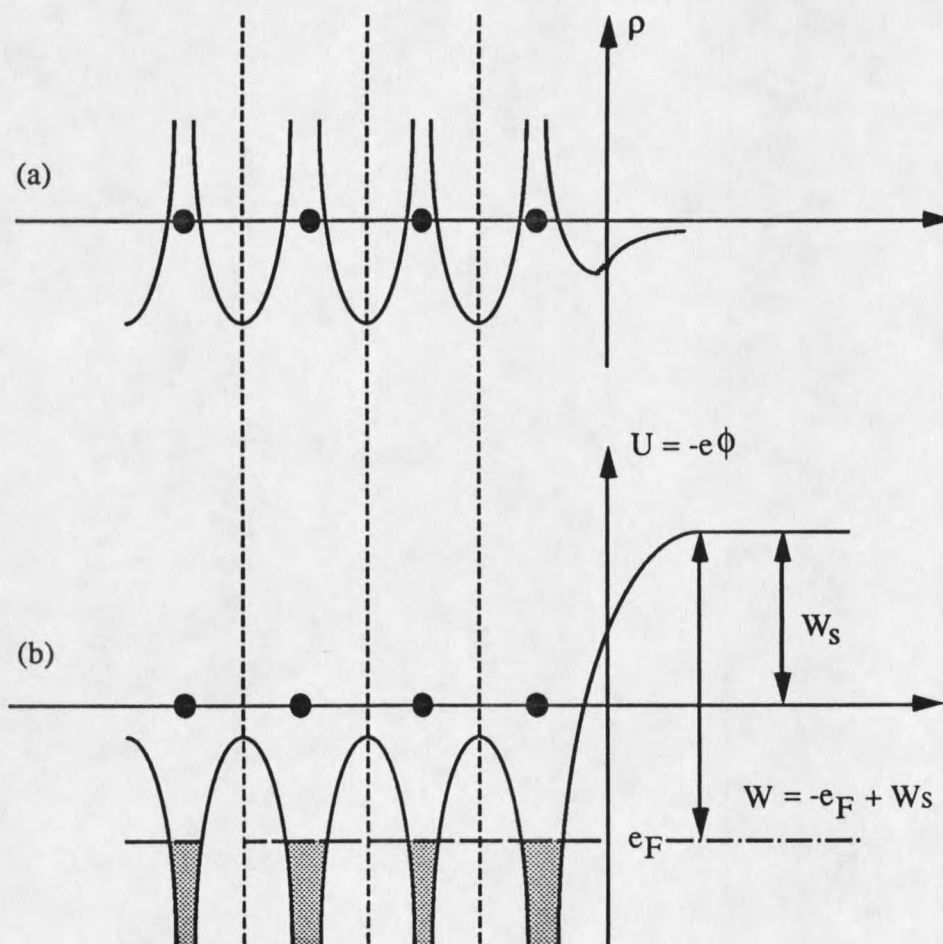


Figure 9. Effect of the surface on the binding energy of an electron.
 ϕ : work function. ρ : electron charge density U : crystal potential.

V_{AB} is known as the CPD between A and B, and can be positive, negative, or zero. If the conductors are at different temperatures, a thermoelectric term must be added to the right-hand side of Equations (15) through (17).

For a surface having patches of different work function values, the potential Φ_i at a point just outside the i^{th} patch, at a distance above the surface small compared with the dimensions of the patch is

$$V_i = -\frac{\Phi_i}{e}, \quad (18)$$

In the absence of applied fields, the potential at a distance large compared with patch dimensions is simply

$$V = -\frac{\Phi}{e}, \quad (19)$$

where Φ defines the average work function of a patchy surface. If the conductors A and B are originally considered to have patchy surfaces, then the CPD is computed from equation (2) using the average values of Φ for each material.

Our CPD measurements were made using the Kelvin probe method, which is illustrated in Figure 10. This method is one of the more common methods in use. In the diagram, the sample A and the probe B are connected via an electric circuit and form a parallel plate capacitor C. The probe paddle is driven to vibrate periodically with respect to the sample so that C is a function of time. A external time-independent potential V_E is introduced into this measurement circuit, as shown in Figure 10. Then, following Kirchoff's second law

$$\Delta V = V_{AB} + V_E. \quad (20)$$

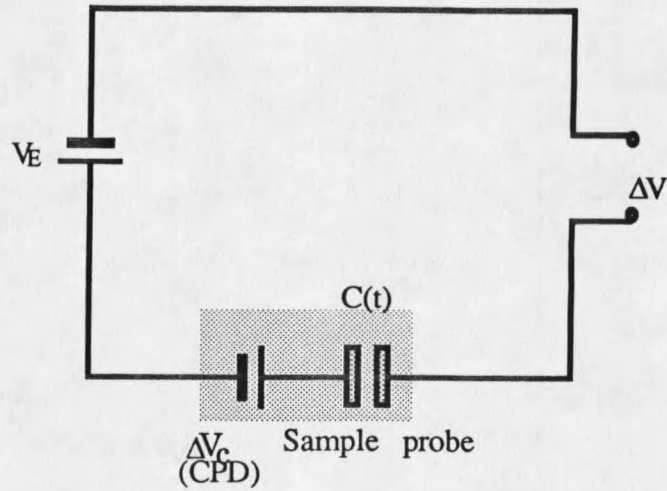


Figure 10. Schematic of contact potential difference method.

By adjusting V_E , ΔV can be made to vanish such that the corresponding value of V_E is the magnitude of the CPD. The charge on the sample in the presence of an applied voltage V_E is given by:

$$Q = C(V_{AB} + V_E) \quad (21)$$

In the Kelvin probe technique the capacitance C is made to vary in time but the V_{AB} and V_E are held constant. If the capacitance changes by an amount ΔC over a time interval Δt , a quantity of charge ΔQ ,

$$\Delta Q = \Delta C (V_{AB} + V_E) \quad (22)$$

will be displaced in the circuit. V_E is adjusted until there is no charge flow in the circuit. That is when $\Delta Q=0$, then

$$V_{AB} = - V_E \quad (23)$$

In this fashion an alternating current is generated which is easily detected and amplified using lock-in amplifier techniques.

Surface nonuniformities can cause an error in measuring the CPD. Craig et al.¹⁴ showed experimentally that the CPD obtained using the Kelvin probe method was a function of the distance between the sample and probe paddle. They found that there existed a range of spacing over which a nulled signal remained essentially zero, but that for larger and smaller spacings a finite signal arises. The origin of this behavior was found in the simultaneous occurrence of two effects: First, the sample and probe may not be precisely parallel, so that as the average plate spacing is changed, the capacitance per unit area varies non-uniformly. This presents a variation in V_E . Second, there may be variations of work function from point to point across the sample and probe surface.

The magnitude of the effect of surface non-uniformity can be estimated following the approximation of the Craig et al.¹⁴ who used a two capacitor model as shown the Figure 11. In the figure, V_1 and V_2 are the CPD values of the two parallel capacitors, Δ is the difference of two spacings of capacitors, and x is a spacing of first capacitor. Thus, the spacing of the second capacitor is $x+\Delta$. The two CPD values are to represent the non-uniformity that may exist in a typical experimental system. The nulling voltage refers to the external voltage V_E required to eliminate charge flow through the detector. This model was used in the Ref. [14] to derive the formula that describes how the error in the CPD measurements varies with experimental parameters. Figure 12 demonstrates the variation for one set of choices of experimental parameters. The plot shows the calculated output of the apparent CPD measurements as a function of plate spacing for different nulling voltages using $V_1=150$ mV, $V_2=250$ mV, $\Delta=0.5$ mm, and the amplitude of the probe vibration $\partial x=0.01$ mm. Curves (a)-(d) correspond to the nulling voltages from 150 mV to 225 mV. From the figure one can conclude that a value of nulling voltage lying

between the values of V_1 and V_2 provides the least error in CPD over the largest range of sample-probe distances.

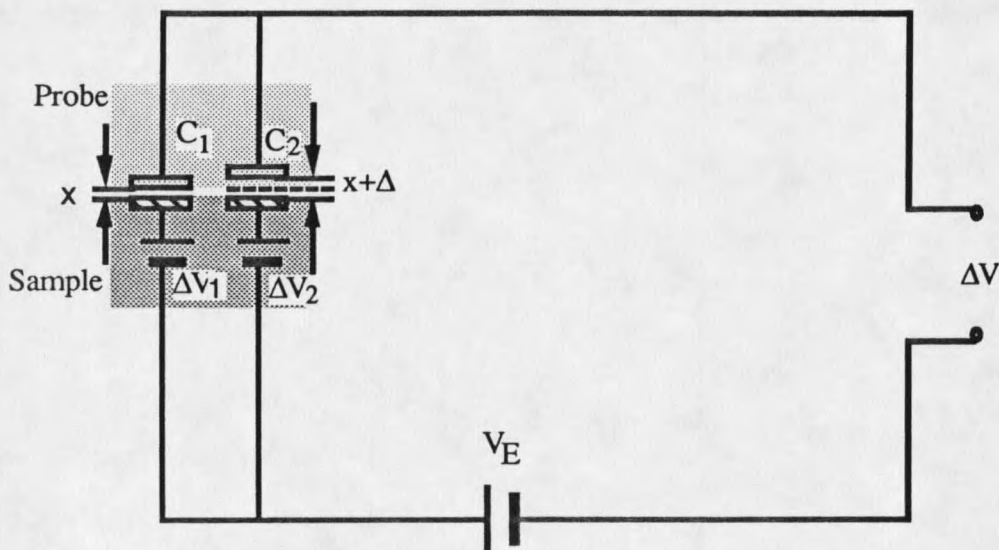


Figure 11. Schematic of two capacitor model for CPD measurements.

In order to test experimentally this simple model we measured the CPD signal vs sample-probe spacing for our system, maintaining a constant probe modulation amplitude. The result of this measurement is shown in Figure 13. There is a strong dependence in apparent CPD with decreasing sample-probe distance for values less than 0.5 mm. The qualitative agreement with the model calculation is excellent. The size of the CPD variation is large and suggests a difficulty in using the Kelvin probe technique for determining absolute values of CPD. In our experiments, however, only changes in CPD due to chemisorption are measured and will not suffer the same sensitivity to sample-probe spacing.

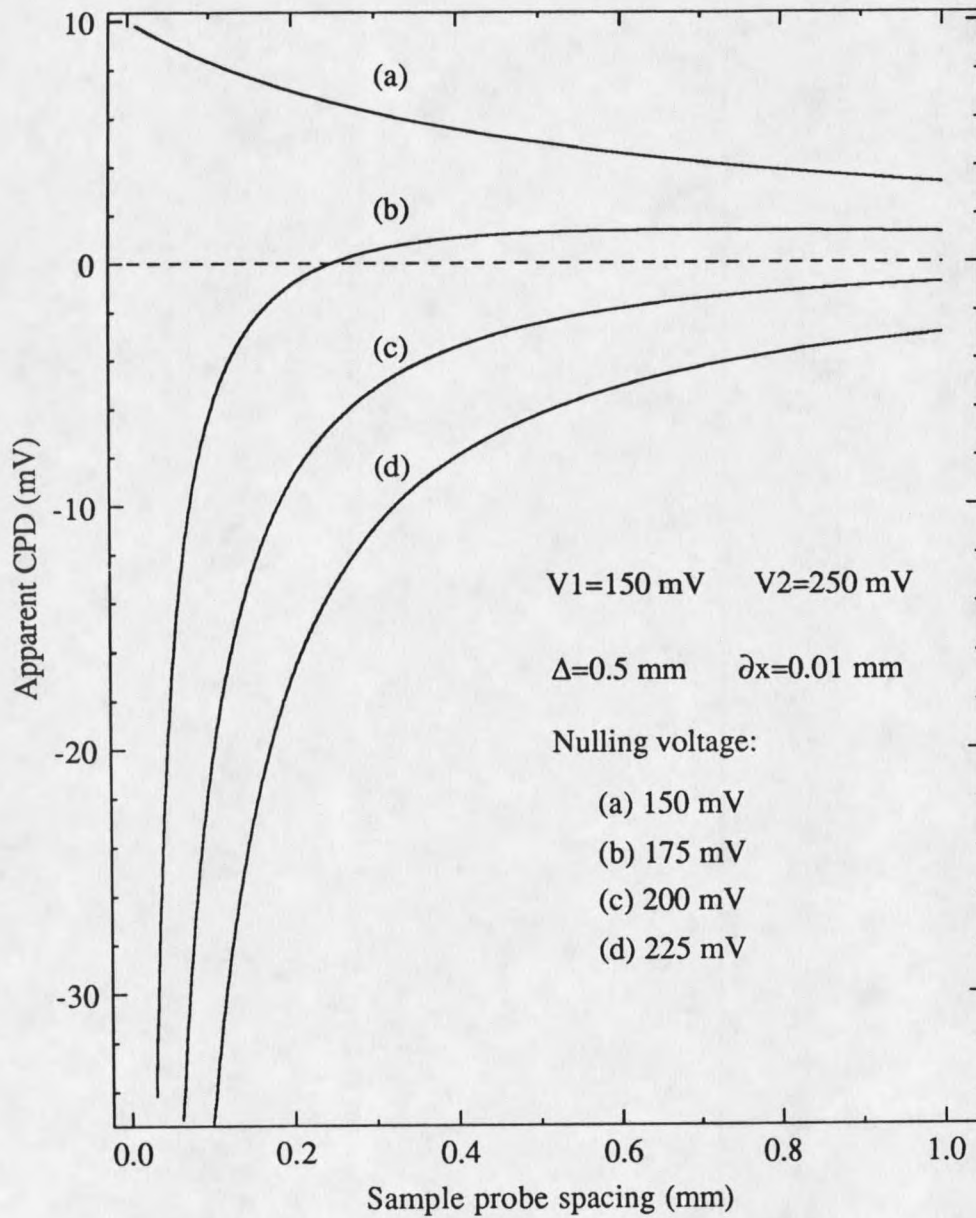


Figure 12. Two capacitor model results for CPD measurements.

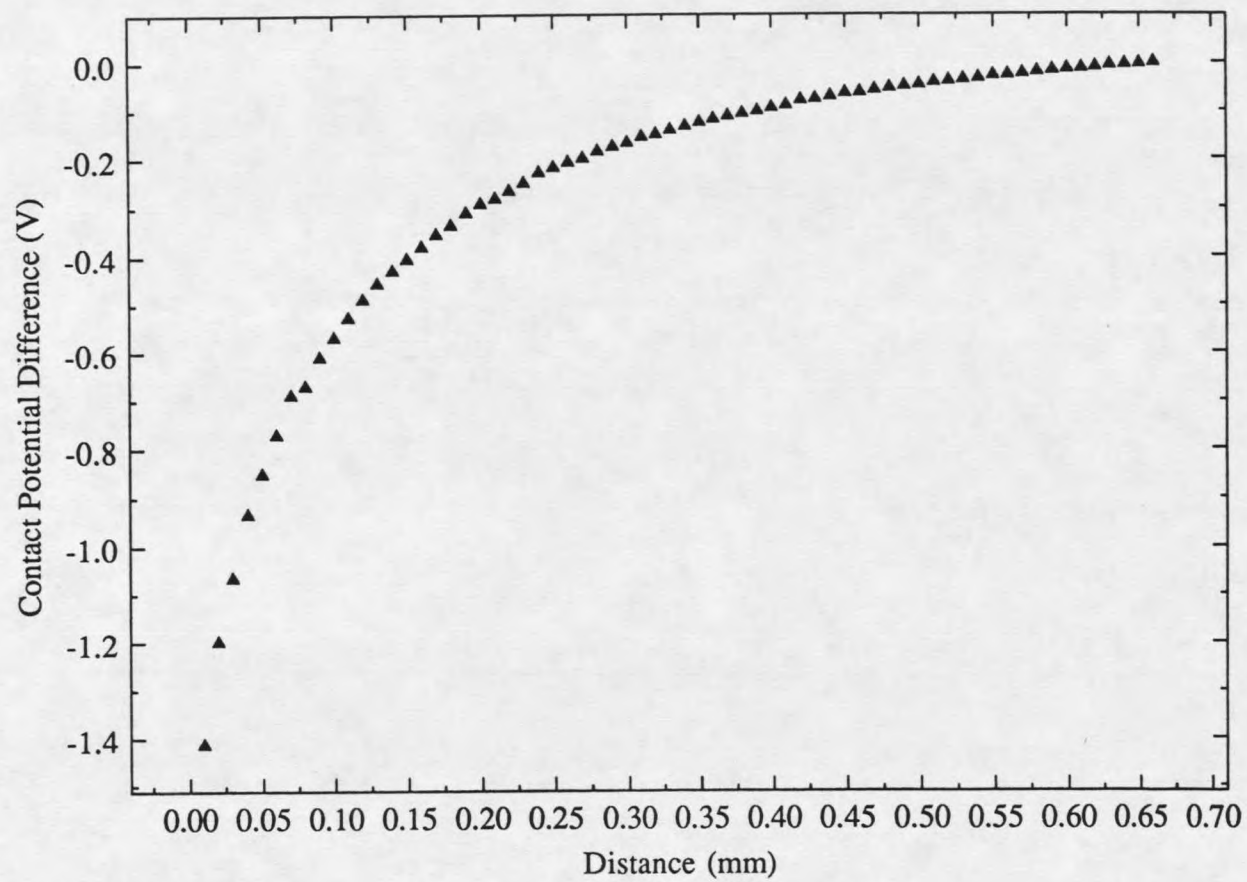


Figure 13. Measured dependence of CPD with sample-probe distance.

Surface Second Harmonic Generation

Surface enhanced second harmonic generation (SHG) is a promising new surface science technique. Two appealing features of SHG are its experimental simplicity and its sensitivity to the surface region. This sensitivity is a result of the fact that under the electric dipole approximation, SHG is forbidden in the bulk of centrosymmetric media and is only allowed at the surface, where the inversion symmetry is broken. A nonintrusive probe, SHG has been shown to have submonolayer sensitivity to molecular adsorbates, and can be used to probe *in situ* the kinetics of molecular adsorption, molecular orientation, and surface geometric structure.¹⁵ Because of its fast time response, it has been demonstrated to be an effective probe of surface phenomena into the femtosecond time regime.¹⁵

Early second harmonic generation experiments dealt primarily with the bulk response of non-centrosymmetric materials or that of non-ultra high vacuum surfaces. Most experimental results suffered from experimental limitations such as low power and broad energy width of laser source or inconsistencies in sample preparation. The theoretical development of SHG dates back to 1948, but before the 1980's the emphasis was on the understanding of surface nonlinear optical process itself and the origins of surface nonlinearities, not on materials dependence. Generally there still remain difficulties in theoretical calculation. Although the submonolayer sensitivity of SHG was recognized in a few cases, little effort was made to develop SHG into a viable surface tool.¹⁶ More recently, the second harmonic generation technique has been applied to the surface of metals and semiconductors in ultra high vacuum. This decade, Y. R. Shen and co-workers at Berkeley have performed a number of experiments demonstrating the versatility of the SHG as surface probe.³

The experimental concept is shown in Figure 14. A monochromatic laser beam at frequency ω is incident onto a sample surface. It induces a nonlinear polarization which then

radiates an additional component at frequency 2ω . The experimental measurement consists of separating the 2ω component from the reflected ω component.

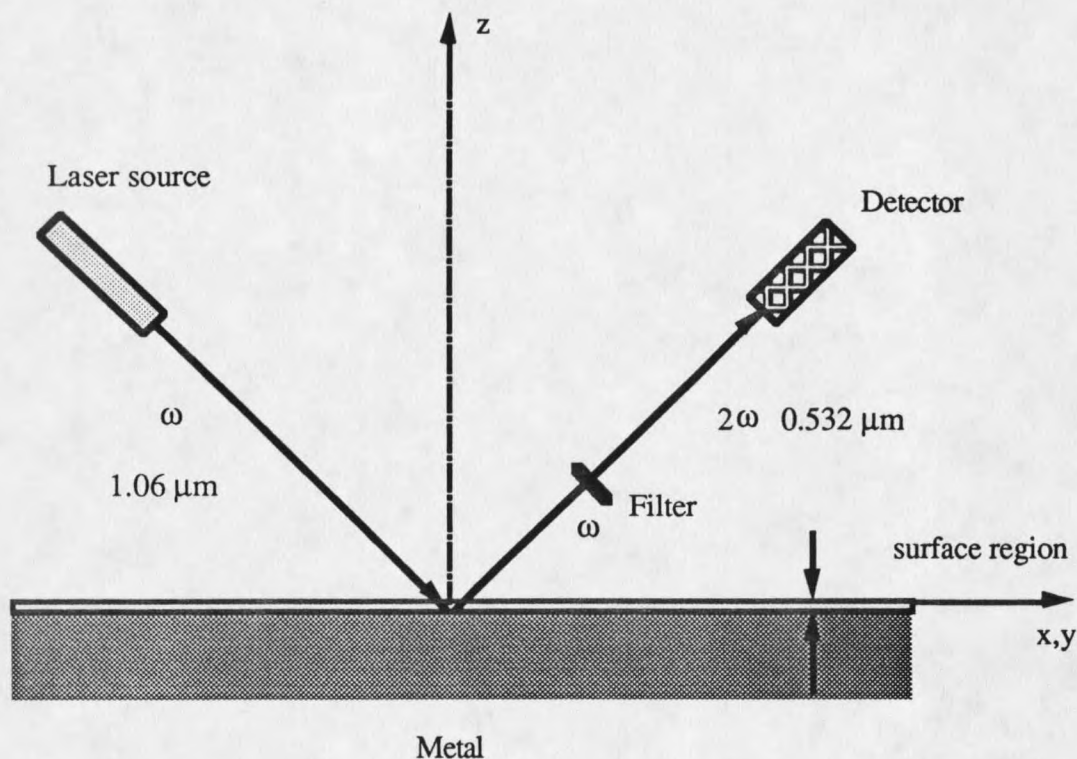


Figure 14. Schematic of SHG measurements.

The extreme surface sensitivity of SHG has been demonstrated by Weber and Liebsch who calculated the non-linear polarization of a surface using the jellium model for metals.¹⁷ Those authors found that the second order polarization response to the incident laser beam electric field originates in a very narrow region of the surface about $1 \text{ \AA}$ wide, centered actually above the surface by about $0.5 \text{ \AA}$. This result is consistent with experimental results that suggest that SHG provides an extremely sensitive tool for chemisorption studies.

The theoretical description of SHG requires a consideration of the optical properties of solids. All electromagnetic phenomena are governed by the Maxwell's equations for the electric and magnetic fields $\mathbf{E}(\mathbf{r},t)$, $\mathbf{B}(\mathbf{r},t)$ and for the source terms $\mathbf{J}$, ρ . In many cases, for example in metals and semiconductors, it is convenient to describe SHG using a generalized electric polarization $\mathbf{P}$ defined by

$$\mathbf{j} = \mathbf{j}_{dc} + \frac{\partial \mathbf{P}}{\partial t}. \quad (24)$$

where $\mathbf{j}_{dc}$ is the time-independent dc current density. Then Maxwell's equations appear in the form

$$\begin{aligned} \nabla \times \mathbf{E} &= -\frac{1}{c} \frac{\partial \mathbf{B}}{\partial t}, \\ \nabla \times \mathbf{B} &= \frac{1}{c} \frac{\partial (\mathbf{E} + 4\pi \mathbf{P})}{\partial t} + \frac{4\pi}{c} \mathbf{j}_{dc}, \\ \nabla \cdot (\mathbf{E} + 4\pi \mathbf{P}) &= 0, \\ \nabla \cdot \mathbf{B} &= 0. \end{aligned} \quad (25)$$

where $\mathbf{P}$ is now the only time-varying source term.

The polarization $\mathbf{P}(\mathbf{r},t)$ is usually a complicated nonlinear function of $\mathbf{E}$. However, it can be expressed in the frequency domain as a sum of a linear term and a nonlinear term.

$$\mathbf{P}(\omega) = \mathbf{P}^{(l)}(\omega) + \mathbf{P}^{(n)}(\omega). \quad (26)$$

Inserting Eq.(26) into Maxwell's equations (25), a wave equation is obtained:

$$\nabla^2 \mathbf{E}(\omega) + \frac{\omega^2}{c^2} \epsilon(\omega) \mathbf{E}(\omega) = -\frac{4\pi\omega^2}{c^2} \mathbf{P}^{(n)}, \quad (27)$$

where $P^{(1)}(\omega) = \chi^{(1)}(\omega) E(\omega)$ and the linear dielectric constant $\epsilon(\omega)$ is related to $\chi^{(1)}(\omega)$ by $\epsilon(\omega) = 1 + 4\pi \chi^{(1)}(\omega)$.

The wave equation can be solved, in principle, given a physical approximation to $P^{(n)}$. In general, $\mathbf{P}$ is a function of $\mathbf{E}$ and describes fully the response of the medium to the field. The equation connecting $\mathbf{E}$ and $\mathbf{P}$ is often known as the constitutive equation. Knowledge of the constitutive equation with appropriate boundary conditions would provide the solution for the resulting set of Maxwell's equations.

Physical approximations are often expressed in terms of the choice of constitutive equation. One approximation is the electric dipole approximation.¹⁵ In this approximation $\mathbf{P}$ is expressed in the frequency domain to second-order in the electric field as

$$\mathbf{P}(\omega) = \mathbf{P}^{(1)}(\omega) + \mathbf{P}^{(2)}(\omega). \quad (28)$$

where the constitutive equations for the first and second order terms are

$$\begin{aligned} \mathbf{P}^{(1)} &= \chi^{(1)} \cdot \mathbf{E}, \\ \mathbf{P}^{(2)} &= \chi^{(2)} : \mathbf{E}\mathbf{E} \end{aligned} \quad (29)$$

$\chi^{(1)}$ and $\chi^{(2)}$ are the linear and the second-order nonlinear susceptibility tensors. The susceptibility $\chi^{(2)}$ is a characteristic property of the medium and depends on the detailed electronic structure. Although quantum theory is needed to obtain the microscopic expressions for nonlinear susceptibility, to lowest order in the field, $P^{(2)}$ is dominated by the contribution from the surface region.¹⁸

A phenomenological model that treats the nonlinear response from the metal surface can be constructed from the free-electron gas theory. At excitation energies below that required for interband transitions, an isotropic electron gas has been used as a simple model of a metal, called the Drude-Lorentz model. In this approach the equation of motion for the electron is given by

$$\ddot{x} + \gamma \dot{x} + \omega_0^2 x + Dx^2 = \frac{eE_0}{2m} (e^{i\omega t} + e^{-i\omega t}), \quad (30)$$

where x is the electron position, mDx^2 is an anharmonic restoring force, γ is a damping term and $E_0 \cos \omega t$ is the driving electric field. The anharmonic potential for this model is given by³

$$V^{(\text{anh})}(x) = \frac{Dx^3}{3} \quad (31)$$

SH is radiated due to the motion of the electron at twice the driving frequency. Assuming a solution of the form

$$x = \frac{1}{2} (Ae^{i\omega t} + Be^{i2\omega t} + \text{c.c.}) \quad (32)$$

and keeping only the first and second order terms yields equations for A and B :

$$A = \frac{eE_0}{[m(\omega_0^2 - \omega^2) + i\omega\gamma]} \\ B = \frac{De^2 E_0^2}{2m^2 [(\omega_0^2 - \omega^2) + i\omega\gamma]^2 [(\omega_0^2 - \omega^2) + 2i\omega\gamma]}. \quad (33)$$

The nonlinear polarization $P(2\omega)$ is given as

$$P^{(2\omega)}(t) = N e x(t) \quad (34)$$

where N is the number of electrons per unit volume. The polarization at 2ω implies there is a collection of dipoles oscillating at 2ω . These oscillating dipoles are sources of radiation at the same frequency.

Another model used to treat the nonlinear source currents at a metal surface is the hydrodynamic model.¹⁹⁻²¹ In this model the electrons are no longer treated as noninteracting as in the free-electron model, and collective electron motions are considered. The effect of the breaking of the inversion symmetry at the surface-vacuum interface is modeled by a semi-infinite medium with a current sheet placed at the surface. To estimate the effects of real surface conditions on the size of the perpendicular and parallel source current, two phenomenological constants, a and b are introduced. The parameters a and b are on the order of unity, and serve as adjustable parameters. Suggested for this application by Rudnick and Stern in an early work,¹⁹ the hydrodynamic model has been applied in detail to the problem of second harmonic generation at a metal surface by Sipe and co-workers.^{20,21}

Once the form of the nonlinear polarization is established, the radiated SH fields can be found by the wave equation (27). The intensity of the reflected second harmonic signal from excitation by a plane wave of frequency ω and polarization $e(\omega)$ is given as³

$$I(2\omega) = \frac{32\pi^3 \omega^2 \sec^2 \theta_{2\omega}}{c^3 \epsilon(\omega) \sqrt{\epsilon(2\omega)}} \left| e_{2\omega} \cdot \chi_{s,eff}^{(2)} : e_{\omega} e_{\omega} \right|^2 I^2(\omega). \quad (35)$$

In this equation $\theta_{2\omega}$ represents the angle of the radiated SH light with respect to the surface normal, $I(\omega)$ is the pump intensity, and $e(2\omega)$ is the polarization at the SH frequency. The vectors $e(\omega)$ and $e(2\omega)$ are related to the unit polarization vectors $\hat{e}(\omega)$ and $\hat{e}(2\omega)$ inside the material by Fresnel coefficients. The effective surface nonlinear susceptibility $\chi_{s,eff}^{(2)}$ incorporates the

surface nonlinear susceptibility $\chi^{(2)}_s$ and the bulk magnetic dipole contributions to the nonlinearity, which are each of second order in the electric field. The result simplifies since, for isotropic media, there are only three nonzero independent elements of $\chi^{(2)}_s$. These are $\chi^{(2)}_{s,zii}$, $\chi^{(2)}_{s,iiz}$ and $\chi^{(2)}_{s,zzz}$ where $i=x,y$. In Eq.(35), the surface nonlinear susceptibility reflects the electronic properties of the surface. Consequently, SHG can be used to detect molecular monolayers adsorbed on metal surface,²² probe electronic transition and orientation of the adsorbate,²³ and study the dynamical interactions between the substrate and adsorbate which governs the chemisorption process.²⁴

A crude estimate of the relative contributions of the surface and the bulk to the second harmonic signal is approximately $(d/a)^2$, where d is the surface layer thickness and a is the size of the atoms or unit cells. This quantity is on the order of, or larger than unity. So it happens in many cases that the surface term actually dominates, and SHG becomes highly surface-specific. One may apply Eq. (35) to estimate how sensitive the technique. A typical pulsed laser used in surface work has an intensity of $I(\omega) \sim 10 \text{ MW/cm}^2$ at $1.06 \mu\text{m}$, a pulse width of 10 nsec, a beam cross-section is about 0.2 cm^2 , and an angle of incidence of $\theta \sim 45^\circ$. For study of adsorbates with non-linear surface susceptibilities $|\chi^{(2)}_{s,\text{eff}}|$ in the range of atomic values, 10^{-14} – 10^{-16} esu, one finds $I(2\omega)$ to be in the range 10^6 – 10^2 photons/pulse. Such a signal is easily detectable so that SHG should have submonolayer sensitivity.

EXPERIMENTAL RESULTS

In this chapter we present the results of our study of the initial phases of oxidation of the (111) crystalline face of aluminum. These techniques were used: Auger electron spectroscopy, contact potential difference measurements, and surface second harmonic generation. The sample was prepared prior to each measurements using the procedure described in the previous chapter.

Auger Electron Spectroscopy

Auger electron spectroscopy was used to monitor the adsorption of oxygen and the formation of the aluminum oxide. Many other workers have reported similar studies. Thus our measurements allow us to make explicit contact with previous studies when interpreting our other results.

Auger spectra were recorded in the $E(dN(E)/dE)$ mode at a primary beam energy of 3.0 kV. The sample was cleaned by Ar^+ sputtering and annealed as described in previous section and surface order was verified by low energy electron diffraction. We measured oxygen uptake using Auger spectroscopy at several oxygen exposure pressures: $(0.013, 0.05, 0.1, 0.11, 0.7, 1) \times 10^{-6}$ Torr.

Figure 15 shows a series of Auger spectra recorded for an oxygen pressure 1×10^{-7} torr at room temperature. The vertical axis is the Auger intensity in arbitrary units and the horizontal axis is the electron kinetic energy. Each curve is labeled in units of Langmuirs of oxygen exposure. One Langmuir (L) is an exposure of one second under pressure 10^{-6} Torr. Metallic aluminum has two dominant Auger electron kinetic energy peaks, one at 68 eV and the other at 1396 eV. When oxygen interacts with aluminum forming, e.g., Al_2O_3 , two additional peaks are present, one at 51 eV the other at 1378 eV.⁸ With increasing oxygen exposure, several points can be noted in Figure 15: First, there is a rapid decrease of the Al (68 eV) signal and an increase of the O (503 eV) peak; Second, the shape of the Al (68) peak at high oxygen coverage is somewhat different than that obtained at initial oxygen exposure; Third, no Al (54 eV) line could be resolved. This spectral line

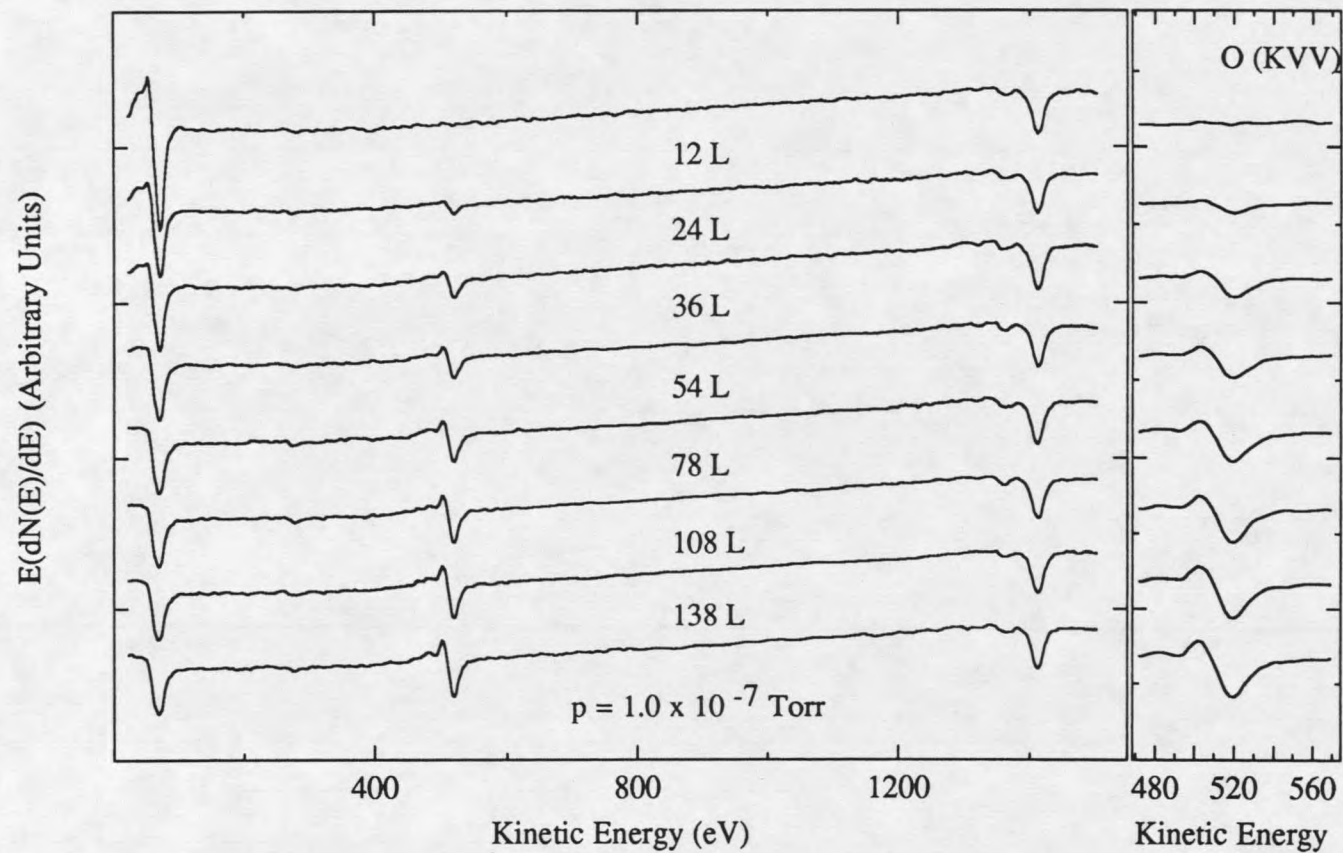


Figure 15. $O_2/Al(111)$ Auger uptake.

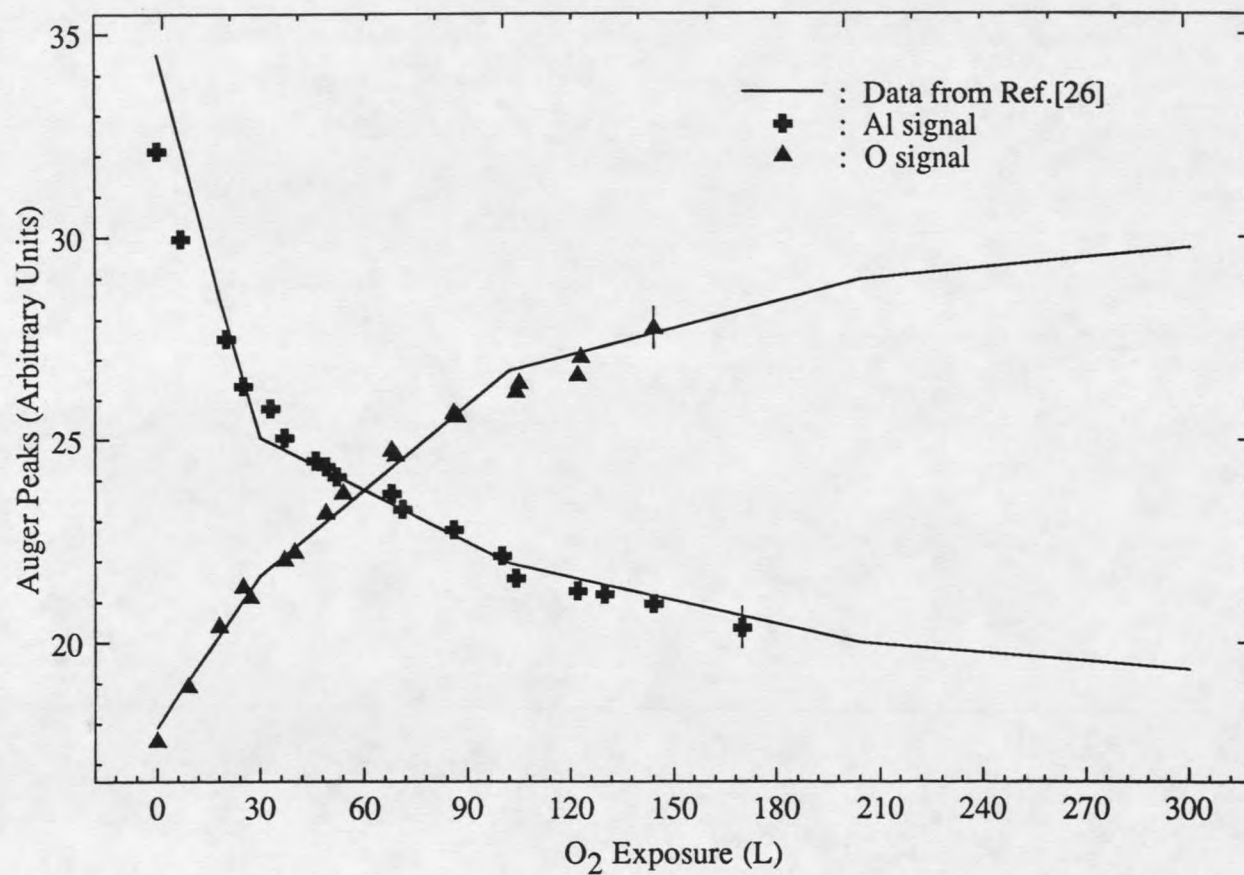


Figure 16. Auger oxygen uptake.

corresponds to oxide formation. However, this result can be explained as due to the low resolution mode we chose for our electron analyzer; Fourth, we observed a small shift in the O (503 eV) feature, which is consistent with the results of Bradshaw et al.²⁵ who observed that the O peak maximum shifts from about 506 eV to 504 eV over the range 0 - 50 L. We observed a ~ 2 eV shift over the range 0 - 138 L. Note that the energy scale in Figure 15 has not been adjusted for errors in the analyzer calibration.

Figure 16 displays the O (503 eV) KLL and Al (68 eV) LMM Auger peak heights versus oxygen exposure. Two independent measurement sequences are represented with the sample held at room temperature. Due to the complex line shape of the 68 eV peak, however, the signal intensity was estimated using the size of the negative dip compared to the background signal at a slightly higher kinetic energy. There is a smooth decay of the Al (68eV) signal (triangles) and a complementary build up of the O (503 eV) peak (squares) with increasing oxygen exposure. The solid curve was digitized from the work of Soria et al.²⁶ Our data agrees with that of Soria et al. within the error of the measurement.

Contact Potential Difference Measurements

Contact potential difference measurements were obtained via the Kelvin vibrating capacitor method using a molybdenum reference electrode. The sample was prepared in the same way as described above. The oxygen exposure was produced by a leak valve connected to the oxygen source. All the measurements were performed at room temperature. Measurements were made with different oxygen exposure pressures: $[1, 0.5, 0.3, 0.1] \times 10^{-6}$ Torr.

Considerable variation was obtained in the work function change ($\Delta\Phi$) during first 50 L oxygen exposure. In most cases the $\Delta\Phi$ increased to reach a maximum and then dropped down, cf. Figure 17. However it was also observed in some cases that the $\Delta\Phi$ decreased first, then increased to maximum, and finally decreased at high oxygen coverage. After about 50 L, all the

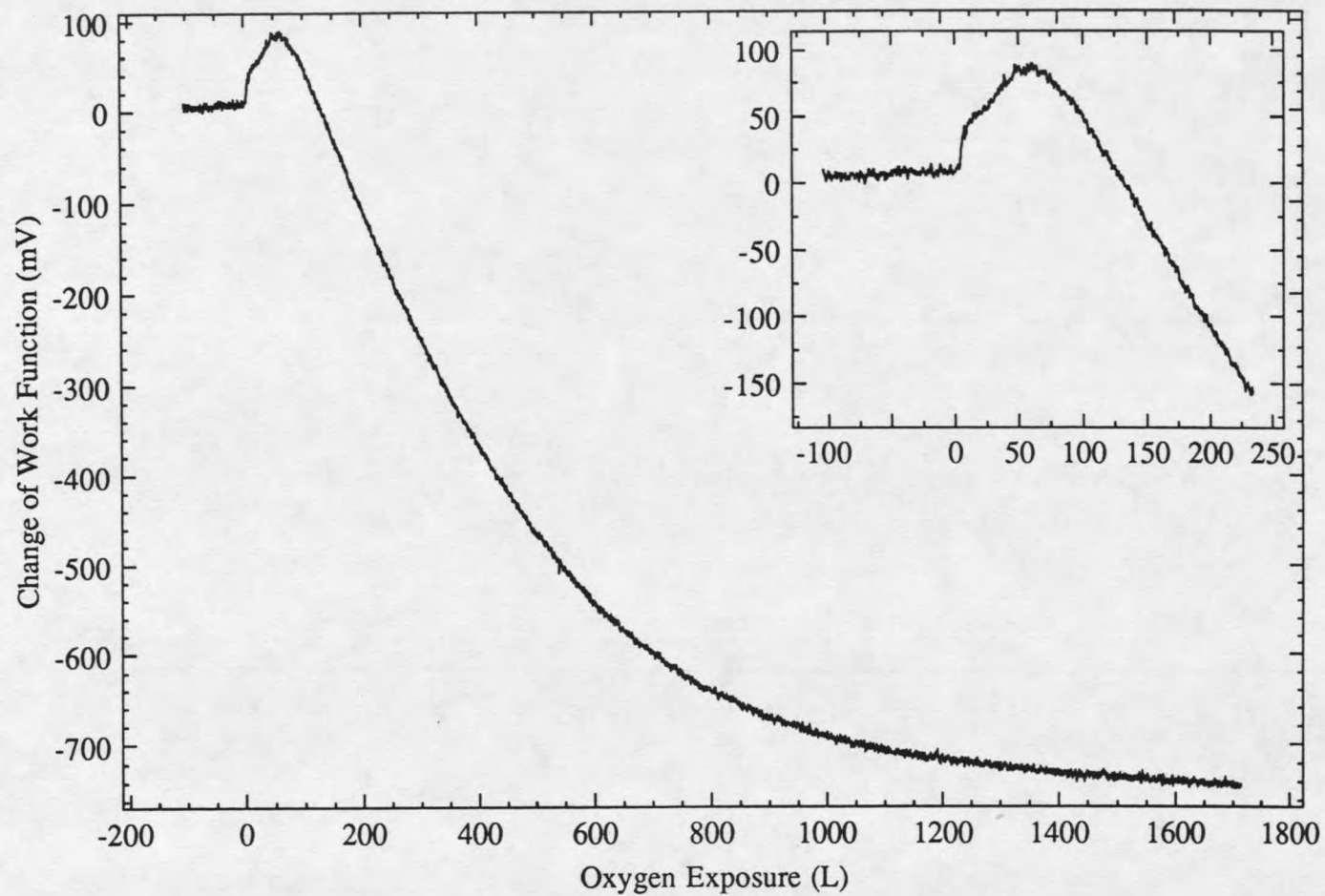


Figure 17. Kelvin probe result.

measurements showed the same property: that $\Delta\Phi$ decreased to a value of approximate 800 eV for high oxygen exposure (1000 - 2000 L). One explanation for the variation in results is that variations in experimental conditions exist. For instance, the $\Delta\Phi$ measurements taken at 10^{-8} Torr differ from those obtained for the same exposure at 10^{-7} Torr. This pressure dependence was similar to that observed by P. Hofmann et al.²⁷ It has previously been observed that the sign and the magnitude of the work function change depends on the conditions of surface preparation, the temperature, and other experimental parameters.²⁷⁻³⁰

We eventually arrived at a prescription for obtaining reproducible results for different exposure pressures. Figure 17 depicts the typical change in work function with oxygen exposure at 1×10^{-7} Torr. Because this is a convenient experimental choice of pressure, most work reported in the literature has been done at this pressure. The curve in Figure 17 was recorded from original experiment data by calibrating the exposure to Langmuirs. Two main points should be noted in the figure. First, the maximum positive $\Delta\Phi$ was about 80 meV. Although this value was dependent on the experimental condition it occurred reproducibly at about 50 L, presumably close to monolayer coverage. Second, after the ~ 50 L point the work function falls gradually toward saturation. The total change of work function at saturation, ~ 1000 L, was reproducibly about 720 mV.

Figure 18 shows that a kinetic effect can also be observed in the work function measurements. During the measurement, stopping the oxygen exposure at any point beyond ~ 50 L resulted in a increase of the net work function change; i.e., a further decrease of the work function. In Figure 18, the oxygen supply was stopped after 310 L of exposure for about 10 minutes and then restored to the original pressure of 1×10^{-7} Torr. The data shows an immediate reduction in work function. This effect was also observed by Hofmann et al.²⁷

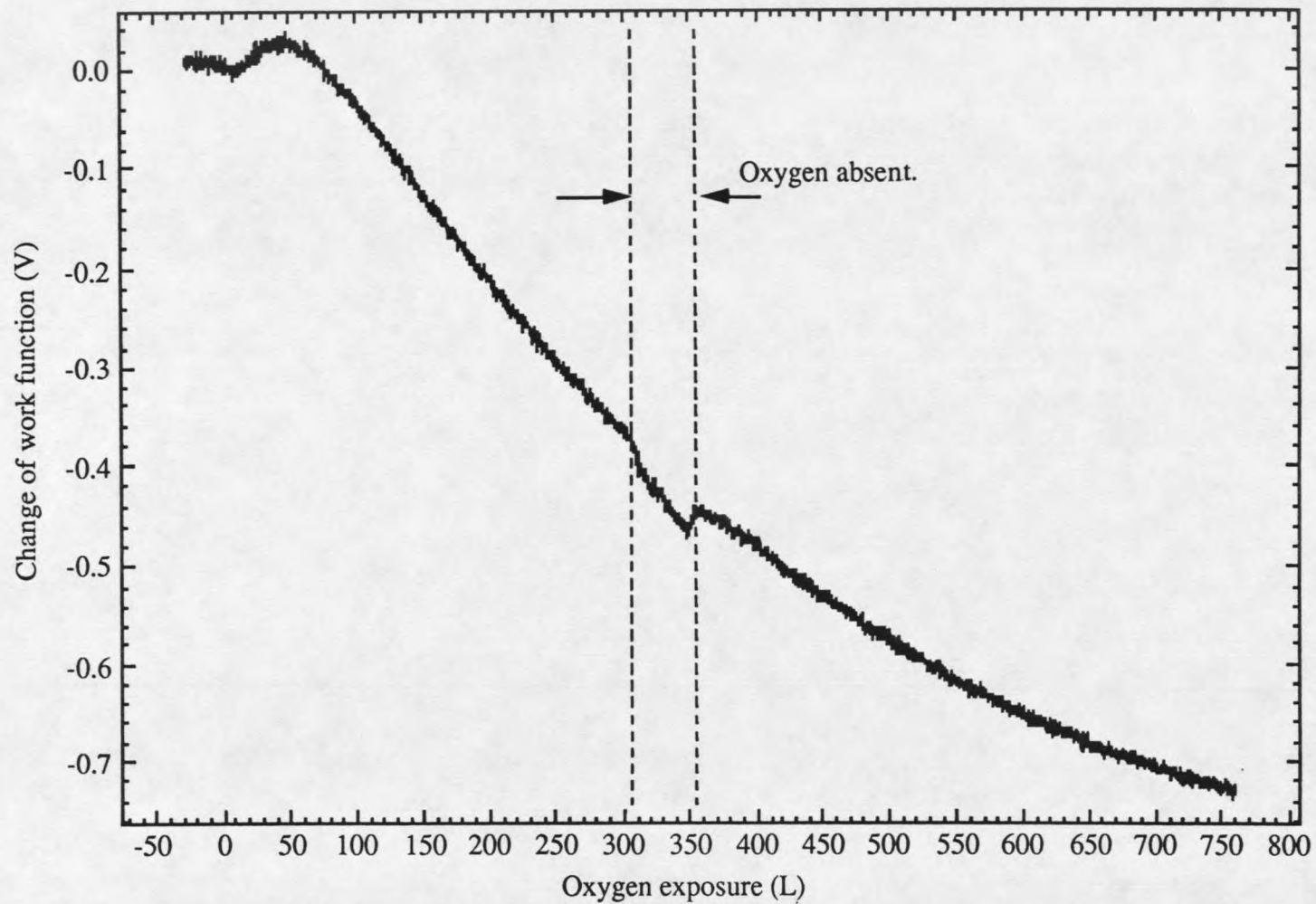


Figure 18: Stop oxygen exposure the work function response.

Surface Second Harmonic Generation Measurements

Surface second harmonic generation measurements were performed in the same experimental chamber and under the same condition as the previous measurements. The sample was cleaned by Ar^+ ion sputtering, annealed at 425 C and slowly cooled down to room temperature. Sharp (1 $\times$ 1) LEED patterns were observed each time, indicating that the surface was well ordered. Five exposures at room temperature with an exposure pressure of 1×10^{-7} Torr oxygen were made, yielding reproducible results. Other trials using different pressures gave behavior similar to the 1×10^{-7} Torr case.

The SH signal from the sample was normalized by a reference signal as described in the previous chapter. The output was plotted as the ratio of sample signal to the reference signal vs the oxygen exposure. Performing this division was important to eliminate the effect of fluctuations of laser intensity. Figure 19 displays the normalized second harmonic (SH) signal intensity vs O_2 exposure at a pressure 1×10^{-7} Torr for the clean surface. It represents the SH intensity measured using p-polarized light. In this geometry a component of the incident photon electric field vector lies in the incident plane and perpendicular to the sample surface. Figure 20 shows the same data except that a s-polarized geometry was used. In this case, the field vector is perpendicular to the incident plane and in the plane of the sample. Figure 21 is data obtained using p-polarized light but with a roughened surface, prepared by argon ion sputtering with four to ten microamps for thirty minutes but not followed by the customary 425 C annealing. About 10^{15} - 10^{16} argon ions per square centimeter hit the sample surface, or approximately 10 - 20 Argon ions per surface unit cell.

The p- and s-polarized measurements from a properly annealed surface in Figure 19 and 20 have some similarities. Upon initial oxygen exposure the SH intensity drops to a minimum at ~ 5 L. This is a fall-off of 18 $\sim$ 25% compared to the clean surface SHG signal level. It then rises rapidly up to about 15 L. At that point the rate of change slows down and a maximum SHG signal appears at ~ 50 L. At the maximum point, the SHG signal has increased by approximately 100%

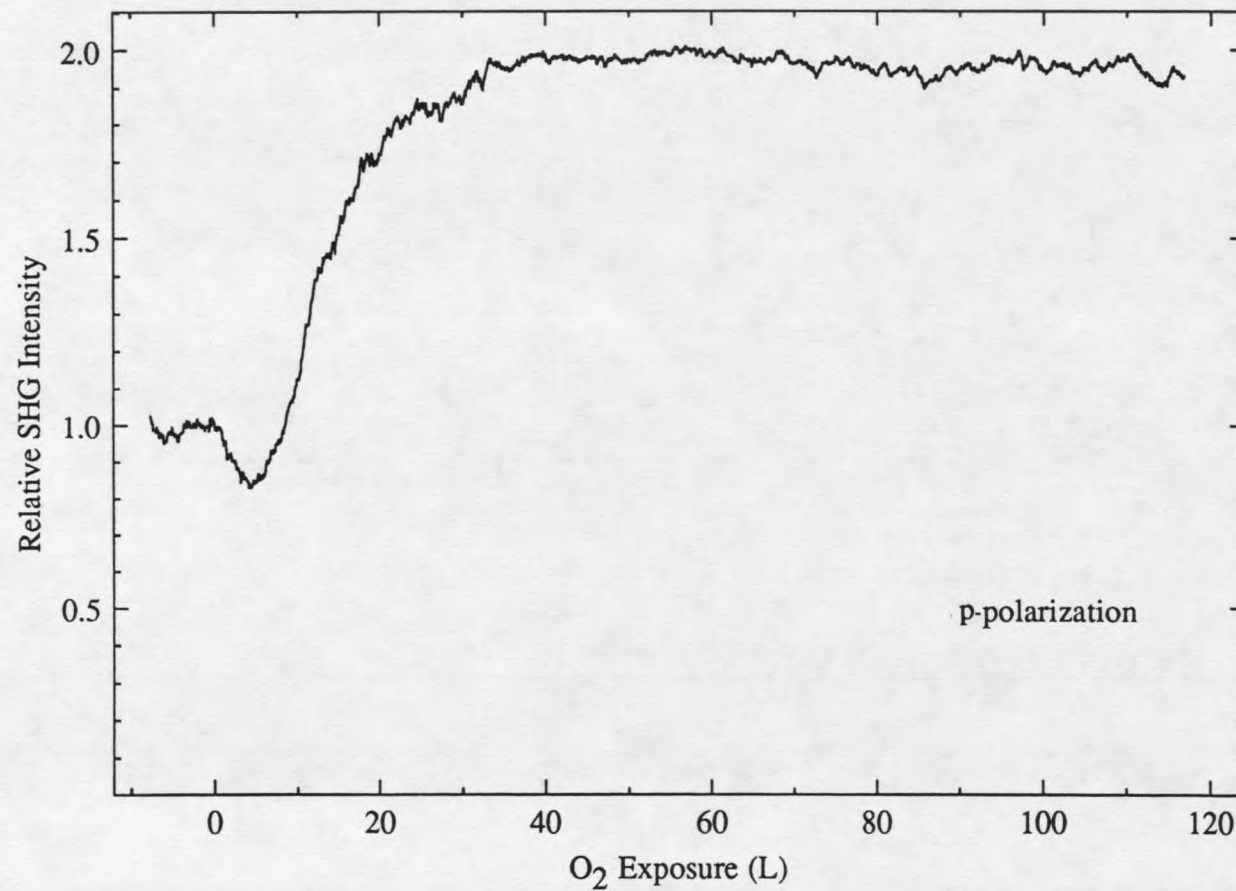


Figure 19. SHG results p-polarized incident.

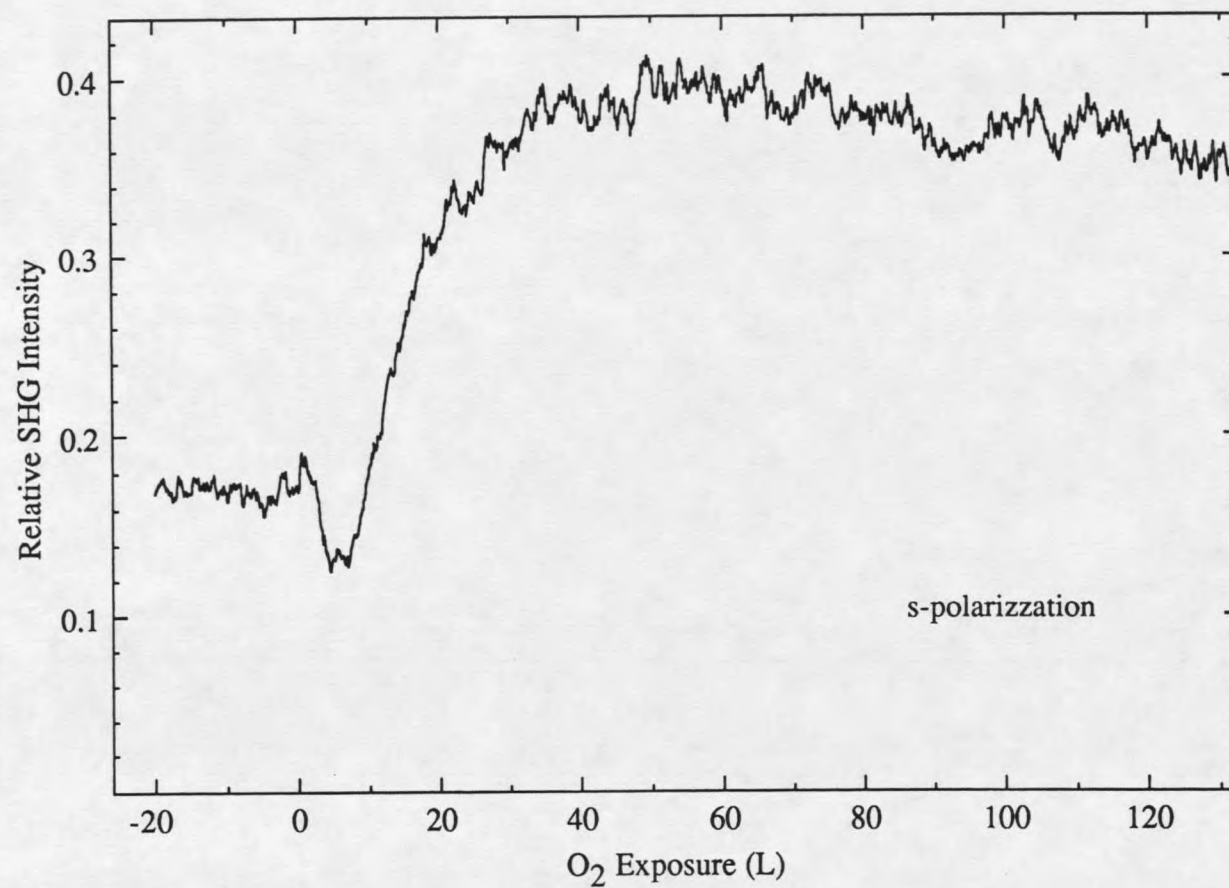


Figure 20. SHG results s-polarized incident.

compared to the clean surface signal level. At higher exposures, the SH intensity falls off by 10% and 4% of the maximum signal up to 120 L exposure for s- and p-polarized data respectively, with the s-polarized data dropping slightly more rapidly than the p-polarized data. The s-polarized data has a weaker signal level.

For the disordered surface in Figure 21, no initial dip at ~ 5 L oxygen exposure was observed. The SHG signal just increased as function of exposure. At about 15 L the rate of increase slows down but there is no maximum point observed at 50 L. Up to 120 L exposure, the signal increase is about 90% of that for the clean surface, and there no fall-off was observed for up to 240 L.

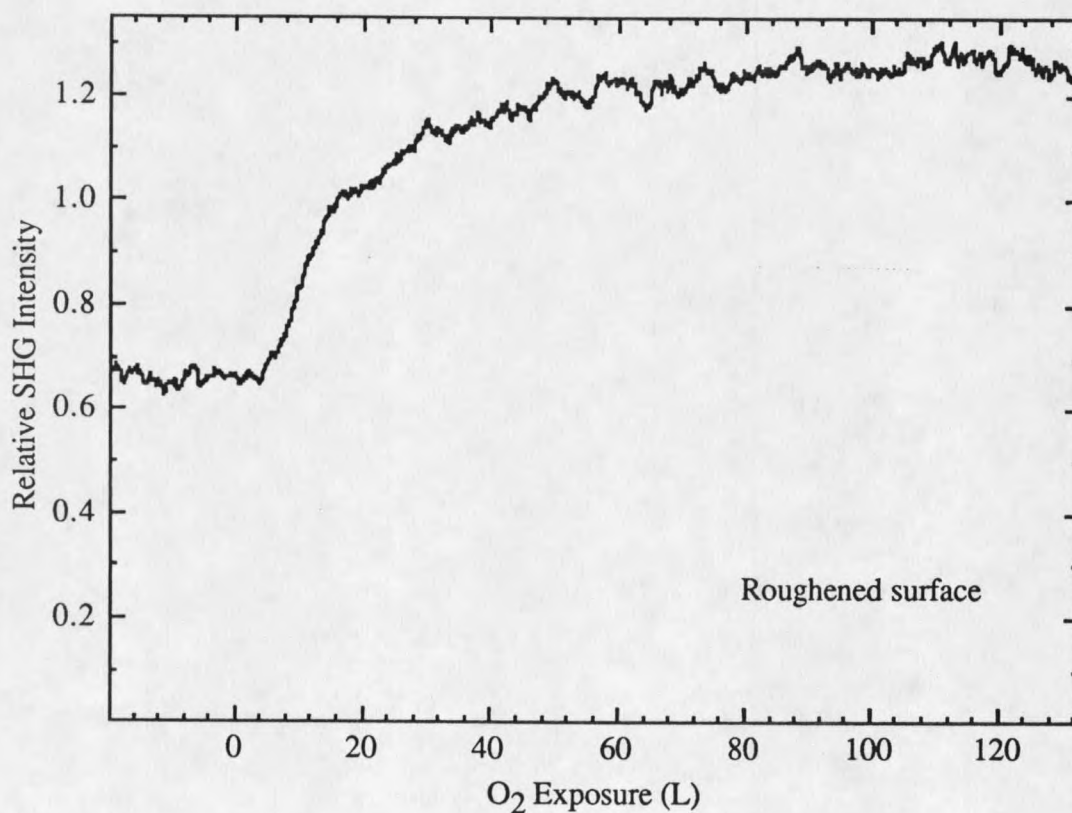


Figure 21. SHG results for sputtering surface.

DISCUSSION

Oxidation of Al (111)

The interaction of oxygen with the aluminum (111) surface has been studied extensively using a wide variety of surface sensitive techniques. Batra and Kleinman have critically reviewed this literature.² There are currently two models of the oxidation process. The first is the ordered overlayer formation model, which asserts that oxidation occurs in distinct phases.²⁸⁻³³ In this model the oxygen atoms bond in both surface and subsurface oxygen sites prior to the formation of Al_2O_3 . Initially, only the surface chemisorption sites are occupied until a certain coverage is attained, at which point subsurface oxygen binding sites begin to be populated. At even higher coverage, Al_2O_3 forms. The second model is that of random overlayer formation. It asserts that both surface and subsurface sites are populated simultaneously.³⁴⁻³⁷ An increase of exposure or a change in temperature only alters their relative occupancy. In both models, the surface bonded oxygen atoms occupy a threefold hollow site in ordered islands and the subsurface oxygen atoms occupy threefold sites below the top layer of the sample.

Many experiments have been done which suggest that the ordered overlayer formation model correctly describes the chemisorption process. There is substantial evidence that oxidation occurs in distinct stages despite the uncertainty about how the bonding sites are occupied by the oxygen atoms. However, a recent high resolution electron energy loss spectroscopy study has given evidence in favor of the random overlayer formation.³⁸ Despite the uncertainty about how the bonding sites are occupied by the oxygen atoms, there is substantial evidence that oxidation occurs in distinct stages. In order to address this issue one must understand when and at what oxygen coverage the surface phase ends and the subsurface phase begins.

Auger electron spectroscopy has been used to study the adsorption kinetics of oxygen on Al(111) and the results support the first model. The plots of Auger peak height versus oxygen exposure in Figure 16, show changes in slope at ~ 30 L and 110 L for both the aluminum 68 eV

and oxygen 510 eV peaks. The Auger spectroscopic model^{9,39} suggests that the substrate is incorporated in layers during the adsorption. The kinks in the slopes indicate the breaks between different adsorbed layers. During the formation of a monolayer the Auger intensity changes linearly. The general features of our own AES data can be partially explained with this model. For instance, the Al 68 eV Auger peak showed a rapid attenuation during the initial 30 L exposure. The slope of this attenuation then decreases and saturation follows. The oxygen Auger peak increases with initial exposure and saturates at high exposure. This behaviour indicates that different chemisorption sites have been occupied sequentially and provides evidence of a multiple phase oxide growth. Although our data was not collected with high resolution and did not convincingly display the kinks observed in Ref. [26], our data serves to demonstrate that our experiments are at least qualitatively consistent with other studies including those presented in the following paragraphs.

The majority of the Auger electron spectroscopy data indicate that the oxide grows in several stages, i.e., a chemisorbed surface phase, a chemisorbed subsurface phase, and finally an oxidized phase. The exposure dependence of the Auger peak heights in the exposure range between 0 L and 50 L is qualitatively different from that of higher coverages. However, the actual coverages corresponding to these chemisorption phases has generally not been well determined, with a large discrepancy still remaining in the literature. From their data, Soria et al.²⁶ suggested that there should be four steps in the oxide development. Below 30 L exposure they detected a surface phase during which the oxygen is thought to occupy the three-fold hollow subsurface sites of the (111) surface, with the surface top layer expanded by about 10%. Between 30 L and 100 L, subsurface phases with different interlayer spacings were found. They observed a kink in their AES curve at 30 L and again at 100 L exposure. Within this interval chemisorption continues. Their conclusion is that by 100 L the chemisorbed species have completed a monolayer and occupy the three-fold surface sites in a 1×1 overlayer. Beyond 100 L the aluminum oxide 55 eV AES line appears. The formation of γ -Al₂O₃ was thought to occur after 200 L exposure, leading to a third

kink in the AES oxygen uptake curves. The piecewise linear fit of their Auger data is presented above in Figure 16.

Contrary to the result of Soria et al., Michel et al.⁶ found that the kinks in the Auger uptake curves occur at 54 L and 120 L in both the aluminum 68 eV and oxygen 510 eV Auger peaks. Furthermore, an oxidized aluminum 54 eV peak occurs at 54 L with a slope change at 120 L. The authors concluded that the completion of an overlayer phase and the beginning of a subsurface phase occurs at 54 L. The subsurface phase is completed at 120 L. They inferred that a monolayer was reached at 54 L exposure, not the 100 L value reported by Soria et al.²⁶ Chen et al.³⁶ observed a change in oxygen uptake rate at 15 L but no other changes up to 120 L, which was the extent of their measurements. Testoni et al.⁷ observed no AES kink over the range from 0 L to 100 L on their annealed surface.

Work function measurements may be used to infer the positions of oxygen atoms on the (111) surface. The basic idea discussed in the preceding chapter is that an electronegative species lying above the surface should introduce an electric dipole with the positive end pointing into the surface. This surface dipole should act to increase the work function because an additional potential is produced by the dipole layer to shield the electrons inside the material. If instead, the oxygen is incorporated below the surface, this electrostatic dipole would be oppositely directed and the work function should decrease.

One would expect that if the oxide grows as a single phase the work function change would be a monotonic function of overlayer coverage. Conversely, if a maximum or minimum is observed a multiple phase growth must occur. Our own Kelvin probe data is consistent with the conclusion that there are two phases. We found that chemisorption initially produced an increase in work function which reached a maximum of approximately +100 mV at 50 L exposure, followed by a decrease, and then saturation after about 1700 L oxygen exposure. The saturation value was about 900 mV lower than the work function of the clean surface. The increase in $\Delta\phi$ is ascribed to the negative outwards dipole caused by the adsorption of oxygen atoms outside the surface.⁴⁰ $\Delta\phi$

reached a maximum at about 50 L, which can be interpreted as the completion of an oxygen monolayer on the substrate. After 50 L exposure, the reduction in work function is usually attributed to the penetration (incorporation) of oxygen atoms into the immediate subsurface region. From the Kelvin probe data, we can draw the following conclusions: between 0 L to 50 L the oxygen adsorbs on the substrate in the overlayer surface phase with a monolayer forming at 50 L oxygen exposure. After 50 L the subsurface chemisorption phase occurs. This leads to the picture that the oxygen first adsorbs into sites on top of the surface and then migrates to subsurface sites.

A variety of groups have produced various inconsistent results for the work function change. Eberhardt and Himpsel⁴¹ measured a 0.10 ± 0.05 eV increase in work function upon exposure to 24 L of O₂. By using the Fowler photocurrent measurement technique, Gartland et al.⁴² observed no change in the work function for 20-30 L O₂ exposure, after which $\Delta\phi$ rose to a plateau of about +0.13 eV at 100 L with no further change up to 250 L. Bradshaw et al.²⁵ using a Kelvin probe, initially found a reduction in $\Delta\phi$ by 0.2 eV at 50 L followed by a saturation plateau. However, later those same workers⁴³ reported for the same system an increase in $\Delta\phi$ after the initial 0.2 eV decrease toward a new plateau that begins at 150 L with a net $\Delta\phi$ of approximately -0.15 eV. Michel et al.⁶ found that the work function measured by a Kelvin probe decreased by 0.033 eV after 54 L exposure, decreased to 0.25 eV after another 125 L exposure, and then remained constant for the next 200 L exposure. Ramsey⁴⁴ observed an immediate increase in $\Delta\phi$ to +0.1 eV at 60 L followed by a gradual fall ultimately taking $\Delta\phi$ to negative values beyond 1000 L. Even though the $\Delta\phi$ measurements of different groups are contradictory in detail, they are all consistent with there being at least two phases for oxygen adsorption.²

As we mentioned in the previous chapter, various measurements give inconsistent results during first 50 L oxygen exposure. The change in CPD has been found to be both positive and negative at the onset of exposure. The most likely explanation for the inconsistencies between published results and our measurements lies in the differences in surface preparation as well as difference in the working oxygen pressures. To illustrate this dependence, Testoni and Stair⁷

showed, using Auger electron spectroscopy and LEED, that the initial oxidation rates can be significantly affected by surface faceting. The faceted surface was produced by rapidly heating the planar (111) surface to 500 C. The planar surface first showed oxide-like aluminum at 50 L, whereas the faceted surface first showed the oxide at less than 10 L. Since work function changes should reflect the initial oxidation conditions it is clear that faceting could affect the work function measurements. Such faceting can occur, for example, when sputtered surfaces are annealed too quickly. It is possible that the inconsistencies in the experimental results exhibited from author to author is the result of differences in both sample preparation and exposure methods.

Another interesting feature in our work function measurements indicate that oxygen adsorbs on the Al (111) surface in a kinetically balanced process which involves both in-diffusion and surface adsorption. This is evidenced by the fact that the uptake of oxygen does not stop at saturation surface coverage. We observed this experimentally by noting a decrease in the work function upon removal of oxygen exposure after saturation had occurred. In addition, Figure 18 shows that when the oxygen supply was interrupted at some point after 50 L exposure, (at which we believe a monolayer is completed), an additional small decrease of 10-20 meV occurred during the 9 minutes that the oxygen was absent. This immediately shows that the change in work function is both exposure-rate dependent and time-dependent, indicating the kinetic nature of the oxidation process. This result is in good agreement with Ref. [43] which also showed that the work function is exposure-rate and time dependent.

Our second harmonic generation data also provides information about the oxidation process. Currently the surface SHG technique is in an early stage of development. Essential to its progress is the development of a clearer understanding and theoretical description of the source of the nonlinear polarizability as it relates to microscopic properties of the surface components. We will not attempt to provide a detailed theoretical explanation here but rather provide a simple physical interpretation of our second harmonic data. Figures 19 and 20 show the general features of the second harmonic intensity as a function of oxygen exposure for both s and p-polarized

incident beam. Both polarizations exhibit an initial decrease in the SHG signal during the first 5 L exposure, followed by an increase to a maximum at about 50 L exposure. After 50 L exposure the signal steadily decreases to the extent of our experimental coverages. These SH intensity versus oxygen exposure measurements were also performed for a disordered surface. The disordered surface was prepared by sputtering with Argon ions at a beam current of 4 μA for 30 minutes without subsequent annealing. The results from the disordered surface differed from those of the ordered surface in that there was no initial signal decrease. In addition, for coverages beyond 50 L, the signal did not decrease as rapidly as that of the ordered surface. Our interpretation of the Second Harmonic data can be summarized as following. The initial decline of the signal to a minimum indicates the incipient oxidation of the surface which we will discuss in detail later. The subsequent increase in SH intensity can be interpreted as the oxygen being adsorbed on the Al(111) surface to form an electric dipole layer, with the maximum at 50 L corresponding to the completion of one monolayer. The following steady decline is then attributed to the penetration of the oxygen into the subsurface region. The SHG data and this interpretation then supports the conclusion of ordered overlayer formation.

The results of other experiments such as ultraviolet photoemission (UPS),^{40,41,43,45-47} and extended X-ray adsorption fine structure (EXAFS)^{46,48,49} also support the ordered overlayer formation model. UPS data shows that at $\sim 50\text{L}$ of oxygen exposure (at room temperature) only an Al 2p peak shifted 1.4 eV toward higher binding energy is present. When higher exposures of 100 L to 200 L are reached a new Al 2p core level feature appears which is chemically shifted 2.7 eV toward higher binding energy relative to the clean-surface Al 2p peak. This 2.7 eV shift is the same as that found for bulk Al_2O_3 . In addition, these shifts suggest that oxidation occurs in stages at room temperature. For low exposures of less than 50 L, only a surface chemisorbed phase exists, while at higher exposures of more than 125 L phases having bulk-like oxide properties exists. At intermediate exposures of about 100L the two phases can coexist,⁴³ with a monolayer forming between 50 L - 125 L. UPS data also showed that when a sample which had been exposed to $\sim$

50L of oxygen was heated to 273 C for 10 min, the Al 2p peak (which was shifted 1.4 eV toward higher binding energy) was replaced by the Al 2p feature shifted by 2.7 eV.⁴⁶ Angle-resolved photoemission from the oxygen p bands on the Al(111) surface has also been observed by three groups.^{40,41,47} Two of the groups^{40,41} concluded that when the sample was exposed to 24 L oxygen and then heated, the surface phase was converted to the subsurface phase. Hofmann et al.⁴⁰ concluded that in the 4 L - 30 L oxygen exposure range, the surface phase occurred and a monolayer was completed at about 30 L after which the subsurface phase appeared. The EXAFS studies^{46,48,49} combined with photoemission measurements have been used to determine that for exposures between 50 L - 150 L, the O-Al separation remains unchanged at $R=1.76 \text{ \AA}$ which is considerably less than those values found for oxidized Al(111) or bulk $\alpha\text{-Al}_2\text{O}_3$, which have O-Al separations of $1.88 \text{ \AA}$ and $1.91 \text{ \AA}$, respectively.⁴⁶ These results also suggest that the surface phase occurs first. Then when exposed to 50 L of oxygen and heated to 200 C, the subsurface phase with a 2.7 eV chemical shift and an oxide-like phase coexist.

High-Resolution Electron Energy Loss Spectroscopy (EELS) seems to be the only experimental technique which supports the assertion of the second model.^{35,36-38,50,51} The EELS data suggests that at low oxygen coverages a mixed phase exists with oxygen atoms adsorbing in both surface and subsurface three-fold hollow sites. Three of these studies^{38,50,51} showed that both surface and subsurface phases coexist, with only the relative proportions varying with exposure and temperature. Low coverage tended to favor the surface sites, while heating increased the subsurface-to-surface oxygen ratio. Higher coverage, at any temperature, yielded a higher concentration of subsurface oxygen atoms. Two other papers^{36,37} concluded that oxygen dissociatively adsorbed and occupied both surface and subsurface binding sites under all exposure conditions in the temperature range of -151 C to +427 C. These studies also concluded that the surface phase was preferentially occupied at low exposures, while higher exposures (> 50 L) increasingly favor population of subsurface phase. In addition, it was found that at temperatures as

low as -142 C formation of Al_2O_3 occurred at oxygen exposures above 15 L, while at room temperature the formation of Al_2O_3 began at 50 L.³⁵

In summary, our Auger Electron Spectroscopy, Kelvin Probe, and second harmonic generation data are in good agreement with the conclusion that multiple phases exist for oxygen adsorption on the Al(111) surface.

Incipient Oxidation Process

In the previous section we discussed the existence of multiple oxide growth phases for oxygen adsorption on the Al(111) surface. In this section, we will describe more fully the incipient oxidation process for coverages of less than one monolayer. Specifically, we will discuss the type of bonding between the oxygen and aluminum atoms when oxygen atoms first begin to adsorb onto the substrate.

Many experiments have been done to elucidate the initial stages of oxygen adsorption on the Al(111) surface. However, it is in this initial range that the experimental results are most inconsistent, to the extent that there is not an agreed-upon model for adsorption up to 30 L exposure. Auger electron spectroscopy data characterizes this stage by a rapid attenuation of the 68 eV metallic Al Auger peak accompanied by an equally rapid increase in the 506 eV oxygen peak. These features were observed by Soria et al.²⁶ for the exposure range from 0 L - 2 L with a break in the attenuation curve at 30 L exposure. They concluded that this strong attenuation is due to surface variations in the structure of the valence band. Using AES and LEED Testoni et al.⁶ measured the differences in oxygen adsorption behavior between the planar surface and the faceted surface during the first 10 L exposure. It is of interest that the metallic Al Auger peak increased slightly in the first 10 L for the planar surface, but decreased monotonically for the faceted surface. The planar surface first showed growth of the 55 eV Auger peak at 50 L, while the faceted surface first showed the oxide at < 10 L exposure. They interpreted this as being due to an increased

oxygen sticking coefficient because of the greater number of steps in the faceted surface. The incipient oxidation studies using EELS experiments³⁶ showed that oxygen dissociatively chemisorbs on the Al(111) occupying both surface and subsurface binding sites even at the low oxygen exposure of <10 L with surface sites preferentially occupied. Our AES data was similar to that of Soria et al., in that we also observed the rapid attenuation of the metallic Al peak and fast increase in the oxygen peak for the initial exposures. Unfortunately due to the low resolution of this data, we were unable to make any further detailed observations.

Perhaps more interesting than our Auger data, were the Kelvin Probe work function measurements. This data showed that the work function increased abruptly by ~ 40 meV in the range 0 L - 5 L and then increased more slowly to a maximum of ~ 80 meV at about 50 L. This increase in the work function is in agreement with the theoretical calculation by Kleinman et al.⁵² Using a self-consistent calculation, they considered a six-layer Al film with a (1×1) ordered oxygen overlayer on the surface in the hollow threefold sites and determined that the work function should be greater than for the clean surface.

While AES, LEED, and work function measurements have provided useful information about the incipient oxidation of Al(111), this process is still far from being well understood. In order to obtain a better understanding of the oxidation process during these early coverages, it is necessary to gain insight into the the electronic structure of the oxygen-aluminum bonding occurring in the surface region. It is our contention that our second harmonic generation results have provided just such information about the oxygen-aluminum bonding. Specifically, we observed a conversion between the metallic bonding and ionic bonding of the oxygen and aluminum atoms. However, before discussing our experimental results, it is necessary to examine some basic concepts concerning the second harmonic generation technique as a surface sensitive experimental probe.

Numerous experiments have sought to identify and understand the origin of second harmonic generation from a clean surface in vacuum as well as the effect of the adsorbate on this

response. The results of these investigations of clean and dosed metals were summarized in Shen's review paper.¹⁵ To understand SHG as a chemisorption probe for investigation of incipient oxidation of the Al(111) surface, we analyzed two models dealing with the adsorbate-substrate interaction.

The first model is the metallic bonding of adsorbate and substrate atoms. Tom et al.²⁴ monitored $I(2\omega)$ from Rh(111) exposed to O_2 and CO. Experiments showed that the electronegative oxygen atoms and carbon monoxide molecules monotonically reduce the SH signal. In that experiment the oxygen did not form an oxide but merely saturated the surface as chemisorbed oxygen atoms. It is presumed that the chemisorption bond of the molecules to the surface depleted the free electrons of the metal thus reducing the nonlinear electronic polarizability of the surface which is proportional to the number of free electrons near the surface. Therefore, since the second harmonic intensity is proportional to this polarizability, it should decrease monotonically from the clean surface value to some final value as a function of oxygen exposure. This is all consistent with the knowledge that O_2 or CO, as electronegative atoms, adsorb strongly to the substrate, having a large sticking coefficient, linearly reducing the number of free electrons near the Fermi level in surface region, thus decreasing the SH signal accordingly. Similar results have also been observed in which the SH intensity is reduced due to the presence of an adsorbate on a metal surface for O_2 on Ag(110),^{53,54} CO on Cu(111),⁵⁵ CO,^{55,56} H_2 ⁵⁶ and O_2 ⁵⁶ on Ni(111), as well as pyridine and benzene on Rh(111).⁵⁷

The second model for surface enhanced second harmonic generation is the electronic dipole layer model in which the formation of a dipole layer increases the SH signal from an originally clean metal surface. Such a change in SH intensity resulting from the appearance and disappearance of a dipole layer has been observed experimentally. C. K. Chen et al. observed this for the adsorption of AgCl on a silver electrode surface.²² Their data showed that at the start of an electrolytic cycle (when the electrolytic current began) the SH signal rose rapidly and then fell rapidly at the end of the cycle. This result suggests that the SH signal originated mainly from a few

layers of AgCl which formed at the silver surface as a result of charge transfer. This Ag-Cl dipole layer increased the surface nonlinear polarizability thus enhancing the SH signal.

Oxygen on Al (111) is perhaps the first system studied that requires the use of both two models. This can be understood by the fact that the oxygen not only adsorbs onto the substrate forming metallic oxygen-aluminum surface bonds, but also in-diffuses forming a different phase which incorporates ionic bonding like that seen in Al_2O_3 . Since the interaction between the adsorbate and substrate will play an important role in the surface nonlinearity, these different oxidation phases should be reflected in the Second Harmonic signal. This was in fact seen experimentally in the data shown in Figures 19 and 20. Initially the SH signal dropped to a minimum at an exposure of 5 L. This corresponds to the metallic bonding phase. Basically, oxygen atoms on the Al surface act as electron acceptors which tend to localize the nearly free electrons of the aluminum surface. This localization of the surface charge density has the effect of reducing the number of free electrons near the Fermi level at the aluminum surface. Now recall that the second harmonic intensity is proportional to the surface nonlinear susceptibility $\chi^{(2)}$ which in turn is dependent upon the number of the free electrons near the Fermi level. The reduction in the number of free electrons at the surface is therefore manifested as a reduction in the observed second harmonic intensity. It also interesting to note that the 5 L minimum occurs at an exposure of less than 0.1% of the saturation coverage. This gives evidence of the extreme surface sensitivity of the SHG technique.

Between 5 L and 15 L, the second harmonic intensity abruptly increases. This is the second phase, during which the oxygen begins to in-diffuse forming ionic bonds with the aluminum like those found in Al_2O_3 . Thus the surface become a system of electrostatic dipole layer. When the dipole moment is driven by the oscillating electric field of laser beam it will display some sort of anharmonicity and thus will radiate second harmonic radiation. As described by the second model, the nonlinear polarization $P^{(n)}$ is proportional to the electric dipole moment.³ This transition to ionic bonding causes the formation of an electrostatic dipole layer which increases

the net dipole moment resulting in a larger second harmonic signal. As the number of surface dipoles grows with increased oxygen exposure the SH signal also increases as demonstrated in Figures 19 and 20.

One final feature of interest in the SHG data is an apparent change in slope at ~ 15 L, at this point the rate of increase in the signal is diminished. We associate this break with the onset of Al-O bonding as described in the HREELS-AES study by Chen et al.³⁶ At this exposure those authors believe that the surface not only has surface and subsurface chemisorbed species but that Al_2O_3 clustering begins. The maximum in the SH data at about 50 L exposure, is believed to correspond to saturated surface coverage, normally associated with the completion of one monolayer. Beyond exposures of 50 L the second harmonic signal is relatively stable because further surface adsorption does not seem probable. That is, while SHG is surface sensitive it is not bulk sensitive, and further changes in the substrate conditions should produce a relatively modest effect in the SH data. The slight decline in SH intensity beyond 50 L is expected because the continual in-diffusion of oxygen reduces the relative surface-bulk asymmetry. Since this asymmetry is reflected in the nonlinear polarizability, its decrease is seen as a reduction in the observed SH intensity.

Thus far our analysis has only been for an initially ordered surface. As mentioned previously the data for the disordered surface shown in Figure 21 differed from that of the ordered surface for early exposures in that there is no initial decrease to a minimum at 5 L. This decrease observed for the ordered surface is believed to be the result of a reduction in the nearly free electrons near the fermi level due to localization of the charge density by the electronegative oxygen atoms. However, when oxygen is adsorbed on a disordered surface, surface has many more bonding sites available than the annealed surface does and it is unable to localize the charge density to the extent seen for an ordered surface. This can be explained by the fact that the surface electrons are no longer in periodic Bloch states which allow them to move "freely" because the roughened surface has broken the two dimensional symmetry of the ordered surface. For example, the blurred

low energy electron diffraction pattern was observed for the roughened surface. In fact, this reduced oxygen aluminum interaction distance would cause these bonding sites to act as a surface dipole which tends to increase the second harmonic response as explained above. This is consistent with the experimental results, in which the SH intensity exhibits an immediate increase upon exposure to oxygen.

In the previous paragraphs we have discussed both our Kelvin Probe and SHG data. However, since the work function change and second harmonic response characterize similar properties of the surface, it is necessary to examine the relations between these two physical phenomena. In order to qualitatively explain our results it is sufficient to introduce a simple finite well model. Figure 22, a) and b) represent the surface potential barriers with work function Φ_I and Φ_{II} respectively, where $\Phi_I > \Phi_{II}$. An electron with energy E below the Fermi level has a sinusoidal wave function in the bulk and an exponentially decaying wave function in the surface region with tails δ_I and δ_{II} respectively. From quantum mechanics, we know that the decay tails have the form:⁵⁸

$$\begin{aligned}\delta_I &= \frac{\hbar}{\sqrt{2m(\Phi_I + E)}} \\ \delta_{II} &= \frac{\hbar}{\sqrt{2m(\Phi_{II} + E)}}.\end{aligned}\tag{1}$$

Obviously $\delta_I < \delta_{II}$ since $\Phi_I > \Phi_{II}$. Thus a smaller wave tail δ corresponds to a larger work function. On the other hand, since the region δ characterizes the response of nonlinear polarization to the incident laser electric field, a smaller δ would correspond to a smaller nonlinear response. Thus we would be able to detect a decrease in the SH signal. Therefore, from the above discussion we can conclude that a larger work function would cause a smaller SH signal. This model is consistent with our experimental results during the oxygen exposure range from 0 L to 5 L during

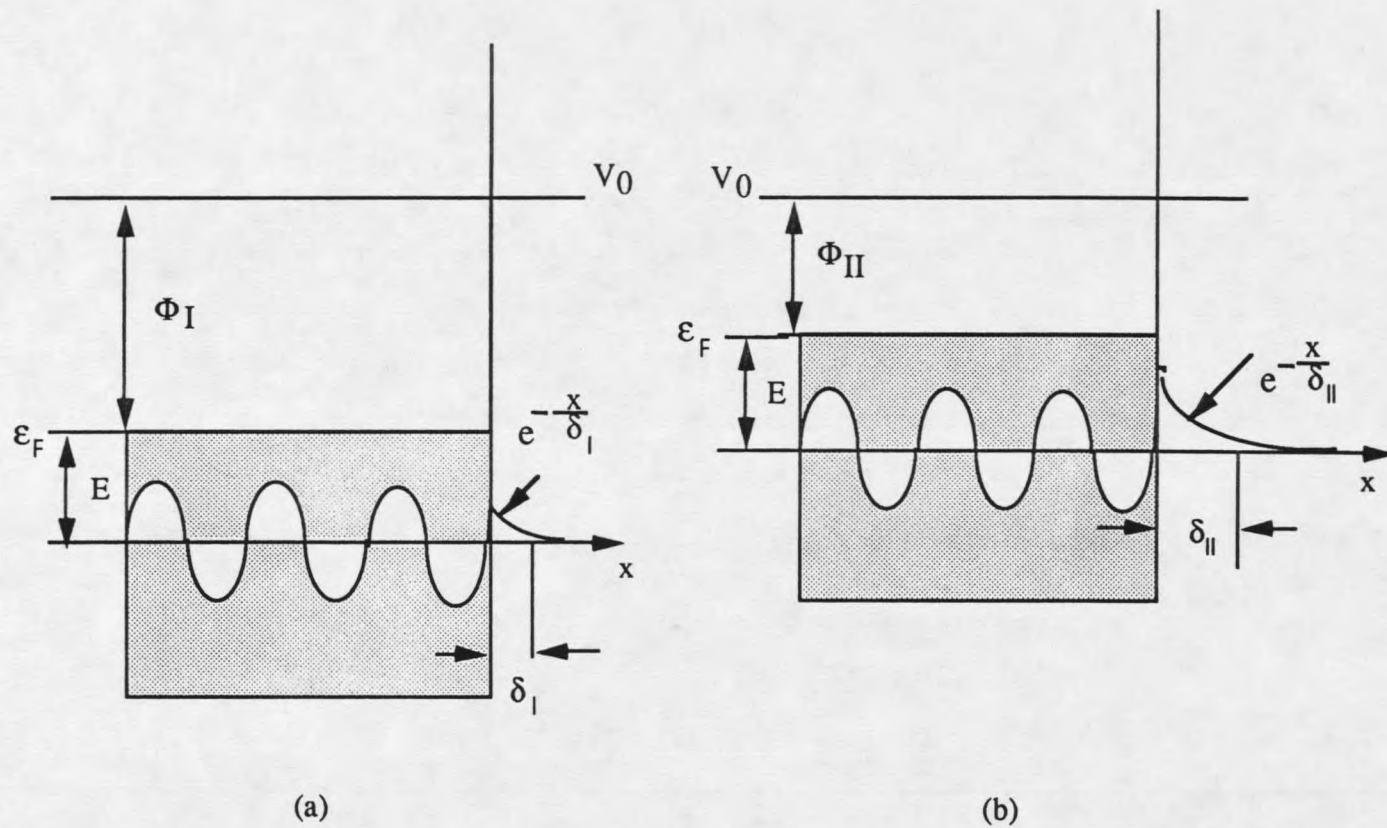


Figure 22. Finite well model for the surface electrons, work function (a) Φ_I and (b) Φ_{II} .

which metallic bonding dominated the surface nonlinearity. However, beyond these very low coverages, this simple finite well model does not apply.

Summary and Final Considerations

In summary, we have investigated the exposure of the Al(111) surface to oxygen, and concluded that multiple phases exist during the oxidation process. The initial phase is characterized by occupation of only surface chemisorption sites. This is followed by population of subsurface binding sites, and finally formation of Al_2O_3 . In addition we have used the relatively new experimental technique of surface enhanced second harmonic generation as a highly surface sensitive probe of the oxidation process. This technique has yielded valuable dynamic information concerning the nature of the oxygen-aluminum bonding during oxidation. Specifically, we observed the conversion from metallic bonding between the adsorbate and substrate to ionic bonding.

Complete understanding of the surface nonlinear polarizability due to the surface interactions needs detailed theoretical calculations. For further considerations, we feel that our SH data has some bearing on the theory of catalytic poisoning. The basic idea involved in the poisoning effects are the electrostatic interaction between two adsorbates, and between an adsorbed atom and an adsorbed or adsorbing molecule. Feibelman and Hamann⁵⁹ have suggested that the long range effect of surface modifiers is best characterized by examining the density of states available at the Fermi level at the surface. For example, a reduction in such states would reduce the chemisorption activity of the surface. Electronegative adsorbates are known to have profound poisoning effects on many metals. Experimentally, studies of adsorption and kinetics on transition metal surfaces show that the addition of less than 0.1 monolayer of a "poison" species such as S, can cause drastic reductions in reactant sticking probabilities, coverages, and reactivities.⁶⁰ It is believed that our SHG data for low exposures presented previously can best be understood using the Feibelman-

Hamann theory.⁵⁹ The most polarizable electrons of a free electron gas are those at the Fermi level. Poisoning by an electronegative atom such as oxygen is the result of its removing charge from the surface that would otherwise facilitate a rate-limiting reaction step. Thus, it reduces the number of surface free electron near the Fermi level. To the extent that aluminum is a free electron metal the SH signal derives from the electrons at the Fermi level at the surface. A decrease in the density of these electrons necessarily will reduce the SH signal. Hence, we suggest that the O/Al(111) system displays a poisoning effect at low coverages.

REFERENCES CITED

REFERENCES

1. D. W. Goodman and J. E. Houston, *Science* 236, 403(1987)
2. I. P. Batra and L. Kleinman, *J. Elect. Spect. Rel. Phen.* 33, 1750 (1984)
3. Y. R. Shen, in " The principles of Nonlinear Optics" New York: Wiley (1984)
4. PHI Model 545 "Instruction Manual", Physical Electronics Industries, Inc.
5. Model 162, Boxcar Averager Operating and Service Manual, EG&G Princeton Applied Research
6. R. Michel, J. Gastaldi, C. Allasta, C. Jourdan and J. Derrien, *Surf. Sci.* 95, 309(1980)
and R. Michel, C. Jourdar, J. Gastaldi and J. Derrien, *Surf. Sci.* 84, L509 (1979)
7. A. L. Testoni and P. C. Stair, *Surf. Sci.* 171, L491 (1986)
8. L. E. Davis et al., *Handbook of Auger Electron Spectroscopy*, Perkin-Elmer Corporation, Eden Prairie, (1979)
9. D. Briggs and M. P. Seah in *Practical Surface Analysis* (Wiley, New York) (1983)
10. H. E. Bishop and J. C. Riviere, *J. Appl. Phys.* 40, 1740 (1969)
11. C. R. Worthington and S. G. Tomlin, *Proc. Phys. Soc.* A69, 401 (1956)
12. W. Reuter in *Proc. Sixth Int. Conf. on X-ray Ptics and Microanalysis*, (Eds. G. Shinoda, K. Kohra and T. Lchinokawa Univ. of Tokyo Press) (1972)
13. N. W. Ashcroft and N. D. Mermin in " Solid State Physics" (Holt, Rinehart and Winstar) (1976)
14. P. P. Craig and V. Radeka, *Rev. Sci. Instr.* 41, 258 (1970)
15. Y. R. Shen, *Ann. Rev. Mater. Sci.* 16, 69 (1986)
16. Y. R. Shen, *J. Vac. Sci. Tech.* B3(5),1464 (1985)
17. M. Weber and A. Liebsch, *Phy. Rev.* B35,7411(1987)

18. J. A. Armstrong, N. Bloembergen, J. Ducuing, and P. S. Pershan, Phys. Rev. 127, 1918 (1962)
19. J. Rudnick and E. A. Stern in Polaritons eds., E. Barstern and F. Demartim, Perhammon Press, New York (1974) P329
20. J. E. Sipe, V. C. Y. So, M. Fukui and G. I. Stegeman, Phys. Rev. B21, 4389 (1980)
21. J. E. Sipe, V. C. Y. So, M. Fukui and G. I. Stegeman, Solid State Comminu. 34, 523 (1980)
22. C. K. Chen, T. F. Heinz, D. Richard and Y. R. Shen, Phys. Rev. Lett. 46, 1010 (1981), H. W. K. Tom, T. F. Heinz and Y. R. Shen, Phys. Rev. Lett. 51, 1983 (1983) and T. F. Heinz, M. M. T. Loy and W. A. Thompson, Phys. Rev. Lett. 54, 63 (1985)
23. T. F. Heinz, C. K. Chen, D. Richard and Y. R. Shen, Phys. Rev. Lett. 48, 478 (1982)
24. H. W. K. Tom, C. M. Mate, X. D. Zhu, J. E. Crowell, T. F. Heinz, G. A. Somorjai and Y. R. Shen, Phys. Rev. Lett. 52, 348 (1984)
25. A. M. Bradshaw, P. Hofmamm and W. Wyrobisch, Surf. Sci. 68, 269 (1977) , A. M. Bradshaw, W. Domcke and L. S. Cederbaum, Phys. Rev. B16, 1480 (1977) 71. and P. Hofmann, K. Horn, A. M. Bradshaw and K. Jacobi, Surf. Sci. Lett. 82, L610 (1979)
26. F. Soria, V. Martinez, M. C. Munoz and J. L. Sacedon, Phys. Rev. B24, 6926 (1981)
27. P. Hofmann, W. Wyrobisch and A. N. Bradshaw, Surf. Sci. 80, 344 (1979)
28. C. W. B. Martinson and S. A. Flodstrom, Surf. Sci. 80, 306 (1979)
29. R. Michel, J. Gastaldi, C. Jourdan and J. Derrien, Surf. Sci. 95, 309 (1979)
30. S. A. Flodstrom, C. W. B. Martinson, R. Z. Bachrach, S. B. M. Hagstrom and R. S. Bauer, Phys. Rev. Lett. 40, 907 (1978)
31. C. W. B. Martinson and S. A. Flodstrom, J. Rundgren, and P. Westrin, Surf. Sci. 89, 102 (1979)

32. A. Bianconi, R. Z. Bachrach, S. B. M. Hagstrom and S. A. Flodstrom, Phys. Rev. B19, 2837 (1979)
33. W. Eberhardt and F. J. Himpsel, Phys. Rev. Lett. 42, 1375 (1979)
34. P. O. Gartland, Surf. Sci. 62, 183 (1977)
35. F. Jona, J. Phys. Chem. Solids 28, 2155 (1967)
36. J. G. Chen, J. E. Crowell and J. T. Yates, Phys. Rev. B33, 1436 (1986)
37. J. E. Crowell, J. G. Chen and J. T. Yates, Surf. Sci. 165, 37 (1986)
38. J. L. Erskine and R. L. Strong, Phys. Rev. B25, 5547 (1982)
39. M. Ece and R. W. Vook, J. Vac.Sci. Tech. A6(3), 1036(1988)
40. P. Hofmann, C. V. Muschwitz, K. Horn, K. Jacobi, A. M. Bradshaw, K. Kambe and M. Scheffler, Surf. Sci. 93, 310 (1980)
44. W. Eberhardt and F. J. Himpsel, Phys. Rev. Lett. 42, 1375 (1979)
42. P. O. Gartland and B. J. Slagsvold, Solid State Commun. 25,489(1978)
43. P. Hofmann, W. Wyrobisch and A. M. Bradshaw, Surf. Sci. 80,344 (1979)
44. J. A. Ramsey, Appl. Surf. Sci. 13, 159 (1982)
45. S. A. Flodstrom, R. Z. Bachrach, R. S. Bauer and S. B. M. Hagstrom, Phys. Rev. Lett. 37, 1282 (1976)
46. D. Norman, R. Jaeger and J. Stohr, Surf. Sci. 105, L297 (1981)
47. C. W. B. Martinson and S. A. Flodstrom, Solid State Commun.30, 671 (671)
48. L. I. Johansson and J. Stohr, Phys. Rev. Lett. 43, 1882 (1979)
49. J. Stohr, L. I. Johansson, S. Brennan, M. Hecht and J. N. Miller, Phys. Rev. B22, 4052 (1980)

50. R. L. Strong, B. Firey, F. W. deWette and J. L. Erskine, Phys. Rev. B26, 3483 (1982)
51. R. L. Strong, B. Firey, F. W. deWette and J. L. Erskine, J. Elec. Spect. Related Phenomena,29, 187 (1983)
52. L. Kleinman and K. Mednick, Phy. Rev. B23,4960(1981)
53. D. Heskett, K. J. Song, A. Burns, E. W. Plummer and H. L. Dai, J. Chem. Phys. 85, 7490 (1986)
54. D. Heskett, J. E. Urbach, K. J. Song, E. W. Plummer and H. L. Dai, Surf. Sci. 197, 225 (1988)
55. X. D. Zhu, Y. R. Shen and R. Carr, Surf. Sci. 163, 114 (1985)
56. S. G. Grubb, A. M. DeSantolo and R. B. Hall, J. Phys. Chem. 92, 1419 (1988)
57. C. C. Mate, G. A. Sommorjai, H. W. K. Tom, X. D. Zhu and Y. R. Shen, J. Chem. Phys. 88, 441 (1988)
58. C. Cohen-Tannoudji, B. Diu and F. Laloe, "Quantum Mechnics", New York,Wiley (1977)
59. P. J. Feibelman and D. R. Hamann, Phys. Rev. Lett. 52, 61 (1984)
60. M. Kiskinova and D. W. Goodman, Surf. Sci. 108, 64 (1981)

APPENDIX

Computer Interface

The package of routines used to collect data from experiment was written in C source code by W. K. Ford, W. Hough, and K. J. Wan. The routines were written with the mood of simple source code adjustments to produce acquisition programs for comprehensive data scans. Thus it should require minimum effort of the programmer to write an acquisition program to be used on a different experiment, therefore reducing implementation overhead. The block diagram in Figure 15 illustrates the frame work of this package of routines.

The connection between the computer and experimental hardware (CMA and high voltage power supply for the ramp) is shown in Figure 16. As can be seen by referring to the figure, the GPIB (IEEE488 General Purpose Interface Bus) interfaces the software (data acquisition program) to the hardware package (A/D converter of SR510 lockin amplifier in our case). So GPIB command was used to control the experiments. For the purpose of our experiments, three disk directories were used to monitor the Auger measurement, Kelvin probe work function measurement, and second harmonic generation measurement. The only difference among them was the scanning parameters. For the Auger case, the Auger signal was collected versus the kinetic energy. For the other measurements the Kelvin probe nulling voltage, or nulling phase, and the second harmonic intensity were measured versus time. Three software libraries were used: the graphic library allowed us to plot the experimental data, the GPIB library was used for calling the GPIB commands, and the OGRE library was used to process the data. The output data file is where the data from the scanned segment is stored. The acquisition program automatically creates a new file name for each output data set and then it stores the data in a file of that name which resided in the current working directory. Every end of the working day, we transferred the data files into the "A1111" subdirectory. Some additional data files were used as part of the software. 'Config.dat' contains information used to configure the experiment devices. 'Calib.dat' contains two parameters per DAC device which are used to form a linear calibration of a DAC device and the

high voltage power supply. Three directories called "aug" , "kevpob" and "shg" were used for Auger electron spectroscopy, Kelvin probe work function measurement, and second harmonic generation measurements, respectively. The source code for the software system is listed below. The following table organizes the code according to function. and by specific experiment.

Common to all programs	gpib.h, globals.h arrows.c, augplot.c, drparpg.c, drscapg.c, gtohv.c, helpag.c, ogfio.c, parser.c, recpage.c, redraw.c, rfainit.c, rfaio.c, rfasave.c, rfaud.c, scdump.c, setdisp.c, stateio.c, syscmd.c, time.c, tprintf.c
Auger spectroscopy	auger.h auger.c, augscan.c, hvcal.c, hvset.c
Kelvin probe measurements	kevpob.h kevpob.c, kevscan.c
Second hamonic generation	shg.h shg.c, shgscan.c

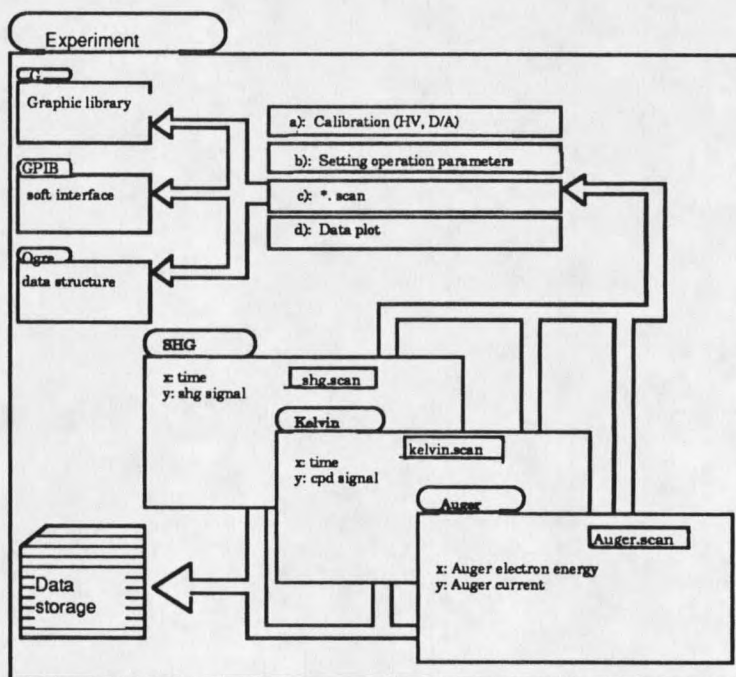
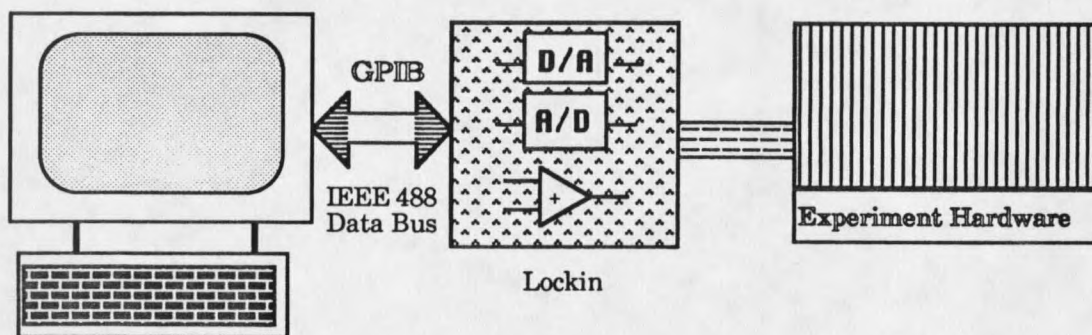


Figure 23. Diagram of computer programs.

Figure 24. Software listing.

```

/*NAME: gpib.h
/* GPIB Commands
#define UNL 0x3f /* GPIB unlisten command
#define UNT 0x5f /* GPIB untalk command
#define GTL 0x01 /* GPIB go to local
#define SDC 0x04 /* GPIB selected device clear
#define PPC 0x05 /* GPIB parallel poll configure
#define GET 0x08 /* GPIB group execute trigger
#define TCT 0x09 /* GPIB take control
#define LLO 0x11 /* GPIB local lock out
#define DCL 0x14 /* GPIB device clear
#define PPU 0x15 /* GPIB parallel poll unconfigure
#define SPE 0x18 /* GPIB serial poll enable
#define SPD 0x19 /* GPIB serial poll disable
#define PPE 0x60 /* GPIB parallel poll enable
#define PPD 0x70 /* GPIB parallel poll disable
/* GPIB status bit vector :
/* global variable ibsta and wait mask
#define ERR (1<<15) /* Error detected
#define TIMO (1<<14) /* Timeout
#define END (1<<13) /* EOI or EOS detected
#define SRQI (1<<12) /* SRQ detected by CIC
#define RQS (1<<11) /* Device needs service
#define CMPL (1<<8) /* I/O completed
#define LOK (1<<7) /* Local lockout state
#define REM (1<<6) /* Remote state
#define CIC (1<<5) /* Controller-in-Charge
#define ATN (1<<4) /* Attention asserted
#define TACS (1<<3) /* Talker active
#define LACS (1<<2) /* Listener active
#define DTAS (1<<1) /* Device trigger state
#define DCAS (1<<0) /* Device clear state
/* Error messages returned in global variable iberr
#define EDVR 0 /* DOS error
#define ECIC 1 /* Function requires GPIB board to be CIC
#define ENOL 2 /* Write function detected no Listeners
#define EADR 3 /* Interface board not addressed correctly
#define EARG 4 /* Invalid argument to function call
#define ESAC 5 /* Function requires GPIB board to be SAC
#define EABO 6 /* I/O operation aborted
#define ENEB 7 /* Non-existent interface board
#define EOIP 10 /* I/O operation started before previous
/* operation completed
#define ECAP 11 /* No capability for intended operation
#define EFSO 12 /* File system operation error
#define EBUS 14 /* Command error during device call
#define ESTB 15 /* Serial poll status byte lost
#define ESRQ 16 /* SRQ remains asserted
/* EOS mode bits
#define BIN (1<<12) /* Eight bit compare
#define XEOS (1<<11) /* Send EOI with EOS byte
#define REOS (1<<10) /* Terminate read on EOS
/* Timeout values and meanings
#define TNONE 0 /* Infinite timeout (disabled)
#define T10us 1 /* Timeout of 10 us (ideal)
#define T30us 2 /* Timeout of 30 us (ideal)
#define T100us 3 /* Timeout of 100 us (ideal)
#define T300us 4 /* Timeout of 300 us (ideal)
#define T1ms 5 /* Timeout of 1 ms (ideal)
#define T3ms 6 /* Timeout of 3 ms (ideal)
#define T10ms 7 /* Timeout of 10 ms (ideal)
#define T30ms 8 /* Timeout of 30 ms (ideal)
#define T100ms 9 /* Timeout of 100 ms (ideal)
#define T300ms 10 /* Timeout of 300 ms (ideal)
#define T1s 11 /* Timeout of 1 s (ideal)
#define T3s 12 /* Timeout of 3 s (ideal)
#define T10s 13 /* Timeout of 10 s (ideal)
#define T30s 14 /* Timeout of 30 s (ideal)
#define T100s 15 /* Timeout of 100 s (ideal)
#define T300s 16 /* Timeout of 300 s (ideal)
#define T1000s 17 /* Timeout of 1000 s (maximum)
/* Miscellaneous
#define S 0x08 /* parallel poll sense bit
#define LF 0x0a /* ASCII linefeed character

/*NAME: globals.h
FUNCTION: Contains the global definitions and initializations
/* Simple data types...
char recfile[] /* Filename containing expt. recipes
errfile[] /* Filename for errors
/* "kevprob.err";
int pgindx, /* Index of currently displayed page
wreci; /* Index of current recipe
/* Complex data types...
RFASRS wu; /* in-memory data structure
SWEEP dirrec, /* Presnt directory rec. selected
recipe; /* Current sweep template
PAGE *augpg[4]; /* Video display pages
FLAGS flag; /* Program current status flags
/* Command Parsing Definitions
CTABLE rec tab[] = { /* Recipe Page commands
{0/* NULL command */},
{"Ex","Exit Program",exit},
{F1,"Go to Scan page",Scapg},
{F3,"Go to Parameter page",Parpag},

```

Figure 24: Software listing (Continued)

```

{"P","Redraw Current Screen",Redrawsc},
{"sc","Initiate/Continue Scan",Kevscan},
{"pa","Pause",Hold},
{"sy","Execute System Command",Syscmd},
{"?","This Page",Helpag},
{"**","Screen Dump",scrndmp},
/* Unique... */
UP,"Move Menu Selector Up",reclast},
DOWN,"Move Menu Selector Down",recnext},
{"e0","Set Initial Energy",Sete0},
{"e1","Set Final Energy",Sete1},
{"t","Set Dwell Time",Setau},
{"np","Set Number of Points",Setnp},
{"T","Set Recipe Title",Setit},
{"sv","Save Recipe",recsave},
{"D","Delete Record",recdelete},
{"Rp","Replace Recipe",recreplace},
{"se","Select Directory Recipe",recselect}
};
extern int unplot();
CTABLE par_tab[] = {
/* Common... */
{0/* NULL command */},
{"Cm","Enter comment",comment},
{"Ex","Exit Program",exit},
{"F1","Go to Scan page",Scapag},
{"F2","Go to Recipe Page",Recpag},
{"Ig","Ignore Spectrum",ignore},
{"sy","Execute System Command",Syscmd},
{"P","Redraw Current Screen",Redrawsc},
{"pa","Pause",Hold},
{"sc","Initiate/Continue Scan",Kevscan},
{"sv","Save Spectrum",RFAsave},
{"?","This Page",Helpag},
{"**","Screen Dump",scrndmp},
};
CTABLE sca_tab[] = {
/* Common... */
{0/* NULL command */},
{"Ex","Exit Program",exit},
{"D","Double Scan Time",Again},
{"F2","Go to Recipe Page",Recpag},
{"F3","Go to Parameter page",Farpag},
{"Ig","Ignore Spectrum",ignore},
{"P","Redraw Current Screen",Redrawsc},
{"pa","Pause",Hold},
{"sc","Initiate/Continue Scan",Kevscan},
{"sv","Save Current Scan",RFAsave},
{"sy","Execute System Command",Syscmd},
{"**","Screen Dump",scrndmp},
{"?","This Page",Helpag},
/* Unique... */
{"pl","Plot Spectrum",augplot},
{"Up1","Unplot current sweep",unplot},
};
/* Lengths of command tables... */
int cmd_tab_len[] = {
sizeof(par_tab)/sizeof(CTABLE),
sizeof(rec_tab)/sizeof(CTABLE),
sizeof(sca_tab)/sizeof(CTABLE),
/* hel_tab*/ 0
};
/* Pointers to the parser tables... */
CTABLE *tabptr[] = { par_tab,rec_tab,sca_tab,0/*hel_tab*/ };
/* Menus for display pages... */
MENU pmenu[] = {
{30,24,"PARAMETER PAGE"}, /* (0) Program parm. title*/
{0,6,"Comment:"}, /* (1) Comment */
{9,17,"t0: "}, /* (2) e0 */
{9,16,"t1: "}, /* (3) e1 */
{9,15,"tau: "}, /* (4) tau */
{9,14,"npts: "}, /* (5) npts */
{40,19,"Line: "}, /* (6) line */
{40,18,"Linex2: "}, /* (7) line x2 */
{40,17,"Bandpass: "}, /* (8) Bandpass */
{40,16,"Sens: "}, /* (9) Sensitivity */
{40,15,"Dyn.Res: "}, /* (10) Dynamic reserve */
{40,14,"Pre. T.C: "}, /* (11) Pre time */
{40,13,"Post T.C: "}, /* (12) Post time */
{40,12,"Phase: "}, /* (13) Phase */
{40,11,"Ref.Freq: "}, /* (14) Reference frequency*/
{9,19,"data: "}, /* (15) Current data point*/
{0,8,"File? "}, /* (16) Filename prompt */
{9,21,"Kevprob Program"}, /* (17) Lockin parm. title*/
{40,21,"SRS510 Lockin"}, /* (18) Lockin parm. title*/
};
MENU smenu[] = {
/* Scan 1.111 */
{30,48," SHG Sweep "}, /* (0) Page title */
{2,48,"File: "}, /* (1) Filename */
{3,46,"t="}, /* (2) e0 */
{69,46,"t1="}, /* (3) e1 */
{1,3,""}, /* (4) y0 */
{1,35,""}, /* (5) y1 */
{20,1,"\0x0"}, /* (6) Message line */
{30,46,"sec/Div."}, /* (7) Volts/Division */
{1,16,"ptop="}, /* (8) Peak to peak */
{45,48,""}, /* (9) Time stamp */
};

```

Figure 24: Software listing (Continued)

```

/*
NAME: augplot.c/
#include <string.h>
#include "auger.h"
static AutoScale(SEGMENT*);
static void case2(char*);
static AutoScale(s) /* Define new plotting window using passed segment */
register SEGMENT *s;
{
register int i;
double miny, maxy;
double x0,x1,y0,y1;
seewd(&x0,&y0,&x1,&y1);
miny = 1e37;
maxy = -miny;
for(i=0;i<s->npts;i++) /* search for ordinate extremes */
{
if( miny>s->d[i] ) miny = s->d[i];
if( maxy<s->d[i] ) maxy = s->d[i];
}
if( !flag.x.sweeping ) /* Set new window */
setscl(s->e0, miny, (s->e1<=s->e0)?(s->e0+1.):(s->e1), /*
(maxy<=miny)?(miny+1.):maxy );
else /* but do not rescale ongoing data */
setscl(x0, miny, x1, (maxy<=miny)?(miny+1.):maxy);
}
RFAPlot(v) /* Draw the data vector contained in the RFASRS structure */
{
RFASRS *v;
DATA TYPE x,del;
register int i;
register SEGMENT *s;

setcap(scagp); /* set up */
udscale();
s = &(v->spectrum);
x = s->e0;
del = (s->e1 - s->e0)/(s->npts-1);
move(x,s->d[0]); /* plotting loop */
for(i=1;i<(s->npts);i++)
draw(x+=del,s->d[i]);
}
static void case2(s) /* Scale and plot current wu structure */
char *s; /* File name to use in label */
{
char bf[64];

AutoScale(&wu.spectrum); /* Autoscale the spectrum */

/* Update display page... */
tprintf(smenu[1].x,smenu[1].y,0,
strcat( strcpy(bf,smenu[1].s),s ) );
tprintf(smenu[9].x,smenu[9].y,0,
strcat( strcpy(bf,smenu[9].s),wu.tstamp));
RFAPlot(&wu); /* Plot Current wu structure */
/* with current scale */
}
static void case4() /* Read new disk file and plot */
{
FILE *fd;
char bf[64];

tprintf(smenu[6].x,smenu[6].y,1,"Filename? "); /* File Prompt */

if( (*gets(bf) == '\0') && (wu.spectrum.d != (DATA_TYPE*)NULL) )
case2("CURRENT"); /* Use current spectrum */

else if ((fd = fopen(bf,"rb")) == NULL )
{
tprintf(smenu[6].x,smenu[6].y,0,"Bad Filename:");
Pause(1.);
}

else /* Plot from disk file */
{
RFARD(fd,&wu);
fclose(fd);
case2(bf);
}

tclr(smenu[6].x,smenu[6].y,30);
}
augplot()
{
Scapag(); /* Switch to graphics page */
switch( flag.value&0x6 )
{
case 4: /* Current file saved. Plot any file */
case4(); break;
case 2: /* Sweeping, plot only current file */
case 0: /* Not sweeping, Not saved */
case2("CURRENT"); break;
}
}
unplot()
{

```

Figure 24: Software listing (Continued)

```

        setcm(1);          /* Set undraw cursor mode          */
        AutoScale(&wu.spectrum); /* Autoscale the spectrum */
        RFAPlot(&wu);       /* Plot Current wu structure */
        setcm(0);          /* Set draw cursor mode   */
    }

/*NAME: arrows*/
#include <math.h>
#include "auger.h"
slower()
{
    wu.dwell *= 2.;
    udednp(); /* Update dwell time on parameter page */
}

faster()
{
    wu.dwell /= 2.;
    if( wu.dwell < 10e-3 ) wu.dwell = 10e-3;
    udednp(); /* Update dwell time on parameter page */
}

expand()
{
    double x0,x1,y0,y1,ctr;
    double d1, d2;
    /* blow up by 3 */
    seedd(&x0,&y0,&x1,&y1);

    ctr = (y1+y0)/2.;
    d1 = (ctr-y0)/3.;
    d2 = (y1-ctr)/3.;
    setscl(x0,ctr - d1, x1, ctr + d2 );
    udscale();
    if( flag.x.sweeping )
        move( wu.spectrum.e1, wu.spectrum.d[wu.spectrum.npts-1] );
}

compress()
{
    double x0,x1,y0,y1,ctr;
    seedd(&x0,&y0,&x1,&y1);
    ctr = (y1+y0)/2.;
    setscl(x0,ctr - (ctr-y0)*3., x1, ctr + (y1-ctr)*3. );
    udscale();
    move( wu.spectrum.e1, wu.spectrum.d[wu.spectrum.npts-1] );
}

/*
NAME:      drparpg.c
SYNOPSIS:  drparpg()
FUNCTION:   Draws the parameter page template.*/
#include <string.h>
#include <malloc.h>
#include "auger.h"
drparpg()
{
    char bf[128];
    setcap(pargp);
    erase(0.,0.,0.);
    tprintf(pmenu[0].x,pmenu[0].y,0x71,pmenu[0].s);
    tprintf(pmenu[17].x,pmenu[17].y,0x06,pmenu[17].s);
    tprintf(pmenu[18].x,pmenu[18].y,0x06,pmenu[18].s);
    sprintf(bf,"%s%-3g",pmenu[2].s,wu.spectrum.e0);
    tprintf(pmenu[2].x,pmenu[2].y,0x03,bf);
    udednp();
    sprintf(bf,"%s%s",pmenu[6].s,(wu.filter40x01)>0?"in":"out");
    tprintf(pmenu[6].x,pmenu[6].y,0x03,bf);
    sprintf(bf,"%s%s",pmenu[7].s,(wu.filter40x02)>0?"in":"out");
    tprintf(pmenu[7].x,pmenu[7].y,0x03,bf);

    sprintf(bf,"%s%s",pmenu[8].s,(wu.filter40x04)>0?"in":"out");
    tprintf(pmenu[8].x,pmenu[8].y,0x03,bf);

    sprintf(bf,"%s%-0e",pmenu[9].s,wu.sens);
    tprintf(pmenu[9].x,pmenu[9].y,0x03,bf);

    sprintf(bf,"%s%-d",pmenu[10].s,wu.dynres);
    tprintf(pmenu[10].x,pmenu[10].y,0x03,bf);

    sprintf(bf,"%s%-3g",pmenu[11].s,wu.pretc);
    tprintf(pmenu[11].x,pmenu[11].y,0x03,bf);

    sprintf(bf,"%s%-3g",pmenu[12].s,wu.posttc);
    tprintf(pmenu[12].x,pmenu[12].y,0x03,bf);

    sprintf(bf,"%s%-3g",pmenu[13].s,wu.phase);
    tprintf(pmenu[13].x,pmenu[13].y,0x03,bf);

    sprintf(bf,"%s%-3g",pmenu[14].s,wu.reffreq);
    tprintf(pmenu[14].x,pmenu[14].y,0x03,bf);

    tprintf( pmenu[1].x,pmenu[1].y,0x2,pmenu[1].s);
    tprintf( pmenu[1].x,pmenu[1].y-2,0x07,wu.comment);
}

udednp()
{
    PAGE      *cp;

```

Figure 24: Software listing (Continued)

```
char    bf[32];
cp = seecap(); setcap(parpg);

sprintf(bf, "%s%-3g", pmenu[3].s, wu.spectrum.el);
tprintf(pmenu[3].x, pmenu[3].y, 0x03, bf);

if( flag.x.sweeping )
    sprintf(bf, "%s%.3e", pmenu[15].s, wu.spectrum.d[wu.spectrum.npts-1]);
else
    sprintf(bf, "%sunused", pmenu[15].s);
tprintf(pmenu[15].x, pmenu[15].y, 0x03, bf);

sprintf(bf, "%s%-3g", pmenu[4].s, wu.dwell);
tprintf(pmenu[4].x, pmenu[4].y, 0x03, bf);

sprintf(bf, "%s%-d", pmenu[5].s, wu.spectrum.npts);
tprintf(pmenu[5].x, pmenu[5].y, 0x03, bf);

setcap(cp);
}

comment()
{
char    bf[256];
if( flag.x.wusaved )      return;
tclr( pmenu[1].x, pmenu[1].y-2, 80*4 );
tprintf( pmenu[1].x, pmenu[1].y, 0x71, pmenu[1].s );
tprintf( pmenu[1].x, pmenu[1].y-2, 0x0, "" );
wu.comment = malloc( strlen( gets(bf) )+1 );
strcpy(wu.comment, bf);
tprintf( pmenu[1].x, pmenu[1].y, 0x2, pmenu[1].s );
}

/*
NAME:      drscapg.c
SYNOPSIS:  drscapg()
FUNCTION:  Draws the scanning page template.*/
#include "auger.h"
drscapg()
{
int      i, del;
setcap(scapg);
erase(0.,0.,0.);
setrgb(0.,1.,1.);
/* Title */
setinv(TRUE);
tprintf(smenu[0].x, smenu[0].y, 0, smenu[0].s);
setinv(FALSE);
/* Outer edge */
setlm(0);
ma(0,0); da(640,0); da(640,400); da(0,400); da(0,0);
/* Plotting box */
ma(V_LEFT-1, V_BOTTOM-1); da(V_RIGHT+1, V_BOTTOM-1);
da(V_RIGHT+1, V_TOP+1); da(V_LEFT-1, V_TOP+1);
da(V_LEFT-1, V_BOTTOM-1);
/* X-axis ticks */
setlm(6);
del = 50;
for(i=V_LEFT+del; i<V_RIGHT; i+=del)
    (ma(i, V_BOTTOM); da(i, V_TOP);)
/* Y-axis ticks */
for(i=V_BOTTOM+del; i<V_TOP; i+=del)
    (ma(V_LEFT, i); da(V_RIGHT, i);)
setlm(0);
setvp(V_LEFT, V_BOTTOM, V_RIGHT, V_TOP);
}

udscale()
{
char    bf[32];
double  e0, el, y0, y1, del;
PAGE    *cp;
cp = seecap(); setcap(scapg);
seewd(iel, y0, el, y1);
sprintf(bf, "%s%.0f", smenu[2].s, e0);
tprintf(smenu[2].x, smenu[2].y, 0, bf);
sprintf(bf, "%s%.0f", smenu[3].s, el);
tprintf(smenu[3].x, smenu[3].y, 0, bf);
setrot(1);
sprintf(bf, "%s%.2e", smenu[4].s, y0);
tprintf(smenu[4].x, smenu[4].y, 0, bf);
sprintf(bf, "%s%.2e", smenu[5].s, y1);
tprintf(smenu[5].x, smenu[5].y, 0, bf);
del = (y1-y0); sprintf(bf, "%s%.2e", smenu[8].s, del);
tprintf(smenu[8].x, smenu[8].y, 0, bf);
setrot(0);

del = (el-e0)/12.; sprintf(bf, "%-5.2f %s", del, smenu[7].s);
tprintf(smenu[7].x, smenu[7].y, 0, bf);
setcap(cp);
}

/*
NAME:      Helpag.c  Help Page
SYNOPSIS:  Helpag()
FUNCTION:  Presents a help display to user.*/
#include "auger.h"
Helpag()
```

Figure 24: Software listing (Continued)

```

{
    PAGE      *p;
    int       i;
    char      b[64];
    augpg[pgindx] = makspg();      /* Save current hardware page */
    setdm(3); setcap(hdwrgpg);

    /* Create new active page */
    /* Display help information... */
    tprintf(0,24,0x74,"Commands...");
    for (i=1;i<cmd_tab_len[pgindx];i++)
    {
        char      *s = tabptr[pgindx][i].cmdst;

        /* replace controls in cmdst's */
        if( strcmp(s,UP) == 0 )
            strcpy(b,"UP");
        else if( strcmp(s,DOWN) == 0 )
            strcpy(b,"DOWN");
        else if( strcmp(s,LEFT) == 0 )
            strcpy(b,"LEFT");
        else if( strcmp(s,RIGHT) == 0 )
            strcpy(b,"RIGHT");
        else if( strcmp(s,F1) == 0 )
            strcpy(b,"F1");
        else if( strcmp(s,F2) == 0 )
            strcpy(b,"F2");
        else if( strcmp(s,F3) == 0 )
            strcpy(b,"F3");
        else
            strcpy(b,s);
        tprintf(10,23-i,0x5,b);

        strcpy(b,tabptr[pgindx][i].help);
        tprintf(20,23-i,0x3,b);
    }

    while( !kbhit() );
    getch();
    setcap( augpg[pgindx] = sethpg(augpg[pgindx]) );
    /* ... Reset System */
}

/*
NAME:      gtohv.c
SYNOPSIS:  GtoHV();
FUNCTION:   Ask user for voltage to set the HV supply.*/
#include "auger.h"
GtoHV()
{
    DATA_TYPE hv;
    char      buff[64];
    tprintf(smenu[6].x, smenu[6].y, 0, "Set: ");
    sscanf(gets(buff),"%f",&hv);
    HVset(hv);
    clr(smenu[6].x,smenu[6].y,10);
}

/*
NAME:      ogfio.c
SYNOPSIS:  o_rd(fd, v)FILE*fd;VECTOR*v;o_wrt(v, fd)FILE*fd; VECTOR
FUNCTION:  o_rd():Read an ogre type file structure fromfd.o_wrt():Write an ogre type file structure to fd.*/
#include <malloc.h>
#include <string.h>
#include <stdio.h>
#include "log\ogre_ds.h"
o_rd(fd,v)
    FILE      *fd;      /* input binary stream descriptor */
    VECTOR     *v;       /* in-memory data file structure */
{
    SEGMENT *s;
    s = &v->seg;
    if( v->v_head != NULL )
    {
        free((char*)s->d);
        free(v->v_head);
    }
    fread((char*)&v->n_vhead,sizeof(v->n_vhead),1,fd);
    v->v_head = malloc(sizeof(char) * v->n_vhead);
    fread(v->v_head,sizeof(char) * v->n_vhead,1,fd);
    fread((char*)&s->npts,sizeof(s->npts),1,fd);
    fread((char*)&s->e0,sizeof(s->e0),1,fd);
    fread((char*)&s->e1,sizeof(s->e1),1,fd);
    s->d = (DATA_TYPE *) malloc(s->npts * sizeof(*s->d));
    /* now read each data point */
    fread((char*)s->d,sizeof(*s->d)* s->npts,1,fd);
}
o_wrt(v, strm)
    VECTOR     *v;       /* data file */
    FILE      *strm;      /* output data file name */
{
    SEGMENT *s;

    fwrite((char*)&v->n_vhead,sizeof(v->n_vhead),1,strm);
    fwrite(v->v_head,sizeof(char) * v->n_vhead,1,strm);
}

```

Figure 24: Software listing (Continued)

```
s = &v->seg;
fwrite((char*)&s->npts,sizeof(s->npts),1,strm);
fwrite((char*)&s->e0,sizeof(DATA_TYPE),1,strm);
fwrite((char*)&s->e1,sizeof(DATA_TYPE),1,strm);
    fwrite((char*)&s->d,sizeof(DATA_TYPE)*(s->npts),1,strm);
}

/*NAME:   parser.c
SYNOPSIS: Parse();
FUNCTION: Parse out any valid commands from stdin.
RETURNS: Address to the function to execute.*/
#include <string.h>
#include "auger.h"
static char pstring[8] = "\0"; /* parse buffer */
static char *p = pstring; /* current string position */
Parse(p_tab, len)
    CTABLE *p_tab;
    int len;
{
    int ans;
    if (kbhit())
        switch( ans=_parse(p_tab,len) )
        {
            case 0: /* Unresolved */
                break;
            case -1: /* Error in command */
                ParseError(); break;
            default: /* Execute command function */
                (*(p_tab)[ans].fn)(); break;
        }
}

static int _parse(p_tab,len)
    int len;
    CTABLE *p_tab;
{
    unsigned int i,l,index,count;
    char ch;
    *p++ = (ch=getch())==NULL ? '\xlf' : ch; /* Assume one is waiting */
    *p = NULL; /* Terminate string */
    index = count = 0; /* Index number of matches */
    l = strlen(pstring);
    for (i=1;i<len;++i)
        if( strncmp(pstring,p_tab[i].cmdst,l)==0 )
            { index = i; count++; }
    if( l < count ) return 0; /* undecided */
    p = pstring; *p = NULL; /* reset for next command */
    if( count==0 )
        return -1; /* error found */
    else if (count == 1)
        return index; /* unique command found */
}

ParseError()
{
    PAGE *p;
    int i,f=0;
    char c;
    augpg[pgindx] = makspg(); /* Save current page */
    setcap( p=sethpg(pgcreate(1)) );
    for (i=0;i<p->rsize;++i)
        {
            if (f==0) {c=0;f=1;}
            else {c=0x40;f=0;}
            *(p->raster+i)=c;
        }
    Pause(.1);
    setcap( augpg[pgindx] = sethpg(augpg[pgindx]) );
    /* ... Reset System */
}

/*NAME:   sweep.c
SYNOPSIS: SWEEP *sweepcpy(from, to)SWEEP from, toSWEEP *sweeprd(ds,fd)SWEEPds:FILEfd;SWEEP *sweepwt(ds,fd)SWEEP
          *ds: FILE*fd;
FUNCTION: sweepcpy Copy one sweep structure to another.
          sweeprd Read a sweep structure from a file.
          sweepwt Write a sweep structure from a file.
RETURNS: sweepcpy pointer to receiving structure.
          sweeprd pointer to structure read.
          sweepwt pointer to structure written.
ERROR: All return NULL pointer if operation failed
#include <memory.h>
SWEEP *sweepcpy(to,frm)
    SWEEP *frm, *to;
{
    return (SWEEP*)memcpy((char*)to,(char*)frm,sizeof(SWEEP));
}

SWEEP *sweeprd(ds, fd)
    SWEEP *ds;
    FILE *fd;
{
    return (fread((char*)ds,sizeof(SWEEP),1,fd)==1) ? ds : (SWEEP*)NULL;
}

SWEEP *sweepwt(ds, fd)
    SWEEP *ds;
    FILE *fd;
{

```

Figure 24: Software listing (Continued)

```

return (fwrite((char*)ds, sizeof(SWEEP), 1, fd) == 1) ? ds : (SWEEP*)NULL;
}

/*
NAME:      setrec          Set Recipes
SYNOPSIS:  (As Below)
FUNCTION:  Set the various components of the current recipe. (To be included in the recpage.c file.*/
Sete0()
{
    char    buff[64];
    tclr(cur[2].x, cur[2].y, 10);
    tprintf(cur[2].x, cur[2].y, 0x71, cur[2].s);
    sscanf(gets(buff), "%f", &recipe.e0);
    udlrc();
}

Sete1()
{
    char    buff[64];
    tclr(cur[3].x, cur[3].y, 10);
    tprintf(cur[3].x, cur[3].y, 0x71, cur[3].s);
    sscanf(gets(buff), "%f", &recipe.e1);
    udlrc();
}

Sete16()
{
    char    buff[64];
    tclr(cur[1].x, cur[1].y, 16);
    tprintf(cur[1].x, cur[1].y, 0x71, cur[1].s);
    strncpy(recipe.label, gets(buff), 15);
    udlrc();
}

Setnp()
{
    char    buff[64];
    tclr(cur[5].x, cur[5].y, 10);
    tprintf(cur[5].x, cur[5].y, 0x71, cur[5].s);
    sscanf(gets(buff), "%d", &recipe.npts);
    udlrc();
}

Setau()
{
    char    buff[64];
    tclr(cur[4].x, cur[4].y, 10);
    tprintf(cur[4].x, cur[4].y, 0x71, cur[4].s);
    sscanf(gets(buff), "%f", &recipe.tau);
    udlrc();
}

/*NAME:      recpage.c Recipe Page routines
SYNOPSIS:    (as below)
FUNCTION:    To handle operations on the experimental recipe page*/
#include <io.h>
#include "auger.h"
static FILE  udlrc(), udlrc();
static FILE  *fd; /* Local file descriptor for rec. list */
static MENU
title = /* Recipe Page Title */
delmsg = /* Delete message */
cur[] = { /* Current recipe menu */
    {30, 22, "Current Configuration"},
    {10, 20, ""},
    {10, 19, "e0:"},
    {10, 18, "e1:"},
    {10, 17, "tau:"},
    {10, 16, "npts:"}
},
dir[] = { /* Directory entry menu */
    {30, 12, "Directory Selection"},
    {10, 10, ""},
    {10, 9, "e0:"},
    {10, 8, "e1:"},
    {10, 7, "tau:"},
    {10, 6, "npts:"}
};
#include "setrec.c" /* Routines to set the recipe parameters */
#include "sweep.c" /* Routines to read, write and copy sweeps */
recinit() /* Called once. Loads the first recipe as current */
{
    setcap(recpg); /* Set up for recipe page */
    erase(0., 0., 0.);
    if (access(recfile, 0x6))
    {
        fd = fopen(recfile, "w+b"); /* New experiment, no file found... */
        /* Set up default scan */
        strncpy(recipe.label, "Wide Scan", 16);
        recipe.e0 = 25.;
        recipe.e1 = 1025.;
        recipe.tau = 0.300;
        recipe.npts = 512;
        recipe.vers = VERSION;
        recsave();
        sweepcpy(&dirrec, &recipe);
    }
    else
    {
        /* Open previous file */
    }
}

```

Figure 24: Software listing (Continued)

```
        fd = fopen(recfile,"r+b");
        sweeprd(&recipe,fd);
        sweepcpy(&dirrec,&recipe);
    }

    recdraw():          /* Set complete new page */
    {
recdraw() /* Called to update current and directory recipes */
    {
        tclr(title.x,title.y,sizeof(title.s));
        tprintf(title.x,title.y,0x71,title.s); setcat(0);
        udrac();
        uddir();
    }
recnext() /* Display the next directory recipe */
    {
        if( sweeprd(&dirrec,fd) == ((SWEEP*)NULL) )
        {
            /* End of file */
            fseek(fd,0L,SEEK_SET); /* Find beginning */
            sweeprd(&dirrec,fd);
        }
        uddir();
    }
reclast() /* Display the previous directory recipe */
    {
        /* Go back past the current recipe */
        fseek(fd, (long)(-sizeof(SWEEP)), SEEK_CUR);

        /* Try to read the recipe before the current one */
        if( fseek(fd, (long)(-sizeof(SWEEP)), SEEK_CUR) )
            /* Beginning of file found. Go to beginning of last rec. */
            fseek(fd, (long)(-sizeof(SWEEP)), SEEK_END);
        sweeprd(&dirrec,fd);
        uddir();
    }
recselect() /* Select the directory rec. as the new current rec. */
    {
        sweepcpy(&recipe,&dirrec);
        udrac();
    }

recsave() /* Enter the current recipe into the directory file */
    {
        /* Position at EOF and append new recipe */
        fseek(fd,0L,SEEK_END);
        sweepwt(&recipe,fd);

        /* Make new recipe the current directory entry */
        fseek(fd, (long)(-sizeof(SWEEP)), SEEK_CUR);
        sweeprd(&recipe,fd);
    }
recdelete() /* Delete the current directory entry */
    {
        SWEEP          a, b;
        FILE           *tfd;
        int            n=0;
        char           bf[64];
        static char    tfn[] = "$.tmp";

        tprintf(delmsg.x,delmsg.y,0x70,delmsg.s); setcat(0);
        tfd = fopen(tfn,"wb");

        /* Bump all following recipes up one, overwriting recipe to delete */
        while( sweeprd(&a,fd) != (SWEEP*)NULL )
        {
            fseek(fd, (long)(-sizeof(SWEEP)), SEEK_CUR);
            fseek(fd, (long)(-sizeof(SWEEP)), SEEK_CUR);
            sweepwt(&a,fd); ++n;
            fseek(fd, (long)(sizeof(SWEEP)), SEEK_CUR);
        }

        /* write all but last recipe to temporary file */
        /* last two recipes will be duplicates */
        fseek(fd, 0L, SEEK_SET);
        sweeprd(&b,fd);
        while( sweeprd(&a,fd) != (SWEEP*)NULL )
        {
            sweepwt(&b,tfd);
            sweepcpy(&b,&a);
        }
        fclose(tfd);          fclose(fd);
        /* rename temporary file to recipe file */
        sprintf(bf,"del %s",recfile);
        system(bf);
        sprintf(bf,"ren %s %s",tfn, recfile);
        system(bf);
        sprintf(bf,"del %s",tfn);
        system(bf);
        fd = fopen(recfile,"r+b");
        /* Position directory on recipe that followed deleted one. */
        fseek(fd, (long)(-n*sizeof(SWEEP)), SEEK_END);
        sweeprd(&dirrec,fd); uddir();
        tclr(delmsg.x,delmsg.y,strlen(delmsg.s));
    }
recreplace()
    {
        /* Overwrite current directory entry */
    }
```

Figure 24: Software listing (Continued)

```

fseek(fd, (long)(-sizeof(SWEEP)), SEEK_CUR);
sweepwt(&recipe,fd);
/* Reposition on replaced recipe */
fseek(fd, (long)(-sizeof(SWEEP)), SEEK_CUR);
sweeprd(&dirrec,fd); uddir();
}

static udrac() /* Update recipe section */
{
    char bf[64];
    tclr( cur[0].x,cur[0].y,sizeof(cur[0].s) );
    tprintf( cur[0].x,cur[0].y,0x6,cur[0].s );
    tclr( cur[1].x,cur[1].y,sizeof(cur[1].s) );
    tprintf( cur[1].x,cur[1].y,0x6,recipe.label );
    tclr( cur[2].x,cur[2].y,sizeof(cur[2].s) );
    tprintf( bf,"%s %5.0f",cur[2].s,recipe.e0 );
    tprintf( cur[2].x,cur[2].y,0x3,bf );
    tclr( cur[3].x,cur[3].y,sizeof(cur[3].s) );
    tprintf( bf,"%s %5.0f",cur[3].s,recipe.e1 );
    tprintf( cur[3].x,cur[3].y,0x3,bf );
    tclr( cur[4].x,cur[4].y,sizeof(cur[4].s) );
    tprintf( bf,"%s %5.2f",cur[4].s,recipe.tau );
    tprintf( cur[4].x,cur[4].y,0x3,bf );
    tclr( cur[5].x,cur[5].y,sizeof(cur[5].s) );
    tprintf( bf,"%s %5d",cur[5].s,recipe.npts );
    tprintf( cur[5].x,cur[5].y,0x3,bf );
}

static uddir() /* Update directory section */
{
    char bf[64];
    tclr( dir[0].x,dir[0].y,sizeof(dir[0].s) );
    tprintf( dir[0].x,dir[0].y,0x6,dir[0].s );
    tclr( dir[1].x,dir[1].y,sizeof(dirrec.label) );
    tprintf( dir[1].x,dir[1].y,0x6,dirrec.label );
    tclr( dir[2].x,dir[2].y,sizeof(dir[2].s) );
    tprintf( bf,"%s %5.0f",dir[2].s,dirrec.e0 );
    tprintf( dir[2].x,dir[2].y,0x3,bf );
    tclr( dir[3].x,dir[3].y,sizeof(dir[3].s) );
    tprintf( bf,"%s %5.0f",dir[3].s,dirrec.e1 );
    tprintf( dir[3].x,dir[3].y,0x3,bf );
    tclr( dir[4].x,dir[4].y,sizeof(dir[4].s) );
    tprintf( bf,"%s %5.2f",dir[4].s,dirrec.tau );
    tprintf( dir[4].x,dir[4].y,0x3,bf );
    tclr( dir[5].x,dir[5].y,sizeof(dir[5].s) );
    tprintf( bf,"%s %5d",dir[5].s,dirrec.npts );
    tprintf( dir[5].x,dir[5].y,0x3,bf );
}

/*NAME: redraw.c
SYNOPSIS: Redrawsc()*/
#include "auger.h"
Redrawsc()
{
    erase(0.,0.,0.);
    switch(pgindx)
    {
        case N_PPG:
            drparpg(); break;
        case N_RPG:
            recdraw(); break;
        case N_SPG:
            drscapg();
            move(wu.spectrum.e1,wu.spectrum.d[wu.spectrum.npts-1]);
            break;
    }
}

/*NAME: rfainit();
SYNOPSIS: int RFAinit(ds);RFASRS*ds;
FUNCTION: Initialize the RFASRS data structure.
#include <malloc.h>
#include "auger.h"
RFAinit(ds)
    RFASRS *ds;
{
    ds->version = VERSION; ds->dwell = recipe.tau;
    ds->pretc = 0.; ds->posttc = 0.;
    ds->phase = 0.;
    ds->reffreq = 0.; ds->sens = 0.;
    ds->filter = -1; ds->dynres = -1;

    ds->comment = malloc(1); *ds->comment = NULL;
    ds->spectrum.npts = recipe.npts;ds->spectrum.e0 = ds->spectrum.e1 = 0.;
    ds->spectrum.d = (DATA_TYPE *)NULL;
}

/*
NAME: rfaio.c Input and Output routines for RFA prog.
SYNOPSIS: RFASRS *RFArd(fd,df)FILE*fd;RFASRS*df;RFASRS *RFAwt(fd,df)FILE* fd;File descriptor of binary file RFASRS
*df:Points to RFA/auger datastructure NOTES: The RFA files follow the OGRE format that dictates that thedata file consists
of a single VECTOR.The VECTOR contains a vector header and a data SEGMENT.*/
#include <memory.h>
#include <malloc.h>
#include <string.h>
#include "auger.h"
RFASRS *RFArd(fd,df)
    FILE *fd;
    RFASRS *df;
{

```

Figure 24: Software listing (Continued)

```

VECTOR          v;
char            *p,*pv;
int             i;
DATA_TYPE *pd;
v.v_head = (char*)NULL; /* o_rd will alloc() memory here */
v.seg.d = (DATA_TYPE*)NULL;
o_rd(fd,&v);
/* Check if it is the correct version, return if not */
/* Clean the RFASRS structure if not empty */
if( df->comment != (char*)NULL ) free(df->comment);
if( df->spectrum.d != (DATA_TYPE*)NULL ) free((char*)df->spectrum.d);
/* Transfer into RFASRS format */
p = (char*)df;
pv = v.v_head;
i = sizeof(df->version)
    + sizeof(df->tstamp) + sizeof(df->dwell)
    + sizeof(df->prctc) + sizeof(df->posttc)
    + sizeof(df->phase) + sizeof(df->reffreq)
    + sizeof(df->sens) + sizeof(df->filter)
    + sizeof(df->dynres);
memcpy(p,pv,i);
free(df->comment);
strcpy( df->comment = malloc( strlen(pv)+1 ), pv);
df->spectrum.e0 = v.seg.e0;
df->spectrum.e1 = v.seg.e1;
df->spectrum.npts = v.seg.npts;
df->spectrum.d = (DATA_TYPE*)malloc((v.seg.npts)*sizeof(DATA_TYPE));
memcpy( (char*)df->spectrum.d,
        (char*)v.seg.d, v.seg.npts*sizeof(DATA_TYPE) );
/* Clean up... */
free( (char*)v.seg.d ); free( v.v_head );
return df;
}
RFASRS *RFAwt(fd,df) /* Save the current RFA data structure */
FILE *fd; /* File descriptor of binary file */
RFASRS *df; /* Points to RFA/aufer data structure */
{
    VECTOR v;
    int count, ccount;
    /* Compute the space needed for the VECTOR header */
    count = sizeof(df->version)
        + sizeof(df->tstamp) + sizeof(df->dwell)
        + sizeof(df->prctc) + sizeof(df->posttc)
        + sizeof(df->phase) + sizeof(df->reffreq)
        + sizeof(df->sens) + sizeof(df->filter)
        + sizeof(df->dynres);
    count += (ccount = strlen( df->comment ) + 1); /* Includes NULL*/
    /* Allocate space and transfer information */
    v.v_head = malloc(v.n.vhead = count);
    memcpy( v.v_head, (char*)df, (count-ccount) );
    memcpy( v.v_head+count-ccount, df->comment, ccount);
    v.seg.e0 = df->spectrum.e0;
    v.seg.e1 = df->spectrum.e1;
    v.seg.npts = df->spectrum.npts;
    /* ...v.seg.d = df->spectrum.d... */
    count = df->spectrum.npts * sizeof(DATA_TYPE);
    v.seg.d = (DATA_TYPE*)malloc(count);
    memcpy( (char*)v.seg.d, (char*)df->spectrum.d, count);
    o_wrt(&v,fd); /* Write the data vector in Ogre format */
    free(v.v_head); /* Clean up and return */
    free((char*)v.seg.d);
    return df;
}

/*NAME: rfasave.c
SYNOPSIS: RFAsave();
FUNCTION: Save the RFA structure wu, as a binary file.*/
#include <io.h>
#include <string.h>
#include "auger.h"
RFAsave() /* Open appropriate file & save */
{
    FILE *fd;
    char buff[32], bf[64];
    if( flag.x.wusaved ) return; /* Dont save twice */
    if( pgindx != N_PPG ) /* Switch to parameter page */
        Parpag();
    if( strlen(wu.comment) == 0 )
        comment();
    tprintf(pmenu[16].x,pmenu[16].y,0x74,pmenu[16].s); /* Get file name,*/
    if( access(gets(buff),0) == 0 ) /* and check for existence */
    {
        tprintf(pmenu[16].x,pmenu[16].y,0x71, "File already exists");
        getch();
    }
    else if( (fd=fopen(buff,"wb")) ) /* NULL return is error */
    {
        RFAwt(fd,wu); fclose(fd); /* write and close file */
        flag.x.wusaved = TRUE;
    }
    else perror("Error in RFAsave()");
    clr(pmenu[16].x,pmenu[16].y,30);
}
ignore() /* Set the saved flag to indicate a saved status without
        actually saving */

```

Figure 24: Software listing (Continued)

```

    {
        if( !flag.x.sweeping )          flag.x.wusaved = TRUE;
    }
/*NAME:      rfaud.c
SYNOPSIS:    RFAUD()
FUNCTION:    Initialize and update the mapping from the srs510 lockin, and the recipe template, to the RFASRS structure. */
#include "auger.h"
#include "srslib/lockin.h"
RFAUD()
/* Update the RFASRS data structure */
/* Current lockin settings */
wu.pretc = Prtcrd();
wu.posttc = Potcrd();
wu.phase = Phasrd();
wu.reffreq = Refqrd();
wu.sens = Sensrd()/Expdrd();
wu.filter = Filtrd();
wu.dynres = Dyrvrd();
/* Copy recipe template to RFA structure */
wu.spectrum.e0 = recipe.e0;
wu.spectrum.e1 = recipe.e1;
wu.dwell = recipe.tau;
wu.spectrum.npts = recipe.npts;
/* Stamp wu with current time */
strcpy(wu.timestamp, getstamp());
}

/*NAME:      scdump.c*/
#include "auger.h"
Scdump()
{
    char    bf[128];

    /* Assume the page is at the top of form */
    scrndmp();
    fprintf(stderr, "\n\n\n");
}

/*NAME:      setdisp.c
SYNOPSIS:    Parpag() Parameter Page Recpag() Recipe Page Scapag() Scan/Graphics Page
FUNCTION:    Select a display page. */
#include "auger.h"
Parpag()
{
    drparpg();
    if( pgindx != N_PPG );
    {
        /* Move the current hardware page into software */
        augpg[pgindx] = makspg();
        /* Move the parameter page into display page */
        setcap(parpg = sethpg(parpg));
        /* Set the program display page pointer to the parm. page */
        pgindx = N_PPG;
    }
}

Recpag()
{
    if( pgindx != N_RPG )
    {
        /* Move the current hardware page into software */
        augpg[pgindx] = makspg();

        /* Move the recipe page into display page */
        setcap(recpg = sethpg(recpg));

        /* Set the program display page pointer to the recipe page */
        pgindx = N_RPG;
    }
}

Scapag()
{
    if( pgindx != N_SPG )
    {
        /* Move the current hardware page into software */
        augpg[pgindx] = makspg();

        /* Move the scan page into display page */
        setcap(scapg = sethpg(scapg));

        /* Set the program display page pointer to the scan page */
        pgindx = N_SPG;
    }
}

/*NAME:      stateio.c*/
#include "auger.h"
static char    datfn[] = "hvcad.dat";
staterd(ds)
{
    STATE    *ds;
    {
        FILE    *fd;
        /* Retrieve previous machine state */
        if( (fd = fopen(datfn, "r")) != NULL )
        {
            fscanf(fd, "%f %f %f", &(ds->hvsup.m), &(ds->hvsup.b), &(ds->ac));
            fclose(fd);
        }
    }
}
```

Figure 24: Software listing (Continued)

```

        else
            perror("ExpInit: Couldn't open state data file");
    }
}
statewt(ds)
{
    STATE    ds;
    {
        FILE    *fd;
        /* Erase old file and write new conversion factors */
        fd = fopen(datfn,"w");
        fprintf(fd,"%f %f %f\n",
            ds.hvsup.m,ds.hvsup.b,ds.ac);
        fclose(fd);
    }
}

/*NAME:    syscmd.c
SYNOPSIS:    Syscmd()
FUNCTION:    Process MS-DOS level commands while leaving the calling routine(s) on hold.
*/
#include "auger.h"
Syscmd()
{
    char        buf[64], pathw[64];
    int        more=TRUE;
    FILE        *fd;
    /* Don't waste real time while sweeping */
    if( flag.x.sweeping && !flag.x.paused )
        return;
    /* Redirect stderr a stdout */
    dup2(fileno(stdout),fileno(stderr));
    augpg[pgindx] = makspg();
    setcap(sethpg(pgcreate(3)));
    tprintf(0,0,0x70,"auger' to return");
    while(more)
    {
        printf("\r%s>>",getcwd(pathw,63));
        if( strcmp(gets(buf),"auger") == 0 )
            more = FALSE;
        else if( strlen(buf) > 0 )
            {system(buf);cputs("\n");}
    }
    setcap( augpg[pgindx] = sethpg(augpg[pgindx]) );
    /* Redirect stderr into debug file */
    freopen(errfile,"a",stderr);
}

/*NAME:    time.c*/
#include <stdio.h>
#include <dos.h>
#include <time.h>
#define GTMADD    (0x2c)
static long    waittime = 0;    /* time to wait in hundredths of secs. */
char *getstamp()
{
    long        t;
    char        *p;
    time(&t);
    p = ctime(&t);
    p[24]=NULL;    /* Clobber newline character */
    return p;
}

static long    htime()    /* return number of 100ths of seconds */
{
    union REGS    r;
    r.h.ah = GTMADD;
    intdos(&r,&r);
    return
        (r.h.dl+100L*(r.h.dh+60L*(r.h.cl+60*r.h.ch)));
}

Pause(t)
{
    double        t;    /* Duration in Seconds */
    {
        /* only hundredths resolution */
        strtclk(t*1000.);
        while( tstclk() );
    }
}

strtclk(t)
{
    double        t;    /* time to count in milliseconds */
    {
        waittime = t/10 + htime();
    }
}

tstclk()
/* return 1 while still counting, 0 when time up */
{
    return (htime())>waittime?0:1;
}

/*NAME:    tpintf.c*/
#include <string.h>
#include "auger.h"
tclr(x,y,n)    /* Print <n> spaces after a menu element */
{
    unsigned int    x,y,n;
    {
        char        bf[82];
        int        i;
        setcat(0);
        n = (n<82)?n:81;
        for(i=0;i<n;i++) bf[i] = ' ';    /* space */
        bf[i] = NULL;
    }
}

```

Figure 24: Software listing (Continued)

```

        tprintf( x,y, stdpg->att.cat,bf );
    }
    tprintf(x,y,att,s)
        unsigned int      x,y,          /* coordinate to write at */
        att:              /* text attributes */
        char              *s;          /* text string */
    {
        ta(x,y);
        setcat(att);
        text(s);
    }

/*NAME:      auger.h
FUNCTION:    Header file for Auger type RFA, Microsoft C Programs.*/
#include <stdio.h>
#include "augdecl.h"
#define VERSION          (1)          /* Program version number*/
/* Screen definitions */
#define V_LEFT           (35)
#define V_RIGHT          (635)
#define V_BOTTOM        (20)
#define V_TOP            (360)
/* Keyboard Code Strings
#define F1               "\x1f\x3b"
#define F2               "\x1f\x3c"
#define F3               "\x1f\x3d"
#define UP               "\x1f\x48"
#define DOWN             "\x1f\x50"
#define RIGHT            "\x1f\x4d"
#define LEFT             "\x1f\x4b"
/* Some common definitions...
#define TRUE             (1)
#define FALSE           (0)
#define HVPORT          (5)          /* SRS dac port used for ramp */
#define N_PPG            (0)          /* Index for parameter page */
#define N_RPG            (1)          /* Index for recipe page */
#define N_SPG            (2)          /* Index for scan page */
#define parpg            (augpg[N_PPG]) /* parameter page pointer */
#define recpg            (augpg[N_RPG]) /* recipe page pointer */
#define scapg            (augpg[N_SPG]) /* scan/graphics page pointer*/
/* Global declarations...
#include "g\gdecl.h"          /* Graphics structures and defs.
#include "augtypes.h"
/* Variables....
extern char              recfile[], /* filename of the recipe list
errfile[],              /* filename for errors
pgindx;                /* index of current hardware page
extern int               wu;         /* in-memory data structure
extern RFASRS            dirrec;     /* Present rec. direct. entry
extern SWEEP             recipe;     /* Current sweep template
extern STATE             equip;      /* Equipment status
extern PAGE              *augpg[];   /* pointers to display pages
extern FLAGS             flag;        /* Program status flags
extern CTABLE            sca_tab[],rec_tab[], /* Command tables
par_tab[],
*tabptr[];             /* Pointers to those tables
extern int               cmd_tab_len[]; /* Lengths of Command Tables
extern MENU              rmenu[], /* Text and coordinates
pmenu[],
smenu[];

/* Function names...
extern                  /* Recipe page routines
reclast(),             recselect(),     recreate(),
recnext(),             recsave(),       recdelete(),
Sete0(), Sete1(),      Setnp(),
Setau(), Setit();

extern char             *getstamp();
extern SWEEP            *sweepcpy(SWEEP *,SWEEP *),
*sweepwt(SWEEP *, FILE *),
*sweeprd(SWEEP *, FILE *);
extern RFASRS           *RFAwt(FILE *, RFASRS *),
*RFArd(FILE *, RFASRS *);
extern int              RFAinit(RFASRS *),
RFAplot(RFASRS *),
o_rd(FILE *, VECTOR *),
o_wrt(VECTOR *, FILE *),
Parse(CTABLE *, int);
extern int              /* Parser function definitions...
augabort(),            Augfine(),      augplot(),      Augscan(),
comment(),              compress(),
faster(), GtoHV(),      hvcal(),      Helpag(),
Hold(),                 ignore(),
Parpag(),
RecDef(), Recpag(), Redrawsc(), RFAsave(),
Syscmd(), slower(), Scapag()
;

extern int              tprintf(unsigned,unsigned,unsigned,char*),
extern int              tclr(unsigned,unsigned,unsigned);
extern int              strtclk(double),tstclk();

/*Name:      augtypes.h
Data structure for the Retarding Field Analyzer Auger
spectroscopy using the SRS-510 Lockin Amplifier.
/* This file defines the Ogre data structures: SEGMENT and VECTOR */
#include "og/ogre_ds.h"
typedef struct

```

Figure 24: Software listing (Continued)

```

short    version; /* current version code */
char     tstamp[26]; /* time of day data was taken */
float    dwell, /* seconds per point (sec.) */
pretc, /* pre-time constant setting (sec.) */
posttc, /* post time constant setting (sec.) */
phase, /* lockin phase setting (degrees) */
reffreq, /* reference frequency (Hz) */
sens; /* sensitivity setting (V) */
short    filter, /* 1: line in, 2: 2x line in, 4: bandpass in
dynres; /* dynamic reserve code */
char     *comment; /* pointer to commentary array */
SEGMENT  spectrum; /* data structure containing spectrum data */
}
/* Sweep recipe data structure WKF/WH 12 May 86 */
typedef struct {
char     label[16]; /* label of the sweep setup */
float    e0, /* beginning of sweep */
el, /* ending "voltage" of sweep */
tau, /* seconds per point */
npts; /* number of points in sweep */
short    vers; /* RFA version number */
} SWEEP;
/* Command table structure */
typedef struct {
char     cmdst[4]; /* Unique identifier for command */
char     help[25]; /* Help string, displayed by help fnct. */
int      (*fn)(); /* Command function to execute */
} CTABLE;
/* Linear fitting parameters */
typedef struct {
float    m, /* High voltage calibration points */
b;
} LINCAL;
/* Data structure defining status of equipment besides SRS510 Lockin */
typedef struct {
LINCAL   hvsup;
float    ac;
} STATE;
/* Structure defining a unit of text */
typedef struct menu {
unsigned int    x,y; /* coordinate of text */
char           s[32];
} MENU;
/* Bit field program status flags */
typedef union {
struct {
unsigned    paused:1, /* (bit0)sweep paused flag */
sweeping:1, /* (bit1)sweep sweeping. */
wusaved:1; /* (bit2)sweep saved flag. */
} x;
int         value;
} FLAGS;

/*NAME:      auger.c
FUNCTION: Main drive for Auger Spectrometry.
#include <signal.h>
#include <stdlib.h>
#include "auger.h" /* General stuff including */
/* structures and global definitions */
#include "globals.h" /* Global definitions */
augabort() /* What to do on abort */
{
signal(SIGINT,augabort);
flag.x.sweeping = flag.x.paused = FALSE;
}
main()
{
int     ans;
ExpInit(); /* Initialize the experiment */
onexit(Augfine); /* Set up normal escape route */
signal(SIGINT,augabort); /* Set up abnormal escape route */
while(1) /* Parsing Loop */
Parse(tabptr[pgindx],cmd_tab_len[pgindx]);
/* Update parameter page */
}
static ExpInit() /* Initialize */
{
FILE     *fd;
freopen(errfile,"a",stderr); /* Error file for debugging */
perror("\nNEW RUN... ");
perror(getstamp()); /* Write time of run */
perror("\n");
equip.hvsup.m = 1.;
equip.hvsup.b = 0.;
staterd(&equip); /* Read the 'hvcal.dat' file */
RFAinit(&wu); /* Initialize the RFA data structure */
ibinit(); /* Init GPIB port */
HVset(50.);
RFAud(); /* Read current lockin config. */
/* Set up software rasters used in experiment */
rdinit(3); /* Initialize raster display */
pgindx = N_RPG; /* Recipes come up first */
recpg = hdwrgpg;
recinit(); /* Draw current batch of recipes */
}

```

Figure 24: Software listing (Continued)

```

scapg = pgcreate(8);          /* For graphics data          */
drscapg();                    /* Draw the scan page        */
parpg = pgcreate(3);          /* Parameters                  */
drparpg();
setcap(recpg);                /* Set first interactive page */
/* Set up flags */
flag.value = 0x04;            /* Saved, not sweeping, not paused */
}
Augfine()                      /* Exit the program gracefully */
{
    flushall();                /* Clean up explicitly        */
    if( !flag.x.wusaved )
    {
        augpg(pgindx) = savhpg();
        setdm(3); setcap(hdwirpg); erase(0,0,0.);

        tprintf(2,10,4,"The current sweep is NOT saved");
        tprintf(5,8,4,"Enter 'n' if you do NOT wish to save it> ");
        if( getch() != 'n' ) RFAsave();
    }
    HVset(50.);
    rdfine();
}

/*NAME: augscan.c
SYNOPSIS: augscan();
FUNCTION: Perform an Auger sweep.
#include <malloc.h>
#include "/srslib/lockin.h"
#include "auger.h"
static int ScInit(), ScFine();
static int Delay(double);
static double el, dwell, delta, np;
/* desired ending sweep energy */
/* starting dwell time */
/* sweep step size */
/* desired number of data points */
static int DATA_TYPE *p; /* Array of data points */
augscan() /* Ramp the HV supply for Auger spectrums */
{
    PAGE *cp;
    switch( flag.value&0x07 )
    {
        case 0: /* Sweep is not running and RFA is not saved */
            tprintf(smenu[6].x,smenu[6].y,0,
                "Last spectrum NOT saved...Continue? [y/n]");
            if( getch() != 'y' )
            {
                tclr(smenu[6].x,smenu[6].y,45);
                return;
            }
            tclr(smenu[6].x,smenu[6].y,45);
            flag.x.wusaved = TRUE;
        case 4: /* Sweep is not running and RFA is saved */
            ScInit();
            break;
        case 3: /* Sweep is paused */
            move( wu.spectrum.el, p[wu.spectrum.npts-1]);
            flag.x.paused = FALSE;
            break;
        default:
            putch(7); perror("RFA flag error");
            return;
    }
    /* Main sweeping loop */
    while( (wu.spectrum.el<el) && (wu.spectrum.npts<np)
        && (flag.x.sweeping) && (!flag.x.paused) )
    {
        HVset(wu.spectrum.el+delta); /* Next retarding energy*/
        Delay(wu.dwell); /* Stabilize the system */
        cp = seecap(); setcap(scapg);
        draw
        {
            wu.spectrum.el+=delta,
            p[wu.spectrum.npts++] = Disprd();
        };
        setcap(cp);
    }
    ScFine();
}
static ScInit()
{
    PAGE *cp;
    RFAud(); /* Update the rfa data structure with lockin
                settings. */
    el = wu.spectrum.el;
    np = wu.spectrum.npts;
    dwell = wu.dwell;
    delta = (wu.spectrum.el-wu.spectrum.e0)/(wu.spectrum.npts-1);
    if( wu.spectrum.d != (DATA_TYPE *)NULL )
        free( (char*)wu.spectrum.d );
    p = wu.spectrum.d
        = (DATA_TYPE *)malloc(wu.spectrum.npts*sizeof(DATA_TYPE));
    cp = seecap();
    setcap(scapg);
    setscl(wu.spectrum.e0, -wu.sens, el, wu.sens); udscale();
    HVset(wu.spectrum.el-wu.spectrum.e0); Pause(3.);
}

```

Figure 24: Software listing (Continued)

```

move( wu.spectrum.e0, p[(wu.spectrum.npts-1)-1]-Disprd() );
setcap(cp);
/* Initialize flags */
flag.x.wusaved = flag.x.paused = FALSE;
flag.x.sweeping = TRUE;

static int Delay(tau)
double tau; /* dwell time in seconds */
{
    strtclk(tau*1000.);
    while(tstclk()) /* Loop until time is up */
        Parse(tabptr[pgindx], cmd_tab_len[pgindx]);
    if ( pgindx == N_PPG)
        udednp(); /* Only when viewing the parameters */
}

Hold()
{
    flag.x.sweeping = flag.x.paused = TRUE;
}

static ScFine()
{
    switch (flag.value&0x03)
    {
        default: /* Sweep has been aborted */
        case 2: /* Sweep ended normally */
            wu.dwell = dwell; /* Reset dwell */
            HVset(wu.spectrum.e0); /* Reset to a safe state */
            flag.x.sweeping = flag.x.paused = flag.x.wusaved = FALSE;
            break;
        case 3: /* Sweep has been paused */
            break;
    }
    udednp(); /* One last time */
}

/*NAME: hvcal*/
#include "auger.h"
hvcal()
{
    FILE *fd, *fopen();
    STATE ds;
    short flag=1;
    char buf[64];
    augpg[pgindx] = makspg();
    setcap( setpg( pgcreate(3) ) );
    ds.hvsup.m = 1; ds.hvsup.b = 0; ds.ac = 1.76;
    staterd(&ds); /* Get old conversion parameters */
    tprintf(5,17,6,"c Calibrate");
    tprintf(5,16,6,"s Save");
    tprintf(5,15,6,"t Test");
    tprintf(5,14,6,"a Analyzer Constant");
    tprintf(5,13,6,"E Exit");
    while (flag)
    {
        tprintf(10,20,0x70,"Current calibration factors");
        sprintf(buf, " m = %f b = %f Analyzer Const = %f",
            ds.hvsup.m, ds.hvsup.b, ds.ac);
        tprintf(15,19,0x03,buf);
        switch(getch())
        {
            case 'c':
                Cal(&ds);
                break;
            case 's':
                tprintf(10,11,0x70,"Saving...");
                staterw(ds);
                equip.hvsup.m = ds.hvsup.m;
                equip.hvsup.b = ds.hvsup.b;
                tclr(10,11,20);
                break;
            case 't':
                test(ds);
                break;
            case 'a':
                tprintf(10,11,0x70,"Enter new analyzer constant:");
                scanf("%f",&(ds.ac));
                equip.ac = ds.ac;
                tclr(10,11,40);
                break;
            case 'E':
                flag=0;
                break;
        }
    }
    setcap( augpg[pgindx]=setpg( augpg[pgindx] ) );
}

static Cal(ds)
STATE *ds;
{
    int i;
    float data[4];
    static char pmpt[4][32] = {
        "Enter low voltage to set: ",
        "Enter supply reading : ",
        "Enter high voltage to set: "
    }

```

Figure 24: Software listing (Continued)

```

                                "Enter supply reading : "
                                );
                                tprintf(10,11,0x70,"Calibrating");
#define at1 (data[0])
#define pa1 (data[1])
#define at2 (data[2])
#define pa2 (data[3])
                                for( i=0;i<4;++i)
                                {
                                    tprintf(10,9-i,0x03,prompt[i]);
                                    scanf("%f",&data[i]);
                                    if( i==0 || i==2) HVset(ds->ac*data[i]);
                                }

                                if( pa2 != pa1 )
                                {
                                    ds->hvsup.m = (at2 - at1)/(pa2 - pa1);
                                    ds->hvsup.b = (at1*pa2 - at2*pa1)/(pa2 - pa1);
                                }

                                for( i=0;i<4;++i)
                                    tclr(10,9-i,70);
                                tclr(10,11,15);
                                }
static test(ds)
STATE ds;
{
    float hv, m,b;
    m=equip.hvsup.m;
    b=equip.hvsup.b;
    equip.hvsup.m = ds.hvsup.m;
    equip.hvsup.b = ds.hvsup.b;
    tprintf(10,11,0x70,"Testing: (Enter # < 0 to quit.)");
    while(1)
    {
        float atof();
        char buf[30];
        tclr(10,9,20);
        tprintf(10,9,0x6,"Set: ");
        gets(buf);
        hv = atof(buf);
        /*scanf("%f",&hv);*/
        if( 0.<hv && hv<1500. ) HVset(ds.ac*hv);
        else break;
    }
    equip.hvsup.b = b;
    equip.hvsup.m = m;
    tclr(10,11,35); tclr(10,9,20);
}

/*NAME: hvset*/
#include "auger.h"
int HVset(e) /* Set a D/A port to correspond to a high voltage */
double e;
{
    double daval;
    if( (0. > e) && (e > 2000) ) return 0;
    daval = ((e*equip.hvsup.m+equip.hvsup.b)/equip.ac); /* Adjust high voltage*/
    return (Dtoawt(HVPORT,daval));
}

/*
NAME kevtypes.h Data structure for Kelvin Probe measurements
using the SRS-510 Lockin Amplifier. */
/* This file defines the OGRE data structures: SEGMENT and VECTOR */
#include "og/ogre_ds.h"
typedef struct
{
    short version; /* current version code */
    char tstamp[26]; /* time of day data was taken */
    float dwell, /* seconds per point (sec.) */
    pretc, /* pre-time constant setting (sec.) */
    posttc, /* post time constant setting (sec.) */
    phase, /* lockin phase setting (degrees) */
    reffreq, /* reference frequency (Hz) */
    sens; /* sensitivity setting (V) */
    short filter; /* 1: line in, 2: 2x line in, 4: bandpass in */
    dynres; /* dynamic reserve code */
    char *comment; /* pointer to commentary array */
    SEGMENT spectrum; /* data structure containing spectrum data */
    RFASRS;
}

/* Sweep recipe data structure WK/WH 12 May 86 */
typedef struct
{
    char label[16]; /* label of the sweep setup */
    float e0, /* beginning of sweep */
    e1, /* ending "voltage" of sweep */
    tau; /* seconds per point */
    int npts; /* number of points in sweep */
    short vers; /* RFA version number */
} SWEEP;

/* Command table structure */
typedef struct
{
    char cmdst[4]; /* Unique identifier for command */
    char help[25]; /* Help string, displayed by help fnct. */
    int (*fn)(); /* Command function to execute */
} CTABLE;

/* Structure defining a unit of text */
typedef struct menu {
    unsigned int x,y; /* coordinate of text */
}

```

Figure 24: Software listing (Continued)

```

char          s[32];
}MENU;
/* Bit field program status flags */
typedef union
{
    struct
    {
        unsigned    paused:1, /* (bit0)sweep paused flag.*/
                    sweeping:1, /* (bit1)sweep sweeping. */
                    wusaved:1; /* (bit2)sweep saved flag. */
    } x;
    int    value;
} FLAGS;

/*NAME:      kevp.h
FUNCTION:    Header file for Kelvin probe data acquisition
             Microsoft C Programs.
#include <stdio.h>
#include "kevdecl.h"
#define      VERSION      (1) /* Program version number */
/* Screen definitions */
#define      V_LEFT      (35)
#define      V_RIGHT      (635)
#define      V_BOTTOM      (20)
#define      V_TOP      (360)
/* Keyboard Code Strings */
#define      F1      "\x1f\x3b"
#define      F2      "\x1f\x3c"
#define      F3      "\x1f\x3d"
#define      UP      "\x1f\x48"
#define      DOWN      "\x1f\x50"
#define      RIGHT      "\x1f\x4d"
#define      LEFT      "\x1f\x4b"
/* Some common definitions...
#define      TRUE      (1)
#define      FALSE      (0)
#define      N_PPG      (0) /* Index for parameter page */
#define      N_RPG      (1) /* Index for recipe page */
#define      N_SPG      (2) /* Index for scan page */
#define      parpg      (augpg[N_PPG]) /* parameter page pointer */
#define      recpg      (augpg[N_RPG]) /* recipe page pointer */
#define      scapg      (augpg[N_SPG]) /* scan/graphics page pointer */
/* Global declarations...
#include "g\src\gldecl.h"
#include "kevtypes.h"
/* Variables....
extern char      recfile[], /* filename of the recipe list */
errfile[], /* filename for errors */
extern int      pgindx, /* index of current hardware page */
cslope;
extern double      wu; /* in-memory data structure */
extern RFASRS      dirrec, /* Present rec. direct. entry */
SWEEP      recipe; /* Current sweep template */
extern PAGE      *augpg[], /* pointers to display pages */
extern FLAGS      flag; /* Program status flags */
extern CTABLE      sca_tab[], rec_tab[], /* Command tables */
par_tab[],
*tabptr[]; /* Pointers to those tables */
extern int      cmd_tab_len[], /* Lengths of Command Tables */
extern MENU      rmenu[], /* Text and coordinates */
pmenu[],
smenu[];

/* Function names...
extern      reclast(), /* Recipe page routines */
recnext(), recselect(), recreplace(),
Sete0(), Setel(), recsave(), recdelete(),
Setau(), Setit();
extern char      *getstamp();
extern SWEEP      *sweepcpy(SWEEP *, SWEEP *),
*sweepwt(SWEEP *, FILE *),
*sweepprd(SWEEP *, FILE *);
extern RFASRS      *RFAwt(FILE *, RFASRS *),
*RFArd(FILE *, RFASRS *);
extern int      RFAinit(RFASRS *),
RFAplot(RFASRS *),
o_rd(FILE *, VECTOR *),
o_wrt(VECTOR *, FILE *),
Parse(CTABLE *, int);
extern int      /* Parser function definitions...
augabort(), Augfine(), augplot(), /* Kevscan(),
comment(), Helpag(), Hold(), ignore(),
Parpag(), Again(), RecDef(), Recpag(),
Redrawsc(), RFAsave(), Syscmd(), Scapag():
extern int      tprintf(unsigned,unsigned,unsigned,char*),
tclr(unsigned,unsigned,unsigned);
extern int      strtclk(double),tstclk();

/*NAME:      kevscan.c
SYNOPSIS:    kevscan();
FUNCTION:    Perform an kevsweep.
#include <malloc.h>
#include "srslib/lockin.h"
#include "kevprob.h"
static      ScInit(), ScFine();
static int      Delay(double);

```

Figure 24: Software listing (Continued)

```

static double      el,          /* desired ending sweep energy */
                  dwell,       /* starting dwell time */
static int         delta;      /* sweep step size */
static DATA_TYPE  np;         /* desired number of data points */
extern double      *p;         /* Array of data points */
kevsan() /* Ramp the HV supply for Auger spectrums */
{
    PAGE
    switch( flag.value&0x07 )
    {
        case 0: /* Sweep is not running and RFA is not saved */
            tprintf(smenu[6].x,smenu[6].y,0,
                "Last spectrum NOT saved...Continue? [y/n]");
            if( getch() != 'y' )
                tclr(smenu[6].x,smenu[6].y,45);
            return;
            tclr(smenu[6].x,smenu[6].y,45);
            flag.x.wusaved = TRUE;
        case 4: /* Sweep is not running and RFA is saved */
            ScInit();
            break;
        case 3: /* Sweep is paused */
            move( wu.spectrum.el, p[wu.spectrum.npts-1]);
            flag.x.paused = FALSE;
            break;
        default:
            putch(7); perror("RFA flag error");
            return;
    }
    /* Main sweeping loop */
    while( (wu.spectrum.el<el) && (wu.spectrum.npts<np)
        && (flag.x.sweeping) && (!flag.x.paused))
    {
        Delay(wu.dwell); /* Stabilize the system */
        cp = seecap(); setcap(scapg); /* Draw data point */
        draw
        {
            wu.spectrum.el+=delta,
            p[wu.spectrum.npts++]--Disprd() * cslope;
        };
        setcap(cp);
    }
    ScFine();
}
static ScInit()
{
    PAGE
    *cp;
    RFAud(); /* Update the rfa data structure with lockin settings. */
    el = wu.spectrum.el;
    np = wu.spectrum.npts;
    dwell = wu.dwell;
    delta = (wu.spectrum.el-wu.spectrum.e0)/(wu.spectrum.npts-1);
    if( wu.spectrum.d != (DATA_TYPE *)NULL )
        free( (char*)wu.spectrum.d );
    p = wu.spectrum.d
        = (DATA_TYPE *)malloc(wu.spectrum.npts*sizeof(DATA_TYPE));
    cp = seecap();
    setcap(scapg);
    setsci(wu.spectrum.e0, -wu.sens * cslope, el, wu.sens*cslope);
    udscale();
    wu.spectrum.el=wu.spectrum.e0;
    move( wu.spectrum.e0, p[(wu.spectrum.npts-1)-1]--Disprd()*cslope );
    setcap(cp);
    /* Initialize flags */
    flag.x.wusaved = flag.x.paused = FALSE;
    flag.x.sweeping = TRUE;
}
static int Delay(tau)
double tau; /* dwell time in seconds */
{
    strtclk(tau*1000.);
    while(tstclk()) /* Loop until time is up */
        Parse(tabptr[pgindx], cmd_tab_len[pgindx]);
    if ( pgindx == N PPG)
        udednp(); /* Only when viewing the parameters */
}
Hold()
{
    flag.x.sweeping = flag.x.paused = TRUE;
}
static ScFine()
{
    switch (flag.value&0x03)
    {
        default: /* Sweep has been aborted */
        case 2: /* Sweep ended normally */
            wu.dwell = dwell; /* Reset dwell */
            flag.x.sweeping = flag.x.paused = flag.x.wusaved =

```

Figure 24: Software listing (Continued)

```

                                FALSE;
                                break;
                                /* Sweep has been paused */
                                break;
                                }
                                udednp(); /* One last time */
                                )
/*
Again()
{
    el = 2. * el;
    np = 2 * np;
    setscl(wu.spectrum.e0, -wu.sens, el, wu.sens);
    p = wu.spectrum.d
        = (DATA_TYPE *)realloc((char*)p,np*sizeof(DATA_TYPE));
    udscale();
    RFAplot(&wu);
}

/*NAME: kevprob.c
FUNCTION: Main drive for Kelvin Probe analysis.
#include <signal.h>
#include <stdlib.h>
#include "kevprob.h"
/* General stuff including
/* structures and global definitions
/* Global definitions
#include "globals.h"
augabort()
{
    signal(SIGINT,augabort);
    flag.x.sweeping = flag.x.paused = FALSE;
}

main()
{
    int    ans;
    Expinit(); /* Initialize the experiment */
    onexit(Augfine); /* Set up normal escape route */
    signal(SIGINT,augabort); /* Set up abnormal escape route */
    while(1) /* Parsing Loop */
        Parse(tabptr[pgindx],cmd_tab_len[pgindx]);
        /* Update parameter page */
}

static Expinit() /* Initialize */
{
    FILE    *fd;
    freopen(errfile,"a",stderr); /* Error file for debugging */
    perror("\nNEW RUN... ");
    perror(getstamp()); /* Write time of run */
    perror("\n");
    RFAinit(&wu); /* initialize the RFA data structure */
    ibinit(); /* Init GPIB port */
    RFAud(); /* Read current lockin config. */
    /* Calculate calibration slope. */
    Kevcalb();
    /* Set up software rasters used in experiment */
    rdinit(3); /* Initialize raster display */
    pgindx = N RPG; /* Recipes come up first */
    recpg = hdwrgpg;
    recinit(); /* Draw current batch of recipes */
    scapg = pgcreate(8); /* For graphics data */
    drscapg(); /* Draw the scan page */
    parpg = pgcreate(3); /* Parameters */
    drparpg();
    setcap(recpg); /* Set first interactive page */
    /* Set up flags */
    flag.value = 0x04; /* Saved, not sweeping, not paused */
}

Augfine() /* Exit the program gracefully */
{
    fflush(); /* Clean up explicitly */
    if( !flag.x.wusaved )
    {
        augpg[pgindx] = savhpg();
        setdm(3); setcap(hdwrgpg); erase(0.,0.,0.);
        tprintf(2,10,4,"The current sweep is NOT saved");
        tprintf(5,8,4,"Enter 'n' if you do NOT wish to save it> ");
        if( getch() != 'n' ) RFAsave();
    }
    rdfine();
}

/*NAME:KEVCAL
CALIBRATE THE KELVIN PROBE.

METHOD:
RAMP THE BIAS ON THE PADDLE OF THE PROBE FROM -1.0V
TO 1.0V, WHILE READING THE LOCK IN DISPLAY. THEN CALCULATE
THE INTERCEPT AND THE SLOPE WHERE THE SET OF X DATA IS
LOCK IN RESPONSE AND THE Y DATA IS THE BIAS VOLTAGE.
THE SLOPE GIVES ME A PROPORTIONALITY CONSTANT FOR THE
SCAN SO I KNOW WHAT THE CONTACT POTENTIAL DIFFERENCE
IS GIVEN SOME LOCK IN READING.

#include "kevprob.h"
#define num (21) /* NUMBER OF DATA POINTS */
#define bport (6) /* DA PORT ON SR510 USED FOR BIASING */
double cslope;

```

Figure 24: Software listing (Continued)

```

Kevcalb()
{
    FILE      *fp,*fopen();
    double    detime,Disprd(),volt,delvolt;
    float     x[num],y[num],sx,sy,st2,sxoss,ss,t,b;
    int       i;
    printf("\nHAVE PATIENCE I AM CALIBRATING THE BIAS ON THE LOCKIN\n");
    printf("WHEN DONE I WILL STORE THE DATA IN CAL.DAT FOR PERUSAL.\n");
    delvolt = .1;volt = -1.;
    /*
    DETIME IS THE "SETTLING" TIME FOR THE LOCK IN TO STABILIZE AFTER
    CHANGING THE BIAS.
    */
    detime = 10. * wu.pretc;
    fp = fopen("cal.dat","w");
    for ( i = 0; i < num; i++ )
    {
        Dtoawts(bport,volt);          /* APPLY BIAS VOLTAGE          */
        pause(detime);                /* STABILIZE SYSTEM          */
        /* READ LOCK IN RESPONSE          */
        fprintf(fp,"%e %e\n",x[i] = (float)Disprd(),y[i] = volt);
        volt += delvolt;
    }

    fclose(fp);

    /* CALCULATE SLOPE AND INTERCEPT          */
    sx = 0.; sy = 0.; st2 = 0.;cslope = 0.;b = 0;
    for ( i = 0; i < num; i++ )
    {
        sx += x[i];
        sy += y[i];
    }
    ss = (float)num;
    sxoss = sx/ss;
    for ( i = 0; i < num; i++ )
    {
        t = x[i] - sxoss;
        st2 += t * t;
        b += t * y[i];
    }
    cslope = b/st2;
    b = (sy - sx * cslope)/ss;
    fprintf(fp,"nslope %e and the intercept is %e ",cslope,b);
    /* NULL CURRENT BY APPLYING BIAS VOLTAGE EQUAL TO Y-INTERCEPT          */
    dtoawts(6,b);
}

/*NAME:      shgscan.c
SYNOPSIS:   shgscan();
FUNCTION:   Perform an SHG sweep.
#include <malloc.h>
#include "srslib/lockin.h"
#include "kevpob.h"
#define APORT 1
#define BPORT 2
static int   ScInit(), ScFine();
static int   Delay(double);
static double el,                /* desired ending sweep energy          */
              dwell,             /* starting dwell time                  */
              delta;             /* sweep step size                      */
static int   np;                /* desired number of data points        */
static DATA_TYPE *p;           /* Array of data points                */
extern double cslope;
kevsan() /* Ramp the HV supply for Auger spectrums          */
{
    PAGE      *cp;
    switch( flag.value&0x07 )
    {
        case 0: /* Sweep is not running and RFA is not saved          */
            tprintf(smenu[6].x,smenu[6].y,0,
                "Last spectrum NOT saved...Continue? [y/n]");
            if( getch() != 'y' )
                tclr(smenu[6].x,smenu[6].y,45);
            return;
            tclr(smenu[6].x,smenu[6].y,45);
            flag.x.wusaved = TRUE;
        case 4: /* Sweep is not running and RFA is saved          */
            ScInit();
            break;
        case 3: /* Sweep is paused          */
            move( wu.spectrum.el, p[wu.spectrum.npts-1]);
            flag.x.paused = FALSE;
            break;
        default:
            putch(7); perror("RFA flag error");
            return;
    }
}

/* Main sweeping loop          */
while( (wu.spectrum.el<el) && (wu.spectrum.npts<np)
    && (flag.x.sweeping) && (!flag.x.paused) )
{
    Delay(wu.dwell);          /* Stablize the system          */
    /* Draw data point          */
    cp = seecap(); setcap(scapg);
    draw
    {
        wu.spectrum.el+=delta;
        p[wu.spectrum.npts++] = Dtoard(APORT)/Dtoard(BPORT);
    };
    setcap(cp);
}

```

Figure 24: Software listing (Continued)

```

        ScFine();
    }
static ScInit()
{
    PAGE      *cp;
    RFAud();   /* Update the rfa data structure with lockin
                settings. */
    el = wu.spectrum.el;
    np = wu.spectrum.npts;
    dwell = wu.dwell;
    delta = (wu.spectrum.el-wu.spectrum.e0)/(wu.spectrum.npts-1);
    if( wu.spectrum.d != (DATA_TYPE *)NULL )
        free( (char*)wu.spectrum.d );
    p = wu.spectrum.d
        =(DATA_TYPE *)malloc(wu.spectrum.npts*sizeof(DATA_TYPE));
    cp = seecap();
    setcap(scapg);
    setscl(wu.spectrum.e0, -2., el, 2.);
    udscale();
    wu.spectrum.el=wu.spectrum.e0;
    move( wu.spectrum.e0, p[(wu.spectrum.npts-1)-1]=
        Dtoard(APORT)/Dtoard(BPORT) );
    setcap(cp);
    /* Initialize flags */
    flag.x.wusaved = flag.x.paused = FALSE;
    flag.x.sweeping = TRUE;
}
static int Delay(tau)
double tau; /* dwell time in seconds */
{
    strtclk(tau*1000.);
    while(tstclk()) /* Loop until time is up */
        Parse(tabptr[pgindx], cmd_tab_len[pgindx]);
    if ( pgindx == N PPG)
        udednp(); /* Only when viewing the parameters */
}
Hold()
{
    flag.x.sweeping = flag.x.paused = TRUE;
}
static ScFine()
{
    switch (flag.values0x03)
    {
        default: /* Sweep has been aborted */
        case 2: /* Sweep ended normally */
            wu.dwell = dwell; /* Reset dwell */
            flag.x.sweeping = flag.x.paused = flag.x.wusaved =
                FALSE;
            break;
        case 3: /* Sweep has been paused */
            break;
    }
    udednp(); /* One last time */
}
/*
Again()
{
    el = 2. * el;
    np = 2 * np;
    setscl(wu.spectrum.e0, -wu.sens, el, wu.sens);
    p = wu.spectrum.d
        =(DATA_TYPE *)realloc((char*)p,np*sizeof(DATA_TYPE));
    udscale();
    RFAplot(&wu);
}
/*NAME: shg.c
FUNCTION: Main drive for SHG analysis.
#include <signal.h>
#include <stdlib.h>
#include "kevpob.h" /* General stuff including
                    /* structures and global definitions
#include "globals.h" /* Global definitions
augabort()
{
    signal(SIGINT,augabort); /* What to do on abort
    flag.x.sweeping = flag.x.paused = FALSE;
}
main()
{
    int ans;
    ExpInit(); /* Initialize the experiment
    onexit(Augfine); /* Set up normal escape route
    signal(SIGINT,augabort); /* Set up abnormal escape route
    while(1) /* Parsing Loop
        Parse(tabptr[pgindx],cmd_tab_len[pgindx]);
        /* Update parameter page
}
static ExpInit() /* Initialize */
{
    FILE *fd;
    freopen(errfile,"a",stderr);/* Error file for debugging
    perror("\nNEW RUN... ");
    perror(getstamp()); /* Write time of run

```

Figure 24: Software listing (Continued)

```

perror("\n");
RFAinit(&wu); /* initialize the RFA data structure */
ibinit(); /* Init GPIB port */
RFAud(); /* Read current lockin config. */
/* Set up software rasters used in experiment */
rdinit(3); /* Initialize raster display */
pgindx = N RPG; /* Recipes come up first */
recpg = hwrpg;
recinit(); /* Draw current batch of recipes */
scap = pgcreate(8); /* For graphics data */
drscap(); /* Draw the scan page */
parpg = pgcreate(3); /* Parameters */
drparpg();
setcap(recpg); /* Set first interactive page */
/* Set up flags */
flag.value = 0x04; /* Saved, not sweeping, not paused */
}
Augfine() /* Exit the program gracefully */
{
flushall(); /* Clean up explicitly */
if (!flag.x.wusaved)
{
augpg[pgindx] = savhpg();
setdm(3); setcap(hwrpg); erase(0.,0.,0.);
tprintf(2,10,4,"The current sweep is NOT saved");
tprintf(5,8,4,"Enter 'n' if you do NOT wish to save it> ");
if (getch() != 'n') RFAsave();
}
rdfine();
}

/*kevtypes.h Data structure for SHG Probe measurements
using the SRS-510 Lockin Amplifier. */
/* This file defines the OGRE data structures: SEGMENT and VECTOR */
#include "og/ogre.ds.h"
typedef struct
{
short version; /* current version code */
char tstamp[26]; /* time of day data was taken */
float dwell; /* seconds per point (sec.) */
float pretc; /* pre-time constant setting (sec.) */
float posttc; /* post time constant setting (sec.) */
float phase; /* lockin phase setting (degrees) */
float reffreq; /* reference frequency (Hz) */
float sens; /* sensitivity setting (V) */
short filter; /* 1: line in, 2: 2x line in, 4: bandpass in */
float dynres; /* dynamic reserve code */
char *comment; /* pointer to commentary array */
SEGMENT spectrum; /* data structure containing spectrum data */
} RFA_SRS;

/* Sweep recipe data structure WKF/WH 12 May 86
typedef struct
{
char label[16]; /* label of the sweep setup */
float e0; /* beginning of sweep */
float e1; /* ending "voltage" of sweep */
float tau; /* seconds per point */
int npts; /* number of points in sweep */
short vers; /* RFA version number */
} SWEEP;

/* Command table structure
typedef struct
{
char cmdst[4]; /* Unique identifier for command */
char help[25]; /* Help string, displayed by help fnct. */
int (*fn)(); /* Command function to execute */
} CTABLE;

/* Structure defining a unit of text
typedef struct menu
{
unsigned int x,y; /* coordinate of text */
char s[32];
} MENU;

/* Bit field program status flags
typedef union
{
struct
{
unsigned paused:1, /* (bit0)sweep paused flag.*/
sweeping:1, /* (bit1)sweep sweeping. */
wusaved:1; /* (bit2)sweep saved flag. */
} x;
int value;
} FLAGS;

/*NAME: kevprob.h
FUNCTION: Header file for Kelvin probe data acquisitionMicrosoft C Programs.*/
#include <stdio.h>
#include "kevdecl.h"
#define VERSION (1) /* Program version number */
/* Updated Aug/20/1986 (time string) */

/* Screen definitions */
#define V_LEFT (35)
#define V_RIGHT (635)
#define V_BOTTOM (20)
#define V_TOP (360)

/* Keyboard Code Strings */
#define F1 "\x1f\x3b"
#define F2 "\x1f\x3c"
#define F3 "\x1f\x3d"
#define UP "\x1f\x48"

```

Figure 24: Software listing (Continued)

```

#define DOWN          "\x1f\x50"
#define RIGHT         "\x1f\x4d"
#define LEFT          "\x1f\x4b"
/* Some common definitions... */
#define TRUE          (1)
#define FALSE         (0)
#define N_PPG         (0) /* Index for parameter page */
#define N_RPG         (1) /* Index for recipe page */
#define N_SPG         (2) /* Index for scan page */
#define parpg         (augpg[N_PPG]) /* parameter page pointer */
#define recpg         (augpg[N_RPG]) /* recipe page pointer */
#define scapg         (augpg[N_SPG]) /* scan/graphics page pointer */
/* Global declarations... */
#include "g\src\gldecl.h" /* Graphics structures and defs. */
#include "kevtypes.h"
/* Variables.... */
extern char          recfile[], /* filename of the recipe list */
errfile[], /* filename for errors */
pgindx; /* index of current hardware page */
extern int
extern double
extern RFASRS
extern SWEEP
wu; /* in-memory data structure */
dirrec; /* Present rec. direct. entry */
recipe; /* Current sweep template */
extern PAGE
extern FLAGS
extern CTABLE
augpg[]; /* pointers to display pages */
flag; /* Program status flags */
sca_tab[], rec_tab[], /* Command tables */
par_tab[], /* Pointers to those tables */
tabptr[]; /* Lengths of Command Tables */
extern int
extern MENU
cmd_tab_len[]; /* Text and coordinates */
rmenu[],
pmenu[],
smenu[];

/* Function names... */
extern
reclast(), /* Recipe page routines */
recnext(), recselect(), recreplace(),
Sete0(), Sete1(), recsave(), recdelete(),
Setau(), Setit(), Setnp(),

extern char
extern SWEEP
*getstamp();
*sweepcpy(SWEEP *, SWEEP *),
*sweepwt(SWEEP *, FILE *),
*sweeprd(SWEEP *, FILE *);

extern RFASRS
*RFawt(FILE *, RFASRS *),
*RFard(FILE *, RFASRS *);

extern int
RFAinit(RFASRS *),
RFAplot(RFASRS *),
o_rd(FILE *, VECTOR *),
o_wrt(VECTOR *, FILE *),
Parse(CTABLE *, int);

extern int
/* Parser function definitions... */
augabort(), Augfine(), augplot(), Kevscan(),
comment(), Helpag(), Hold(), ignore(),
Parpag(), Again(), RecDef(), Recpag(),
Redrawsc(), RFAsave(), Syscmd(), Scapag();

extern int
tprintf(unsigned,unsigned,unsigned,char*),
tclr(unsigned,unsigned,unsigned);

extern int
strtclk(double),tstclk();

```

MONTANA STATE UNIVERSITY LIBRARIES



3 1762 10032266 6