CHEMICAL RESEARCH, DEVELOPMENT & ENGINEERING CENTER

CRDEC-TR-318

MID INFRARED POLARIZED LIGHT SCATTERING
Applications for the Remote Detection of Chemical and Biological Contaminations

A. H. Carrieri
D. J. Owens
C. E. Henry
K. E. Schmidt
J. L. Jensen
DETECTION DIRECTORATE

J. R. Bottiger
J. O. Jensen
RESEARCH DIRECTORATE

C. M. Herzinger
S. M. Haugland
UNIVERSITY OF NEBRASKA AT LINCOLN
Lincoln, NE 68508

January 1992

Approved for public release; distribution is unlimited.

U.S. ARMY ARMAMENT MUNITIONS CHEMICAL COMMAND

Aberdeen Proving Ground, Maryland 21010-5423

Disclaimer

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorizing documents.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*

2. REPORT DATE
1992 January

3. REPORT TYPE AND DATES COVERED
Final, Dec 86 - Dec 91

4. TITLE AND SUBTITLE
Mid Infrared Polarized Light Scattering Applications for the Remote Detection of Chemical and Biological Contaminations

5. FUNDING NUMBERS
PR-1C162706A553C

6. AUTHOR(S) Carrieri, A.H., Owens, D.J., Henry, C.E., Schmidt, K.E.,* Jensen, J.L.,* Bottiger, J.R., and Jensen, J.O. (CRDEC), and Herzinger, C.M., and Haugland, S.M. (Univ. of Nebraska at Lincoln)

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
CDR, CRDEC, ATTN: SMCCR-DD/SMCCR-RSP-B, APG, MD 21010-5423
University of Nebraska at Lincoln, Lincoln, NE 68508

8. PERFORMING ORGANIZATION REPORT NUMBER
CRDEC-TR-318

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

A polarized IR scattering facility that measures all elements of the Mueller matrix was built here at the U.S. Army Chemical Research, Development and Engineering Center (CRDEC) for remote detection of contaminants (analytes) spread on natural and manufactured surfaces (background scattering interferent). The ellipsometer is a two-modulator design that when interfaced to a mathematical algorithm can be trained to emit beam CO_2 laser energies at incident angles that best contrast various analyte target backscattering signatures from background signatures. For a probable detection event, sets of Mueller matrices are measured at beam energies coinciding with resonant IR absorptions by the analyte and at incident angle where backscattering by that analyte surface is strong. Identification of the contaminant(s) in situ is determined by another algorithm that operates on a vector whose $N \leq 16$ components are the independent Mueller elements measured at the susceptible beam energy/angle parameters. This work documents the phase sensitive detection program at CRDEC from past developments to future applications.

14. SUBJECT TERMS
Phase sensitive detection
Remote sensing
Mueller matrix
Neural network
Vibrational Circular Dichroism
(continued on page ii)

15. NUMBER OF PAGES
483

16. PRICE CODE

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

14. Subject Terms (continued)

Molecular absorption
Ellipsometer
Backscattering

PREFACE

The work described in this report was authorized under Project No. 1C162706A553C, Reconnaissance, Detection, and Identification. This work was started in December 1986 and completed in December 1991.

The use of trade names or manufacturers names in this report does not constitute an official endorsement of any commercial products. This report may not be cited for purposes of advertisement.

This report has been approved for release to the public.

Acknowledgments

The authors recognize C. N. Merrow (STC, Incorporated, Hampton, VA) and D. Zeroka* (Lehigh University, Bethlehem, PA) for suggesting the biological and controlled substance simulants compiled in Table 5. We also thank our colleagues E. Bahar (University of Nebraska at Lincoln, Lincoln NE), C. Chabalowski (U.S. Army Ballistic Research Laboratory), and H. Hameka (University of Pennsylvania, Philadelphia, PA) for stimulating and fruitful discussions on *full wave* scattering theory and on the *ab initio* calculations of quantum chemistry on molecular electronic structures, equilibrium geometries, and vibrational modes in the analyte materials.

* **Professor Zeroka is now involved in theoretical VCD predictions of sugar and amine compounds.**

Blank

CONTENTS

LIST OF FIGURES

LIST OF TABLES

MID INFRARED POLARIZED LIGHT SCATTERING
Applications for the Remote Detection of Chemical and Biological Contaminations

1. INTRODUCTION

The work reported here originated with earlier measurements of bi-directional reflectance.[1] Those experiments measured the depolarized infrared reflectance component (a multiple-scattering effect referred to as volume reflectance) from contaminated soil, sand and other terrains and manufactured samples over the discrete energies of a grating-tunable CO_2 laser. We suspected the depolarized radiance to originate within the subsurface volume of the scattering substrate,[2] and be selectively absorbed by the interstitial liquid contaminant coatings when tuned through the analyte's IR resonance absorption frequencies. Consequently, the detected depolarized radiation component would have been transmitted through the liquid coatings and attenuated in exponential proportion to the product of coating thickness and absorption coefficient. Thus, the volume reflectance signals on resonance frequencies of the analyte can be related to film thickness of the contamination layers. (Separating the depolarized component reduces the problem to transmission of radiance from extended source, i.e. below the irradiation zone and through the contaminant layer.) We thus sought from the separated depolarized radiance a means to detect with finite probability (and certainly not uniquely) the presence of liquid contaminations on terrain, and approximate thickness of the coating layers through a set of differential volume reflectance measurements.

Our conclusion from analysis of these measurements was not encouraging. The minor depolarized IR scattering component (about 10% of the total scattering power) from agent simulant wetted soil and sand samples did not qualitatively reveal absorption by the contaminant at concentrations that would have proven fatal to life had it been an actual chemical agent. Furthermore, once absorption was detected at the highest concentrations, this method of detection could not singularly characterize the contaminant. It is unlikely that separate contaminants with different toxicity but overlapping extinction energies could be resolved through analysis of CO_2 laser reflectance spectra alone -- be it in total reflectance or its separated coherent and incoherent components.

New concepts were necessary, and in 1985 experimental and theoretical research programs were started on developing a phase-sensitive infrared scattering solution of chemical and biological warfare (CBW) agent detection problems. This new technology was certain to improve detection thresholds and provide quantitative information on physical and geometrical properties of the scatterer -- with interstitial contamination layers!! (We are developing techniques to use the rich phase information in the scattering EM waves that present reflectance-based systems, i.e., DISC/DIAL, do not or cannot measure.)

In the theoretical program, quantum chemistry codes are used to predict energies and absorption strengths of the contaminants at the molecular level. These infrared spectral intensities are converted into refractive indices that are accessed by a **Full Wave** electromagnetic wave scattering model used to predict the scatterer's 4×4 Mueller matrix. The Mueller matrix is a complete optical characterization of the scatterer, and is computed at beam excitation energies and backscattering angles that can best contrast a contaminant (referred to as the chemical analyte with an IR absorption moiety) from all other scatterers (background). In the experimental program, we are developing three ellipsometer sensors for production of a data base of Mueller elements representing scattering by aerosols and by liquid coatings spread across various surfaces. With guidance from a valid theoretical scattering model, the ellipsometer sensors can be made to operate at beam energies and backscatter angles that produce sets of Mueller elements that are susceptible to the contaminant and only that contaminant. The set of independent elements most sensitive to the analyte(s) are inputs to an algorithm designed to identify it, or establish non-presence.

The purpose, then, of this work is to characterize contaminated surfaces *in toto* through their Mueller matrix signatures, interpret these data elements, discern targeted contaminant information immediately (near real-time alarm), and quantify the threat target mass concentration (map). The analyte compounds of interest include chiral sugars and other enantiomers that preferentially absorb right- and left-circularly polarized light, and thus simulate Vibrational Circular Dichroism (VCD) in more complex biological structures. Background (interferent) materials of interest include manufactured and terrestrial (scattering) interferent surfaces such as soil, sand, concrete, asphalt, and treated metallics commonly used in military hardware. Other analytes we wish to target for identification are the chemical agent simulant class of phosphonated hydrocarbons and other liquids that exhibit at least one strong resonant IR vibrational normal mode.

Presented in Section 2 are fundamental definitions of photopolarimetry on which the ellipsometer is based. In Section 3, the types of measurements to be conducted and the important experimental parameters are discussed. In Section 4, the ellipsometer and its theory of operation is presented, and in Section 5, a Full-Wave light scattering model for rough surface scattering is introduced, currently under development at the University of Nebraska in collaboration with CRDEC. In this same section, three quantum chemistry software packages are briefly reviewed. These *ab initio* models predict least-energy group configurations in the analyte molecules, and their corresponding absorption spectrum. We will merge quantum molecular codes with the Full Wave polarized EM wave scattering code, so that a comprehensive model can be used to predict linear and circular birefringence; VCD, depolarization, and other polarization dependent scattering phenomena.

The integrated **Full Wave** scattering and quantum molecular codes will be tested through a systematic set of experiments and if found valid guide the development of a field ellipsometer sensor capable of multi-target detections, by simulating the entire experiment under various field scenarios. These simulations of Mueller backscattering elements will direct us toward optimizing those parameters most crucial in development of a prototype version of the 9-channel analog laboratory phase sensitive detection system (first generation, Section 4), a digital data acquisition system counterpart (second generation, Section 6.2), and neural network (third generation, Section 6.4). In Section 6 and the concluding Section 7, present development status of the experimental ellipsometer systems is updated, methods of advanced digital data acquisition and processing techniques are suggested for a future detection module of the ellipsometer, the structure of an initial data base is outlined, and a brief discussion on our initial work with neural network computing is addressed.

2. KEY DEFINITIONS USED IN PHOTOPOLARIMETRY

References 3 through 7 cover in some detail the definitions and conventions used in ellipsometry. Among these, **Shurcliff's** book[4] describes polarized light best at the introductory level, and includes useful Mueller matrices for standard birefringent and polarizing optics that make up these ellipsometer instruments. The standard texts by **van de Hulst**[8], **Bohren and Huffman**[9] are frequently referenced sources for Mie and Rayleigh scattering by arbitrary particles. Moreover, Section 5 is reserved for a brief but more focused theoretical treatment of the Mueller elements related to scattering by dielectric and metal surfaces of varied roughness (changing heights and slopes), a subject we later concentrate on for this detection problem.

We begin our discussion by defining a* Stokes vector and Mueller matrix: a column vector representing total and partially-polarized states of propagating electromagnetic laser radiation, and a matrix that transforms these states due to reflection, transmission, scattering, or absorption events. They represent the fundamental operational principles of the ellipsometer instruments. Notation of J.D. Jackson[10] in defining the Stokes parameter is used throughout this work. We also choose f symbols for Mueller matrix elements of the scattering sample, as denoted by R.C. Thompson, J.R. Bottiger and E.S. Fry.[11]

2.1 Stokes Vector.

The Stokes vector is a four-element column matrix that represents a beam of total or partially-polarized light. Consider a homogeneous plane wave propagating in a direction of the vector $\mathbf{k} = \hat{n}k$,

$$\mathbf{E} = (E_1\hat{\epsilon}_1 + E_2\hat{\epsilon}_2)e^{i(\mathbf{k}\cdot\mathbf{x}-\omega t)} \tag{1}$$

where the complex electric field amplitudes, E_1 and E_2, have amplitude (a) and phase (δ).

$$E_1 = a_1 e^{i\delta_1} \;,\; E_2 = a_2 e^{i\delta_2} \tag{2}$$

Elements of the Stokes column matrix are defined in terms of the coordinate basis vectors $\hat{\epsilon}_1$ and $\hat{\epsilon}_2$ as:

$$\mathbf{s} = \begin{pmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{pmatrix} = \begin{pmatrix} <|\hat{\epsilon}_1\cdot\mathbf{E}|^2> + <|\hat{\epsilon}_2\cdot\mathbf{E}|^2> \\ <|\hat{\epsilon}_1\cdot\mathbf{E}|^2> - <|\hat{\epsilon}_2\cdot\mathbf{E}|^2> \\ <2Re(\hat{\epsilon}_1\cdot\mathbf{E})(\hat{\epsilon}_2\cdot\mathbf{E})> \\ <2Im(\hat{\epsilon}_1\cdot\mathbf{E})(\hat{\epsilon}_2\cdot\mathbf{E})> \end{pmatrix} =$$

$$\begin{pmatrix} <a_1^2> + <a_2^2> \\ <a_1^2> - <a_2^2> \\ <2a_1 a_2 \cos(\delta_2-\delta_1)> \\ <2a_1 a_2 \sin(\delta_2-\delta_1)> \end{pmatrix}. \tag{3}$$

* A linearly-polarized beam that is multiply scattered from terrestrial surfaces will has a depolarized component in the IR.[1] The Stokes calculus is necessary when predicting these transformations.

Brackets < > denote time-averaged values. Notice that s_0 is an additive intensity term and suggests total scattering power, while positive (negative) s_1 suggests a majority of horizontal (vertical) linear polarization. Both s_2 and s_3 contain phase difference terms, and can suggest + (clockwise) or - (counter clockwise) handedness of elliptical EM waves. As an illustration, consider the Stokes parameters for three polarization states: (a) *unpolarized*[**] , (b) horizontal linear and (c) right circular. For case (a), by definition of unpolarized light, $< | \hat{\epsilon}_1 \cdot \mathbf{E} |^2 > = < | \hat{\epsilon}_2 \cdot \mathbf{E} |^2 >$, and $s_3 = s_4 = 0$, since both sine and cosine terms average to zero independently of the amplitudes a_1 and a_2. When dividing all parameters by s_0, the normalized Stokes vector for unpolarized light becomes (1,0,0,0). For case (b), $\delta_1 = \delta_2,\ a_2 = 0,\ \hat{\epsilon}_2 \cdot \mathbf{E} = 0$, and therefore the normalized Stokes vector for linearly polarized light in the horizontal direction is (1,1,0,0). For the final case (c), $\delta_1 = \delta_2 - \frac{\pi}{2},\ a_1 = a_2,$ and therefore the normalized Stokes vector for right circular polarized light is (1,0,0,1).

The Stokes vector of the incident beam and the scattered radiance, collected from the irradiation zone in some small solid angle from the sample, changes continuously and periodically when operated on by the ellipsometer's transmitter and receiver photoelastic modulation (PEM) optics, respectively. The following Table 1 lists a selection of six states generated by specific retardation amplitudes in the PEMs, and swept periodically at 33.980 KHz (transmitter) and 31.896 KHz (receiver) transducer frequencies, rates at which the PEM's octagonal ZnSe crystal are stressed-then-relaxed. We later make reference to this table when the topics of system matrices and diagnostics of MCT detector signal are discussed.

Table 1. Normalized Stokes vectors for six polarization states.

HORIZONTAL LINEAR	VERTICAL LINEAR	+45° LINEAR	-45° LINEAR	RIGHT CIRCULAR	LEFT CIRCULAR
$\begin{pmatrix}1\\1\\0\\0\end{pmatrix}$	$\begin{pmatrix}1\\-1\\0\\0\end{pmatrix}$	$\begin{pmatrix}1\\0\\1\\0\end{pmatrix}$	$\begin{pmatrix}1\\0\\-1\\0\end{pmatrix}$	$\begin{pmatrix}1\\0\\0\\1\end{pmatrix}$	$\begin{pmatrix}1\\0\\0\\-1\end{pmatrix}$

2.2 Mueller Matrix.

The polarization state of an electromagnetic wave is generally altered by reflection, transmission, scattering and/or absorption processes. The scattering phase and amplitude information represented by the Mueller matrix depends on the physical properties of the interactive medium and its geometry or topographic detail. A transformation of the incident beam Stokes vector after its backscattered interaction with a material boundary into a new vector defines the Mueller system matrix operator.

[**] Unpolarized light implies nonpreferential electric field directional properties, i.e., the total electric field vector is equally probable of lying in any orientation in the scattering plane over the time in which a measurement is made.

The Mueller Matrix Transforms the Input Stokes Polarization State Vector.

$$\mathbf{s}_j = \mathbf{f}_{jk}\,\mathbf{s}_k \tag{4}$$

In Equation (4), $\mathbf{s}_k$ are Stokes vector components of the incident beam, and $\mathbf{s}_j$ are resultant components after beam → medium interaction. That transformation is given by $\mathbf{f}_{jk}$, a 4x4 operator whose elements represent a complete geometrical and physical description of a linear medium interacted by the beam – it is the Mueller matrix. (The non-linear phenomena of stimulated Raman and Brillouin scattering, second harmonic generation, etc., cannot be interpreted by a linear operator. The medium may require a tensor description of permittivity and (for magnetic materials) permeability. However, the electric field intensity required to produce such effects is far beyond the incident irradiations by these ellipsometer probe beams.) In Figure 1, we graphically illustrate the beam scattering geometry. The Mueller matrix field and a few of its elemental interpretations are schematized in Figure 2.

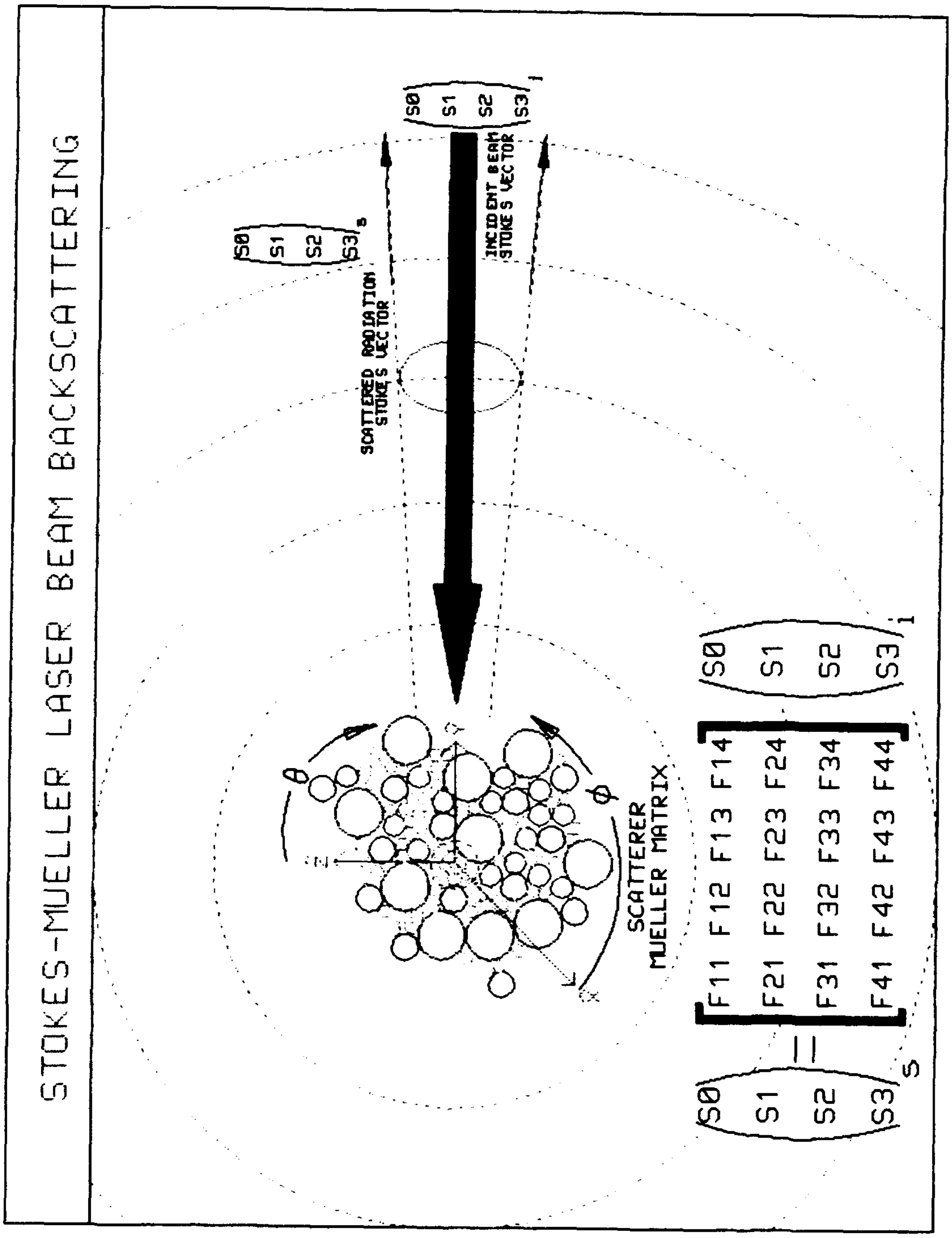


Figure 1. Stokes vector - Mueller matrix laser beam backscattering from a particle conglomerate.

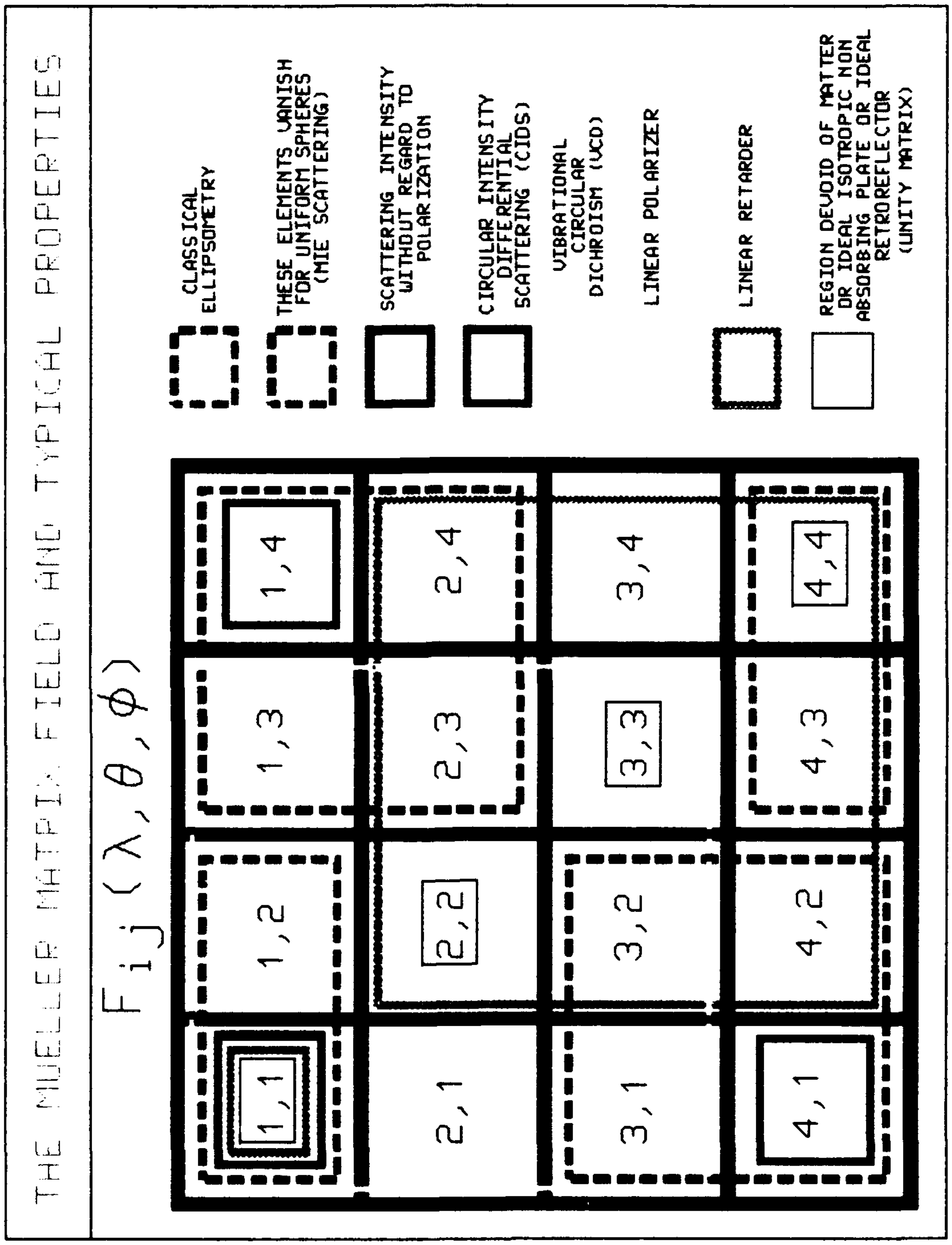


Figure 2. The Mueller matrix and some of its typical properties with active sub-field elements.

In the following Table 2, Mueller matrices of the scattering sample (denoted **F**, the surface and subsurface contaminant layers are embedded in **F**) and optical components of the ellipsometer design are presented. The matrix elements of reflection, retardation, and polarization optics are used in latter sections of this work when all Mueller elements of the contaminated sample are correlated to primary and combination PEM modulator frequencies in the instrument's Fourier transformed MCT detector signal intensity spectrum.

Table 2. The Mueller matrices of individual optic components making up the 2-modulator, mid infrared IR ellipsometer systems. f_{ij} are matrix elements of the scattering sample, α and β are amplitude ratios of reflected-to-incident electric field components for light polarized parallel and perpendicular to the plane of incidence, respectively, σ is the reflection induced phase shift between these two components, θ is the polar (tilt) angle resulting from goniometer rotation, δ_0 and ω are peak retardance and frequency generated by the ZnSe phase modulators, respectively.

MUELLER MATRICES OF THE EXPERIMENTAL SYSTEM	
Scattering Sample	
$\begin{pmatrix} f_{11} & f_{12} & f_{13} & f_{14} \\ f_{21} & f_{22} & f_{23} & f_{24} \\ f_{31} & f_{32} & f_{33} & f_{34} \\ f_{41} & f_{42} & f_{43} & f_{44} \end{pmatrix}$	
Goniometer	
Mirror	Rotation operator
$\frac{1}{2}\begin{pmatrix} \alpha^2+\beta^2 & \alpha^2-\beta^2 & 0 & 0 \\ \alpha^2-\beta^2 & \alpha^2+\beta^2 & 0 & 0 \\ 0 & 0 & -2\alpha\beta\cos\sigma & 2\alpha\beta\sin\sigma \\ 0 & 0 & 2\alpha\beta\sin\sigma & -2\alpha\beta\cos\sigma \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos 2\theta & \sin 2\theta & 0 \\ 0 & -\sin 2\theta & \cos 2\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$
Linear Polarizers	
Horizontal(Vertical)	+(-)45 Degrees
$\frac{1}{2}\begin{pmatrix} 1 & +(-)1 & 0 & 0 \\ +(-)1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$	$\frac{1}{2}\begin{pmatrix} 1 & 0 & +(-)1 & 0 \\ 0 & 0 & 0 & 0 \\ +(-)1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$
Photoelastic Modulators	
Vertical(Horizontal)	+(-)45 Degrees
$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \cos(\delta_0\cos\omega t) & -(+)\sin(\delta_0\cos\omega t) \\ 0 & 0 & +(-)\sin(\delta_0\cos\omega t) & \cos(\delta_0\cos\omega t) \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\delta_0\cos\omega t) & 0 & +(-)\sin(\delta_0\cos\omega t) \\ 0 & 0 & 1 & 0 \\ 0 & -(+)\sin(\delta_0\cos\omega t) & 0 & \cos(\delta_0\cos\omega t) \end{pmatrix}$

3. EXPERIMENTAL APPROACH

In addition to Reference 11, the works of Williams[12], Roseler[13] and Vorburger et al.[14] seem relevant to this specific detection problem. Williams describes depolarization, cross polarization, and changes in ellipticity through rough surface scattering signatures by a Poincare sphere representation, while Roseler introduces a Fourier transform spectrometer formalism to yield a spectroscopic IR phase matrix measurement. Vorburger et al., discuss the ellipsometric parameters ψ and Δ for various textured surfaces, presenting data at 0.6328 μm and 0.5461 μm. They conclude that topographical roughness is a major random error source for inversion methods that map ψ and Δ parameters to physical properties of the scatterer. This is a major concern if polarized scattering can be successfully applied to the IR remote sensing problem: for a quantitative **in situ** infrared detection of amorphous or crystalline contaminants, its physical absorption property (where the imaginary part of its permittivity maximizes) is its key identifier, and that information must manifest itself consistently in the susceptible on/off resonance differential scattering Mueller elements. This issue is addressed again in Section 5.

From our most recent literature search, we have located other visible single-frequency bistatic photopolarimeter systems of one and four modulator design[15], with essentially the same detector analog electronics as first reported by Kemp[16], and later Hunt and Huffman.[17] The work of Whitt and Ulaby's[18] is also noted, reporting millimeter-wave polarimeter measurements, as is van Zyl's[19] paper on radar polarization signatures from rough surfaces in the backscattering plane.

We have not located in the open literature data published on mid IR Mueller matrix elements from rough surfaces of varied optical properties, e.g., the contaminated terrestrial scenarios. The instruments we report here were designed especially for this application, and provide an opportunity to acquire quantitative Mueller data elements, sensitive to perhaps sub-micron layers of contaminant spread across a surface, and conduct the necessary near real-time data processing and analysis that we hope will rapidly identify (alarm) and quantify (map) chemical and biological warfare (CBW) agents dispersed onto terrestrial and manufactured materials.

The ellipsometer remote sensors presented here consist of two photoelastic modulators driven at frequencies offset by 2.06 KHz, one operating on the incident beam the other on collected backscattered radiance. They are the Mueller matrix generators capable of optically computing nine elements simultaneously and all sixteen in sequence according to orientations of modulator-polarizer axes. One instrument will be used to analyze surfaces of a rigid and generally continuous texture; a nonporous military painted panel for instance. A similar laboratory instrument will analyze the natural soils and other granular porous surfaces without mechanical disturbance to the scattering sample. This instrument requires a goniometer optical device. A third instrument is intended for long-range field evaluation.

In conducting laboratory Mueller matrix measurements from soil and other granular-like samples, **in situ**, an opto-mechanical goniometer arm was built and later integrated into one ellipsometer optical design, between linear polarizing optics producing the system's initial and final Stokes vectors. The goniometer introduces three mirror optics to the ellipsometer's system matrix, a necessary but optically and computationally complex addition to the ellipsometer system and its data analysis functions. As we progress toward later sections of this work, the data analysis complications that arise as a result of the presence of these mirror optics will be made more clear. Moreover, we later present both software and hardware methods for compensating all goniometer mirror phase contributions so that the desired sample matrix elements is extracted from the detector system waveforms.

The order in which the incident beam is transmitted and reflected by system optic components, and scattered by the sample, defines the important **system product matrix**. Only in the goniometer-type and field instruments do **system** (measured) elements need to be transformed into **sample** elements ($f_{ij}[\lambda,\theta]$) by software and electronic decoding ($f'_{ij}[\lambda,\theta]$, Appendix II). These **sample** Mueller elements contain the important information on the analyte that we seek in making detection judgements.

3.1 Radiation Wavelength, Backscattering Angle, and Analyte Mass Density Dependencies of the Matrix Elements.

The experimental goal is production of a reliable data bank of Mueller matrix elements that most contrast background (terrain) and chemical/biological contaminant (analyte). These elements will be obtained as functions of wavelength of the probe beams; tuned specifically to the analyte, angle in the backscattering plane of the incident beams, and mass of contaminant deposited to the sample per unit area in the zone of beam exposure.

We reserve the latter Section 6.4 for a discussion of our starting data base structure and its management.

3.1.1 Wavelength Selection of the Irradiating Probe Beams.

The Mueller matrix elements are to be measured alternately at CO_2 beam energies that drive strong vibrational modes in the contaminant to be detected (sometimes referred to as analytical wavelengths) and at an energy where no contaminant molecular excitation is produced (reference).

In addition to a standard $C^{12}O_2^{16}$ laser source, isotopic carbon dioxide gases are part of the gain fill in three other lasers of the ellipsometer transmitter. The isotope lasers are used for purposes of extending the mid IR wavelength range in which the sample can be irradiated, and to fill in wavelength gaps between P- and R-branch $C^{12}O_2^{16}$ transitions (widen and make more continuous the wavelength coverage). The beam wavelength selections (i.e., emissions with enough power for measuring scattered radiance) range from 9.0 μm at the R(40) line 00^01–02^00 band in the $C^{12}O_2^{18}$ laser, to 12.08 μm at the P(44) line 00^0–10^00 band of the $C^{14}O_2^{16}$ laser. For example, consider detection of the liquid chemical agent simulant DIMP, $CH_3PO(OCH(CH_3)_2)_2$. Absorption band assignments of this analyte are two intense ν (P-O-C) vibrational modes at 10.169 μm and 9.884 μm, and a less intense P–CH_3 rocking mode at 10.902 μm. Typically, three of four lasers will be tuned to wavelengths that align to peak maximum in the vibrational bands (analytical wavelengths) of the target, while the fourth laser is *off*-tuned to generate reference (background) Mueller data elements (Appendix I). The full Mueller matrix 16-element field recorded between beams and those that possess susceptible behavior to the analyte (change abruptly during laser switching between resonance and reference wavelengths) are singled out as detection candidates.

3.1.2 Angle to the Sample of the Incident Probe Beams.

Generally, standoff active sensors are monostatic backscattering systems where the transmitter source and collection optical receiver are stationary and co-located as a unit. The ellipsometer facility at CRDEC is also monostatic, measuring the Mueller elements in backscattering directions over all angles of incidence. To accommodate such measurements from granular materials and surfaces of continuous texture, two separate ellipsometers were constructed and are schematized in Figures 4a-b. For the rigid surfaces, like painted metallics and pelletized substrates, the sample is positioned vertically and rotated along an axis perpendicular to the plane of incidence defined by the incident beam at polar angles ranging from -89^0 to $+89^0$ (Figure 4b). For loose particles, like natural soil and sand, the sample lies undisturbed in its horizontal position while a goniometer transceiver arm delivers the beam to the surface over all polar angles in the upper hemisphere and directs backscattering to the MCT detector (Figure 4a) by use of three flat mirrors, each positioned 45^0 to its incident beam. Thus, the reference plane in which the Stokes vectors of transceived radiations are measures vary as the goniometer arm rotates out of the reference plane – a complication this design presents in data analysis. Introduction of these mirrors between linear polarizers producing incident- and final-Stokes vectors of the ellipsometer design causes other complications. Matrix signatures from the **sample** must now be separated from those elements by the ellipsometer **system**. The phase changes imparted by each goniometer mirror optic can be compensated for either optically, or mathematically through a series of calibration experiments (Section 4.6.3, and Appendix II).

A phenomenon of direct backscattering from optically rough surfaces is the so-called **opposition effect**, which may have benefit for these backscattering ellipsometers as detection instruments. The effect is predicted by **Full Wave** theory (see Section 6), and states that an enhanced incoherent *backscattering* component of radiance results from scattering at a randomly rough boundary interface. An experiment conducted by **Mendez and O'Donnell**[20] show results that tend to confirm this prediction. **Bohren and Huffman**[9] show that if the irradiated surface is modeled as a random array of identical dipole oscillators, then the total backscattering radiation field is generally incoherent and dependent on spacing between the dipole radiators, unlike the forward scattering field that is totally in phase and independent of separation between radiating particles. Since enhanced backscattering manifests itself as a noncoherent component of scattering, it would appear, then, that a spatially integrated backscattering signal from randomly rough surfaces could vary dramatically in the scattering phase signature (*viz*, fluctuations in the Mueller matrix backscattering 'picture' as the select irradiation beam spatially scan a contaminated terrestrial surface.) Again, we emphasize that a changing refractive index in the analyte between beams alternating on target resonance and off target absorption resonance must be revealed in a reproducible manner for making a successful, unique, physical detection.

3.1.3 Contaminant Deposition to the Sample.

Three common liquid simulants are chosen as initial detection targets: nonvolatile dimethyl siloxane fluid SF96 (General Electric nomenclature), and the more volatile DMMP and DIMP phosphonated organic compounds. Presented in Figure 3 are infrared imaginary refractive indices of all three compounds, and listed in Table 4 are the compounds' strongest mid IR extinction frequencies. (The imaginary part of refractive index is proportional to the absorption coefficient of the medium. It can be obtained through Kramers-Kronig analysis of either absorption or reflectance spectra[10,21].)

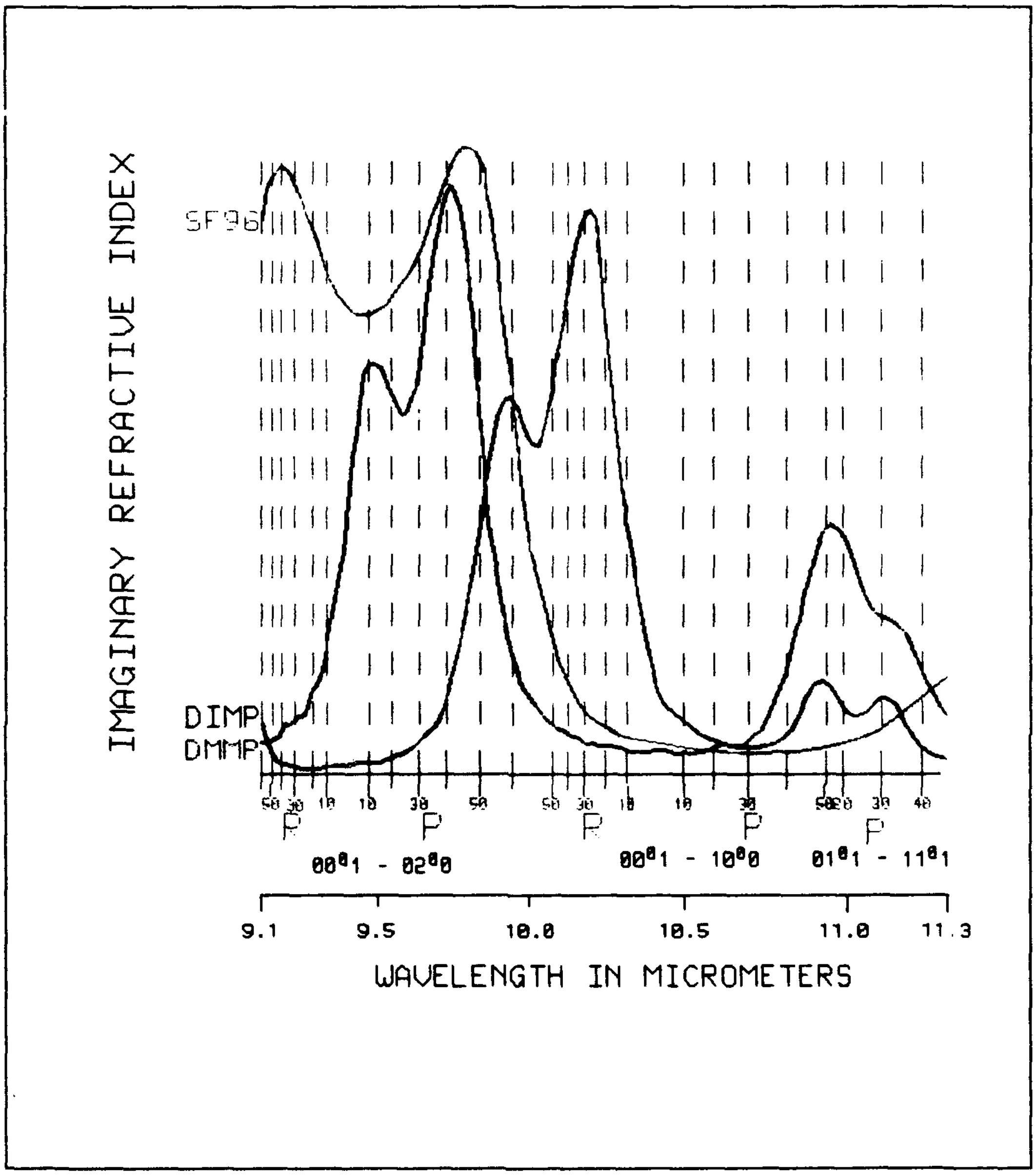


Figure 3. Spectra of the imaginary component of complex refractive index (absorption) in volatile liquid chemical agent simulants dimethyl methyl (DMMP) and diisopropyl methyl (DIMP) phosphonates, and nonvolatile polydimethylsiloxane (SF96 - General Electric nomenclature). The dashed vertical lines are some of the allowed CO_2 laser transitions where the ellipsometer can probe. See Appendix I for a complete list of transition assignments and nomenclature for these ellipsometer's beam sources.

The relationship of applied mass contamination on a surface and the Mueller elements, at angle-wavelength domains specific for detection of the contaminant, could yield useful information to a trained signal processor for quantification of the analyte once detection is established from an appropriate data bank.

This data bank of information can come from a set of experiments that sequential measure all Mueller elements from some scattering substrate when dry, then coated with increasing quantities of the analyte. Consider a soil substrate as the background scatterer. First, all sixteen matrix elements are measured from the unwetted soil sample and referenced as the background matrix set of Mueller elements over a range of beam energies. Next, the soil is contaminated by the analyte compound in low concentration and the elements are once again measured, at beam energies specific to the analyte. The analyte-susceptible Mueller elements are singled out for discriminant features, then the sample is re-wetted with additional analyte solution and the matrix elements remeasured. Measurements continue to an analyte density typically less than 20 gm cm^{-2}. These kinds of controlled analyte dispersion experiments will correlate a pattern in the analyte-susceptible Mueller elements from low to high analyte mass density, with the uncontaminated elements serving as a reference frame of background information. The element data can subsequently be input to an appropriate decision making algorithm (Appendix VI) that electronically filters background data and discerns the informational content of the analyte-specific Mueller element signals, correlating their pattern to quantity of analyte spread across/into the surface. In a typical data trial, contaminant mass densities of 2-20 gm m^{-2} are deposited in 2-5 gm m^{-2} increments.

Experiments for classifying biological contaminants spread across terrain and manufactured surfaces through their Mueller matrix features are being discussed now. We feel some control experiments must first be performed to first recognize whether resonant absorption by these analytes can reveal a matrix signature in the absence of interferent scatterers. Initial experiments would include measurements from aqueous suspensions of simulant organisms, generated bioaerosols in a chamber, and liquid and crystalline compounds like sugars that have known molecular symmetrizations exhibiting dichroism (e.g., vibrational circular dichroism or VCD). If features of detection are clearly evident in differential Mueller element data sets then the experiments are refined (i.e., concentrate on the optimum angle-wavelength domains producing susceptible Mueller elements), redone with interferent scattering (an **in situ** contaminated surface scenario), then analyzed for the element features separate from the terrain element signatures. The experiments would be extended to include other disseminated specimens like sterile B. Anthracis, B. Cereus, B. Thuringiensis, E. Coli, and fungus spores.

3.1.4 Time Dependence of Matrix Elements After Aerosol Ejection.

In the experimental program we monitor the susceptible Mueller elements after deposition of the contaminant. The expected temporal fluctuations in these Mueller elements could conceivably sense diffusion and spread of contaminant across and into the substrate. Evaporation of the volatile liquid DMMP and DIMP analytes would also manifest temporal fluctuations in these elements. Heat liberated by an absorbed beam energy will deplete the thin volatile analyte surface coatings, thereby generating a contaminant vapor cloud above the irradiation zone. It is difficult to predict whether this vapor presence above the surface will cause significant alterations in the ac (phase spectrum) scattering components. (Backscattering from the vapor itself is insignificant, however, scattering from the terrain transmitting through the vapor cloud may indeed be sensitive to some matrix elements.) Transmission of scattered radiance through the vapor plume on analyte resonant wavelengths would have, however, some attenuation effect on f_{11}, i.e., the dc component of the MCT signal, it being a measure of scattering intensity.

Other changes in f_{ij} from volatile simulants, if indeed they are measurable, are likely to result from changing scattering surface topography due to evaporation and diffusion of liquid analyte layers or, in biological specimens, a changing morphology under exposure by an absorbing beam.

4. THE ELLIPSOMETERS: THEORY OF OPERATION AND DESIGN

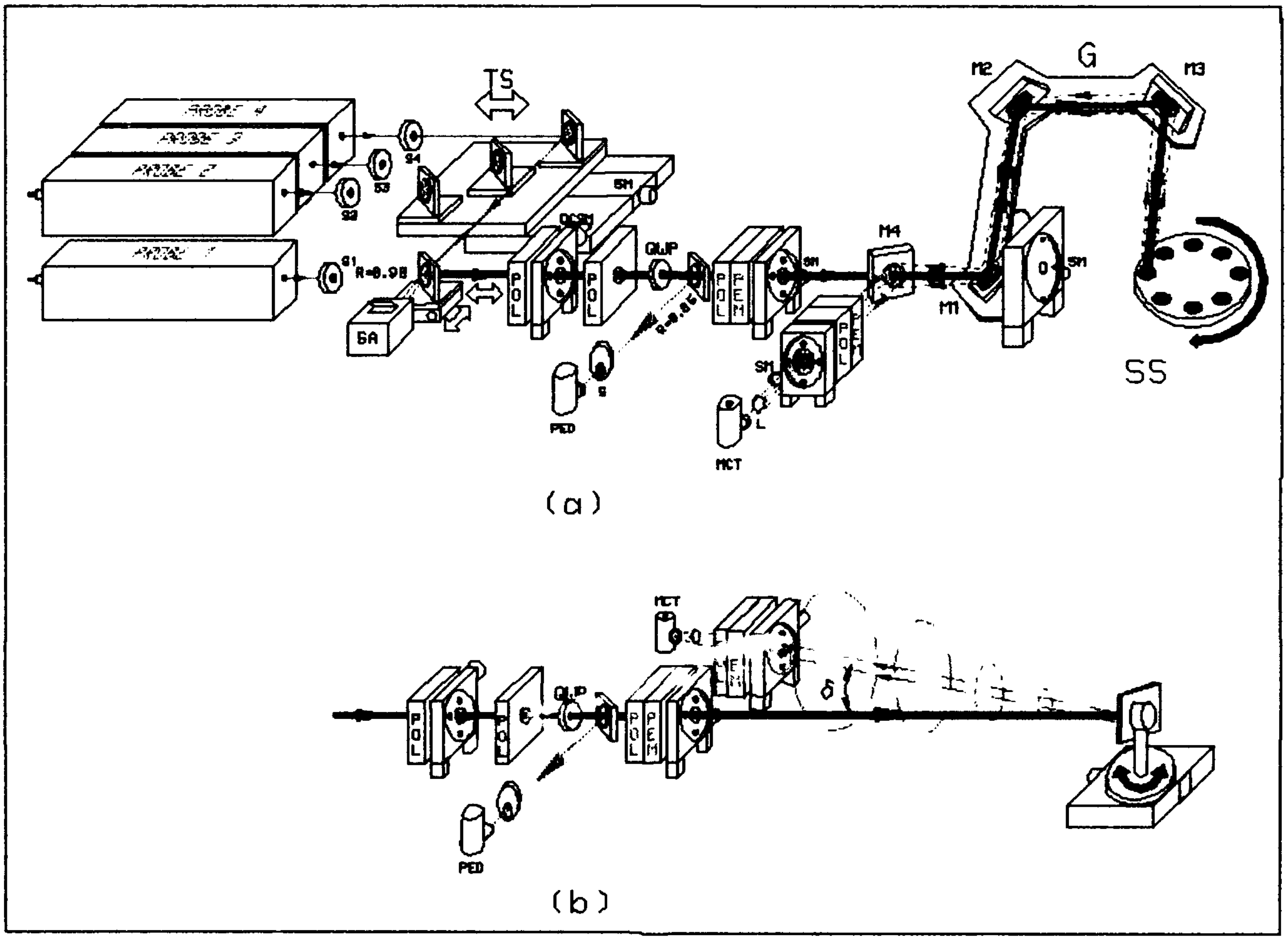
The CRDEC Mueller matrix ellipsometer (MME) systems are similar in optic and analog data acquisition system designs to the facility reported in Reference 11. Our experimental systems are multi-laser and multi-infrared wavelength (9.0-12.1 μm), monostatic backscattering, and of a two modulator design; theirs a single-visible wavelength (0.6328 μm), bistatic, four modulator system. A disadvantage of these two-modulator systems is that only 9 of 16 matrix elements can be simultaneously measured. It requires four sequential angular permutations between each of this ellipsometer's linear polarizer-retarder optic units (Section 4.4) to complete the measurement of the 16-element field. We found it necessary, for precision and repeatable measurements, to stepper-motor control and computer automate all optic translation and rotation sequences. It follows, that computer data collection operations are synchronous to indexing of the optics hardware.

Advantages of the two- verses four-modulator systems include less multiplicative error in the modulation crystals due to: (a) imperfections in the crystal, (b) thermal distortion by heating by the incident IR beam, (c) misalignments in optics, and (c) beam transmission offset from the PEM crystalline axis. A less chance of mixing of the Mueller elements (i.e., overlap of the Fourier intensities in the transformed scattergram) is realized because of the wider separation between primary and overtone modulator frequencies. (The requirements imposed on the phase-sensitive detector boards are less restrict because of the greater frequency separation, and thus less chance of harmonic overlap, between Fourier signal components.)

4.1 Hardware of the Experimental System.

Figures 4a-c shows the ellipsometers' basic hardware components for three types of experiments: (a) short range laboratory matrix element measurements from dry/wetted porous granular materials (the soils and sands) left undisturbed; (b) short range laboratory matrix measurements from surfaces of nonporous composition made to rotate over all backscattering angles (the flat, continuous, and cohesive surfaces like painted metallic panels used in military hardware); and (c) long range matrix measurements, where the beam is sent outside the laboratory to target boards located down range at distances of 500 meters and more. (The switching system of this system's transmitter is reviewed in Section 6.2.) All systems are automated and computer controlled, including: laser switching, beam and sample positioning, power regulation, goniometer and PEM-polarizer rotation; data acquisition, storage, and processing. Wavelength selection and stabilization of the four laser systems is part of an initialization procedure, the only manual optics hands-on task required by the operator of these instruments. (This may also be automated in future prototype systems if tunable laser sources are required.)

The salient components of the ellipsometer system are now discussed.



Figures 4a-b. The optical arrangement of ellipsometer systems for laboratory study of (a) dry/wetted surfaces of a granular and porous texture, like terrestrial surfaces, undisturbed in a natural flat position and (b) dry/wetted surfaces of a non-porous rigid texture. The components of the ellipsometer instrument include: Probes 1-4, four mid infrared lasers with distinct CO_2 gain media (three are isotopic); S1-S4, shutters intercepting the four incident beams; TS translation stage to direct the appropriate beam to the scattering sample; SM, stepper motors providing accurate computer-controlled stage rotations and translations; SA, spectrum analyser for determining beam wavelength; DCSM, dc servo motor for power regulation of the incident beams; POL, linear polarizers; QWP, quarter-wave plate for production of circularly-polarized radiation; PEM, photoelastic modulators; M1-M4, mirrors for reflecting incident radiation 45^0; G, goniometer beam transceiver arm; SS, rotary sample stage; L, focusing lens to the MCT detector chip; PED, pyroelectric detector for monitoring the incident beams; δ, a small deviation from the backscattering angle; and MCT, mercury-cadmium-telluride detector of scattered radiations.

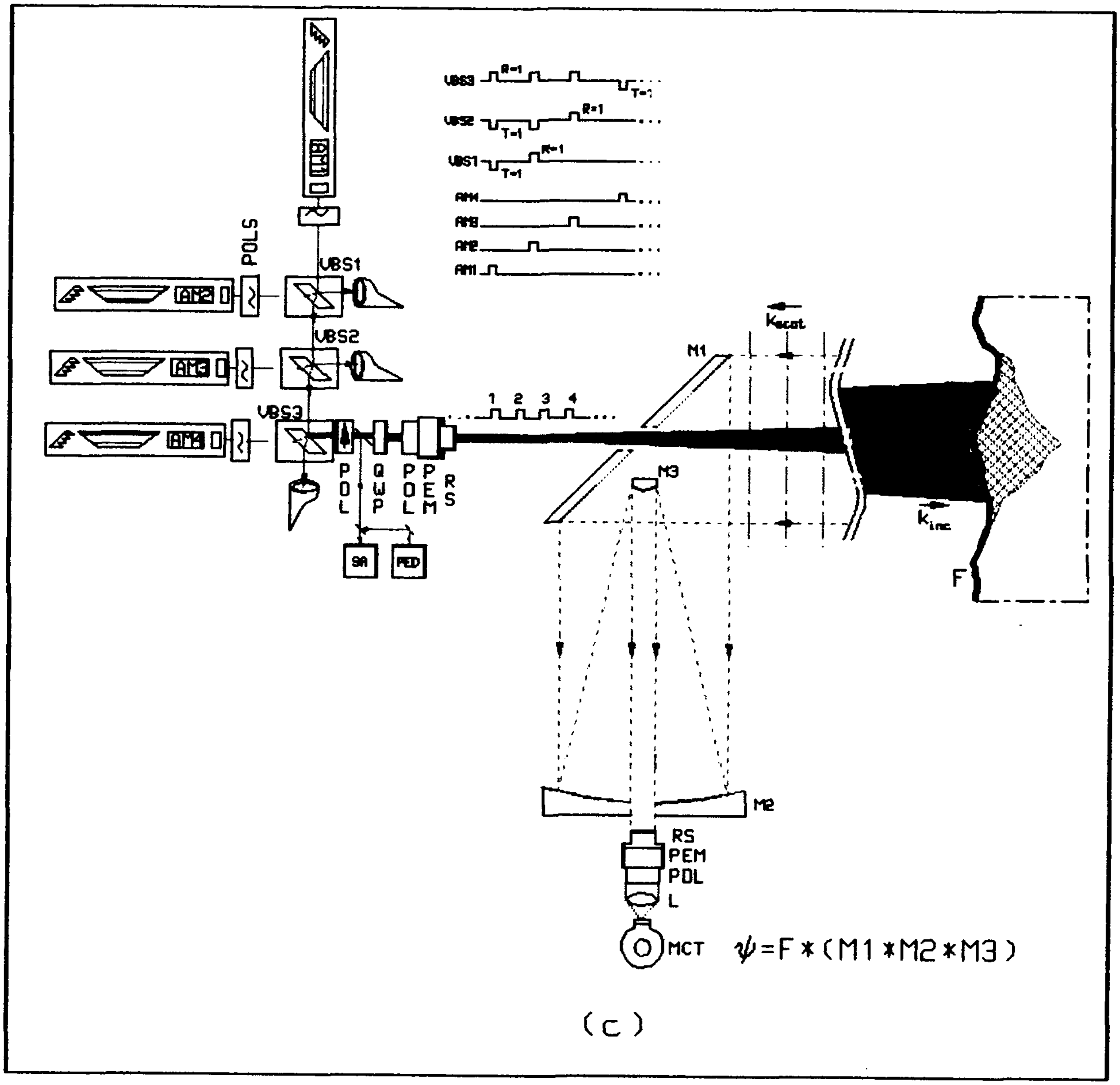


Figure 4c. Field evaluation of contaminated surfaces with a prototype ellipsometer sensor. Additional components are: VBS, variable beam splitters (R=0.1 or .95) and AM laser amplitude modulators for rapid switching of the four beams; and Cassegrain telescope including mirrors M1, M2, and M3.

4.1.1 Probe CO_2 Lasers, Incident Beam Selection, Wavelength Monitor, and Power Regulation.

Four sealed CO_2 continuous-wave (cw) lasers make up the transmitter section of the ellipsometers. Each laser is grating wavelength tunable, and contains piezoelectric circuitry and feedback for mode-locking and beam amplitude and frequency stabilities. The gain media of the four lasers are inert gases mixed with, respectively, nonisotopic $C^{12}O_2^{16}$ and isotopic $C^{13}O_2^{16}$, $C^{14}O_2^{16}$, and $C^{12}O_2^{18}$ gases. Advantages of this four-laser configuration include: (a) a wider wavelength range to probe the sample; (b) a greater selection of discrete wavelengths to probe the surface, therefore, more continuous spectral coverage in the mid IR region; and (c) rapid 4-wavelength matrix measurements without a need for retuning and restabilization. Both (a) and (b), above, follow because vibrational-rotational excited states of the more massive triatomic CO_2 isotopes have shifted P- and R-branch transitions beyond broadening of the $C^{12}O_2^{16}$ laser lines.

For the reader interested in CO_2 spectroscopy, we refer to Herzberg's[22] standard text. For a description of the principal mechanisms and energy level assignments in CO_2 lasers we reference Tyte's[23] book. Listed in Appendix I are the allowed infrared emissions from all four laser systems in these ellipsometer instruments.

Switching between the transmitter's four output beams (of unlike wavelength) is computer controlled by shutter pulsing (S1-S4), and translation of various 90^0 mirror reflectors. The four shutters of Figures 4a-b, one intercepting each laser beam, are open/closed in sequence with and synchronized to the position of three mirrors mounted on a bi-directional translational stage. Either laser beam 1, 2, 3, or 4 traverse the ellipsometer's optical axis (to the scattering sample) by the appropriate mirror position and shutter opening. Figure 4a suggests that just before shutter S4 is switched open (S1-3 are closed), translation stage TS positions to the left where the beam from laser 4 is centered on the outside stage mirror, reflected 90^0 twice, and sent to the sample. After completion of measurement of all 16 Mueller matrix elements at this wavelength, S4 will close, TS will index to the right, S3 will open, and the next Mueller element set is measured at this new wavelength. (It takes about 30-40 minutes for measuring four spectral sets of 16 matrix elements. Thus, when switching from probe laser 4 to probe laser 3, one has enough time to retune and stabilize laser 4 to a different wavelength. This allows one to make multiple laser lines measurements without halting experimental operations.) Future plans call for incorporating a rapid beam switching system for producing near *msec* trains of 3- or 4-wavelength pulses in a next generation high-powered frequency agile ellipsometer configuration (Section 6.2). This future field ellipsometer sensor will contain electro-optic variable beam splitter devices, its lasers an order of magnitude more powerful, and its optic collection aperture considerably larger (Figure 4c) via a Cassegrain (or similar construction) telescope.

Optical spectrum analysers are used to visually display the wavelengths of each beam sent to and scattered by the sample. This instrument (SA in Figures 4a-b) consists mainly of a grating (blazed to diffract most efficiently within a certain band of laser transitions) that diffracts a small power percentage of the incident beam to an order whose intensity is displayed on a thermally sensitive florescent (via ultraviolet illumination) screen, darkening in a region aligned to the angle of diffraction. Above the florescent screen, etched markings delineate each CO_2 beam diffractive order to its P- or R-branch transition assignment and band. Two spectrum analysers were used here to cover the extended wavelength range by the isotopic laser emissions.

In latter sections, we discuss the electronic circuit design that regulates power between switching beams incident to the scattering sample. For now, however, notice the linear polarizer mounted to the dc servo-motor (DCSM) driven rotary stage in Figures 4a-b. Its function is to regulate power of each incident beam to a fixed dc PED detector reference voltage, so that operation in the linear region of the MCT detector is maintained (guard against saturation) for all alternating beams incident to the scattering surface. A feedback loop between pyroelectric detector PED and the DCSM accomplishes this regulation function.

4.1.2 Photoelastic Polarization Modulation.

Anti-reflection coated Zinc Selenide (ZnSe) octagonal windows are the active birefringent optical elements that generate polarization modulation in incident laser beams and their collected scattered radiances. These optics are the heart of the ellipsometer: they generate the primary crystal oscillator frequencies and all combination overtones in the spectrum of the intensity waveform measured by the ellipsometer's MCT photoconductive detector. We refer to this complex waveform as the *scattergram*. It encodes the scatterer's Mueller matrix elements. An oscillating birefringence along the crystal's extraordinary (fast) optical axis is produced when resonant periodic compressions/relaxations are applied via a piezoelectric quartz transducer bonded onto its opposite ends. The greater the applied strain along this crystalline plane (within elastic limits), the greater the phase delay in the beams' EM wave component traversing this fast axis, relative to the orthogonal ordinary (slow) field component: along the axis where no phase delay is experienced by the wave during compression and relaxation periods. The net effect: a coherent plane-polarized laser beam incident 45^0 to the active ZnSe's optical axis (equal fast- and slow-axis EM wave components) becomes polarization modulated (continuous change in Stokes vector with period ν^{-1}, ν the transducer frequency) as components recombine on exiting the crystal.

In Figures 4a-c, stacked Ge-plate linear polarizers are positioned before and after the transmitter and receiver PEM's, respectively. They define incident and final Stokes vectors. Before the transmitter PEM, the polarizer has its transmission axis oriented 45^0 (to fast or slow ZnSe axes) so that beam *E*-field wave components traversing vertical (fast, extraordinary) and horizontal (slow, ordinary) to the PEM optical axis equate. Birefringence, by nature of ZnSe's lattice structure, imparts a relative delay in phase of the wave component along the crystal's optical extraordinary axis. The piezo-induced pressure along the ZnSe's cleaved ends imparts a changing refractive index along this fast axis -- applied strain is proportion to the change of index along the crystal's extraordinary axis and the relative phase difference between components. Vector addition of variably-delayed (fast axis) and unaffected (slow axis) E-field components yields polarization-modulated at the transducer frequency in the beam exiting the PEM. The incident beam modulation from linear to elliptical left- then right-handedness and back to linear has a period of $\tau_{mod} = \nu_{mod}^{-1} = 29.4\mu\ sec$. In collected scattered radiance, the modulation is $31.4\ \mu\ sec$. The variable crystal strain is driven electronically in the PEM's modulator control unit connected to an oscillator circuit located near the modulator head connected to the transducer's bonded quartz plates.

The quarter-wave retardation limit in the ZnSe modulators extends to 19 μm, and its clear aperture diameter is 55 mm (wavelength transmission range is 0.5-19 μm). Compression amplitudes in the crystal by the transducer are monitored by computer so that when switching between beams of different wavelength a peak retardation of 137.74 degrees (2.404 rad) is always maintained (δ_0 in Table 2) by the PEM control unit and oscillator circuit. We later explain in Section 4.3 the importance of maintaining $\delta_0 = 2.404$ rad between laser beams of unlike wavelengths.

Section 4.4 also shows how to interpret and decode the MCT detected scattergram, i.e., a one-to-one mapping assignments between the primary and overtone intensities in the Fourier transformed scattergrams and the elements of the scattering Mueller matrix elements.

The ellipsometer's transmitter POL-PEM optic unit (Figures 4a-c) polarization-modulates the incident beam Stokes vector at linear driving frequency ν_1 = 33.980 KHz, while the instrument's receiver PEM-POL unit operates on collected scattered radiance with a modulation frequency of ν_2 = 31.896 KHz. Together, transmitter and receiver PEM's produce the primary and harmonic combination overtones intensities ($n\omega_1 \pm k\omega_2$ harmonics) in the Fourier-transformed scattergrams (the MCT detector voltage waveform I_f). Modulator frequencies are intentionally offset by 2.064 KHz to insure good separation between primary and overtone frequency components in the scattergram's Fourier power spectrum. The frequencies: $\nu_{1,2}$, $2\nu_{1,2}$, $\nu_1+\nu_2$, $2(\nu_1-\nu_2)$, $2\nu_1 \pm \nu_2$, and $\nu_1+2\nu_2$, and the dc component map onto 9 of 16 Mueller elements simultaneously with no intermixing (see Sections 4.2 and 4.3).

4.1.3. The Goniometer Beam Transceiver Arm For In Situ Analyses of Porous Wetted Surfaces.

As mentioned earlier, additional mirror optics were included in one laboratory ellipsometer design so that measurements on porous and granular surfaces can be performed without mechanically handling the scatterer. The goniometer device was fabricated by Mark Schlein of CRDEC and its intended use was to replicate field Mueller matrix measurements in a laboratory environment.

Dry and wetted soil and other terrestrials are analyzed in their normal horizontal position by use of this 'scanning optical projection arm,' as the ejected contaminant naturally diffuses into and spreads across the strata. The scattering in situ measurements is necessary for determining a field feasibility of this technology: the Mueller elements data base by this ellipsometer instrument must bring out properties characteristic of the analyte consistently, and delineate it from scattering by background particles. A typical field scenario is terrain contaminated via an exploded agent round. A 'trained' ellipsometer sensor would probe the suspected contaminated terrain at selective wavelengths and angles to reveal the targeted contaminant's molecular vibrational properties (absorption, VCD, depolarization, etc.). An additional algorithm in that sensor can be further trained to analyze the contaminants fate, such as liquid analyte diffusion, its evaporation and reactivity with the substrate, a changing morphology in crystalline biological compounds, and so on. The information basis to make such analyzes is, again, the analyte-susceptible field of Mueller elements -- implying a proper selection of the sensor's incident beam angles and beam wavelengths. A set of systematic experiments may identify patterns in the differential Mueller elements that maximize information on the analyte and discern whether the environment effects its fate.

Since the ellipsometer is a backscattering instrument, the goniometer acts as a transceiver to deliver the beam to the surface, and collect backscattered radiation through some small solid angle limited by the size of the smallest mirror mounted on its arm (its limiting aperture), or some other limiting aperture in the receiver design. In these ellipsometers, the clear aperture of the polarizer mounted to the receiver PEM is the limiting aperture. Consisting of an anodized L-shaped aluminum arm with three IR mirrors mounted at 45^0 in three corners (yielding three 90^0 reflections), the goniometer arm is rotated across the upper hemisphere of the sample via a stepper motor rotary stage, computer driven and capable of fractional angle resolution. Details of data acquisition relative to arm movement are provided in Section 4.7.4.

Pyroelectric (Incident Beam) and Photoconductive (Scattergram) Detectors.

Each laser emission line operates at a separate gain, yielding output beam powers from the ellipsometer's laser sources ranging from 2 Watts or slightly less at the weakest P- or R-branch wing transitions, to a maximum of 15 Watts at the 'hot' mid-branch transition. Gain in the branch transitions are specified according to the grating design (*blazed* to yield high efficiency for certain diffractive orders) in the laser cavity. (The grating is the back resonator optic in the laser head.)

Regulation of intensity in beams incident to the scattering sample guards against MCT detector saturation. The incident beam from each laser, vertical in polarization, is regulated in power by a rotating linear polarizing optic in its path, whose transmission angle is controlled by closed-loop feedback. The electronic feedback in this loop is governed from a circuit that draws its input from the (amplified) output of a pyroelectric detector (PED). That detector monitors a split incident-beam percentage of power (less than 1%) transmitted through the regulation polarizer (Figure 4a-b). The current generated by the PED detector is converted into a voltage, amplified, then compared to a fixed reference voltage that is set to correspond to the maximum power of the weakest laser transition selected in the experimental trial. If voltages of reference and PED output are unequal, then the comparator circuit balances it by feedback to the dc servo motor that constantly adjusts the rotary stage on which the beam power regulation polarizer optic is mounted. Since polarization of the incident beam entering this polarizer is vertical, beam power exiting it (the polarizer is made of Ge stacked-plates oriented at the Brewster angle) is governed by Malus' law: $I_t = I_i \cos^2\theta$; where I_i is incident beam intensity, I_t is the exiting beam intensity, and $\theta(t)$ is the angle the polarizer's transmission axis makes with the vertical plane. (Angle $\theta(t)$ is under the control of the PED detector output $\longrightarrow$ feedback circuit loop. This circuit adjusts θ between switched beams, and regulates power for the duration of beam exposure.) Appendix III illustrates the electronic circuits governing power regulation in more detail.

A Laser Precision Corporation model RS-5900 PED radiometer is used in the power regulation feedback circuitry. It works on the principle that when a photon is absorbed on its Lithium Tantalate ($LiNbO_3$) photosensitive surface, its temperature rise causes a spontaneous dipole moment increase (the product of induced charge and polarized separation distance in the lattice) that is converted into a voltage. The electronics convert this voltage to beam power (the detector is equipped with a 1.5 Hz chopper) density the company claims is traceable to NIST standards. Sensitivity in the feedback loop from this detector output is expected to regulate incident beam power to one percent or better of the set reference voltage value.

Light scattered by the sample is detected by a liquid nitrogen cooled Mercury-Cadmium-Telluride (MCT, $Hg_{1-x}Cd_xTe$) photoconductive chip. Since throughput (a measure of scattered laser power collected through the instrument's optical system over some solid angle projected by the receiver MCT chip) is small in these ellipsometer systems, the high detectivity of cryogenic MCT detection technology is required for acquiring scattergrams from the irradiation zone with fast enough response time. (Scattering from most samples of interest is near Lambertian, i.e., isotropic radiance. Also, rough dielectric surfaces are generally good absorbers of the IR beam energy, typical of most terrains.) The responsivity of MCT detectors provide sub microsecond time constants, good enough resolution for capturing the scattergram signal and making the Mueller mappings with $\leq 10\%$ error. The MCT detector response time allows accurate digital recording of the scattergram with satisfactory resolution for phase extractions. (The Fourier power spectrum of the scattergram must clearly demarcates all PEM harmonics up to 100 KHz.)

MCT photoconductive detectors operate on a principle different from PED's. Scattered photons from the sample strike the MCT semiconductor chip generating free carrier electron-hole pairs producing changes in surface electrical resistance. Liquid nitrogen cooling of the chip's surface is required for reducing background noise and increasing detectivity of the photon-induced electron-hole pairs. The ellipsometer's scattergram represents a changing surface resistance of the semiconductor via intensity variation in the focused scattered beam radiance illuminating the chip. The varying surface resistivity is converted into a voltage change (the scattergram) by applying a bias current through the MCT chip. Consequently, the scattergram is preamplified and then sent to a variable gain (voltage controlled) amplifier, filtered and digitized (Section 6.1), or distributed to the phase sensitive detection analog electronics (Section 4.5).

MCT specifications for the ellipsometer systems are: (1) a cutoff wavelength to at least the maximum wavelength output of the $C^{14}O_2^{16}$ laser; (2) a time constant small enough to capture, with good resolution, the real time transient waveform containing primary and harmonic frequencies to 100 KHz; and (3) a detectivity high enough so that, at grazing incident beam angle, signal-to-noise ratio is greater than one. These are met or are exceeded by a MCT detector we have purchased from EG&G Judson. That specific detector has the respective properties: 12.4 μm; 0.5 μsec; and 2.0-5.1 x 10^{10} $cm\ Hz^{0.5}W^{-1}$. Surface area of the MCT chip is 1 mm^2, thus focusing/demagnifying the scattered radiance with lens L onto the detector chip is required to increase intensity I_f to detectable levels in the scattergram.

4.2 Ellipsometer System and Sub-System Matrices.

Signal definition and fundamentals of data processing in these ellipsometer systems are now discussed. Recall from Sections 2.1 and 2.2 the principals of Stokes vector and Mueller matrix: the incident Stokes vector ($\mathbf{s}^i$) is altered upon scattering ($\mathbf{s}^f$) interactions according to the Mueller matrix (F) transformation. Consider the sequence of optical elements in the ellipsometer system of Figure 4a, starting with the transmitter POL-PEM (the linear polarizer mated to this PEM defines the initial Stokes vector components s_j^i; j=0,1,2,3) and ending with the receiver PEM-POL (the linear polarizer mated to this PEM defines the final Stokes vector components s_j^f.)

Each optical element positioned between POL's defining $\mathbf{s}^{i,f}$ are classified by their separate Mueller matrix operator, and thus the transformed Stokes vector through the goniometer-type ellipsometer system is given by:

$$\mathbf{s}^f = \mathbf{F}\mathbf{s}^i \tag{5}$$

$$\mathbf{F} = \mathbf{P}_r\mathbf{G}_r\mathbf{M}_4\mathbf{R}\mathbf{M}_1\mathbf{M}_2\mathbf{M}_3\mathbf{F}\mathbf{M}_3\mathbf{M}_2\mathbf{M}_1\mathbf{R}\mathbf{G}_t\mathbf{P}_t \tag{6a}$$

where subscripts t and r denote transmitter and receiver paths; P and G are matrices of polarizer and PEM optics, respectively; M_1, M_2, and M_3 are matrices of the flat 45^0 mirrors mounted on the goniometer arm; F is the matrix of the scatterer, the unknown of primary importance; and R is a goniometer rotation operator. The product of F matrices do not commute, and **must** appear in proper order from transmitter through receiver optics.

Before proceeding, we should clarify the function of R. Rotation operator R changes the reference measurement plane in which the Stokes vectors are defined (ϵ_1,ϵ_2 in Equations 1-3). In Figures 4a-c, the optic platform is the reference plane defining the basis unit vectors $\hat{\epsilon}_1$ and $\hat{\epsilon}_2$. If the goniometer arm is positioned at an angle other than $\pm 90^0$, the Stokes vector becomes referenced in its new goniometer reference plane defined by the new basis unit vectors $\hat{\epsilon}_1'$,$\hat{\epsilon}_2'$ (the plane defined by beam reflections by the three goniometer mirrors), rotated from the old reference plane by polar angle θ. The R operator third from the end in Equation 6a transforms the Stokes vector from its old reference plane $\hat{\epsilon}_1$,$\hat{\epsilon}_2$ to the new goniometer reference plane. Therefore, $M_{1,2,3}$ and F (Table 2) are now referenced in this new frame. On exiting the goniometer transceiver, scattered light is operated on by R again; forth matrix from the left hand side of Equation 6a, which transforms the Stokes vector back to the old $\hat{\epsilon}_1,\hat{\epsilon}_2$ reference frame. Positioning of R operators in Equation 6 must follow the precise entrance and exit points of the goniometer transceiver.

A new matrix product is defined below, which we shall call the ellipsometer's **system** matrix.

$$\Psi = M_4RM_1M_2M_3FM_3M_2M_1R \tag{6b}$$

By substituting Equation 6b into 6a, the **sample** matrix can now be rewritten as:

$$F = P_rG_r\Psi G_tP_t. \tag{6c}$$

4.3 MCT Detector Waveform of Scattered Radiation.

We proceed in deriving a functional form of the MCT scattergram generated by the 2-modulator ellipsometer systems of Figure 4a-c. Four optical permutations of each POL-PEM pair are illustrated in Figure 5 as defined by Cases A, B, C and D.

Case A. Vertical -45^0:$+45^0$ Vertical

In the notation used above, axes of polarizer and modulator optics in *transmitter* and *receiver* paths of the ellipsometers are separated by a colon. The left part, *Vertical* -45^0, denotes the *receiver* part of this instrument's POL-PEM unit, with polarizer (Vertical) transmission axis followed by its attached PEM retarder (fast) axis (- 45^0). The second entries, $+45^0$ *Vertical* to the right of the colon in Case A, denotes the *transmitter* section PEM-POL unit with retarder axis ($+45^0$) to the left and attached polarizer axis (Vertical) to the right. Angles are measured relative to the plane of the optics table (the reference plane), hence Vertical is 0^0 or perpendicular to the optical platform, and $+(-)45^0$ is measured in the clockwise (counter clockwise) direction one-eighth revolution as viewing each PEM **from the laser source.**

Figure 5 illustrates four optic orientations in the ellipsometer. First consider the configuration given by Case A. Polarizers mated to each modulator set the incident and final Stokes vectors accordingly;

$$\mathbf{s}^i = I_0\begin{pmatrix}1\\-1\\0\\0\end{pmatrix}, \quad \mathbf{s}^f = I_f\begin{pmatrix}1\\-1\\0\\0\end{pmatrix}. \tag{7}$$

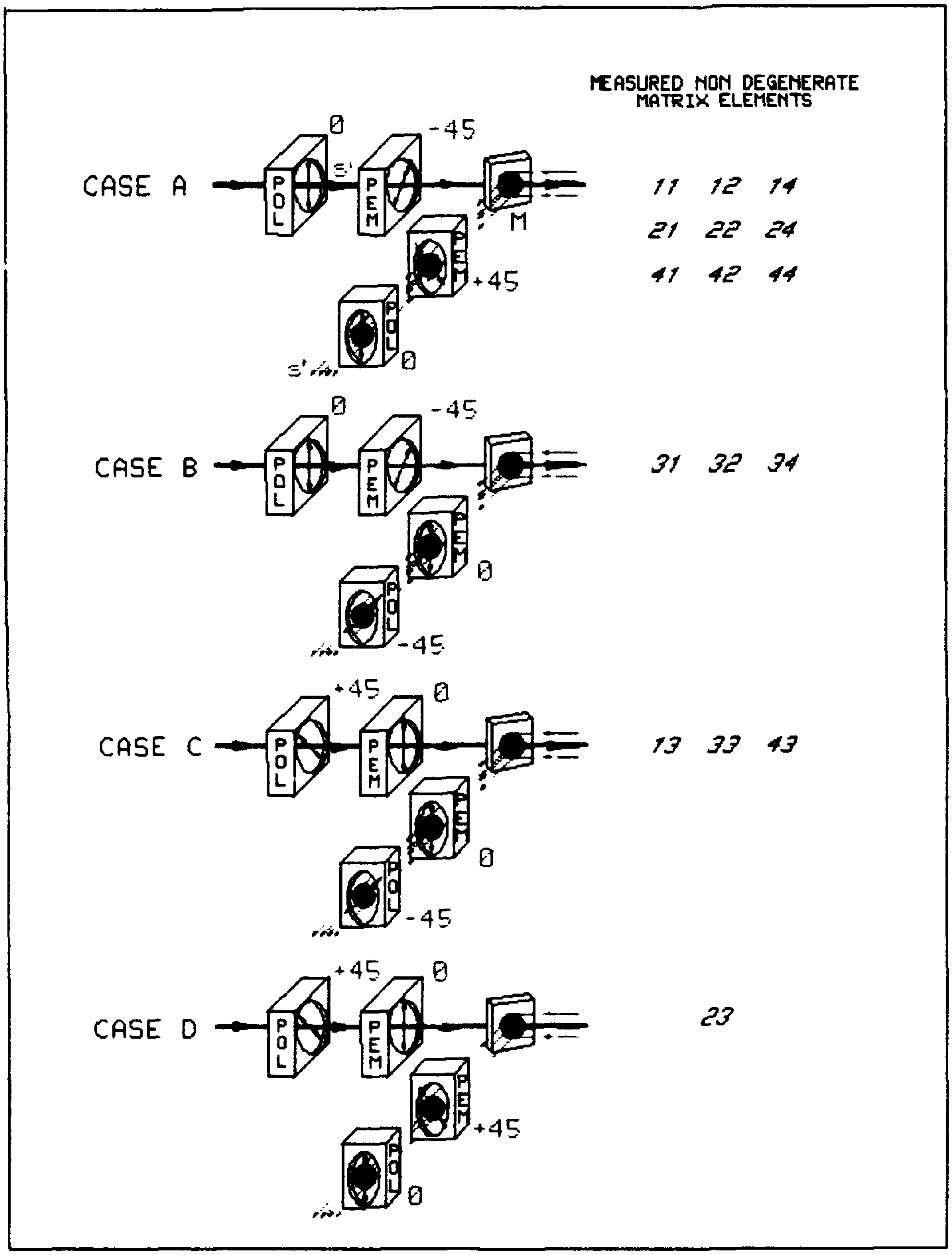


Figure 5. The optical orientations of polarizer-modulator crystal axis producing the system Mueller matrix elements, as measured from the ellipsometer's scattergram by the analog data collection unit. Note that the Mueller element contributions by mirror M can be compensated by rearrangement and insertion of an identical mirror. (See Section 4.6.3.)

Substitution of Equations 7a, 6c, and the appropriate matrices of Table 2 into Equation 5, then left-multiplication by s^f results in the following expression for scattering intensity exiting the receiver polarizer and incident to the MCT detector chip.

$$I_f = \frac{I_0}{8}(1\ -1\ 0\ 0)\begin{pmatrix} 1 & -1 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\delta_0\cos\omega_2 t) & 0 & +\sin(\delta_0\cos\omega_2 t) \\ 0 & 0 & 1 & 0 \\ 0 & -\sin(\delta_0\cos\omega_2 t) & 0 & \cos(\delta_0\cos\omega_2 t) \end{pmatrix} \cdot \tag{8a}$$

$$\cdot \begin{pmatrix} \psi_{11} & \psi_{12} & \psi_{13} & \psi_{14} \\ \psi_{21} & \psi_{22} & \psi_{23} & \psi_{24} \\ \psi_{31} & \psi_{32} & \psi_{33} & \psi_{34} \\ \psi_{41} & \psi_{42} & \psi_{43} & \psi_{44} \end{pmatrix}\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\delta_0\cos\omega_1 t) & 0 & -\sin(\delta_0\cos\omega_1 t) \\ 0 & 0 & 1 & 0 \\ 0 & +\sin(\delta_0\cos\omega_1 t) & 0 & \cos(\delta_0\cos\omega_1 t) \end{pmatrix}\begin{pmatrix} 1 & -1 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}\begin{pmatrix} 1 \\ -1 \\ 0 \\ 0 \end{pmatrix}$$

$$= \frac{I_0}{2}(\psi_{11} - \psi_{12}\cos(\delta_0\cos\omega_1 t) - \psi_{14}\sin(\delta_0\cos\omega_1 t) - \psi_{21}\cos(\delta_0\cos\omega_2 t) \tag{8b}$$

$$+ \psi_{22}\cos(\delta_0\cos\omega_2 t)\cos(\delta_0\cos\omega_1 t) + \psi_{24}\cos(\delta_0\cos\omega_2 t)\sin(\delta_0\cos\omega_1 t)$$

$$- \psi_{41}\sin(\delta_0\cos\omega_2 t) + \psi_{42}\sin(\delta_0\cos\omega_2 t)\cos(\delta_0\cos\omega_1 t)$$

$$+ \psi_{44}\sin(\delta_0\cos\omega_2 t)\sin(\delta_0\cos\omega_1 t))$$

The maximum retardation amplitude along the modulator's fast axis is δ_0. The driving resonance crystal frequencies in receiver and transmitter PEM's are, respectively, ω_1 and ω_2. The δ_0 and ω variables in arguments of the sine and cosine terms of Equation 8b can be separated by substitution of the Bessel generating function[24]:

$$e^{i\delta_0\cos(\omega t)} = J_0(\delta_0) + 2\sum_{k=1}^{\infty} i^k J_k(\delta_0)\cos k\,\omega t \tag{9}$$

where J_k are Bessel functions of k^{th} order, and i is the imaginary number $\sqrt{-1}$. Both PEM transducers are set to yield δ_0=2.404 radians (in the arguments of the Bessel functions). This nulls the zero order Bessel function: $J_0(2.404)=0$. The real and imaginary components of Equation 9 can now be expressed by the following infinite series expansions.

$$\frac{\cos(\delta_0 \cos\omega t)}{2} = -J_2(\delta_0)\cos 2\omega t + J_4(\delta_0)\cos 4\omega t - J_6(\delta_0)\cos 6\omega t + \cdots \tag{10a}$$

$$\frac{\sin(\delta_0 \cos\omega t)}{2} = J_1(\delta_0)\cos \omega t - J_3(\delta_0)\cos 3\omega t + J_5(\delta_0)\cos 5\omega t - \cdots \tag{10b}$$

By substituting Equations 10a-b into Equation 8b and factoring, the amplitude and frequency components can be separated to yield the scattergram intensity waveform expansion I_f.

$$\frac{I_f}{I_0} = \cdots + ½[J_3^2(\delta_0)\cos(3\omega_2 t)\cos(3\omega_1 t) - J_1(\delta_0)J_3(\delta_0)\cos(\omega_2 t)\cos(3\omega_1) \tag{11}$$

$$- J_1(\delta_0)J_3(\delta_0)\cos(3\omega_2 t)\cos(\omega_1 t) + J_1^2(\delta_0)\cos(\omega_2 t)\cos(\omega_1 t)]\psi_{44}$$

$$- ½[J_3(\delta_0)J_4(\delta_0)\cos(3\omega_2 t)\cos(4\omega_1 t) - J_1(\delta_0)J_4(\delta_0)\cos(\omega_2 t)\cos(4\omega_1 t)$$

[illegible] $(\delta_0)J_3(\delta_0)\cos(3\omega_2 t)\cos(2\omega_1 t) + J_1(\delta_0)J_2(\delta_0)\cos(\omega_2 t)\cos(2\omega_1 t)]\psi_{42}$

$$+ [J_3(\delta_0)\cos(3\omega_2 t) - J_1(\delta_0)\cos(\omega_2 t)]\psi_{41} + ½[J_3(\delta_0)J_4(\delta_0)\cos(4\omega_2 t)\cos(3\omega_1 t)$$

$$- J_2(\delta_0)J_3(\delta_0)\cos(2\omega_2 t)\cos(3\omega_1 t) - J_1(\delta_0)J_4(\delta_0)\cos(4\omega_2 t)\cos(\omega_1 t)$$

$$+ J_1(\delta_0)J_2(\delta_0)\cos(2\omega_2 t)\cos(\omega_1 t)]\psi_{24} + ½[J_4^2(\delta_0)\cos(4\omega_2 t)\cos(4\omega_1 t)$$

$$- J_2(\delta_0)J_4(\delta_0)\cos(2\omega_2 t)\cos(4\omega_1 t) - J_2(\delta_0)J_4(\delta_0)\cos(4\omega_2 t)\cos(2\omega_1 t)$$

$$+ J_2^2(\delta_0)\cos(2\omega_2 t)\cos(2\omega_1 t)]\psi_{22} - [J_4(\delta_0)\cos(4\omega_2 t) - J_2(\delta_0)\cos(2\omega_2 t)]\psi_{21}$$

$$- [J_4(\delta_0)\cos(4\omega_1 t) - J_2(\delta_0)\cos(2\omega_1 t)]\psi_{12} + [J_3(\delta_0)\cos(3\omega_1 t)$$

$$- J_1(\delta_0)\cos(\omega_1 t)]\psi_{14} + \psi_{11}$$

Equation 11 consists of a dc component which is always matrix element ψ_{11}, and an infinite number of overtones of the modulator driving frequencies that diminish in amplitude as the order and product of higher-order Bessel function coefficients increase. We truncate the series of Equation 11 for order $k \geq 3$ (a good approximation of the scattergram given the S/N of these instruments) and correlate the remaining nine strongest Fourier intensities one-to-one to their respective Mueller system elements ψ_{ij} (in the coefficients of Equation 11). The transducer primary frequencies for each PEM were chosen as $\omega_1/2\pi$ = 31.896 KHz in the transmitter crystal, and $\omega_2/2\pi$ = 33.960 KHz in the receiver crystal. Furthermore, with these choices of resonant PEM driving frequencies, spectral intensities in I_f are separated by a minimum of 2.064 KHz. Thus, standard lock-in amplifiers can quickly detect and phase-match each scattergram frequency component to its reference primary or strongest overtone modulator frequency without interference from a neighboring harmonic.

The analog data acquisition system separately conducts the dc scattergram component (ψ_{11}) to a separate analog-to-digital (A/D) converter channel in one unit module (Appendix III). For the ac signals (elements other than ψ_{11}), eight phase-sensitive detector circuit cards are set to the reference modulator frequencies of Equation 8b. These lock-in amplifiers electronically multiply their respective reference modulator frequency and real-time scattergram waveforms. The result is an analog output that represents the phase difference between scattergram filtered to the frequency of a reference modulator waveform. The 8 ac Mueller data channels are received simultaneously in another module of the acquisition unit, and all 9 channels are conducted to an A/D converter that strobes through each channel, recording its output in an appropriate file of CPU memory.

In terms of $\frac{I_f}{I_0}$, Equation 11, the most intense primary and overtone frequency components of the scattergram are:*

$$\frac{I_f}{I_0} = \cdots + 0.270\psi_{44}\cos(\omega_2 \pm \omega_1)t - 0.224\psi_{42}\cos(\omega_2 \pm 2\omega_1)t - 0.520\psi_{41}\cos(\omega_2 t) \quad (12)$$
$$- 0.224\psi_{24}\cos(2\omega_2 \pm \omega_1)t + 0.186\psi_{22}\cos(2\omega_2 \pm 2\omega_1)t + 0.431\psi_{21}\cos(2\omega_2 t)$$
$$+ 0.431\psi_{12}\cos(2\omega_1 t) - 0.520\psi_{14}\cos(\omega_1 t) + \psi_{11}$$

where $\cos(A \pm B) \equiv \cos(A+B) + \cos(A-B)$. The sign before each coefficient indicates a relative π-phase difference between elements, i.e., in Equation 12 ψ_{24} and ψ_{42} are 180^0 out of phase with each other.

Case B. Vertical -45^0:Vertical -45^0

After the six Mueller elements of Case A are measured as functions of backscattering angle and wavelength, the ellipsometer rotates the receiver PEM-POL pair -45^0 (as viewed from the laser source) producing the new orientation called Case B, Figure 5. The new incident and scattering Stokes vectors are:

$$\mathbf{s}^i = I_0 \begin{pmatrix} 1 \\ -1 \\ 0 \\ 0 \end{pmatrix}, \quad \mathbf{s}^f = I_f \begin{pmatrix} 1 \\ 0 \\ -1 \\ 0 \end{pmatrix}. \quad (13)$$

* Equation 11 is truncated for Bessel terms greater than 3rd order. Recall that δ_0 = 2.404 rad. The law of cosines was used in deriving Equation 12, as were these tabulated values: $J_1(\delta_0)$ = 0.520, $J_2(\delta_0)$ = 0.431.

Substituting matrices from Tables 1 and 2 produces the following analogous expression to Equation 8.

$$\frac{I_f}{I_0} = \frac{1}{8}(1\ 0\ {-1}\ 0)\begin{pmatrix} 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \cos(\delta_0\cos\omega_2 t) & -\sin(\delta_0\cos\omega_2 t) \\ 0 & 0 & +\sin(\delta_0\cos\omega_2 t) & \cos(\delta_0\cos\omega_2 t) \end{pmatrix} \cdot \tag{14a}$$

$$\cdot \begin{pmatrix} \psi_{11} & \psi_{12} & \psi_{13} & \psi_{14} \\ \psi_{21} & \psi_{22} & \psi_{23} & \psi_{24} \\ \psi_{31} & \psi_{32} & \psi_{33} & \psi_{34} \\ \psi_{41} & \psi_{42} & \psi_{43} & \psi_{44} \end{pmatrix}\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\delta_0\cos\omega_1 t) & 0 & -\sin(\delta_0\cos\omega_1 t) \\ 0 & 0 & 1 & 0 \\ 0 & +\sin(\delta_0\cos\omega_1 t) & 0 & \cos(\delta_0\cos\omega_1 t) \end{pmatrix}\begin{pmatrix} 1 & -1 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}\begin{pmatrix} 1 \\ -1 \\ 0 \\ 0 \end{pmatrix}$$

$$\begin{aligned} = \cdots &+ 0.270\psi_{44}\cos(\omega_2 \pm \omega_1)t + 0.224\psi_{42}\cos(\omega_2 \pm 2\omega_1)t - 0.520\psi_{41}\cos(\omega_2 t) \\ &+ 0.224\psi_{34}\cos(2\omega_2 \pm \omega_1)t + 0.186\psi_{32}\cos(2\omega_2 \pm 2\omega_1)t + 0.431\psi_{31}\cos(2\omega_2 t) \\ &+ 0.431\psi_{12}\cos(2\omega_1 t) - 0.520\psi_{14}\cos(\omega_1 t) + \psi_{11} \end{aligned} \tag{14b}$$

Notice that three new elements appear in the above expression: ψ_{31}, ψ_{32}, and ψ_{34}.

Case C. +45° Vertical:Vertical –45°

By this time, 12 of 16 Mueller elements have been measured. The ellipsometer will now send an index command to the motion controller, causing the transmitter POL-PEM to rotate precisely +45° for producing Case C, Figure 5. Incident and scattered Stokes vectors become:

$$\mathbf{s}^i = I_0\begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \quad \mathbf{s}^f = I_f\begin{pmatrix} 1 \\ 0 \\ -1 \\ 0 \end{pmatrix}, \tag{15}$$

and detector intensity is represented by the product of the following six matrices. (See Tables 1 and 2.)

$$\frac{I_f}{I_0} = \frac{1}{8}(1\ 0\ {-1}\ 0)\begin{pmatrix} 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \cos(\delta_0\cos\omega_2 t) & -\sin(\delta_0\cos\omega_2 t) \\ 0 & 0 & +\sin(\delta_0\cos\omega_2 t) & \cos(\delta_0\cos\omega_2 t) \end{pmatrix} \cdot \tag{16a}$$

$$\cdot \begin{pmatrix} \psi_{11} & \psi_{12} & \psi_{13} & \psi_{14} \\ \psi_{21} & \psi_{22} & \psi_{23} & \psi_{24} \\ \psi_{31} & \psi_{32} & \psi_{33} & \psi_{34} \\ \psi_{41} & \psi_{42} & \psi_{43} & \psi_{44} \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \cos(\delta_0 \cos\omega_1 t) & -\sin(\delta_0 \cos\omega_1 t) \\ 0 & 0 & +\sin(\delta_0 \cos\omega_1 t) & \cos(\delta_0 \cos\omega_1 t) \end{pmatrix} \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

$$= \cdots + 0.270\psi_{44}\cos(\omega_2 \pm \omega_1)t - 0.224\psi_{43}\cos(\omega_2 \pm 2\omega_1)t - 0.520\psi_{41}\cos(\omega_2 t) \quad \text{(16b)}$$
$$+ 0.224\psi_{34}\cos(2\omega_2 \pm \omega_1)t - 0.186\psi_{33}\cos(2\omega_2 \pm 2\omega_1)t + 0.431\psi_{31}\cos(2\omega_2 t)$$
$$- 0.431\psi_{13}\cos(2\omega_1 t) - 0.520\psi_{14}\cos(\omega_1 t) + \psi_{11}$$

Again, three of the nine Mueller matrix elements in the above expression are new mappings according to new polarizer and modulator optic orientations. They are: ψ_{13}, ψ_{33}, and ψ_{43}.

Case D. +45^0 Vertical: +45^0 Vertical

By this time, 15 of 16 Mueller elements over angle and wavelength to the scattering sample have been measured. The software module that controls the system's optical permutation of axes will now cause a +45^0 rotation of the receiver PEM-POL, producing the final Case D in Figure 5. Incident and scattered Stokes vectors are now:

$$\mathbf{s}^i = I_0 \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \quad \mathbf{s}^f = I_f \begin{pmatrix} 1 \\ -1 \\ 0 \\ 0 \end{pmatrix}, \quad \text{(17)}$$

and the detector intensity is represented by the product of these seven matrices. (Consult Tables 1 and 2.)

$$\frac{I_f}{I_0} = \frac{1}{8}(1\ -1\ 0\ 0) \begin{pmatrix} 1 & -1 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\delta_0 \cos\omega_2 t) & 0 & +\sin(\delta_0 \cos\omega_2 t) \\ 0 & 0 & 1 & 0 \\ 0 & -\sin(\delta_0 \cos\omega_2 t) & 0 & \cos(\delta_0 \cos\omega_2 t) \end{pmatrix} \cdot \quad \text{(18a)}$$

$$\cdot \begin{pmatrix} \psi_{11} & \psi_{12} & \psi_{13} & \psi_{14} \\ \psi_{21} & \psi_{22} & \psi_{23} & \psi_{24} \\ \psi_{31} & \psi_{32} & \psi_{33} & \psi_{34} \\ \psi_{41} & \psi_{42} & \psi_{43} & \psi_{44} \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \cos(\delta_0 \cos\omega_1 t) & -\sin(\delta_0 \cos\omega_1 t) \\ 0 & 0 & +\sin(\delta_0 \cos\omega_1 t) & \cos(\delta_0 \cos\omega_1 t) \end{pmatrix} \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

$$= \cdots - 0.270\psi_{44}\cos(\omega_2 \pm \omega_1)t + 0.224\psi_{43}\cos(\omega_2 \pm 2\omega_1)t + 0.520\psi_{41}\cos(\omega_2 t) \quad (18b)$$
$$+ 0.224\psi_{24}\cos(2\omega_2 \pm \omega_1)t - 0.186\psi_{23}\cos(2\omega_2 \pm 2\omega_1)t + 0.431\psi_{21}\cos(2\omega_2 t)$$
$$- 0.431\psi_{13}\cos(2\omega_1 t) - 0.520\psi_{14}\cos(\omega_1 t) + \psi_{11}$$

The new and final 16th element in the above equation that completes the Mueller matrix field of mappings is ψ_{23}. It takes $\approx$ 4 s for the transceiver POL-PEM optics to cycle through Cases A-D per beam wavelength per backscattering angle.

In the ellipsometer configuration of Figure 4b, analysis of the sample scattergram is considerably less complex than from that of configurations Figure 4a or Figure 4c. All **M**, and **R** operators are now the unit matrix. The preceding equations still apply, except now the **system** matrix is the **sample** matrix, i.e., $\psi_{ij} = f_{ij}$, since only the scattering sample lies between linear polarizers defining incident and final Stokes vectors. Sample matrix elements are thereby directly obtained at Fourier intensities of the MCT detector I_f waveform.

4.4 Lock-in Detection Matrix of Primary and Overtone PEM Modulator Frequencies.

The discussion in the previous section on detector waveform production is summarized here by inclusion of the important *frequency matrix*, assigning primary and overtones appearing in the MCT intensities of Equations 12, 14b, 16b, and 18b to Mueller **system** matrix elements according to the four PEM-POL optic configurations cited as Case A through D.

Table 3 contains important information on the ellipsometer's sequence of electronic signal acquisitions. Computer automation at startup initializes the system to Case A in Figure 5. All eight lock-in amplification channels in the analog detector unit (see Appendix III) plus two dc channels (before and after automatic gain operation) are producing outputs that are digitized and stored in CPU memory. These data files are preprocessed and organized into Mueller element files with header blocks containing information on backscattering angles, beam wavelengths, type scatterer, sample surface topography (mean squared heights and slopes), type analyte, contaminant mass density, irradiation time, and other parameters that depend on experiment measurement options.

Consider again the sequence of steps for data collection and storage. The ellipsometer operation starts with measurements of the initial nine Mueller element channels of Case A. A change of optical orientation produces Case B, and the system proceeds with measurements of the next 3 nondegenerate matrix element channels. The A/D converter board recognizes and activates these three channels corresponding to the new Mueller elements, and software appropriately routes the data to three new files. Twelve of sixteen elements have now been acquired and stored in memory. The six channels that are degenerate in Case B, i.e., the duplicated Mueller elements collected from Case A, may be deactivated for efficient use of CPU memory. (All channels are checked in calibration experiments to assure repeatability between degenerate elements from one optical orientation to the next.) The computer next sends an ANSI code to the controller and Case C is produced. Again, three of nine channels contain nondegenerate elements and are active, bringing the number of acquired and stored matrix elements to fifteen. The sixteenth element is acquired after the computer produces the final configuration shown as Case D in Figure 5, viz, one nondegenerate element, one required active channel for collection and storage of data. Therefore, for a complete Mueller matrix measurement per independent experimental variable, the Amplitude and Phase Sensitive Detector (APSD) activates its data channels in sequence 9:3:3:1 corresponding to optic permutations labeled Cases A-D.

Table 3. Lock-in frequencies for phase sensitive detection, and corresponding optical alignments for measurement of the full Mueller matrix. Grouped in part (a) are the primary and major combination frequency components of transmitter (ω_1) and receiver (ω_2) photoelastic modulator's that map onto each Mueller matrix element from the scattergram's Fourier intensities. A parenthetical number next to each frequency implies that component's relative strength compared to component dc=1. The entries for each matrix element are a result of the orientations of linear polarizer and modulator optical axes as grouped in part (b). (See Figure 5.)

SELECT PRIMARY MUELLER MATRIX LOCK-IN FREQUENCIES (KHz)			
ψ_{11} *dc* *dc* *dc* *dc*	ψ_{12} $2\omega_1$ (0.431) $2\omega_1$ (0.431) 0 0	ψ_{13} 0 0 $2\omega_1$ (0.431) $2\omega_1$ (0.431)	ψ_{14} ω_1 (0.520) ω_1 (0.520) ω_1 (0.520) ω_1 (0.520)
ψ_{21} $2\omega_2$ (0.431) 0 0 $2\omega_2$ (0.431)	ψ_{22} $2\omega_2-2\omega_1$ (0.186) 0 0 0	ψ_{23} 0 0 0 $2\omega_2-2\omega_1$ (0.186)	ψ_{24} $\omega_1+2\omega_2$ (0.224) 0 0 $\omega_1+2\omega_2$ (0.224)
ψ_{31} 0 $2\omega_2$ (0.431) $2\omega_2$ (0.431) 0	ψ_{32} 0 $2\omega_2-2\omega_1$ (0.186) 0 0	ψ_{33} 0 0 $2\omega_2-2\omega_1$ (0.186) 0	ψ_{34} 0 $\omega_1+2\omega_2$ (0.224) $\omega_1+2\omega_2$ (0.224) 0
ψ_{41} ω_2 (0.520) ω_2 (0.520) ω_2 (0.520) ω_2 (0.520)	ψ_{42} $2\omega_1+\omega_2$ (0.224) $2\omega_1+\omega_2$ (0.224) 0 0	ψ_{43} 0 0 $2\omega_1+\omega_2$ (0.224) $2\omega_1+\omega_2$ (0.224)	ψ_{44} $\omega_2+\omega_1$ (0.270) $\omega_2+\omega_1$ (0.270) $\omega_2+\omega_1$ (0.270) $\omega_2+\omega_1$ (0.270)

(a)

ANGLES RELATIVE TO REFERENCE PLANE NORMAL			
TRANSMITTER		RECEIVER	
LINEAR POLARIZER	*PHASE MODULATOR*	*PHASE MODULATOR*	*LINEAR POLARIZER*
Vertical	-45^0	$+45^0$	Vertical
Vertical	-45^0	Vertical	-45^0
$+45^0$	Vertical	Vertical	-45^0
$+45^0$	Vertical	$+45^0$	Vertical

(b)

4.5 Analog Amplitude and Phase Sensitive Detection Electronics.

This section presents the assembly of electronic circuits and modules of the ellipsometer analog data acquisition system. We denote the analog circuitry presented here as 'first generation.' Moreover, we now are pursuing development of 'second generation' real time digital data acquisition systems (Section 6.1) that will be compared directly, regarding performance and economy, to this 9-channel analog unit. Furthermore, we have already begun development of a neural network architecture that will eventually interface to the analog port output module, or the output connector of the digital acquisition unit (see Section 6.3). This is designated a 'third generation' phase sensitive detection.

The modular analog signal acquisition unit is now, however, in an advanced engineering stage and the first tested in all ellipsometer configurations. Figure 6 is a basic overlay of how information in the scattergram is mapped into the Mueller elements, showing major frequency synthesizer, APSD channels, and software matrix normalization interfaces. In Appendix III, a more detailed electronic breakdown of the complete APSD unit is provided.

4.5.1 Mueller Matrix Acquisition: Theory of Operation.

The hardware of the analog detection system used to collect and separate various amplitude and phase informations from the detector waveform (Section 4.3) will now be discussed. In constructing a finite set of Mueller matrices, this detector processes the input waveform with eight discrete modules. The basic function of these modules as a unit is to collect and separate the pre-amplified MCT scattergram into nine discrete frequencies (Table 3), and from these generate a scalar (number) which corresponds to the cosine of the difference in phase between these frequencies and the phase of nine respective reference frequencies derived from the two photoelastic modulation oscillator circuits. These resulting scalars are accessed via a RS-232C port that integrates signal processor and host computer mainframe.

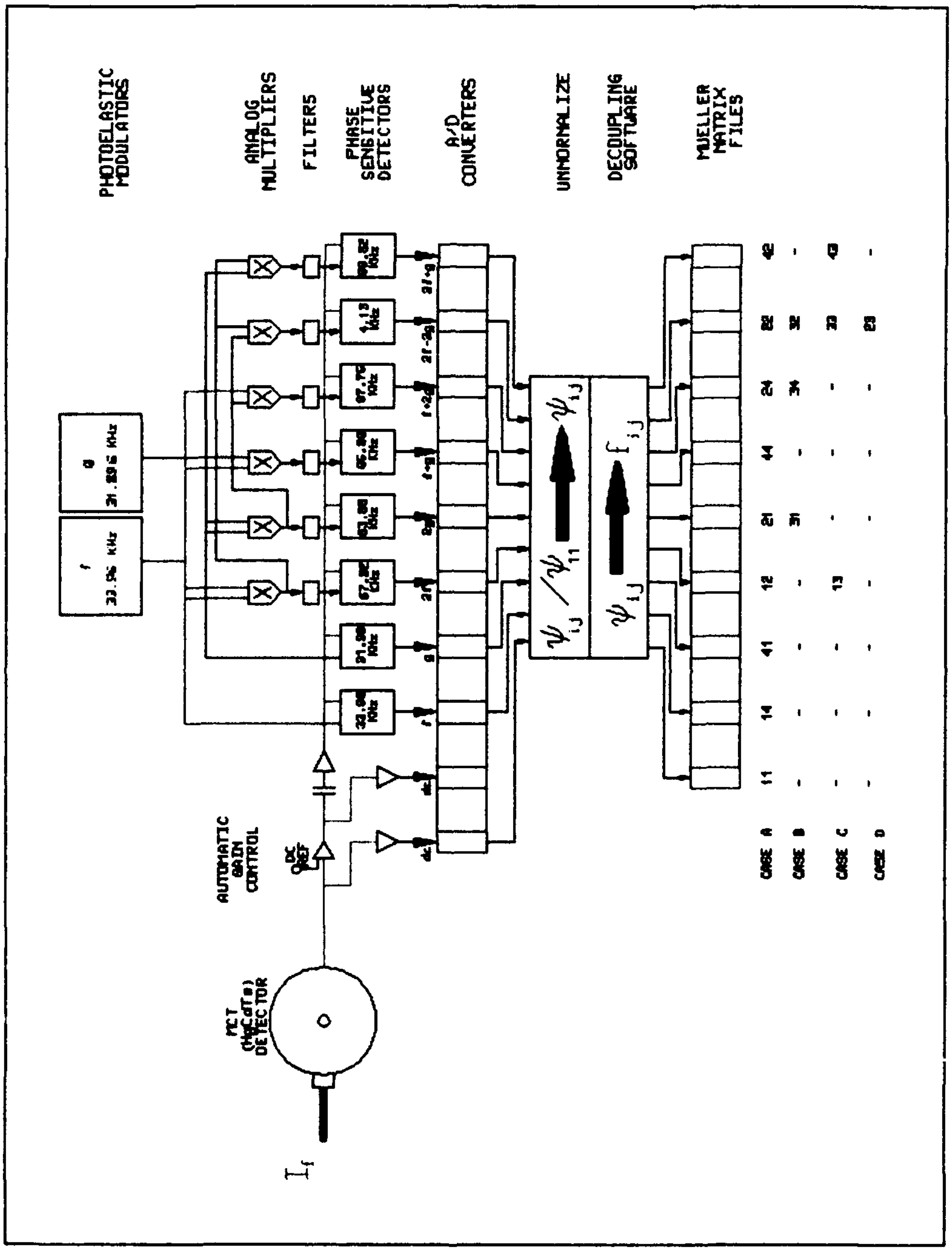


Figure 6. Flow chart of the analog detection electronics. $2\pi f = \omega_1$ and $2\pi g = \omega_2$ are linear frequencies of transmitter and receiver photoelastic modulators, and I_f is the scattergram intensity (the ellipsometer's output waveform containing the two modulators primary frequencies and all combination frequencies).

4.5.2 Reference Frequencies Generator, Mueller Elements Lock-in Amplifications.

The entire signal processing element of the analog Mueller matrix acquisition unit essentially contains a signal reference generator and a signal comparator. The signal reference generator consists of Modules I, II, and III (see Figure AIII.7a). Module I accepts the transistor-transistor logic (TTL) frequencies ω_1, $2\omega_1$ and ω_2, $2\omega_2$ direct from the PEM oscillators, and synthesizes four sinusoid waveforms (3.0-4.0 V_{p-p}) each at a frequency and phase relative to the trigger pulses from the corresponding modulator. These four sinusoidal waveforms are then multiplied by Module II to produce the overtones: $\omega_1+\omega_2$ (65.86 KHz), $\omega_1+2\omega_2$ (97.75 KHz), $2\omega_1+\omega_2$ (99.82 KHz), and $2\omega_1-2\omega_2$ (4.13 KHz). The multiplier board of Module II includes buffering and harmonic filter circuits for the four PEM sinusoid waveform inputs from Module I. The output of Module II thus consists of eight sinusoid waveforms, 2 V_{p-p}, including primary frequencies ω_1 (33.96 KHz), $2\omega_1$(67.92 KHz), ω_2(31.90 KHz), $2\omega_2$(63.79 KHz) and the overtones (Table 3).

Note, that the four overtone product sinusoids consist of both sum and difference values of the multiplied input primary frequencies. We have chosen the specific overtones indicated in Table 3 because they correspond to the eight most intense Fourier amplitudes (greatest signal-to-noise ratio among Mueller components) obtained from the detected scattergram.

Module III conditions the product waveforms (from Module II) by selective bandpass filtering of the desired overtone frequencies, and provides an adjustable phase shift to all eight (primary plus overtones) reference waveforms. These reference waveforms can be obtained through BNC connectors $J_{36} - J_{43}$ (2.0-3.0 V_{p-p} at $Z=50\ \Omega$) for calibration purposes.

Module IV (see Figure AIII.8) consists of the phase sensitive detector (PSD) boards. The eight reference frequencies synthesized from Modules I-III are inputs to the individual PSD circuit cards tuned for that frequency. The PSD circuits multiply (dot product) reference and MCT detector waveforms. With reference (amplitude A) and MCT scattergram (amplitude B) waveforms connect to the input channel of each PSD board, an analog dc output voltage is produced with magnitude proportional to the cosine of the phase angle between input waveforms and amplitude product: $AB\cos(\theta_a-\theta_b)$. All PSD outputs are buffered with gain control potentiometers (R_{17} - R_{24}) located on the unit's front panel, also for calibration purposes. These buffered signals are externally available through eight 50Ω BNC terminators (J_9 - J_{16}).

4.5.3 Digitization of the APSD Outputs (Mueller Matrix Channels).

Module V (see Figure AIII.8) of the analog phase sensitive detector unit consists of a microprocessor controlled model ST701 Analog-to-Digital converter (ADC) manufactured by DATEL, Incorporated. With a 12/20 Intel compatible VME plug-in board, this unit is used as a stand alone processing system that is accessible through the breadboard's RS232 serial link, and controlled through commands issued by the host microvax computer. Among its other tasks, the host computer strobes the ADC for acquisition of all nine analog APSD channels of data synchronous to experimental variable(s), and options of measurements coded in the system control software package (Appendix IV).

4.5.4 Servo Loops for Variable MCT Gain Control and Incident Beam Power Regulation.

Module VI of the analog APSD consists of an automatic-gain-control (AGC) amplifier, fabricated for this system by Analog Modules, Inc., that is connected directly to the ellipsometer's MCT detector output port. The feedback function in the AGC amplifier controls current through the detector's split dc load resistance, to maintain a constant pre-set dc amplitude in the MCT output for all variations in the experimental parameters, including laser wavelength and backscattering angle. As these independent variables are controlled by the instrument's automation software, the AGC regulates the dc scattergram component, thereby causes normalization of all ac matrix components. (Division of phase Mueller elements by the dc element f_{11}. All elements except the dc element are bounded between +1 and -1.)

There is a provision in the AGC amplifier unit to measure the f_{11} element before active gain control with a separate low pass filter and amplification circuit. This data file, transferred directly to CPU memory then permanently stored on hard disk, is required information when converting between normalized and regular elements of the Mueller matrix. The following Figure 7 is a schematic drawing of the basic ellipsometer optical system and MCT detector with AGC circuit module. Figure AIII.11 (Appendix III) is a functional block diagram of an AGC amplifier circuit built for this system by Analog Modules, Inc.

In Figure 7, connections A and B are resonance frequencies driving oscillating birefringence in both transmitter and receiver PEM crystals: they are the phase reference points of the retardations along the ZnSe extraordinary axis. Connection D contains (normalized) phase information in the scattered beam radiance after AGC operation (the ac Mueller matrix components ratioed by the f_{11} element), and C contains the absolute magnitude of the scattergram signal before loop control (i.e., the dc component without AGC operation). The analog data acquisition system compares the phase of D to the reference phases A and B plus its respective combination frequency components (see Section 4.5.2).

The AGC amplifier must not introduce any propagation delay or phase shift of its own as signal strength varies. A beam chopper operating at a rate of $\approx$ 100 Hz is used to produce a dark time so that the detector offset voltage can be eliminated via a closed loop. The amplitude of the MCT signal will vary from 1 μV to 1 mV, due to changes in beam wavelength and backscatter angle experimental variables. The ac signals filtered by the lock-in circuit boards are between frequencies 1 and 100 KHz. The Analog Modules automatic gain control amplifier (Figure AIII.11), has the following salient characteristics: (1) gain control from 60-100 dB, 1% or better regulation from 1 KHz to 200 KHz, noise < 25 μV, frequency bandwidth 1 MHz to < 1 KHz, an input impedance of 100KΩ, and a $\pm$15 VDC supply voltage.

Module VII is an incident beam power regulation circuit. The split incident beam pyroelectric detector output is monitored by the power regulation circuit, and compares this value to a preset desired power reference. Depending on the comparison of these values, the servo motor will rotate the axis of a linear polarizer through which the incident beam transmits. Its function is to regulate the intensity of the beam incident to the scattering sample when switching between probe beams (course adjustment) and during irradiation (fine regulation). The circuits of this module are given in Figures AIII.12a-b.

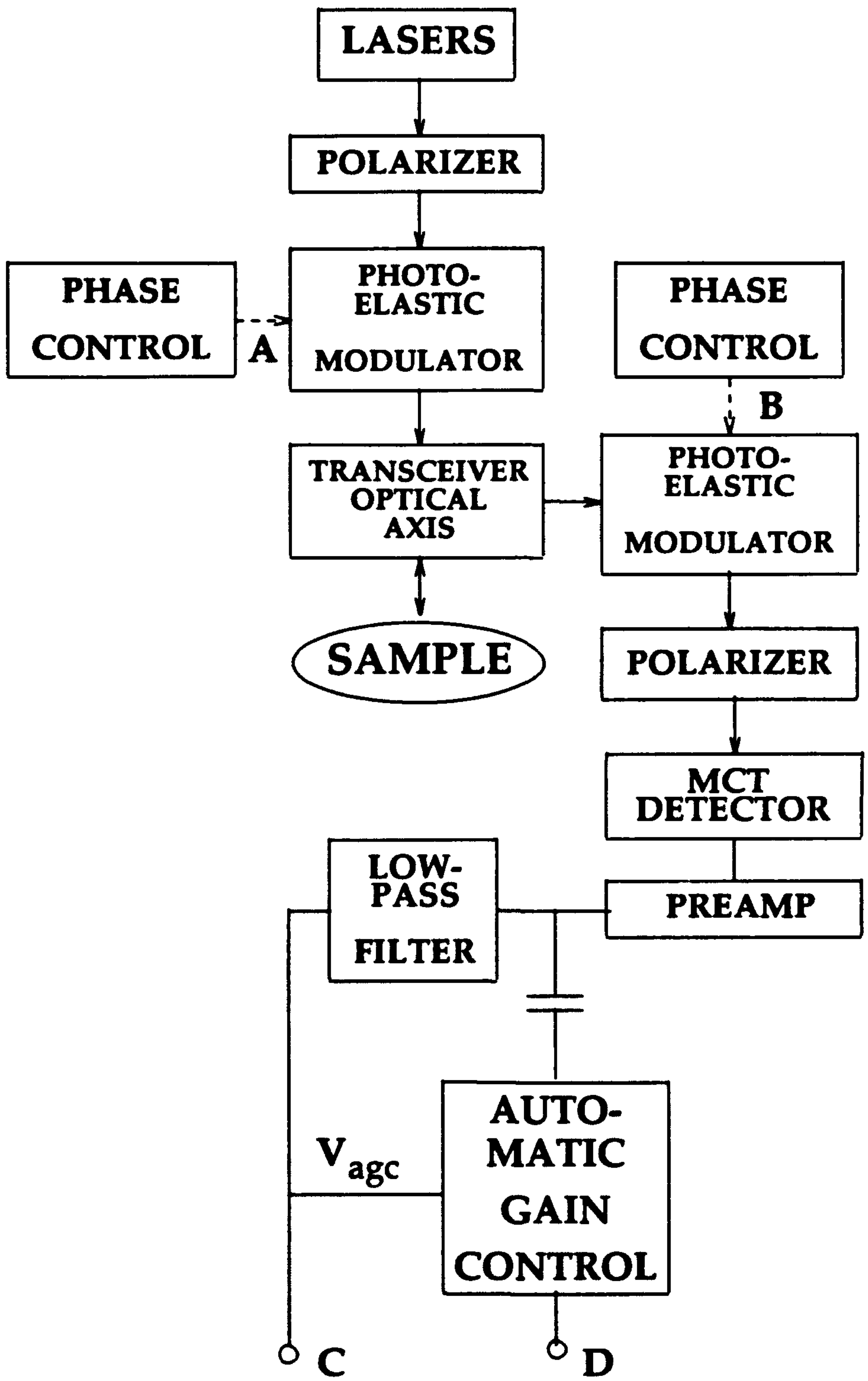


Figure 7. The ellipsometer's optical system, detector, and gain-control modules.

4.5.5 Stepper Motor Control of Optics Hardware, Switching Between Incident Beams, PEM Peak Phase Retardation Selection Per Beam.

We call Module VIII the Serial Addressable Gateway (SAG) system of the analog APSD (Figure AIII.13a). This module is a communications bus consisting of four RS-232C serial ports and sixteen discrete I/O points. The SAG unit provides serial communications to each of the stepper controllers that automate the experimental operation, and to the A/D board that digitizes all data output channels. It also provides for eight control points used for shutter control for switching between beams (Figure AIII.15a), and for maintaining constant 2.404 rad peak modulator retardation in transmitter and receiver PEM's (Figure AIII.14) between switched beams of unlike energies. This module allows bi-directional communications between the host computer and associated devices it wishes to communicate with.

4.6 Alignment and Calibration.

Topics discussed in this section are optic and electronic alignment methods for correct matrix file production and collection, and calibration procedures that transform measured matrix elements collected by the ellipsometer system (this applies only to the goniometer-based and field ellipsometers of Figures 4a and 4c) to the true surface-analyte **sample** Mueller elements.

4.6.1 Alignment of Coupled Polarization Modulator and Linear Polarizer Axes.

Angle between the PEM's ZnSe principal (extraordinary) axis and mated linear polarizer transmission axis must be precisely 45^0 to insure pure **polarization-modulation** in irradiation beam and scattered radiance, thus maximum signal-to-noise ratios in the Fourier intensities (primary and overtone modulator frequencies) of the scattergram I_f. The calibration experiment for accomplishing precision alignment in PEM-polarizer axes is shown in Figure 8.

The quarter-wave plate optic QWP, Figure 8, converts the incident linearly polarized beam to circular polarization. Consequently, rotation of POL1 does not change beam intensity transmitted through the PEM-POL unit when rotated to produce cases A-D. Each PEM-POL unit consists of an IR linear polarizer (stacked Ge plates oriented at the Brewster angle) attached to a micrometer-adjusted rotary stage RSM attached to a photoelastic modulator PEM attached to a stepper-motor controlled rotary stage RSS. Linear polarizer POL2 produces, in conjunction with the active POL-PEM unit, intensity modulation in the beam striking the MCT photoconductive chip at the PEM transducer driving frequency (sine wave generator). This modulated output is pre-amplified through an ac-coupled circuit (A) and sent to the input channel of a lock-in amplifier (LIA). Reference frequency f_1, split from the transducer oscillator driving the ZnSe crystal to resonant vibration (located in the PEM head), is sent to the LIA's reference channel. The LIA electronically multiplies reference and detector waveforms, **viz**, it produces an analog output that tracks the cosine of the phase difference between reference and MCT sinusoids (bounded by $\pm 5V$).

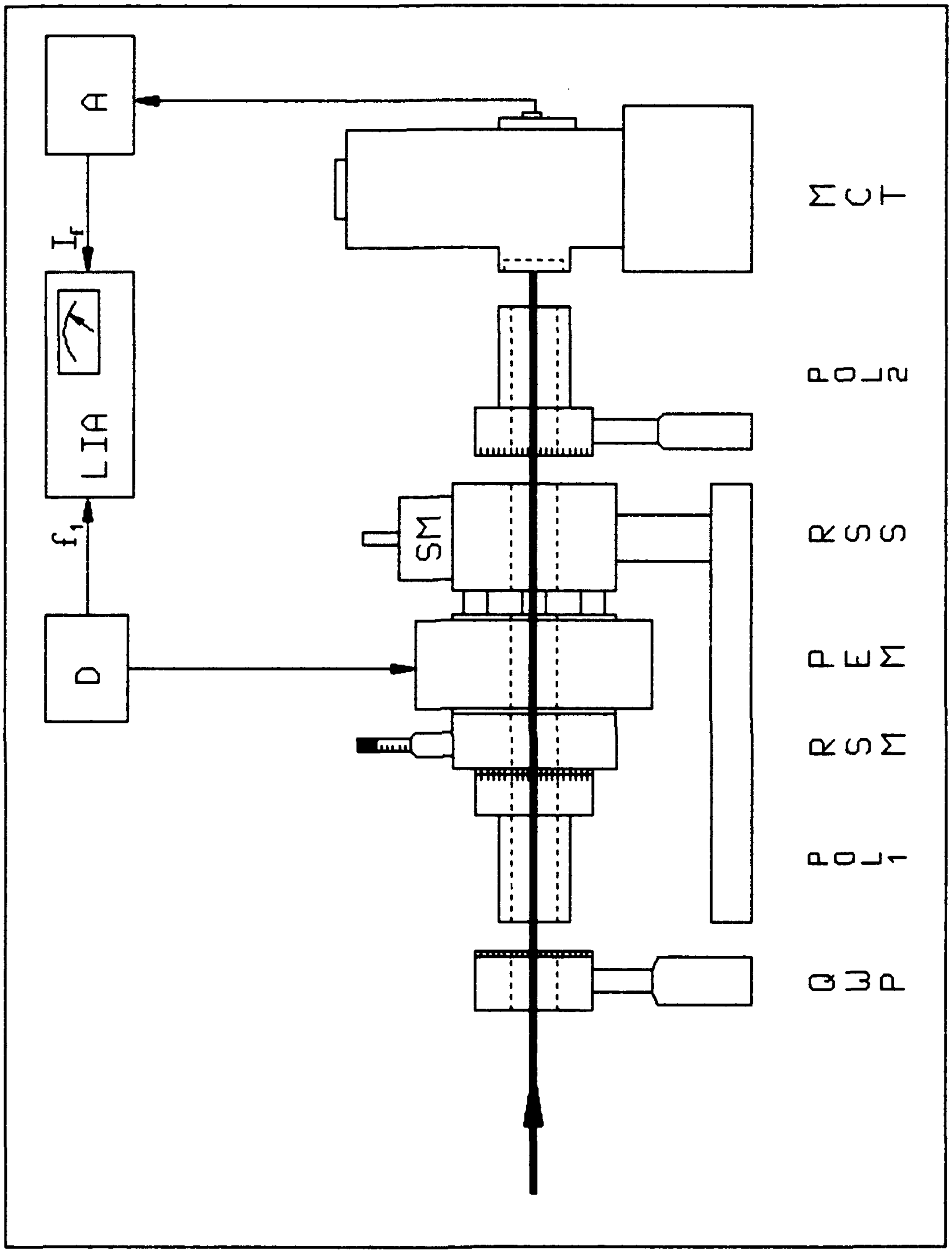


Figure 8. Alignment of the ellipsometer's linear polarizer (POL1) - photoelastic modulator (PEM) pairs. LIA, lock-in amplifier; D, electronic oscillation circuit driving the PEM at frequency f_1; A, MCT detector amplifier; QWP, quarter-wave plate; RSM, precision rotary stage coupling POL1 and PEM; SM, stepper motor; RSS, stage for rotating the POL1-PEM pair; and MCT, liquid nitrogen cooled HgCdTe infrared photoconductive detector.

When aligning the optics between POL1 and PEM axes, the micrometer on RSM is turned until a null output is displayed on the LIA meter: the PEM retarder and POL1 transmission axes are now co-aligned. Two nulls will occur per 2π POL1 revolution. It is good practice to rotate POL1 several times to insure that these nulls appear exactly 180^0 apart. An alignment precision of about 5 arc minutes between nulls is possible with this particular design.

Now that the polarizer transmission and modulator fast axes are co-aligned, rotation of POL1 (course then fine turning of RSM) exactly 45^0 puts the PEM-POL units into final alignment. A precision 5 arc minutes or better is possible, given the precision rotary stage RSM employed here. This alignment can also be performed more directly by rotating POL1 until a maximum output is displayed by the LIA. However, we find that locating nulls in the LIA's analog meter is a more accurate measurement technique, compared to seeking a maximum deflection during rotation of POL1. (Alignment error goes as the cosine of the offset angle between PEM-POL axes.)

4.6.2 Amplitude and Phase Adjustments of the Detector's Nine Element Channels to Transmission Optics of Known Mueller Matrix.

Calibration of the ellipsometer instruments can be performed routinely before and after experimental trials through measurements of spectral intensities in the scattergram (Equations 12, 14b, 16b, and 18b) by the CO_2 beam transmitting three optic calibrators: (a) linear polarizer, (b) quarter waveplate, and (c) combination polarizer-waveplate and waveplate-polarizer; inserted between transmitter and receiver POL-PEM units. The measured Mueller matrix elements of the calibrator (Figure 9) are matched to its known elements by proper phase and gain adjustment of each of nine PSD boards designated Module IV of the analog acquisition system. (The phase per channel of the 15 ac elements are adjusted to match the known calibrator elements, and gain per channel with VGC activation is set to ± 1. See Table 3b, and Section 4.5.) With VGC operation, the f_{11}-element channel is, of course, maintained to a preset voltage within the linear operating range of the MCT detector. The calibration experiment is summarized in Figure 9.

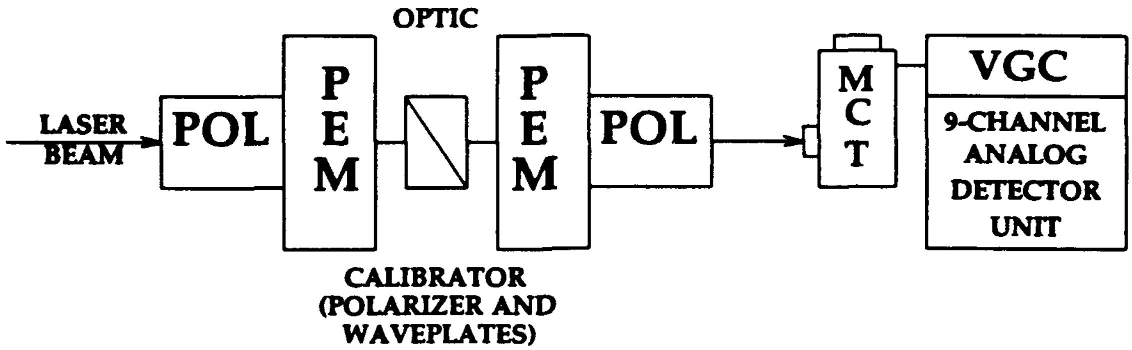


Figure 9. Calibration of the analog data acquisition channels for measurement of all Mueller matrix elements. PEM-POL are the transmitter and receiver photoelastic modulator-polarizer pairs, MCT is the infrared HgCdTe photoconductive detector, and VGC is its variable gain control amplifier (see Figure AIII.11). The calibrators are polarizer and waveplate optics of known Mueller matrix elements. Each channel of the analog detection unit has independent adjustments for phase and amplitude to match the calibrator signatures over the dynamic range of the MCT output waveform (see Section 4.5).

The calibrator optics exhibit Mueller elements of the form:

Linear Polarizer

$$P(\theta) = \tfrac{1}{2}\begin{bmatrix} 1 & \cos(2\theta) & \sin(2\theta) & 0 \\ \cos(2\theta) & \cos^2(2\theta) & \cos(2\theta)\sin(2\theta) & 0 \\ \sin(2\theta) & \cos(2\theta)\sin(2\theta) & \sin^2(2\theta) & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{19a}$$

Quarter-Wave plate

$$Q(\rho) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos^2(2\rho) & \cos(2\rho)\sin(2\rho) & -\sin(2\rho) \\ 0 & \cos(2\rho)\sin(2\rho) & \sin^2(2\rho) & \cos(2\rho) \\ 0 & \sin(2\rho) & -\cos(2\rho) & 0 \end{bmatrix} \tag{19b}$$

where θ is the transmission axis of the polarizer and ρ is the fast axis of the waveplate (quarter-wave retardation).

The rotating polarizer (Equation 19a) calibrates the Mueller elements; f_{31}, f_{12}, f_{21}, and f_{13}. Rotating quarter-wave plate (Equation 19b) calibrates f_{22}, f_{23}, f_{24}, f_{32}, f_{33}, f_{34}, f_{42}, and f_{43}. A combination of polarizer and quarter-wave optics in operator order $P(0)Q(\rho)$ calibrates element f_{41}, while f_{14} can be calibrated in operator order $Q(\rho)P(0)$. A measurement with no optic (air) calibrates the channels for elements f_{11} and f_{44}.

The following Table 4, summarizes how the dc and eight lock-in frequency channels of the analog detection system (Section 4.4) are calibrated to the known rotating optic(s). The calibration of the detector can easily be checked before and after running the ellipsometer system for lengthy periods.

Table 4. Calibration of the PSD's analog electronic channels to the Mueller elements of a rotating quarter-wave plate $Q(\rho)$, rotating linear polarizer $P(\theta)$, and combination optics $Q(\rho)P(0)$ and $P(0)Q(\rho)$.

Channel	Lock-in Frequency	Incident, Final Stokes Vectors [Calibrator, Mueller Element, Signal] V, V	V, -45°	+45°, -45°	+45°, V
1	dc	[*Air*, 11, unity]			
2	ω_2	[$P(0)Q(\rho)$, 41, ½ $\sin 2\rho$]			
3	ω_1	[$Q(\rho)P(0)$, 14, −½ $\sin 2\rho$]			
4	$2\omega_2$			[$P(\theta)$, 31, ½ $\sin 2\theta$]	[$P(\theta)$, 21, ½ $\cos 2\theta$]
5	$2\omega_1$		[$P(\theta)$, 12, ½ $\cos 2\theta$]	[$P(\theta)$, 13, ½ $\sin 2\theta$]	
6	$\omega_1+\omega_2$	[*Air*, 44, unity]			
7	$2\omega_2+\omega_1$	[$Q(\rho)$, 24, $-\sin 2\rho$]	[$Q(\rho)$, 34, $\cos 2\rho$]		
8	$2\omega_1+\omega_2$		[$Q(\rho)$, 42, $\sin 2\rho$]	[$Q(\rho)$, 43, $-\cos 2\rho$]	
9	$2\omega_1-2\omega_2$	[$Q(\rho)$, 22, $\cos^2 2\rho$]	[$Q(\rho)$, 32, $(\cos 2\rho)(\sin 2\rho)$]	[$Q(\rho)$, 33, $\sin^2 2\rho$]	[$Q(\rho)$, 23, $(\cos 2\rho)(\sin 2\rho)$]

4.6.3 Decoupling Sample From System Matrix Elements in the 3-Mirror Goniometer Type Ellipsometer Waveform Output.

Although any discrimination between bare and contaminated surfaces with this instrument will rely principally on the wavelength dependence of the backscattered light, we wish also to investigate the effect of varying the angle at which the IR beam strikes the sample surface. The simplest means for accomplishing this would be to fix and direct the incident beam straight downward onto a sample holder (e.g., a petri dish) and tilt the holder through the desired range of angles with an appropriate mount. Unfortunately we will look at numerous loosely packed samples, such as soils, coated with liquid contaminants, so only a small tilt angle would be allowed. Instead, then, we have chosen to lay our porous granular samples flat on the optical table and vary the light's incident angle using a goniometer - whose mirrors also return the backscattered radiation. By not disturbing the scatterer, the natural effects of liquid diffusion (into and across the porous material bulk) and evaporation of the analyte can be analysed by a screening of elements sensitive to surface geometry.

The drawback to this arrangement is that the instrument measures the net Mueller matrix of everything in the optical path between the two polarization modulators, so the sample's Mueller matrix is buried in the middle of a long product of matrices representing all the goniometer mirrors - both going in and returning (Equation 6b).

We now return to the real-time scattergram obtained from the ellipsometer configuration of Figure 4a, the configuration with the goniometer transceiver arm. This complex waveform requires filtering of the arm's mirror elements for extraction of the Mueller elements of the scattering sample.

In Equation 6b, $\mathbf{M}_i$ represent four metallic mirror matrices oriented in a plane with normal vector 45^0 to the incident beam. Three of these mirrors make up the goniometer arm, and the other directs backscattered radiance from sample to MCT detector. The mirror optics and arm rotation matrices are given in Table 2. The unknown sample Mueller matrix is embedded in the system matrix ψ_{ij} measurements, and needs to be extracted. This can be done analytically by simply inverting M and R from the left and right hand sides of Equation 6b, and substituting element values from Table 2, given values of refractive indices supplied by the manufacturer of the optical surfaces. Each mirror Mueller matrix is an exact function of the optical 'constants' n and k (in Table 2, α, β, and σ are implicit functions of n and k) of its surface coating layers and substrate material. Even under strict quality control procedures from the manufacturer, all mirrors cannot be assumed to have identical σ, β, and σ values. Since their variance in n and k values are not accurately known, three separate experiments are required for empirical determinations of $\mathbf{M}_1\mathbf{M}_2\mathbf{M}_3$, $\mathbf{M}_3\mathbf{M}_2\mathbf{M}_1$, and $\mathbf{M}_3$.

The calibration experiments for decoupling the mirror Mueller elements from the detected signal are schematized in Figure 10. In the top configuration, we define: $\mathbf{B}(\lambda) = \mathbf{M}_3(\lambda)\mathbf{M}_2(\lambda)\mathbf{M}_1(\lambda)$; according to the order in which the incident beam reflects from the mirror flats and transmits through receiver the optics (R) to the MCT detector. The T and R optics are interchanged, and beam direction reversed in the middle configuration of Figure 10, yielding the inverse-order goniometer matrix we define: $\mathbf{C}(\lambda) = \mathbf{M}_1(\lambda)\mathbf{M}_2(\lambda)\mathbf{M}_3(\lambda)$. In the final calibration measurement for $\mathbf{M}_4$, bottom POL-MOD configurations of Figure 10, $\mathbf{D}(\lambda) = \mathbf{M}_4(\lambda)\mathbf{C}(\lambda)$. Note, in the above cases, the goniometer is fixed in the plane of incidence, i.e., $\theta = 0^0$ and $\mathbf{R} = \mathbf{I}$ - the identity matrix. The matrices $\mathbf{M}_i$ are noncommutative, i.e., $\mathbf{B}(\lambda) \neq \mathbf{C}(\lambda)$. By inverting Equation 6b, and substituting in the calibration matrix data, we come to a solution for the Mueller matrix of the contaminated sample with the following form.

$$\mathbf{F}(\lambda,\theta) = \mathbf{C}^{-1}(\lambda)\mathbf{R}^{-1}(\theta)\mathbf{D}^{-1}(\lambda)\mathbf{C}(\lambda)\mathbf{\Psi}(\lambda,\theta)\mathbf{R}^{-1}(\theta)\mathbf{B}^{-1}(\lambda) \tag{20}$$

Presented in Appendix II are term-by-term values the above **F** product matrix. The result was symbolically computed from a LISP coded mathematical algorithm named MACSYMA, and were later checked for accuracy. Fortran 77 code of the element equations, also produced by MACSYMA, is used in the decoupling software conversion operation labeled $\psi_{ij} \rightarrow f_{ij}$ in Figure 6.

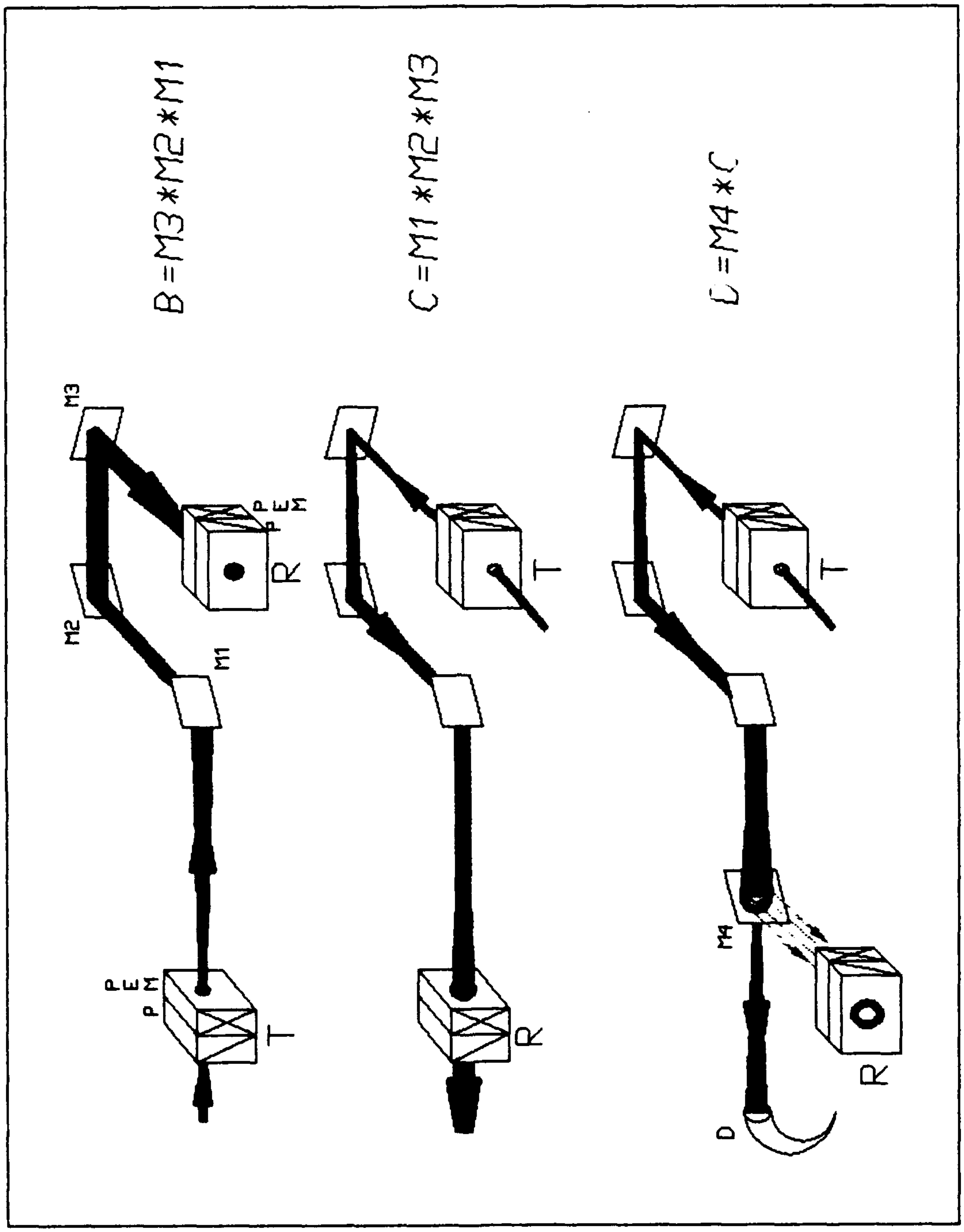


Figure 10. Calibration experiments for decoupling four mirror (M1-4) matrix elements from the ellipsometer **system** matrix of Figure 4a. T and R are the transmitter and receiver linear polarizer(P) - photoelastic modulator(PEM) pairs, respectively, and D is a beam dump. The goniometer arm is oriented + or -90^0, so that the reference plane of measurement of the Stokes vectors in transmitted and received beams are the same. Mueller matrices B, C, and D are produced from the respective optical orientations, and must be measured for each laser wavelength.

Optical Redesign of the 3-Mirror Goniometer Arm

In principle, the Mueller **sample** matrix can be extracted form the measured **system** matrix if the matrices of the mirrors are known. Sections 4.2 and 4.6.3 details the calibration measurements needed - at every wavelength! - for a Mueller description of the goniometer and the subsequent calculations to deconvolve the desired **sample** matrix from the total measured **system** matrix. This process clearly is not satisfactory. It is at best time consuming and inelegant, and, more seriously, there are unresolved questions about the propagation of uncertainties in the goniometer calibration measurements into the calculation of the final matrix. We consider it more sound for the goniometer arm to be redesigned so that the mirror matrices are measured empirically with the instrument rather than to rely on theoretical calculations that, though precise in form, require an exact knowledge of the mirror surfaces' IR optical constants.

Lets now return to the Mueller matrix, M, of a mirror previously expressed in Table 2.

$$\mathbf{M} = \frac{1}{2}\begin{pmatrix} \alpha^2+\beta^2 & \alpha^2-\beta^2 & 0 & 0 \\ \alpha^2-\beta^2 & \alpha^2+\beta^2 & 0 & 0 \\ 0 & 0 & -2\alpha\beta\cos\sigma & 2\alpha\beta\sin\sigma \\ 0 & 0 & 2\alpha\beta\sin\sigma & -2\alpha\beta\cos\sigma \end{pmatrix} \tag{21}$$

As before, the Stokes vectors of the incident and reflected rays are both referred to the plane of incidence containing those two rays (and the mirror normal), α is the ratio of reflected to incident amplitudes for light polarized parallel to the plane of incidence, β is the same ratio for light polarized perpendicular to the plane of incidence, and σ is the reflection induced phase shift between the two components. Note that there is no mixing between parallel and perpendicular polarization components.

Figure 11a illustrates the simple case of a mirror reflecting a light beam upward by 90^0. Let the incident beam, segment 1, have a Stokes vector (s_0,s_1,s_2,s_3) originally referred to the horizontal plane, H. To apply Equation 5, the incident Stokes vector must first be re-referenced to the reflection plane of incidence, V. This is done by operating with the rotation matrix R, defined as:

$$\mathbf{R}(\theta) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos 2\theta & \sin 2\theta & 0 \\ 0 & -\sin 2\theta & \cos 2\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{22}$$

where θ is the angle between the old and new reference planes.

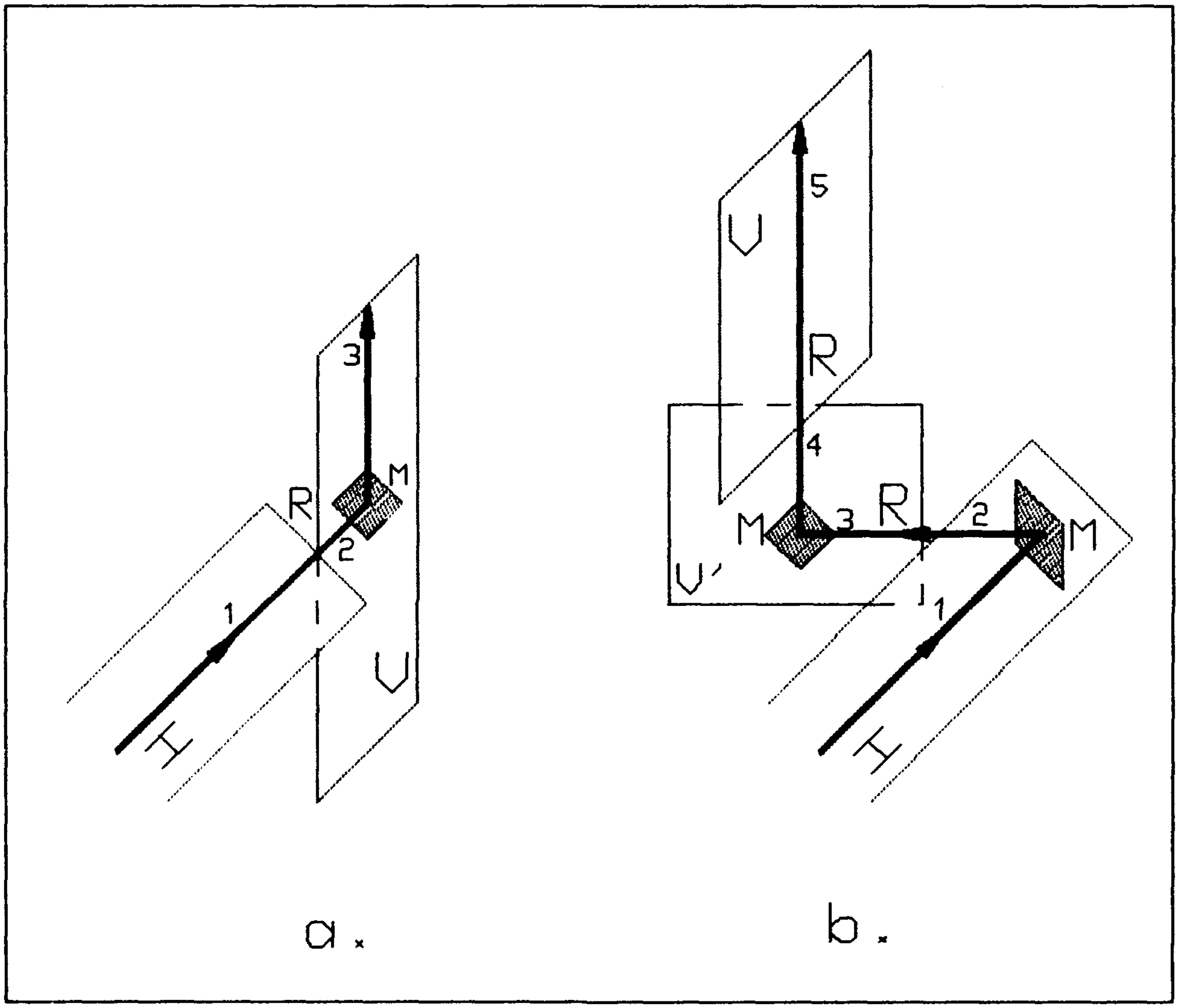


Figure 11. A right angle reflector (a) with a single mirror that changes the Stokes vector and (b) with a pair of mirrors that do not.

Thus, the Stokes vector of the reflected beam, segment 3, referenced to plane V is:

$$\begin{pmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{pmatrix}_3 = \mathbf{M}\ \mathbf{R}(90^0) \begin{pmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{pmatrix}_1 . \tag{23}$$

Notice that $\mathbf{M}\ \mathbf{R}(90^0)$ is just $\mathbf{M}$ with sign changes in the 2nd and 3rd columns.

In Figure 11b we also direct an initially horizontal beam upward, but this time it is first reflected 90^0 in the horizontal plane. A final application of $\mathbf{R}(90^0)$ between segments 4 and 5 makes the final beam (segment 5) identical with respect to direction and reference frame as the final beam (segment 3) in Figure 11a, but now

$$\begin{pmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{pmatrix}_5 = \mathbf{R}(90^0)\ \mathbf{M}\ \mathbf{R}(90^0)\ \mathbf{M} \begin{pmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{pmatrix}_1 . \tag{24}$$

Performing the suggested matrix multiplications, we easily find

$$\mathbf{R}(90^0)\ \mathbf{M}\ \mathbf{R}(90^0)\ \mathbf{M} = \alpha^2\beta^2 \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} . \tag{25}$$

The Stokes vector of the final beam is identical to that of the initial beam, except for an unimportant attenuating factor $\alpha^2\beta^2$. A final reference rotation is not critical and was applied to make Figures 11a and 11b exactly comparable. The only difference without it is a change in two signs of the unity matrix.

The reason a pair of mirrors arranged as in Figure 11b is transparent with respect to Mueller calculations is, of course, that the identity of parallel and perpendicular polarization components is interchanged for the two reflections. The second mirror reverses the relative phase shift of the first mirror and also equalizes the amplitude attenuations. Also note that this result holds for every wavelength. We require only 90^0 reflections and that the mirrors be optically identical.

We believe a goniometer can be constructed on this principle and will allow a direct probing of the sample's Mueller matrix without all the calibrations and inverse matrix calculations of Section 4.6.3.

4.7 System Software: Experiment Automation, Data Collection and Graphics Display.

The development of software for system hardware automation, analog-to-digital conversions for data collection, and graphics presentation for visualization of reduced data sets is updated as new experiments are devised. The first version ellipsometer software package is complete, and is presented in Appendix IV. Linked to the hardware of the analog data acquisition system, the design of this automation and data graphics analysis code is modular. Structured in menu format, it is flexible enough to incorporate changes for accommodating future applications of these ellipsometer systems. A VAXstation II/GPX computer operating under VMS version 5.4 controls automation I/O between it, the SAG, and the DAEDAL MC2000 series stepper motor controllers. All system software in written in the FORTRAN 77 language.

4.7.1 Switching of Laser Shutters, Modulator Retardation Adjustment.

The laser shutters and the modulator retardations are controlled from the 'NEW_TEK.FOR' (Appendix IV) routine as string commands to the main controller relay SAG network (see Appendix III, Figures AIII.13a-b, 15b).

4.7.2 Modulator-Polarizer Permutations.

These are stepper motor controller functions. Movements are predetermined by the user and stored on file. Refer to 'MOV_STAGE.FOR' and associated routines (Appendix IV).

4.7.3 Sample Selection and Stage Rotation.

Sample selection is based on the input order in which a series of dry and wetted surface measurements are made, with a maximum of eight samples. The means by which a sample is selected and rotated about its axis is a function of the 'NEW_TEK.FOR, VV.FOR, V45.FOR, P45V.FOR, and P4545.FOR' routines (Appendix IV). The latter four routines, named for their associated POL-PEM axis orientations, index the sample stage rotation so that Mueller elements can be collected at any range of backscattering angles to and from the various dry and wetted sample scatterers.

4.7.4 Goniometer Rotation and Data Acquisition.

Goniometer rotation operations is performed by 'VV.FOR, V45.FOR, P45V.FOR, and P4545.FOR' routines (Appendix IV). The goniometer is controlled in an identical manner as in the sample stage rotation routines, with the exception that its angle increment can be adjusted to provide measurements at any resolution. Data acquisition is contained in these four subroutines and occurs sequentially and synchronous to stage rotation.

4.7.5 Data Storage and File Management.

The collected sample data is stored in two discrete files. One is an index file that contains information about the sample, the other file contains the Mueller matrix data. Refer to the 'NEW_TEK.FOR' routine (Appendix IV).

4.7.6 Graphics Presentation.

The graphics display for this program was written for a 4111 TEKTRONIX color terminal or equivalent. (Graphics are not required to run the experiment.) The graphics routines that display Mueller elements of the scatterer as functions of backscattering angle and beam wavelength are 'TEK_INPUTS.FOR, TEK_TEXT.FOR and LASER_IN.FOR' (Appendix IV). For real time graphics display refer to 'TEK3.FOR, DRAW_ELE.FOR AND SEE_ELE.FOR' (Appendix IV).

The following Figure 12 is a typical graphics output of the software package given in Appendix IV.

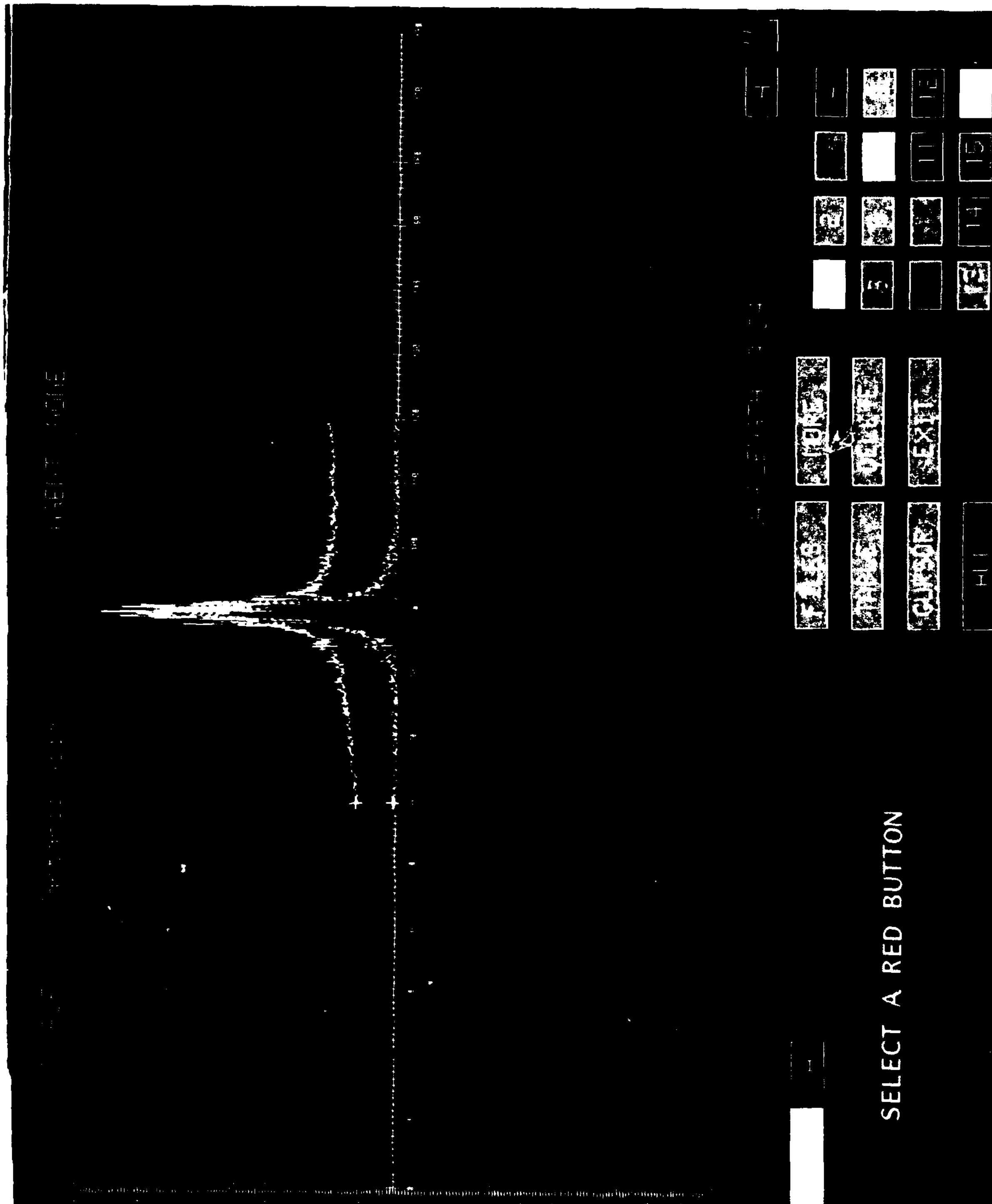


Figure 12 Graphics output from the software package of Appendix IV. In this particular data set, 12 measured elements are displayed as a function of backscattering angle (0.1° resolution) from a water of isomer (-) tartaric acid at $\lambda = 9.24\ \mu m$. The color-coded Mueller field of elements is displayed in the the lower right. Boxes H and V are for horizontal and vertical axes scrolling, box CURSOR activates H and V, box I is for the selection of any of the Mueller elements per PEM optics orientation, box FILES contains the data files, box TABLE contains the raw voltages output from the A/D channels, box DELETE erases element(s) from the screen, box ALL with DELETE erases all elements from the screen, box MORE selects additional elements for display.

5. THEORETICAL MODELLING

Modelling of the Mueller matrix elements is being performed using: (1) RETRO/DISPLAY, a Full Wave theoretical software package that provides closed form expressions for scattering of electromagnetic waves from isotropic surfaces of 'rough' through 'smooth' texture, and graphical analysis of those data; (2) DETECT/DECIDE2, algorithms that select optimum wavelengths and angles from RETRO for maximum probability of contamination detection and; (3) CADPAC/BROOKLYN89/GAUSSIAN88, quantum chemistry software packages that predict infrared, equilibrium geometries, vibrational modes and frequencies, vibrational circular dichroism and other physical properties of the target molecules.

Currently, one of us (J.O. Jensen) along with researchers at the University of Pennsylvania (H. Hameka), Lehigh University (D. Zeroka), the Ballistics Research Laboratory (C. Chabalowski), are modifying the CADPAC/BROOKLYN89/GAUSSIAN88 packages for infrared absorption and VCD spectral interpretations inherent in Mueller elements [1,4], [4,1], and [1,1]. (The elements f_{14} and f_{41} contain information on the VCD property. BROOKLYN89 is now being modified for VCD calculations by various chiral molecules. Also, CADPAC/GAUSSIAN88 is now being used by D. Zeroka for vibrational modes and VCD predictions of linear and ringed sugars and amine molecules that simulate chirality in the more complex biological structures.) There is an interest here to build on a valid quantum chemistry model predicting IR absorption in the more complex molecular systems that behave like chemical/biological agent compounds, and couple it to a tested Full Wave scattering theory. (A purely analytical model of scattering. The present Full Wave model must access physical information on the scatterer from an experimental data bank.) This model's output would in turn transfer it output to a neural network connected to the data channels of the ellipsometer sensor (Section 6.3).

We give a brief summary on quantum modelling approachs in the following section. A more detailed discussion of vibration-rotation, VCD, depolarization and other properties of the analyte compounds of interest will be presented in future papers. Moreover, we do elaborate on and give the source code of model RETRO/DISPLAY and its associated algorithm DETECT/DECIDE2 later in this section and Appendixes V and VI.

5.1 Modelling the Analyte's Resonant Molecular Motions: Applying CADPAC, BROOKLYN89, and GAUSSIAN88 Quantum Chemistry Codes.

The backscattering of polarized light to yield a Mueller matrix depends, in part, on surface geometry. This has been addressed by the work of E. Bahar et al (Sections 5.2.1 - 5.2.4, and Appendix V). Another (coupled) aspect of the Mueller matrix is the interaction of light with specific molecules on the surface, a part associated with the pure physical nature of the scatterer. The molecular phenomena causing these interactions include absorption, depolarization, and circular dichroism. These molecular phenomena will result in an index of refraction that is in a matrix form similar to the Mueller matrix itself. Thus the index of refraction and absorption coefficient are no longer simple scalar quantities. If this matrix form of the refractive index can be extracted from the data, useful chemical data can then be determined.

From the quantum chemistry codes GAUSSIAN90[25], CADPAC[26], and BROOKLYN89[27] we can accurately predict absorption, depolarization, and circular dichroism in the gas phase of molecules. The spectra of a molecule on a surface is similar to a gas phase molecule. Thus we can accurately predict the matrix form of the refractive index by the interaction of light with the given molecule.

Vibrational Circular Dichroism (VCD) is particularly useful in predicting biological contaminants.[28] It is a measure of scattering between left- and right-handed circularly polarized light, interactions that differ with chiral molecules. VCD is related to the [4,1] and [1,4] elements of the Mueller matrix, since these elements are transformations of one circular handedness into another. If the [4,1] and [1,4] elements are measured at two different frequencies on and off resonance, then the difference between the scattering intensity in these elements is indicative of the presence or absence of a chiral molecule.

The flowchart of the calculations that we perform is given in Figure 13. The calculations are usually done at the Hartree-Fock level of theory using a finite basis of Gaussian type wave functions. The minimum energy configuration of the analyte is found within this approximation. The second derivative with respect to all nuclear displacements is then found. The eigenvalues and eigenvectors of the second derivative matrix give the vibrational frequencies and normal modes, respectively. The dipole derivatives along the normal modes give the relative intensities of the peaks.

VCD is calculated using the computer package CADPAC. VCD is a non-Born-Oppenheimer effect caused by the coupling of electronic and nuclear motion. The calculation gives the overlap of the change in the wave function due to a nuclear displacement against the change in the wavefunction due to an external magnetic field. Thus the nuclear motion causes a slight asymmetry in the way that left-handed and right-handed circularly polarized light interact with the molecule.

It is known from the literature[29] that vibrational frequencies calculated at the Hartree-Fock level of theory tend to be slightly higher than experiment. The correction factors needed to make the calculated frequencies agree with the experimental frequencies tend to be constant across a group of similar compounds. Much of the work done in our laboratory involves determining the correction factors to the raw calculated results.

OVERVIEW OF THE SPECTRAL COMPUTATION PROCEDURE

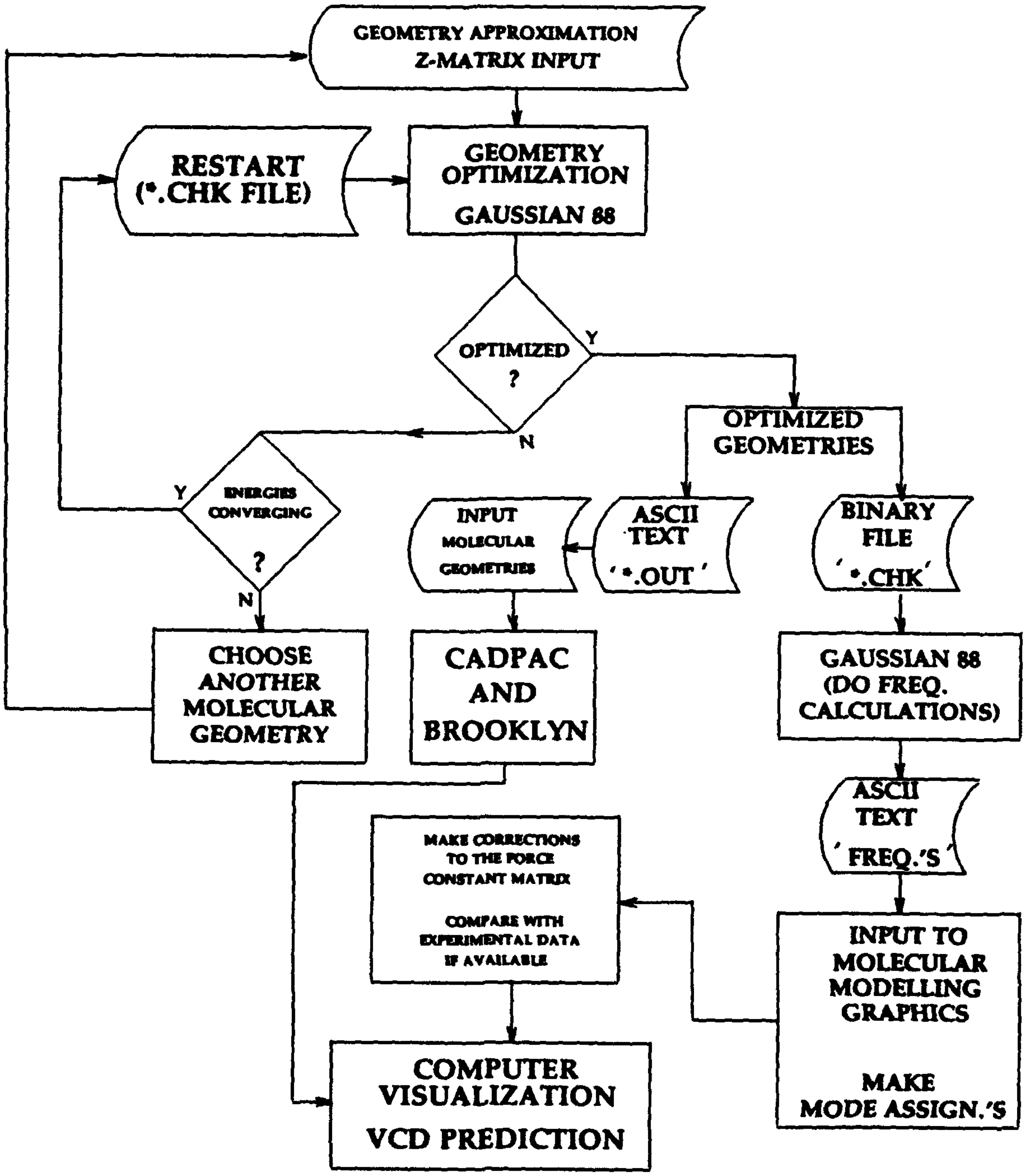


Figure 13. An outline of the computation procedure used to predict the analyte's IR spectrum via Gaussian, CADPAC, and BROOKLYN quantum chemistry software packages.

Linkage to the Full Wave Scattering Code.

In the latter Section 6.3, we discuss a concept in which the ellipsometer's real-time Mueller element outputs are connected to the inputs of a neural network. This network would be designed to partition analyte-specific data, so that presence or absence of the analyte can be established in the scattering zone of the ellipsometer probe beams, as indicated by an appropriate alarm signal at the network's output layer. Inputs to the neural network are weighed according to a valid theoretical model. A candidate model for scattering by a randomly rough surface is the **Full Wave** theory presented in the following section. We anticipate making a linkage of the **CADPAC, GAUSSIAN,** and **BROOKLYN** quantum chemistry codes to the **Full Wave** code. In a successful neural network model, the quantum chemistry codes would predict resonant absorption by the analyte(s). The scattering code accesses this information and outputs Mueller elements of the contaminated surface. That data would then be used to weight the network's input sensor data, and train it to alarm under certain conditions, i.e., when a susceptible set of Mueller elements are present and successfully partitioned from its background. An additional layer can be designed into the neural network that accesses a data bank where Mueller signal strengths and analyte mass densities are correlated, allowing a quantitative map of the threat contaminant to be displayed.

The quantum models consider single molecules, and are used to compute energies driving resonant molecular vibrational motion, assigning an intensity value to the absorption. In the **Full Wave** model, macroscopic boundaries separate material media and each material is characterized by a complex dielectric constant. Obviously, a macroscopic analyte medium containing many, many, molecules would likely have a broadening effect on the quantum predictions and, perhaps, can be predicted by the many-body theories of statistical mechanics.

Once the absorption of a material is determined by the quantum codes (over a wide spectral band), its real and imaginary parts of refractive index can be computed by the **Kramers-Kronig**[10] relationships. These are the data the Full Wave model uses in its surface scattering Mueller matrix predictions.

5.2 Mueller Matrix Predictions by EM Wave Scattering From Rough Surfaces: The Full Wave Model of Physical Optics Theory and its Application for Remote Detection.

An experimental verifiable model for accurate predictions of the Mueller matrix elements is of great value in the development of a detection system. Predicting the Mueller elements as functions of laser beam scattering wavelength, statistical orientation between scatterer and incident beam, optical properties of the scatterer (indices of refraction), and its topography could essentially simulate the entire experimental operation. With these descriptive models, we seek an optimum domain for which these parameters can most readily reveal the analyte, and guide the experimenter toward a most probable detection scattering event.

5.2.1 The Full Wave Model: RETRO.

Program RETRO is a numerical implementation of a Full Wave electromagnetic scattering theory developed by Professor Ezekiel Bahar at the University of Nebraska-Lincoln. This theory bridges the gap between physical optics and perturbation theories.[30,31,32,33,34] RETRO, written by Craig Herzinger, calculates theoretical Mueller matrix elements for light scattering by randomly rough 2-d surfaces. Associated software named DISPLAY presents 3-d, contour, and 2-d graphics from the output of RETRO. RETRO/DISPLAY was written specifically for this experimental program, and, should it be proven feasible, aid us in chosing optimum wavelength, angle, and polarization parameters for characterizing contaminants on various background and interferent terrestrial or manufactured material. Later in this section we discuss what simplifying assumptions are made in **Full Wave** theory to allow numeric results to be calculated in a reasonable period of time. The RETRO/DISPLAY source code, written in Fortran 77 and now running on a CRAY supercomputer, is included Appendix V. Also, notational differences between the code and theory are addressed, as are format and content of input and output data files, and the relationships of many program variables to the theory in symbolic form.

Full Wave theory calculates scattering of an electromagnetic plane wave from a two dimensional, statistically rough, surface between free space (air) and a material with a relative dielectric constant, $\epsilon_r(\lambda_0)$, where λ_0 is the free space wavelength of the radiation.

The surface boundary is defined as $y = h(x,z)$, where $<h> = 0$.
The reference plane is defined as $y = 0$.

The plane of incidence is assumed to be the x-y plane and θ_0^i is the angle between the direction of the wave and the normal to the reference plane. For backscatter, the incident and final directions lie on the same line so that $\theta_0^i = -\theta_0^f = \theta_0$

The mean squared height of the surface is $<h^2>$.
The mean squared slope of the surface is $\sigma_s^2 = <h_x{}^2 + h_z{}^2>$, $h_x = \frac{\partial h}{\partial x}$, $h_z = \frac{\partial h}{\partial z}$.
The correlation length of the surface, l_c, is defined by $l_c^2 = \frac{4<h^2>}{\sigma_s^2}$.
The auto-correlation function of the surface heights is $r_{hh'}(x_d,y_d) = \frac{<hh'>}{<h^2>}$,
where $h = h(x,z)$, $h' = h(x',z')$, $x_d = x-x'$, $z_d = z-z'$.

Full Wave theory allows calculations of elements $F = F(<h^2>,\sigma_s^2,\lambda_0,\theta_0)$.

The assumptions made in this program can be broken into two classes:

- Process assumptions and
- Surface assumptions.

Process Assumptions:

Process assumptions address how the light is scattered, for example how many times it strikes the surface, how the emitter and receiver are oriented, and how diffuse the scattered light is. Surface assumptions deal with the statistical representation of the surface heights and slopes.

This work assumes that scattering can be properly characterized by a single-scatter process. That is, light measured at the detector is assumed to have struck the rough surface exactly once; multiple scattering is not considered. This allows a second order iterative solution to be used in the **Full Wave** theory, additionally, it also limits how rough the surface can be. Another assumption is that the light source and detector are on the same optical path with the same orientation. The common terminology for this is backscattering. Backscattering is considered because the ellipsometer is a proving instrument for a future remote sensing system, where the receiver and detector are at one location far from the target surface.

The Mueller matrix elements are calculated on a per solid unit angle basis, therefore the matrix is correct to within a scalar constant of its experimental counterpart. The scalar constant is based on the size of the solid angle intercepted by the detector. Moreover, for this work to be valid, the detector must look at a small enough range of received radiation centered in the backscattering plane such that pure backscattering can be used as a good approximation of radiance collected from the irradiation cross-sectional area over the entire range of angles and wavelengths. Also, the solid angle intercepted by the detector must be invariant to changes in incident angle, θ_0, and wavelength, λ_0. A third assumption is that the scattered radiation is totally diffuse. This means the coherent portion of the return (the return without the surface roughness, times a constant) is assumed equal to zero. This condition limits how smooth the surface can be.

Surface Assumptions:

The first major assumption is that the surface is isotropic and uniform, i.e., the scattering is independent of the rotational or translational position of the rough surface; the scattering is invariant to a rotation or translation of the x and z axes. This leads to the conclusion that $r_{hh}(x_d, z_d) \rightarrow r_{hh}(r_d)$, $r_d^2 = x_d^2 + z_d^2$.

The second assumption is that the probability density of the heights and slopes are independent, $p(h, h_x, h_z) \rightarrow p_h(h) p_S(h_x, h_z)$. Also, p_h and p_S are assumed, respectively, to be Gaussian and jointly Gaussian probability density functions.

To meet the condition of single scattering, the following restriction is applied: $\sigma_S^2 \leq 2$. To satisfy the condition of purely diffuse scattering, $<h^2>$ is restricted by the relationship $4<h^2>k_0^2$ is much greater than 1, where $k_0 = 2\frac{\pi}{\lambda_0}$. These conditions are not enforced by the program; you must make sure the input data is satisfactory.

Consequences of Assumptions:

Full Wave theory, under the above conditions/assumptions, can be used to calculate the scattering phase (Mueller) matrix F' as defined by the **modified** Stokes vector notation.

$$F' = \begin{pmatrix} \sigma_{VV}^{VV} & \sigma_{VH}^{VH} & 0 & 0 \\ \sigma_{HV}^{HV} & \sigma_{HH}^{HH} & 0 & 0 \\ 0 & 0 & \mathrm{Re}\{\sigma_{VV}^{HH} + \sigma_{VH}^{HV}\} & \mathrm{Im}\{\sigma_{VV}^{HH}\} \\ 0 & 0 & -Im\{\sigma_{VV}^{HH}\} & \mathrm{Re}\{\sigma_{VV}^{HH} - \sigma_{VH}^{HV}\} \end{pmatrix} \tag{26}$$

$$\text{where } \sigma_{ij}^{kl} = Q \int_{-\infty}^{\infty} \frac{D^{ij} D^{kr}}{(\bar{n} \cdot \bar{a}_y)^2} P_2 P_S \, dh_x \, dh_z \tag{27}$$

$$\text{and } Q = 2k_0^2 \int_0^{\infty} \left(exp(v_y^2<h^2>(1-r_{hh})) - exp(-v_y^2<h^2>) \right) J_0(v_{xz} r_d) \, r_d \, dr_d \tag{28}$$

Note that under the process assumptions, eight elements of F' are analytically zero. Measuring these eight elements experimentally should be a good test of theory with the discussed restrictions.

Linkage to Scattering Matrix $\mathbf{F}'$:

In our previous notation, $\mathbf{F}'$ was a modified matrix derived from modified Stoke's vectors. To transform this matrix into the desired Mueller matrix **F**, the notational difference must be addressed. Consider a Stokes vector in the modified notation, $\mathbf{I}_M$, and in the standard notation, $\mathbf{I}_S$, that represent the same light.

$$\mathbf{I}_M = \begin{pmatrix} S_0 \\ S_1 \\ s_3 \\ s_4 \end{pmatrix}, \quad \mathbf{I}_S = \begin{pmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{pmatrix} = \begin{pmatrix} S_0+S_1 \\ S_0-S_1 \\ s_2 \\ s_3 \end{pmatrix} \tag{29}$$

The modified Stoke's vector of the scattered light is $\mathbf{I}_M' = \mathbf{F}'\mathbf{I}_M$ and the standard Stoke's vector of the scattered light $\mathbf{I}_S' = \mathbf{F}\mathbf{I}_S$ clearly must represent the same light. For this to be true

$$\mathbf{F} = \mathbf{A}\mathbf{F}'\mathbf{B}, \quad \text{where} \tag{30}$$

$$\mathbf{A} = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \text{and} \quad \mathbf{B} = \begin{pmatrix} ½ & ½ & 0 & 0 \\ ½ & -½ & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{31}$$

A and **B** are transformation matrices such that $\mathbf{I}_S = \mathbf{A}\mathbf{I}_M$ and $\mathbf{I}_M = \mathbf{B}\mathbf{I}_S$.

RETRO then calculates **F** by computing Q and the necessary $(\sigma_{ij}^{k\,l}/Q)$'s.

5.2.2 DISPLAY: Graphical Analysis of the Full Wave Model.

Program DISPLAY is an interactive program that plots elements of backscatter Mueller matrices. The elements can be displayed in a 3-d, 2-d, or contour format as a function of radiation wavelength, λ_0, and incident angle, θ_0. The program is written in FORTRAN 77 and is set up to run on the UNIX CRAY2 supercomputer. Data files created by RETRO can be used as input for graphical software packages. Also, files containing experimental data, if properly formatted, can be used as input to other programs.

The purpose in writing DISPLAY was to present the theoretical data produced by RETRO in a variety of ways. DISPLAY is to aid in analyzing theoretical and experimental data, and to help in choosing appropriate wavelengths for ellipsometric study of various background and background-analyte surface scenarios. DISPLAY allows for direct comparison of experiment to theory.

DISPLAY uses the plotting package Disspla, version 10.0. Local printing capability is required for interactive work. All calls to Disspla are confined to subroutines that perform specific tasks. These subroutines print headings, draw axes, draw curves, etc. To change plotting packages, these routines (in the Disspla source code) need to be rewritten. Appendix V provides DISPLAY's source code, startup procedure, menu options, plot directives and data analysis options. Its source code is also provided in Appendix V.

We end this section by including **Full Wave** data from dry and wetted clay surfaces of variable roughness via execution of RETRO and DISPLAY programs. The clay sample is an admixture of three minerals in one-third proportion by weight; colloidal montmorillonite, kaolin, and illite. The optical constants and other information on how this clay pellet was fabricated are given in Reference 21. Figure 14a is the clay's calculated $f_{11}(\lambda,\theta)$ element within a backscattering angle range of $0 \leq \theta \leq 88^0$, and wavelength band of $9.0 \leq \lambda \leq 12.2 \mu m$. Mean squared height of the clay pellet's surface is 5.0 μm^2 (smooth), its mean squared slope is 0.05, and the probability density functions of heights and slopes used in the theoretical model are Gaussian.[30,31,32,33] The three maxima arising in the matrix element-surfaces result from Reststrahlen absorption (a narrow wavelength region where a sharp jump occurs in the imaginary part of the complex refractive index in some minerals) in the soil material. In Figures 14b-f, respectively, mean square slope $<\sigma^2>$ of the soil surface is increased in order from 0.05 to 0.10, 0.50, 1.00, 1.50, and finally to 2.00 while $<h^2>$ is held constant. Recall that the f_{11} element is a measure of scattering power, as such, the pattern depicted in Figures 14a-f (i.e., scattering from a smooth to a rough surface as $<\sigma^2>$ increases) is intuitively correct. At small slopes, the surface is spatially slow-varying and therefore most scattered energy occurs at 0^0 specular angle. As the soil surface slope and mean height go to zero, the Matrix element-surface of Figure 14a should reduce to the Fresnel reflection curve at 0 degrees, zero everywhere else. We see from Figures 14a-f, that as surface slopes increase (sharper topographical detail), scattering becomes more Lambertian-like, i.e., scattering energy becomes isotropic as shown by the increasing and broadening of the Mueller element surface $f_{11}(\lambda,\theta)$ for angles beyond normal incidence (0^0 degree), at the expense of decreasing specularly-reflected energy. We also note an intriguing result: the Reststrahlen peaks shift toward higher angle as roughness increases.

This same graphical analysis was conducted on the f_{21} element, results of which are presented in Figures 15a-f. Notice a trend in this Mueller element as the soil-surface roughness increases. The three Reststrahlen bands first appear positive, damp, reverse sign and decrease negative with increasing $<\sigma^2>$. The rate of change in the f_{21} bandhead amplitudes seems more rapid at the largest slopes. This change of sign in the absorption bands in the [2,1] element results from a changing relationship between horizontal and vertical polarized components of backscattered light when going from smooth to rough surfaces, an anomaly of Full-Wave theory[30,31,32].

Our computer animation and visualization of the elements from a comprehensive **Full Wave** data bank show that $<h^2>$ is an insensitive parameter to change in all IR Mueller element signatures between 10 - 100 μm^2.

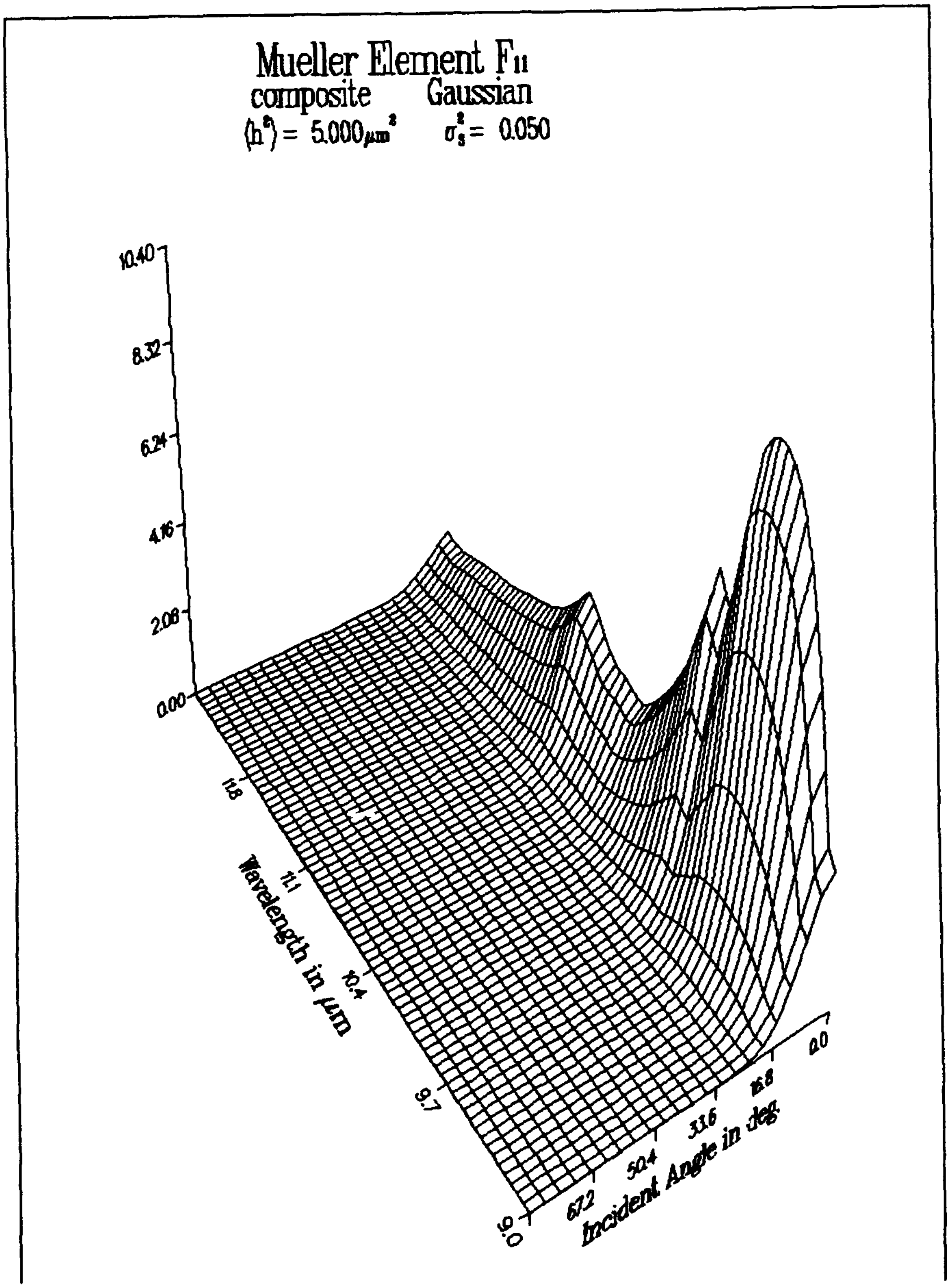


Figure 14a. Full-Wave model prediction of the Mueller matrix element f_{11} in the backscattering direction from a composite soil material assuming a surface structure of Gaussian distributed slopes (σ) and heights (h). The topographical mean-squared surface height (in μm^2) and slope of the soil sample in (a) are, respectively, 5.0 and 0.05. Note that as $<\sigma^2>$ increases in the following five figures b-f, the surface becomes a more Lambertian-like (isotropic) reflector as expected. The model restricts $<\sigma^2> \leq 2$.

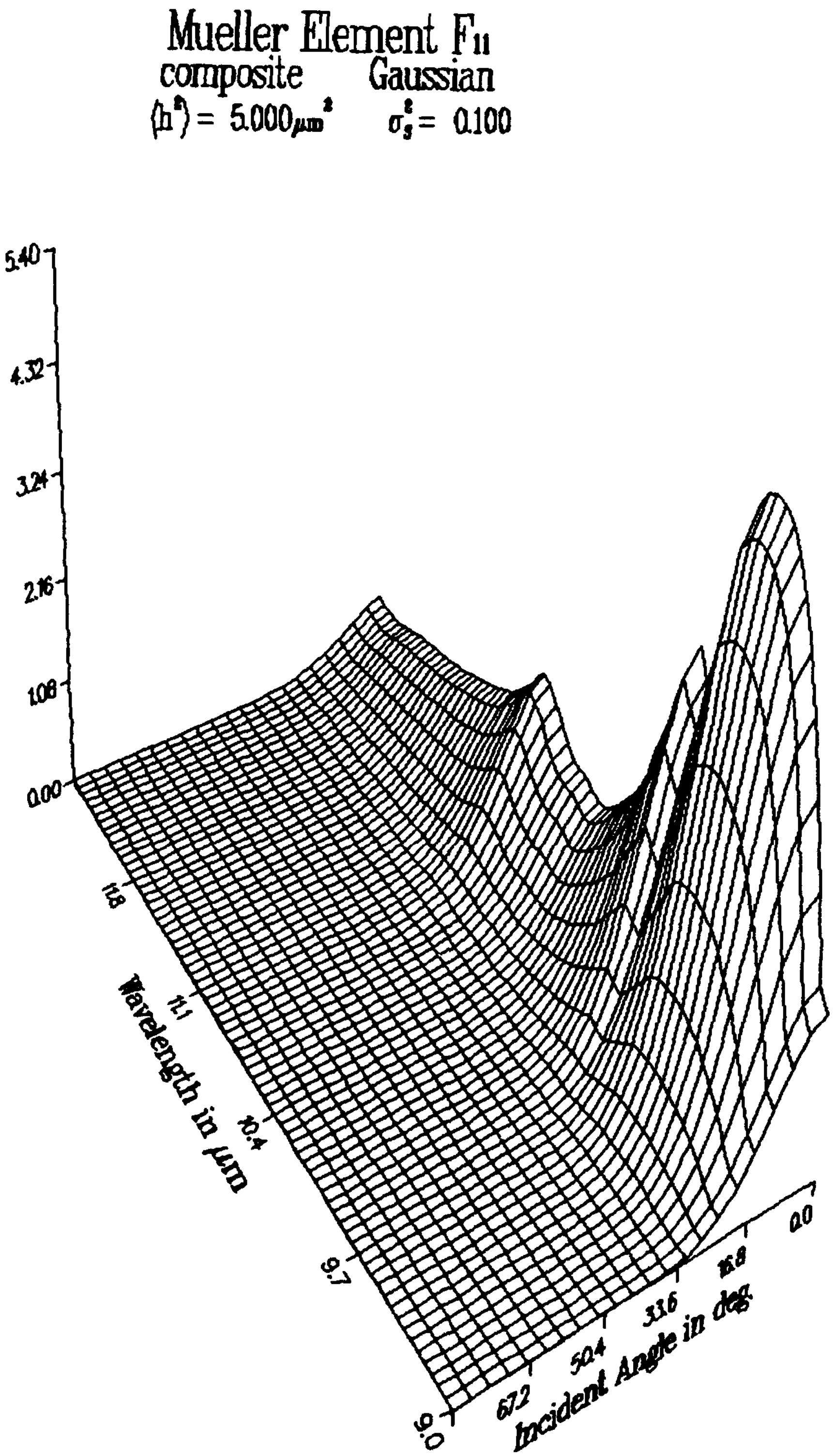


Figure 14b. Full-Wave model prediction of the Mueller matrix element f_{11} in the backscattering direction from a composite soil material assuming a surface structure of Gaussian distributed slopes (σ) and heights (h). The topographical mean-squared surface height (in μm^2) and slope of the soil sample in (a) are, respectively, 5.0 and 0.10.

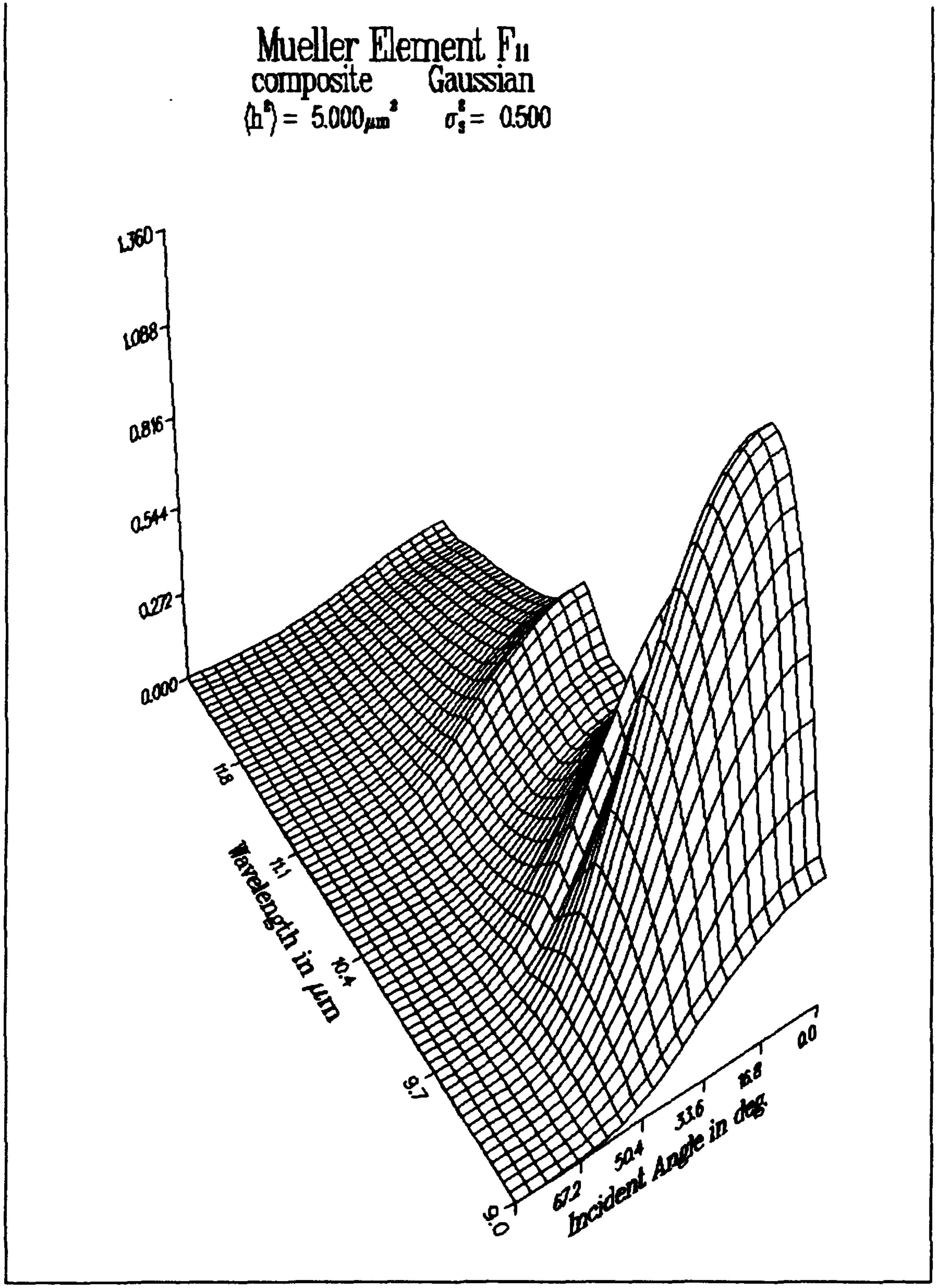


Figure 14c. Full-Wave model prediction of the Mueller matrix element f_{11} in the backscattering direction from a composite soil material assuming a surface structure of Gaussian distributed slopes (σ) and heights (h). The topographical mean-squared surface height (in μm^2) and slope of the soil sample in (a) are, respectively, 5.0 and 0.50.

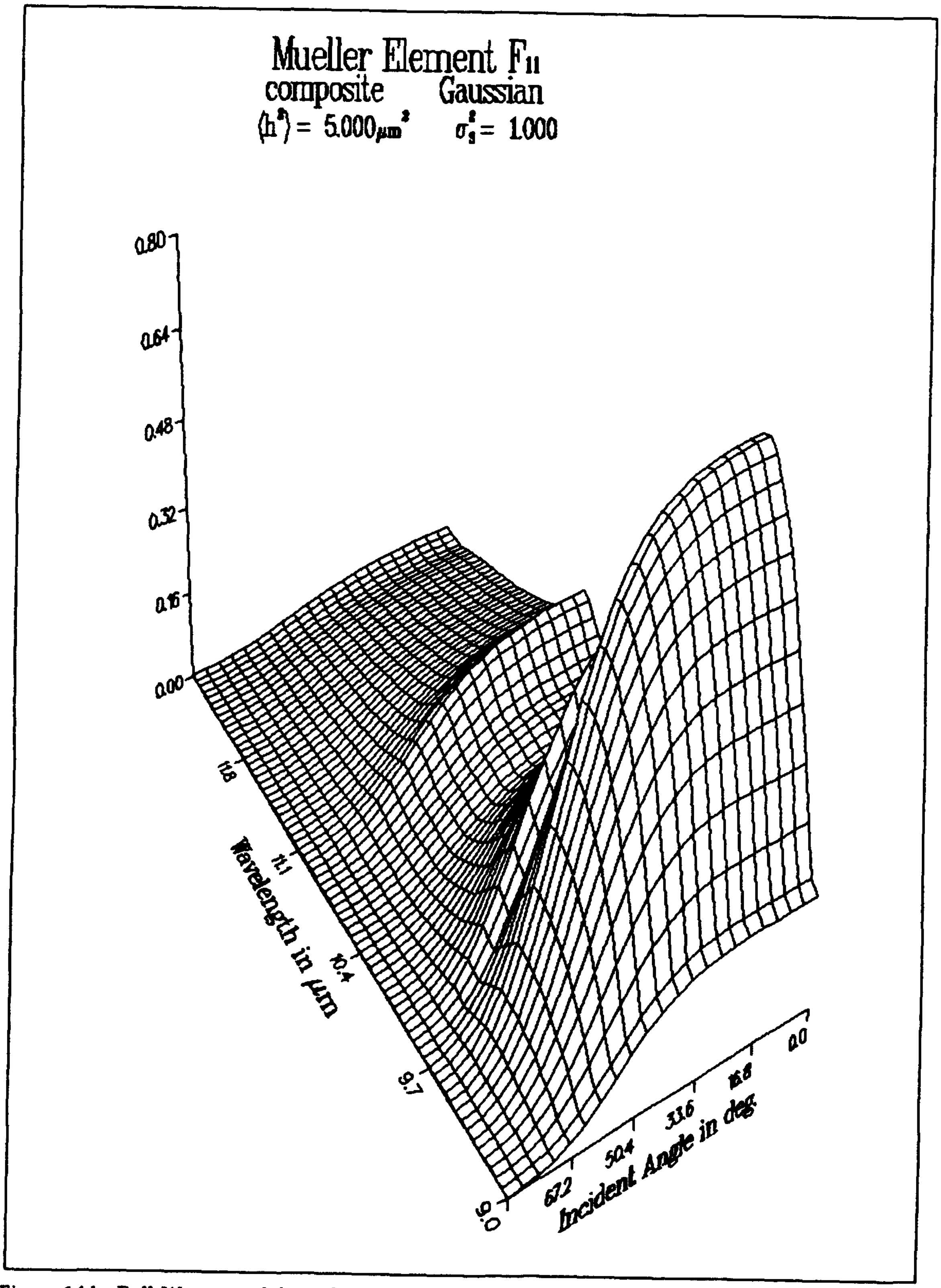


Figure 14d. Full-Wave model prediction of the Mueller matrix element f_{11} in the backscattering direction from a composite soil material assuming a surface structure of Gaussian distributed slopes (σ) and heights (h). The topographical mean-squared surface height (in μm^2) and slope of the soil sample in (a) are, respectively, 5.0 and 1.00.

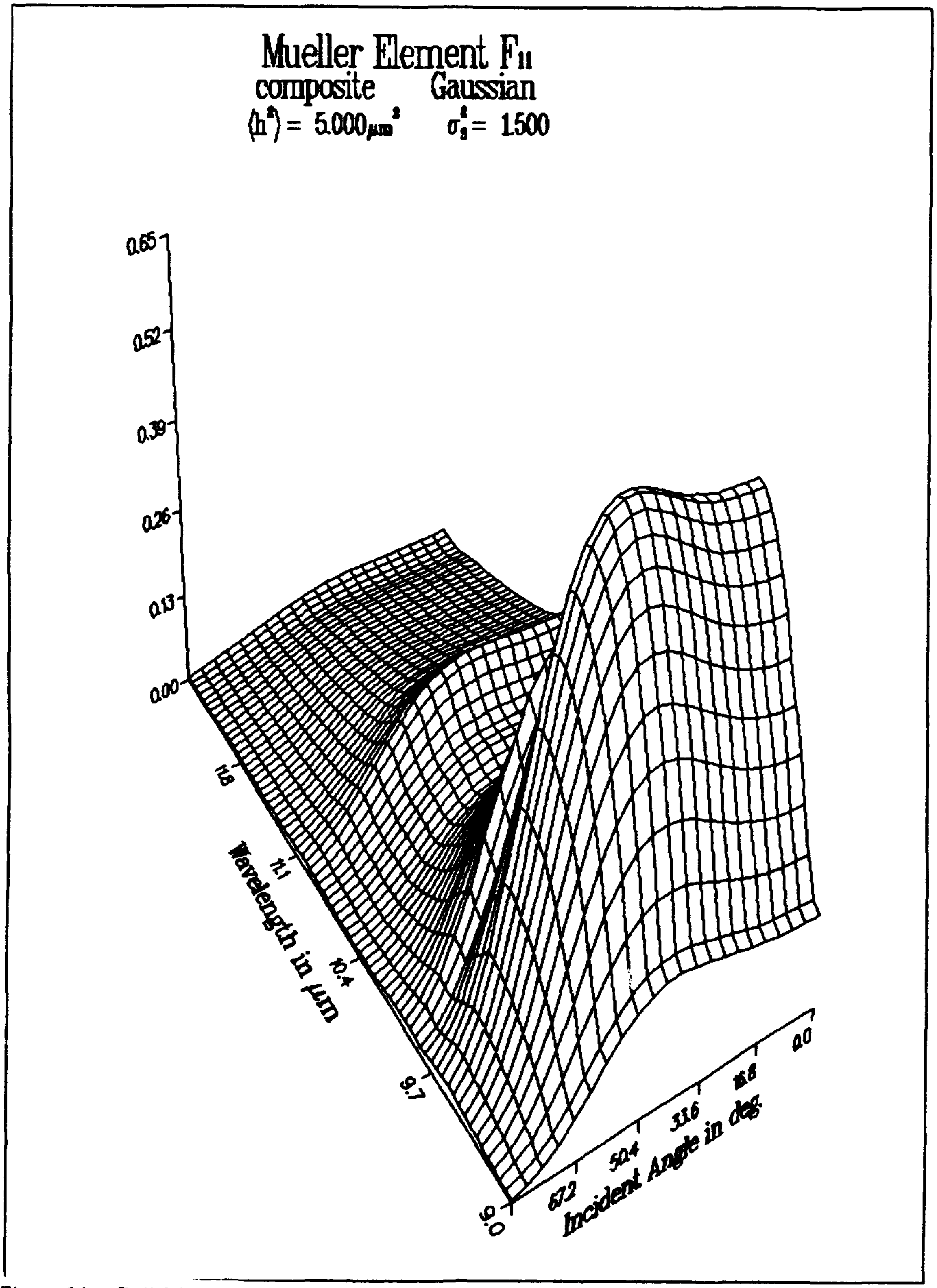


Figure 14e. Full-Wave model prediction of the Mueller matrix element f_{11} in the backscattering direction from a composite soil material assuming a surface structure of Gaussian distributed slopes (σ) and heights (h). The topographical mean-squared surface height (in μm^2) and slope of the soil sample in (a) are, respectively, 5.0 and 1.50.

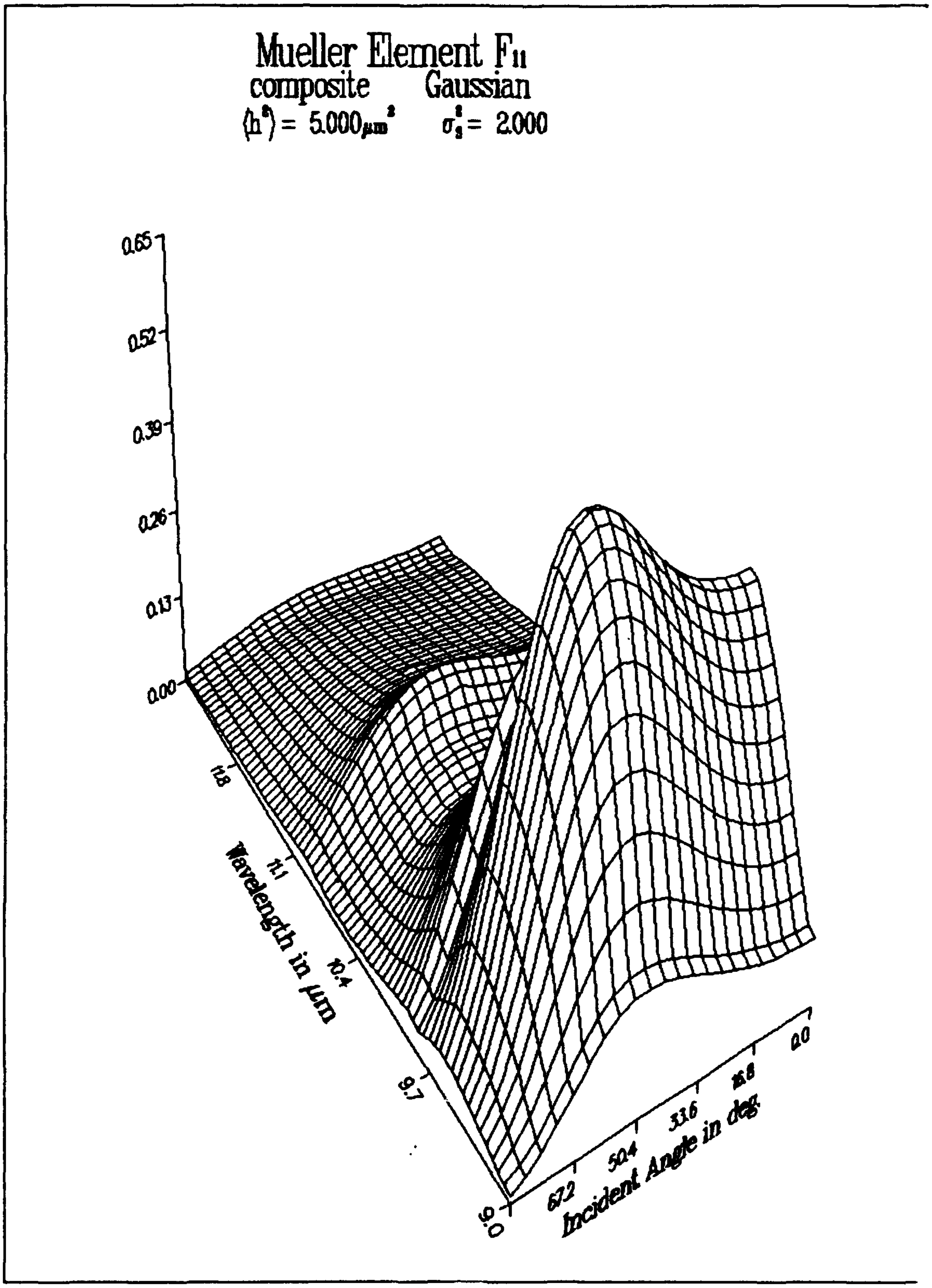


Figure 14f. Full-Wave model prediction of the Mueller matrix element f_{11} in the backscattering direction from a composite soil material assuming a surface structure of Gaussian distributed slopes (σ) and heights (h). The topographical mean-squared surface height (in μm^2) and slope of the soil sample in (a) are, respectively, 5.0 and 2.00.

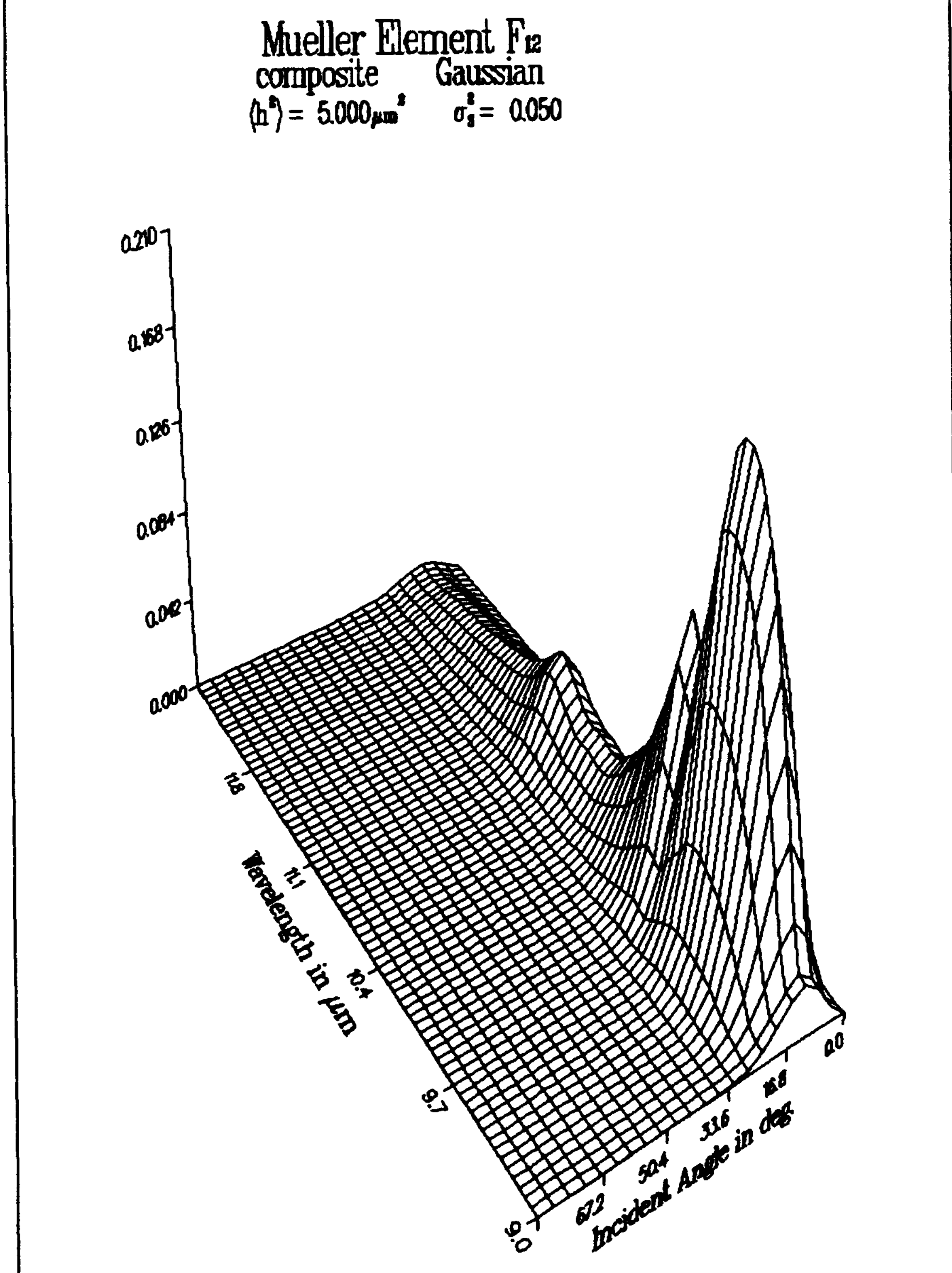


Figure 15a. Full-Wave model prediction of the Mueller matrix element f_{12} in the backscattering direction from a composite soil material assuming a surface structure of Gaussian distributed slopes (σ) and heights (h). The topographical mean-squared surface height (in μm^2) and slope of the soil sample in (a) are, respectively, 5.0 and 0.05. Note the sign reversal in the material's bandheads as $<\sigma^2>$ increases in the following figures b-e. The model restricts $<\sigma^2> \leq 2$.

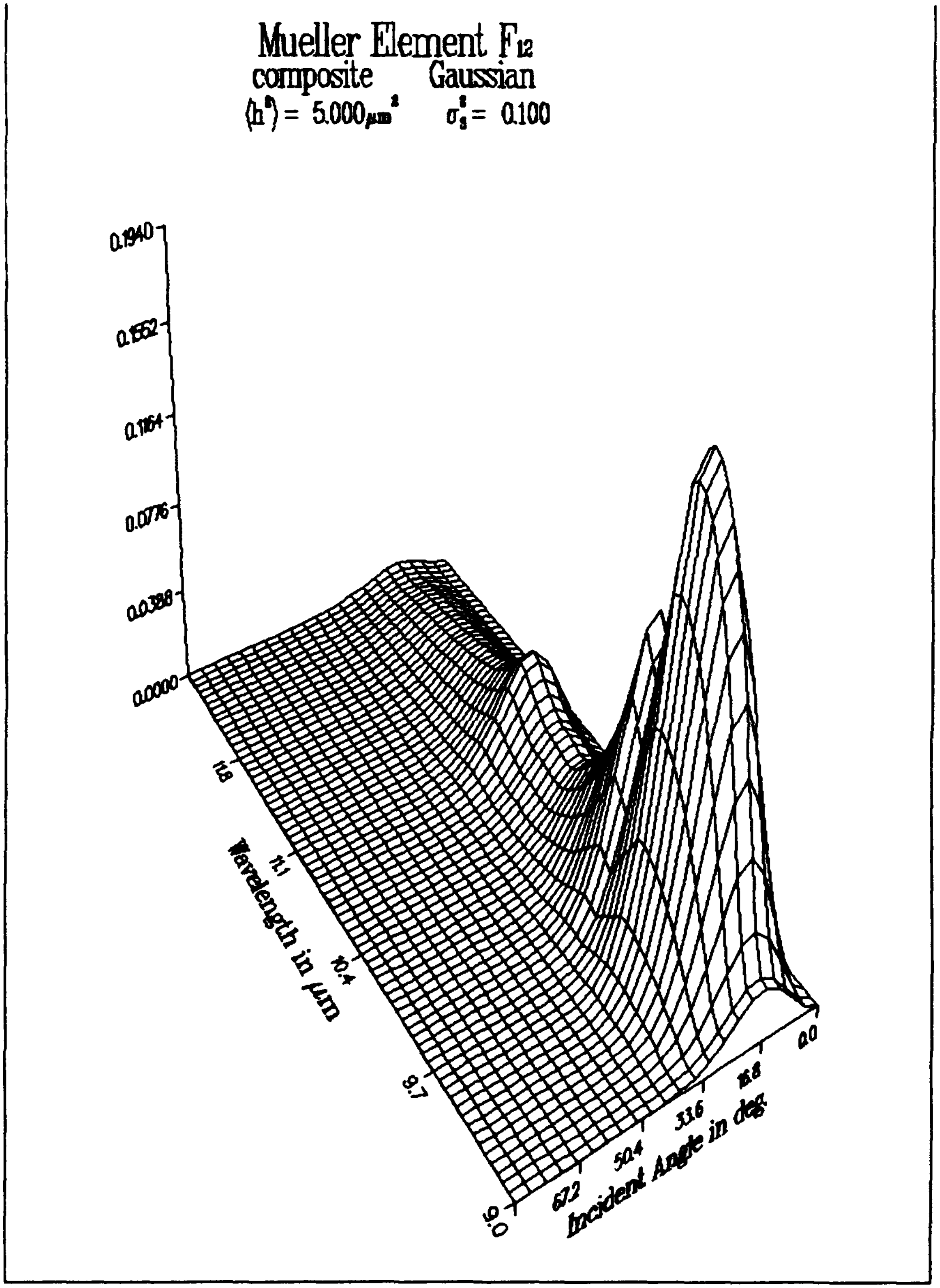


Figure 15b. Full-Wave model prediction of the Mueller matrix element f_{12} in the backscattering direction from a composite soil material assuming a surface structure of Gaussian distributed slopes (σ) and heights (h). The topographical mean-squared surface height (in μm^2) and slope of the soil sample in (a) are, respectively, 5.0 and 0.10.

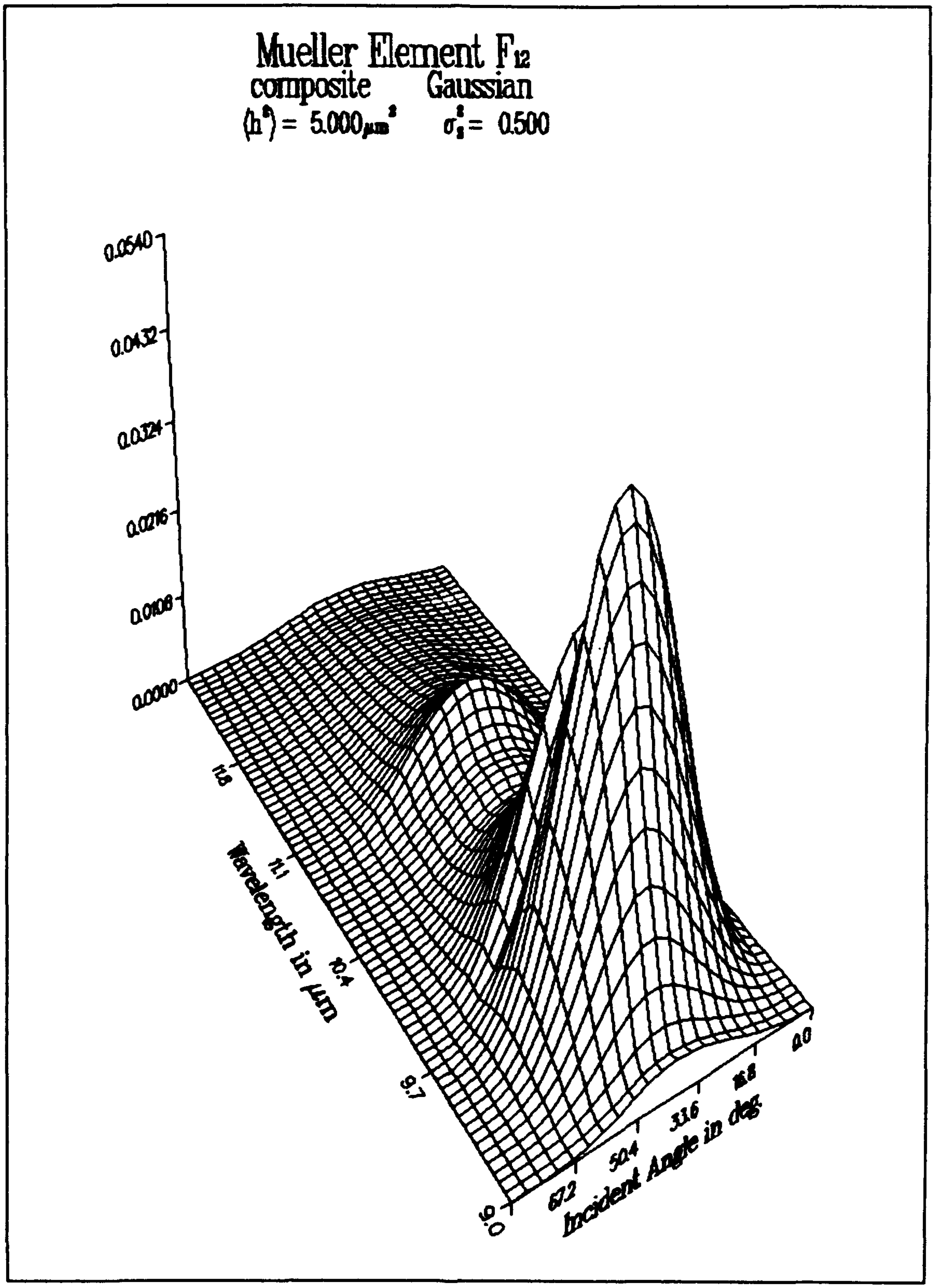


Figure 15c. Full-Wave model prediction of the Mueller matrix element f_{12} in the backscattering direction from a composite soil material assuming a surface structure of Gaussian distributed slopes (σ) and heights (h). The topographical mean-squared surface height (in μm^2) and slope of the soil sample in (a) are, respectively, 5.0 and 0.50.

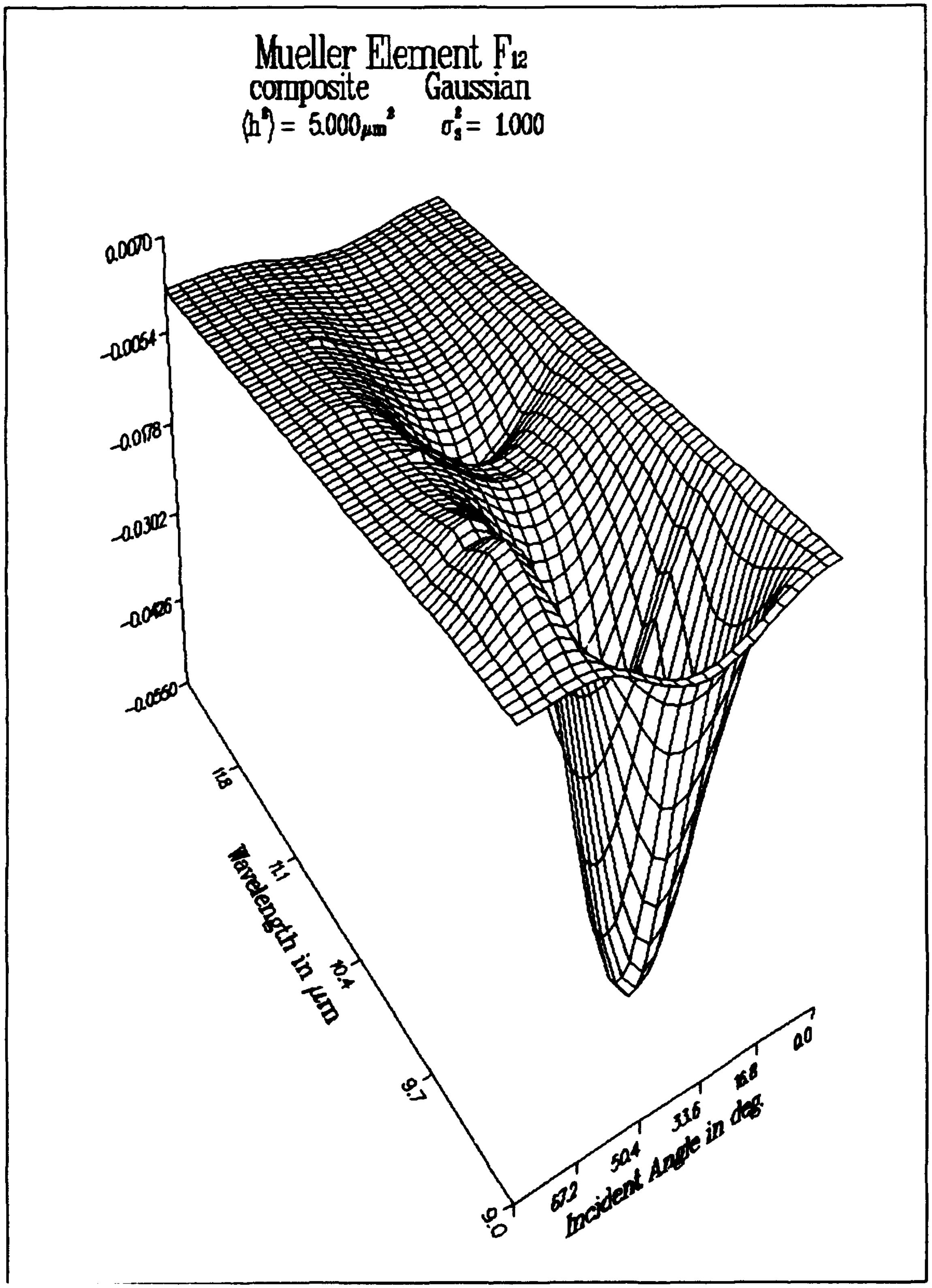


Figure 15d. Full-Wave model prediction of the Mueller matrix element f_{12} in the backscattering direction from a composite soil material assuming a surface structure of Gaussian distributed slopes (σ) and heights (h). The topographical mean-squared surface height (in μm^2) and slope of the soil sample in (a) are, respectively, 5.0 and 1.00.

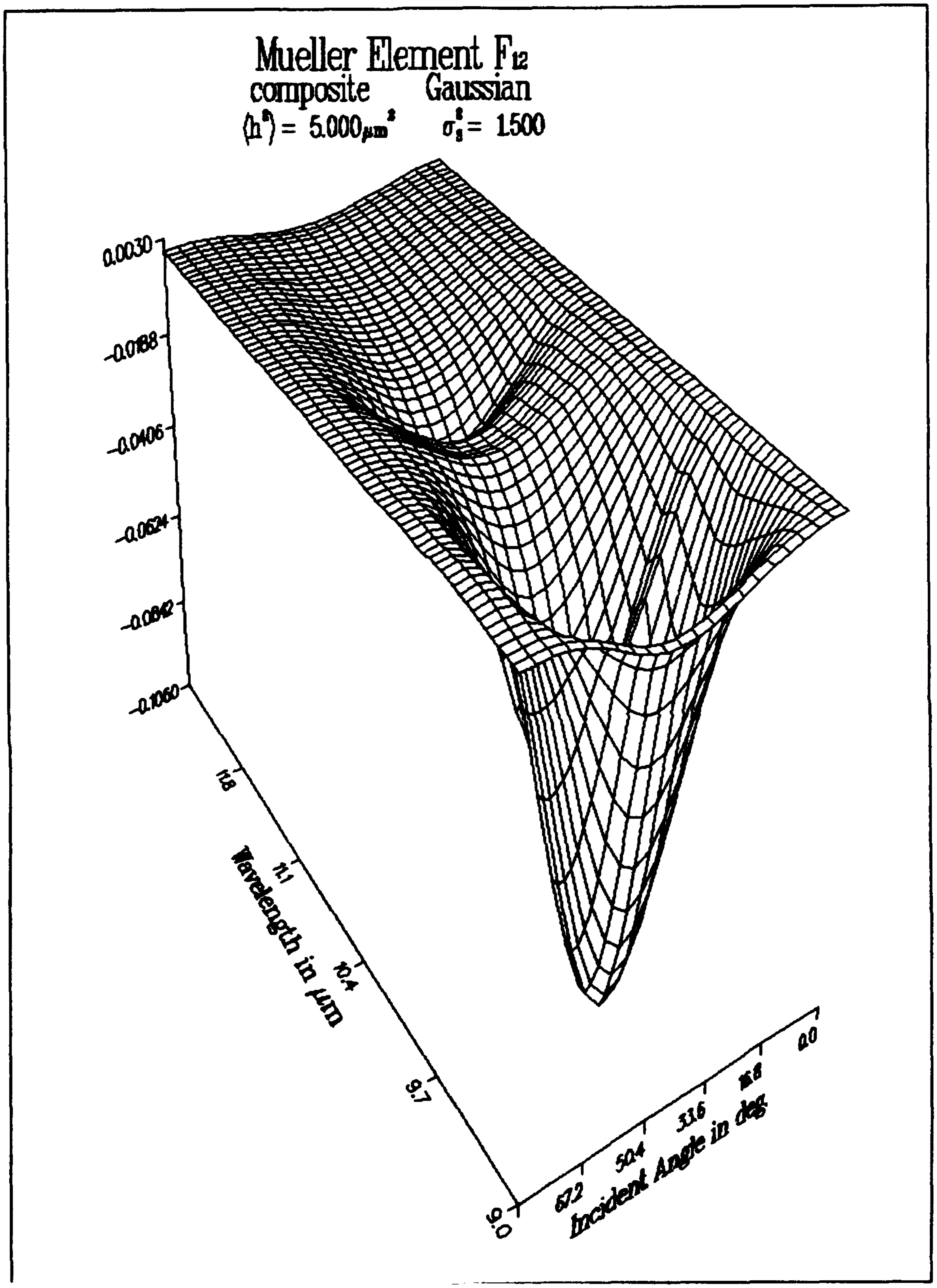


Figure 15e. Full-Wave model prediction of the Mueller matrix element f_{12} in the backscattering direction from a composite soil material assuming a surface structure of Gaussian distributed slopes (σ) and heights (h). The topographical mean-squared surface height (in μm^2) and slope of the soil sample in (a) are, respectively, 5.0 and 1.50.

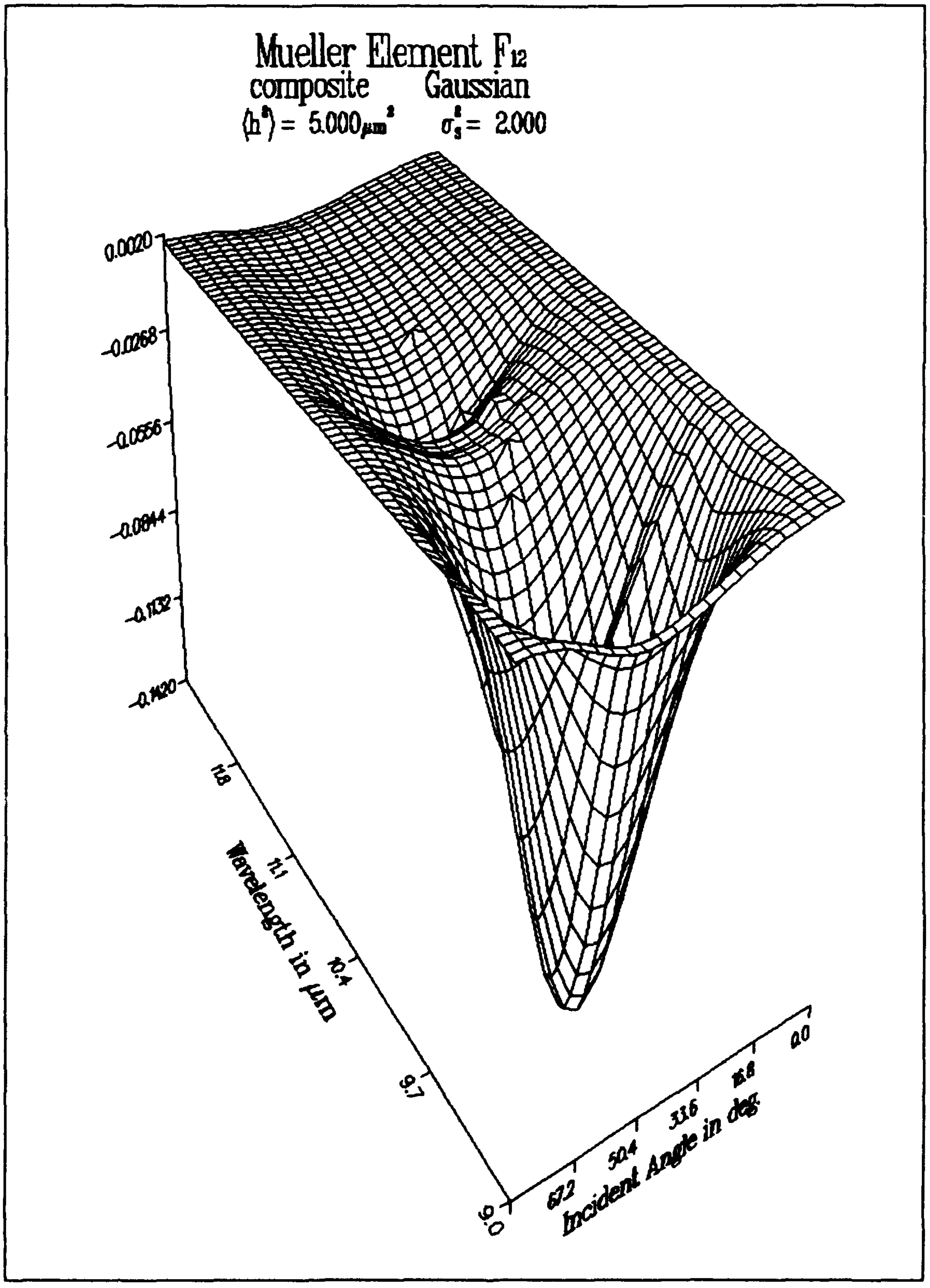


Figure 15f. Full-Wave model prediction of the Mueller matrix element f_{12} in the backscattering direction from a composite soil material assuming a surface structure of Gaussian distributed slopes (σ) and heights (h). The topographical mean-squared surface height (in μm^2) and slope of the soil sample in (a) are, respectively, 5.0 and 2.00.

An example of how one could choose laser beam angles and wavelengths, per Mueller element, for detecting contaminants DMMP, DIMP and SF96 on a soil surface is illustrated in Figures 16 through 18. Contours $F(\lambda,\theta)$ in the $\lambda - \theta$ plane are plotted for Mueller elements f_{11}, f_{12}, and f_{34} for a clay surface with $<h^2> = 20\mu m^2$, and $\sigma_s^2 = 0.5$, and elements of the same surface coated by SF96, DMMP, and DIMP contaminant liquids. (The assumption here is that after the soil is wetted by the liquid contaminant, its surface becomes uniformly coated and conforms to the unwetted soil surface geometry. Also, the coating is optically thick.) The cross-hatched sections in these figures are (θ, λ) regions where the analyte can be discerned from the soil background. These regions are *set* subtractions of data from dry and contaminated soil, and are clearly contrasted in the Mueller elements. The programs RETRO/DISPLAY (Sections 5.2.1-5.2.2, and Appendix V) were executed in producing these data. In the f_{11} element, SF96- and DMMP-contaminated soils yield predominate analyte signals at the higher wavelengths starting $\approx 12.2\mu m$, and at angles not exceeding 48^0, while DIMP cannot be distinguished in this Mueller element. In element f_{12}, Figure 17, SF96- and DMMP-contaminated surfaces are still disjoint from the dry soil surface at the higher wavelengths, but their detection λ–θ domain is more restrictive in angle. DIMP still cannot be detected via f_{12}. Finally, in Figure 18, we see that DIMP can be detected via f_{34} in the cross-hatched region near 10.15 μm and 12^0–48^0, as are SF96 and DMMP in the higher λ–θ domains.

This kind of graphical analysis can be extended to the remaining independent Mueller elements, and a selection of angles and wavelengths can likely be chosen to discriminate against and between a vast group of chemical contaminants that exhibit an IR vibrational spectrum. Moreover, through experiment and theoretical modelling, we hope to establish how these domain patterns change with target concentration, allowing one to map the contaminant once detection is established, and monitor its fate.

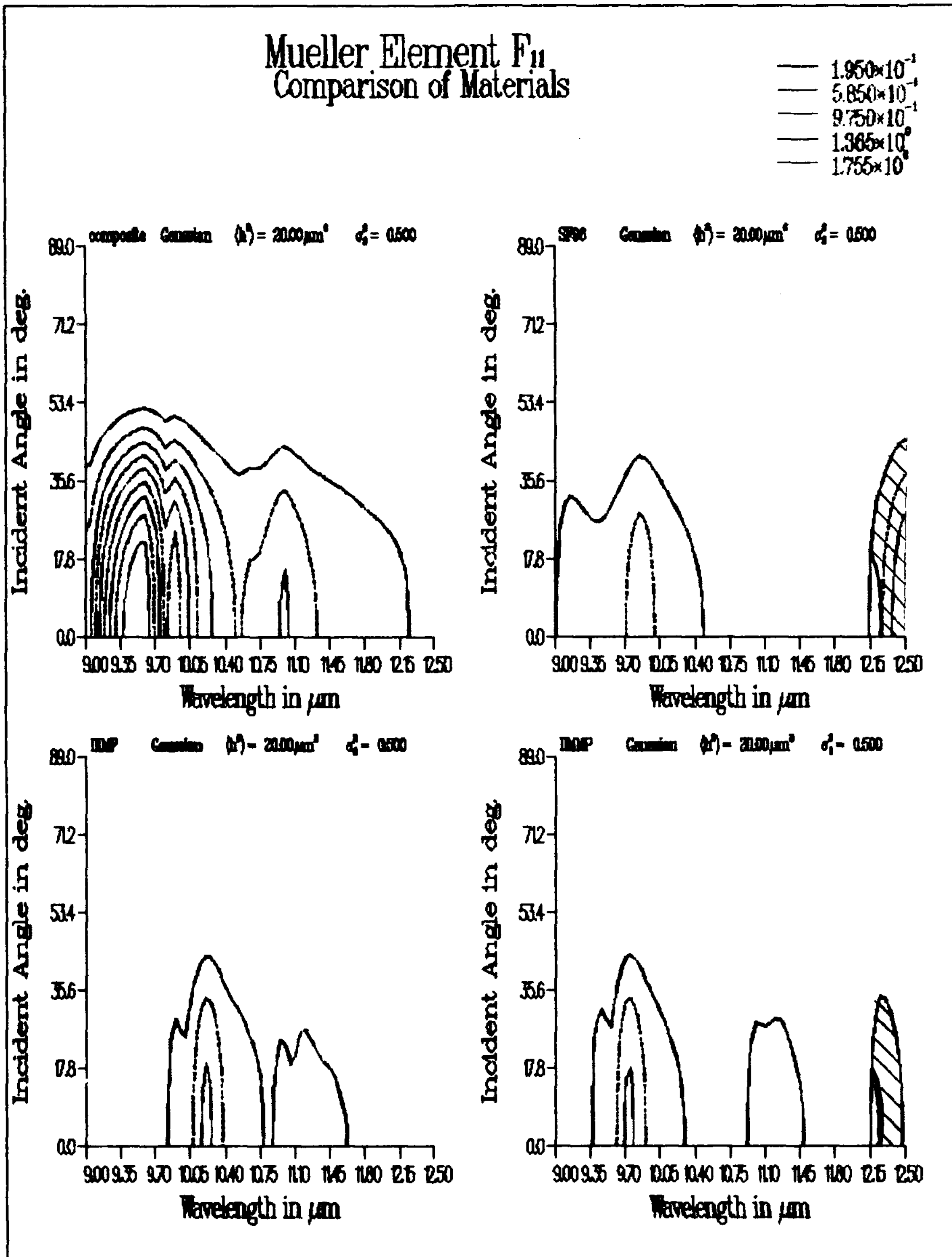


Figure 16. Regions of irradiation wavelength and backscattering angle in the f_{11} Mueller matrix element that are most useful for discriminating against liquid chemical agent simulants SF96, DIMP, and DMMP on a soil surface. The crosshatched wavelength-angle domains are areas where the ellipsometer sensor should be set to, so that a signal from the contaminants can be detected. These regions of maximum analyte detections result from a subtraction of dry- from wet-soil data sets. Note that DIMP cannot be detected in f_{11}.

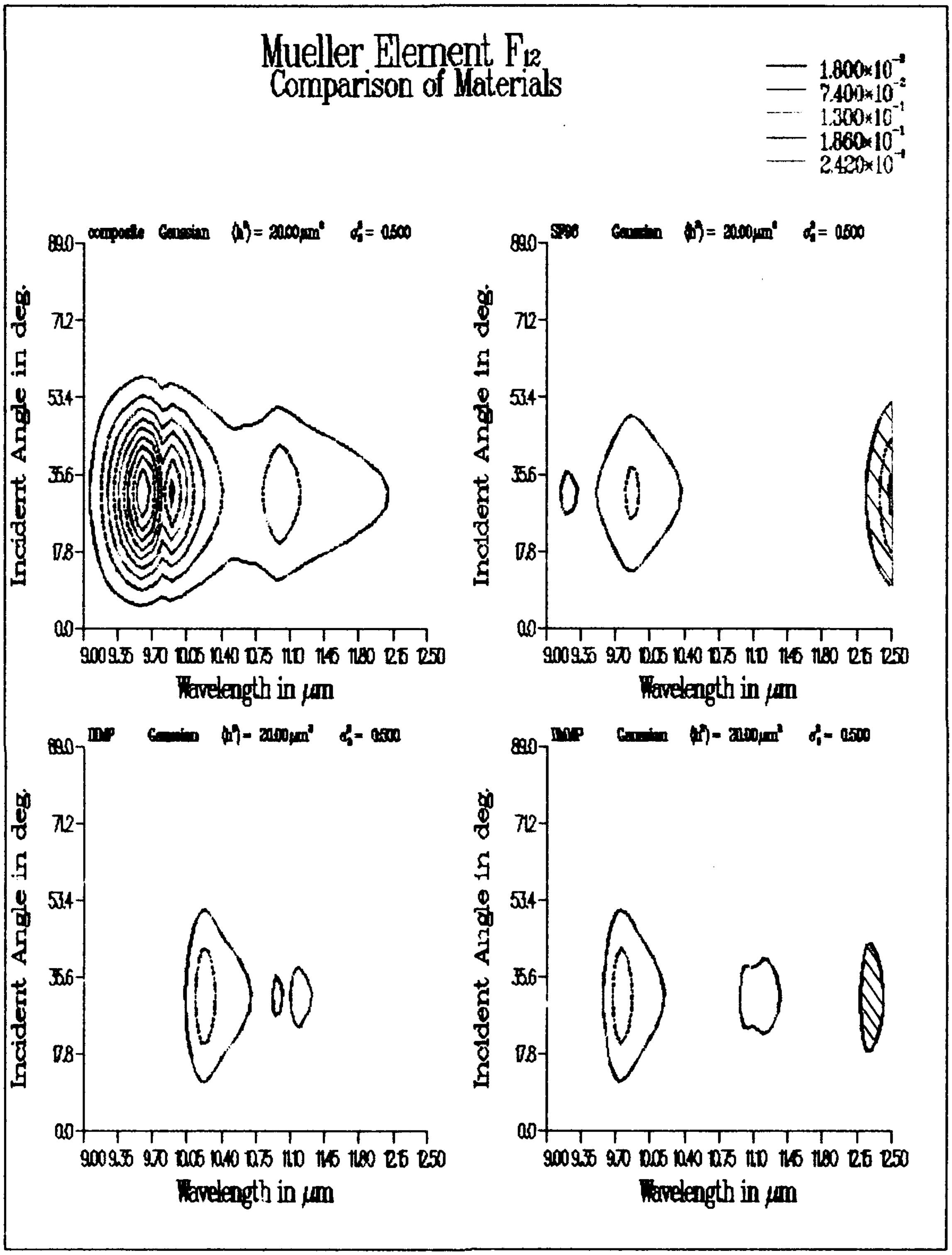


Figure 17. Regions of irradiation wavelength and backscattering angle in the f_{12} Mueller matrix element that are most useful for discriminating against liquid chemical agent simulants SF96, DIMP, and DMMP on a soil surface. The crosshatched wavelength-angle domains are areas where the ellipsometer sensor should be set to, so that a signal from the contaminants can be detected. These regions of maximum analyte detections result from a subtraction of dry- from wet-soil data sets. Note that DIMP cannot be detected in f_{12}.

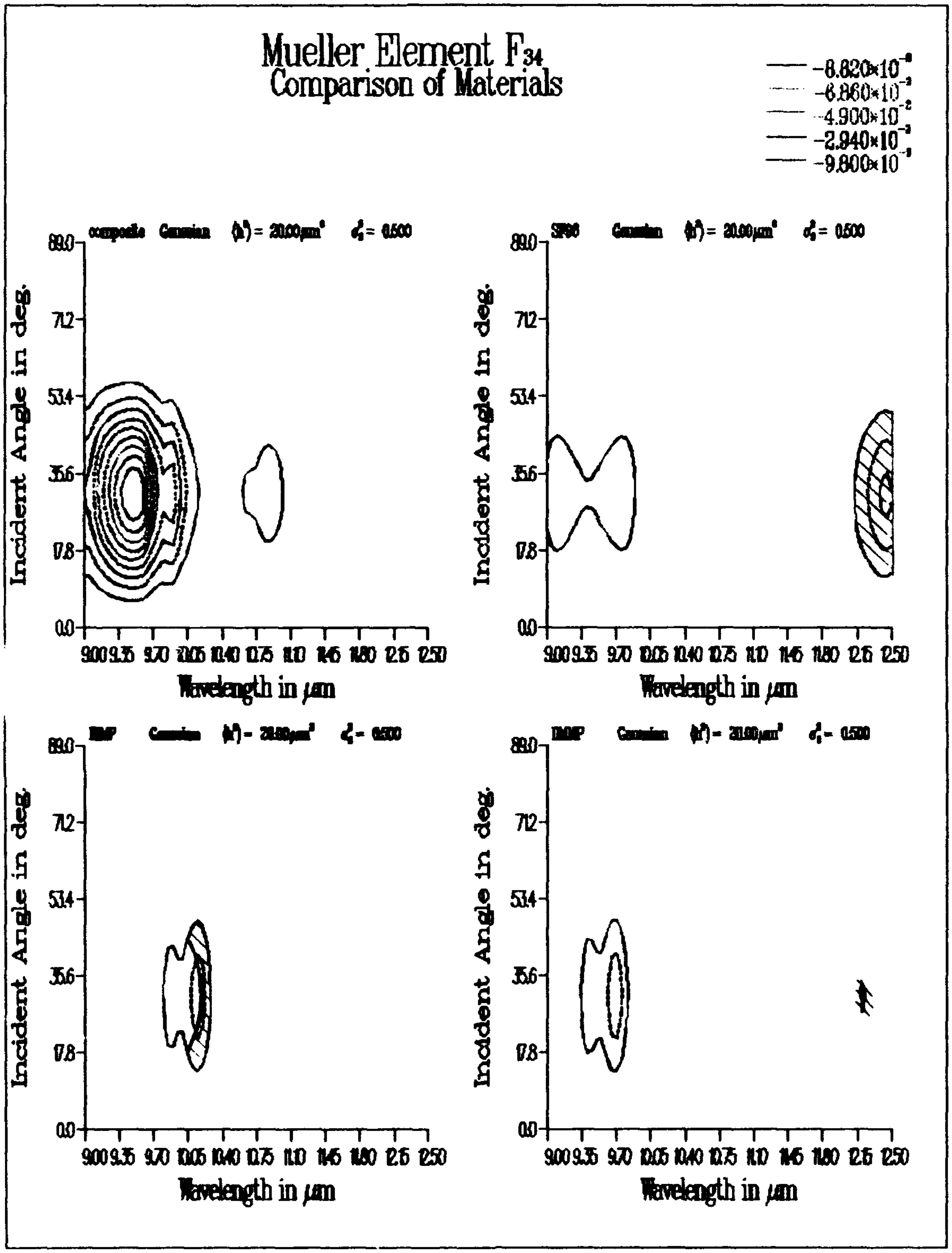


Figure 18. Regions of irradiation wavelength and backscattering angle in the f_{34} Mueller matrix element that are most useful for discriminating against liquid chemical agent simulants SF96, DIMP, and DMMP on a soil surface. The crosshatched wavelength-angle domains are areas where the ellipsometer sensor should be set to, so that a signal from the contaminants can be detected. These regions of maximum analyte detections result from a subtraction of dry- from wet-soil data sets. Note that all three simulants can be detected in f_{34}.

5.2.3 DETECT: Remote Detection Application of Full Wave Theory.

Program DETECT, written by Mark Haugland, is a detection algorithm for this multi-wavelength, two-modulator IR ellipsometer. The program serves 2 purposes: (1) to locate optimum angles of incidence and wavelengths for use in discriminating between a contaminated and a dry surface; and (2) to identify those Mueller matrix elements that can be used to discriminate between the analyte (contaminant) and the background (substrate) at optimum angles of incidence and laser wavelengths.

The Mueller matrix for each scattering surface is a function of wavelength, incident angle, mean square height, and mean square slope. One way to select useful combinations of incident angle and wavelength is to, as described in the previous section, visually inspect DISPLAY plots of the difference of the Mueller matrices for both contaminated and bare materials. Due to the sheer amount of data involved, this method is time consuming and most difficult when extracting quantitative information quickly.

The most current version of Full Wave theory can compute Mueller matrices for stratified media with an optically thick contaminant layer, i.e., one rough interface is considered. Given a layered boundary value problem, the backscattering Mueller matrix **F** for one or more randomly rough interfaces has 6 linearly independent entries, given the assumptions and restrictions on the media prescribed previously in the Full Wave model. These matrix elements are used in the construction of a 6-dimensional vector $\bar{v}$ in the following manner.

$$\bar{v} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{bmatrix} \tag{32}$$

$$v_1 = f_{11},\ v_2 = f_{12},\ v_3 = f_{22},\ v_4 = f_{33},\ v_5 = f_{34},\ v_6 = f_{44}. \tag{33}$$

Let $\bar{v}^b$ represent the background material, and $\bar{v}^t$ represent the target material. As a first step, consider a vector $\bar{d}$ defined by

$$\bar{d} = \bar{v}^t - \bar{v}^b. \tag{34}$$

The magnitude of $\bar{d}$ is given by

$$|\bar{d}| = (\sum_{k=1}^{6} d_k^2)^{\frac{1}{2}}. \tag{35}$$

The magnitude of Equation 35 is the first step in selecting a useful detection discriminant (combination of incident angle and wavelength). Terms in Equation (35) with $|\vec{v}_k^b| > |\vec{v}_k^t|$ are excluded, i.e., relatively strong returns from the background material filtered (similar to the subtracted data sets of Section 5.2.2). The following equation uses step functions to disregard relatively strong returns from the background material.

$$x_k = d_k \, u(|v_k^t| - |v_k^b|) \tag{36}$$

Here, u(.) is a unit step function. Candidate incident angle and wavelength detection parameters are found by calculating

$$|\bar{x}| = (\sum_{k=1}^{6} x_k^2)^{\frac{1}{2}} \tag{37}$$

and enforcing the condition $|\bar{x}| > |\bar{x}|_{max}$. The useful Mueller matrix detection elements at these wavelength and incident angle pairs correspond to the nonzero components of $\bar{x}$.

Consider cases where $x_k \gg x_j$. Often, this results when v_k^t and v_k^b are large in magnitude compared to v_j^t and v_j^b. Consequently, x_j has negligible effect on the value of Equation (37) even though v_j^t and v_j^b may exhibit strikingly different behavior. Scaling each component of x by an appropriate factor will yield a test producing results that depend on the relative size of two matrix elements rather than the magnitude of their difference. The vector $\bar{y}$ has this property

$$y_k = \frac{x_k}{\sqrt{|v_k^t v_k^b|}}. \tag{38}$$

As in Equation (37), combinations of incident angle and wavelength of particular interest may be found by computing

$$|\bar{y}| = (\sum_{k=1}^{6} y_k^2)^{\frac{1}{2}} \tag{39}$$

and requiring $|\bar{y}| > |\bar{y}|_{max}$. Again, useful Mueller matrix elements correspond to the non-zero components of y.

Should a matrix element for either the target or the background tend toward zero, Equation (39) becomes singular (tends to infinity). For this reason, it is necessary to set y_k equal to zero in the DETECT program whenever v_k^b or v_k^t are zero.

The refractive index n(λ) - ik(λ) of a material is a complex function of wavelength which plays an instrumental role in determining a material's response to an incident photon. Peaks in the Mueller matrix elements usually occur at resonant wavelengths. Resonant wavelengths correspond to local maxima in k.

DETECT identifies all resonant wavelengths for both the background and the target materials. The program also identifies all local minima in k. DETECT helps the user identify the correlation between on- and off-resonance wavelengths and numerical results from Equations (36), (37), (38), and (39).

We have thus far developed the criterion for finding useful combinations of incident angle, wavelength, per Mueller matrix element susceptible to the analyte. We have not yet accounted for the variational error of parameters expected in experimental operation. Inputing several sets of experimental data to the current version of DETECT, computing average values for $\bar{v}^b$ and $\bar{v}^t$, and dividing each component of these vectors by their respective variances may account for variability in the experimental measurements.

The remainder of this section discusses a more sophisticated way of accounting for experimental variability than averaging data and dividing by variances. We discuss simulation of experimental uncertainties with the theoretical data base. First, however, we review the definitions of variance, covariance, and the covariance matrix.

The variance of a univariate quantity z is defined by

$$\sigma_z = \frac{1}{N}\sum_{i=1}^{N}(z_i - z_{ave})^2 \tag{40}$$

where N is the number of samples taken and z_{ave} is the mean value of z over N samples. N should be large enough so that increasing it will not change σ_z. The expected or average value $E(\bar{r})$ of a random vector $\bar{r}$ is the vector whose components are the average value of each component of $\bar{r}$, that is:[35]

$$E(\bar{r}) = \begin{bmatrix} E(r_1) \\ E(r_2) \\ \cdots \\ \cdots \\ E(r_k) \end{bmatrix}. \tag{41}$$

When generalizing variance to multidimensional quantities, one defines the covariance of 2 components r_i and r_j of a random vector $\bar{r}$ by:[35]

$$cov(r_i, r_j) = E\,[(r_i - E(r_i))(r_j - E(r_j))]. \tag{42}$$

In Equation (42), the covariance of r_i and r_j is the average of the product of r_i's and r_j's deviation from their respective mean values. For i=j, Equation (42) is the variance of r_j. If r_i and r_j are uncorrelated, then Equation (42) is identically zero. The covariance matrix contains the covariances of all components of $\bar{r}$. The elements of the covariance matrix are arranged according to the following definition:

$$\Sigma = E\,[(\bar{r} - E(\bar{r}))(\bar{r} - E(\bar{r}))^T] \tag{43}$$

where symbol T denotes transposition. From Equation (43), it is clear that $\Sigma_{ij} = cov(r_i, r_j)$ and that the covariance matrix is symmetric.

In the next section it is assumed that the covariance matrix is positive definite. This is necessary to insure that the quadratic forms in question are ellipsoidal.[35]

Hotelling's T-squared method is one way to check if a hypothesis is true or false. For this application, the first hypothesis is that a given backscatter angle/wavelength combination is useful. The second hypothesis is that the contaminant is present. Throughout this section, it is assumed that the covariance matrices for the contaminated and uncontaminated surfaces are equal.

This method uses the boundary of an ellipsoid as the test criterion. The ellipsoidal region is defined by

$$\bar{x}^T \Sigma^{-1} \bar{x} = c^2 \tag{44}$$

where c is a constant, Σ^{-1} is the inverse of the n dimensional covariance matrix for the data contained in $\bar{x}$, and $\bar{x}$ is a $1 \times n$ column vector whose entries represent the average value of the quantity defined by Equation (35). Optimum angles and wavelength are those for which

$$c > c_{max} \tag{45}$$

i.e., $\bar{x}$ lying outside of the ellipsoid defined by Equation (44). As earlier stated, useful Mueller matrix elements correspond to the non-zero components of $\bar{x}$.

Now that a set of useful angle-wavelength pairs have been found, it is time to use them to identify a contaminant. One way to accomplish this is to evaluate Equation (44) at several

* In Equation (36), k is 6. Experimental results may show that there are more than 6 linearly independent Mueller matrix elements. For this reason, n is left as an unknown dimension ≤ 16.

angles and wavelengths, say m, for an unknown sample, then store these values in the $1 \times m$ column vector $\bar{c}$. Let $\bar{c}_{ref}$ denote the value of $\bar{c}$ for an analyte on some surface. Define $\bar{a}$ by:

$$\bar{a} = \bar{c} - \bar{c}_{ref}. \tag{46}$$

The contaminant is present if $|\bar{a}| < |\bar{a}|_{max}$, and its concentration (mass density) is approximated by using $\bar{c}_{ref}$ representing various densities in Equation (46). The $\bar{c}_{ref}$ that results in the smallest $|\bar{a}|$ is the closest approximation to $\bar{c}$. Hence, the unknown sample has approximately the same concentration as the known sample whose $\bar{c}_{ref}$ resulted in the smallest $|\bar{a}|$.

Using Equation (46) to identify contaminants works in principle, but information is not used to a full extent in representing the Mueller matrix data collected at each angle and wavelength by a single scalar. This method may involve using more angle-wavelength pairs than are necessary for ascertaining the analyte. However, using a single scalar to represent the independent Mueller matrices per angle and wavelength demands considerably less computer memory.

Incorporating noise into the theoretical data and substituting in Equations (45) and (46) provides a way to simulate experimental uncertainties. One way to do this is to add a random component to the input variables of RETRO. For example, slightly varying the rough surface geometry (mean square height and slope) in a random manner simulates a scanning incident beam irradiating areas sample-to-sample.

5.2.4 DECIDE2: A Detection Optimization Algorithm.

Program DECIDE2 computes and analyzes backscatter Mueller matrices every time it calls its subroutine RETRO. These data are used to better distinguish between background (base) and target (analyte) materials. In performing its intended function, DECIDE2 determines which Mueller matrix elements are of use at wavelengths and incident angles susceptible to the analyte.

DECIDE2 is an alternative to using the DISPLAY plotting package for graphical discrimination analyses. DECIDE2 locates primary resonant wavelengths for each material. It then locates the beam wavelengths at which the difference in the imaginary part of refractive index between target and background are maximum. At each of these wavelengths, DECIDE2 computes Mueller matrices for both materials as a function of incident angle. Immediately following this computation, each pair of corresponding Mueller matrices is separately analyzed (Equations 37 and 38).

The program DECIDE2 identifies the combination of these wavelengths and incident angles that result in a probable discrimination between the two unlike materials. These angle/wavelength pairs are slightly varied and reexamined. If there is an increase in discrimination characteristics between varied angles and wavelengths, then the program stores those new parameters. A new variation in angle and wavelength about these values are interrogated next, and so on. This 'seeking' program iterates the interrogation process until no further increase in discrimination has been detected. Once the program has located the optimum angle and wavelength, the computer proceeds with its analysis of 121 more Mueller matrices for angles of incidence and wavelengths near other initial optimum pairs. These 121 Mueller matrices, with analysis results, are written in a file of format readable to DISPLAY.

6. DISCUSSION OF FUTURE WORK

An alternate digital method for data acquisition, and a neural network interface to the analog detection output module are presented in this section. The new digital processing methods we are now exploring should result, if successful, in a turn-key data acquisition unit with on-board functions that filter specific frequencies in the scattered light intensity much like the lock-in electronics of the analog data acquisition.

6.1 Digital Data Acquisition and Signal Processing of the Scattergram.

We begin this section by summarizing the current method, which uses separate lock-in amplifiers, for determining the normalized Mueller matrix elements at a given wavelength and scattering angle. Let us assume that the driving amplitudes on the two photoelastic modulators have been properly set for the wavelength in use, and that the angles and orientations of all the optical elements have been correctly adjusted. Then Equation 11 shows that the output from the detector can be represented as the sum of an infinite number of discrete frequencies, namely the sums and differences of all integral multiples of the two modulator frequencies. The amplitude of each frequency component is given by the product of one or two Bessel functions of integer order (which fortunately tend toward zero as the order increases) and a factor that is one of eight Mueller matrix elements. The dc component of the detector output is proportional to a ninth Mueller matrix element, the ψ_{11} element. When the detector system's gain is actively servo-controlled to keep the dc output at a constant level, the ac components are also bounded so that their amplitudes are effectively proportional to the normalized Mueller matrix elements.

The same set of frequencies and Bessel function factors comprise the detector signal in each of the four experimental configurations: Case A,B,C, and D (see Section 4.3). All that differs among the configurations is the identity of the eight normalized Mueller matrix elements that help determine the amplitude of those frequencies. ψ_{11} is proportional to the detector dc level in each configuration.

Measurement of a normalized Mueller matrix element then is equivalent to the measurement of the amplitude of the corresponding frequency component in the detector signal. That is done initially with eight separate lock-in amplifiers, one for each frequency. Eight sinusoidal reference signals - each of which is at the same frequency as and synchronized with one of the eight desired frequencies in the detector signal - are produced by appropriate analog multiplications and filtering among sine waves derived from the reference outputs of the two PEM power supplies. The detector signal is split and sent to each lock-in amplifier board where it is first filtered through a narrow passband filter to reject most of the power except that near the desired frequency for which the particular board is designed. When the enhanced signal is finally multiplied against its corresponding reference frequency by the lock-in amplifier, the dc component of the resulting waveform is a measure of that frequency's amplitude in the detector signal and so is a measure of the corresponding (depending on experimental configuration) normalized Mueller matrix element.

Note that Equations 12, 14b, 16b, and 18b for the Mueller matrix elements are relationships among optical quantities - retardation (radians) of the two modulators and intensity (Watt cm^{-2}) incident on the detector. The voltages presented to the lock-in amplifiers' inputs are, in a sense, representations of those optical quantities. But between the optics and the lock-ins lie a great many electronic components that transform, amplify, and filter signals along the way. As a result, an expression for the Mueller matrix elements analogous to, say Equation 12, but in terms of the voltages at the lock-ins would have to include factors for the gains and phase shifts (both frequency dependent) introduced by the train of electronics. Tracking all this would be impractical; instead, on each lock-in board is included a phase shifter to shift the reference frequency relative to the signal, and a final gain control. These are adjusted and set on each board during the calibration procedure in which measurements are made for optical standards (polarizers and waveplates) whose Mueller matrix elements are known. After calibration, a reference frequency and its corresponding detector signal component will always be in phase or 180 degrees out of phase at the lock-in, depending on the sign of the corresponding Mueller matrix element.

The lock-in amplifiers will operate with a short time constant, probably within a few tenths of a second, depending on the amount of detector noise present. In other words, the (dc) output of the lock-in at any instant depends only on the input voltages over the previous few tenth seconds. It has occurred to us that if we can digitize the detector output waveform over that period of time with adequate resolution, along with waveforms representing the simultaneous modulator retardations, we ought to be able to calculate the same information that the lock-ins give and so eliminate most of the experiment's data acquisition electronics.

Three separate techniques for computing Mueller matrix elements from digitized data have suggested themselves already, and appear plausible to warrant serious investigation. We have not yet worked out all the details for any approach, but the concepts involved will be sketched out below.

The most obvious approach is to measure a fast Fourier transform (FFT) on a data stream sampled from the detector and note the amplitudes at the eight frequencies of interest. A calibration relating each Fourier amplitude to the corresponding normalized Mueller matrix amplitude would need to be performed, but in principle a simple power spectrum of the detector output will yield the magnitudes of the Mueller matrix elements. A greater effort is required to decide the signs of the elements: the FFT must compute the phase of each signal component as well as its amplitude, and the complex FFT must also be performed on simultaneously sampled sine waves synchronized with the two polarization modulators. From those three phases, with perhaps a phase correction determined in the calibration procedure, the sign of the Mueller matrix element can be worked out.

At least two ways of implementing the FFT approach are feasible. A sophisticated multichannel waveform analyser, such as the Analog 6100 or LeCroy 9424 in our laboratory, can acquire the waveforms and measure the FFT's rapidly (Figure 19). The resulting amplitudes and phases would be transferred to a PC for the final arithmetic and display and/or storage of the Mueller matrix elements. Alternately, the entire process could be carried out with a real time microcomputer or PC, using an A/D board with at least three input channels to acquire the waveforms, and software including a FFT routine to analyze them and extract the Mueller matrix elements.

A second approach is to let the computer emulate the system now in use by carrying out numerically the same multiplications and filtering that the analog electronics perform. Here, it would be more convenient to synchronize a pair of reference sawtooth waveforms (rather than sine waves) with the polarization modulators, so that a sampled voltage represented the instantaneous phase (rather than amplitude) of the retardation of its modulator. Then the simultaneous phases of the remaining six reference frequencies could be quickly calculated from sums and differences of the two sampled phases.

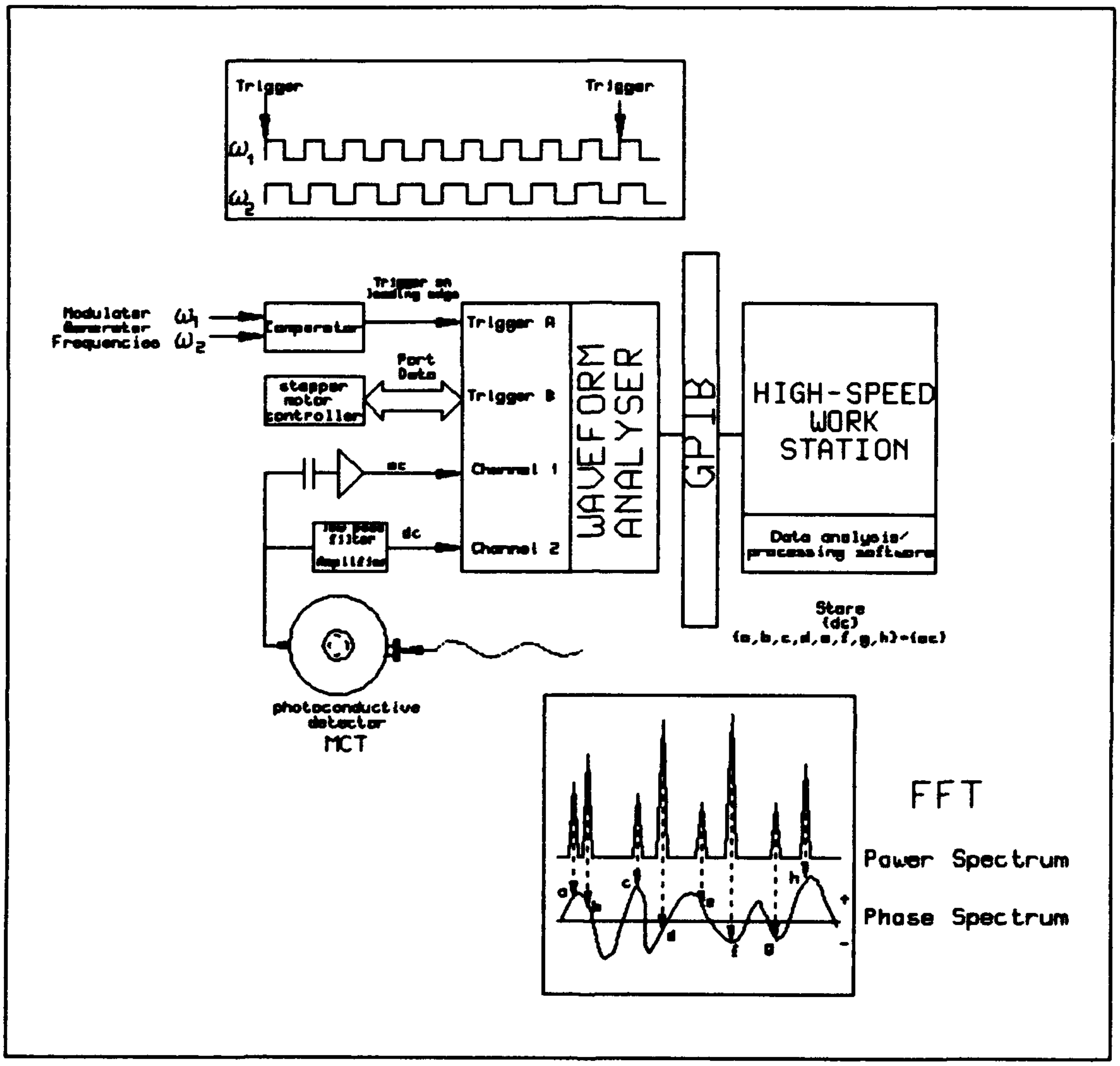


Figure 19. Mueller matrix digital acquisition using a standard Analogic 6100/650 or LeCroy 9424 waveform analyser. This instrument computes Fourier intensity and phase spectra of the MCT-detected scattergram. Acquisition of Mueller elements are triggered when the modulator reference frequencies are aligned as shown at the top of the figure. They are correlated to peak Fourier intensities a, b, c, d, e, f, g, h and a dc value that make up the scattergram waveform. The nine Mueller elements are transferred to computer memory via an ANSI command from the CPU routed through the stepper motor controller, used to control the experimental operation.

We envision something like the following sequence of operations. Three voltages are read in and scaled via a A/D board, representing, at the same instant, the phase of transmitter modulator (ω_1 radians), the phase of receiver modulator (ω_2 radians), and the detector output. From the two phases we form the instantaneous phases of the other six references: $2\omega_1$, $2\omega_2$, $\omega_1-\omega_2$, $2\omega_1-\omega_2$, $\omega_1-2\omega_2$, and $2\omega_1-2\omega_2$. (In some cases sums instead of differences may be chosen.) At this point compensating phase shifts, previously determined in a calibration procedure, may be applied to the eight phases. The cosine of each net phase is then calculated, giving eight numbers that represent the instantaneous values of the eight reference frequency voltages that (in the present analog system) would be found on the reference inputs of the lock-in amplifiers. Note that the amplitude of a reference frequency is not a relevant quantity, so long as it's constant, and is here taken to be unity.

When each of the cosine terms is multiplied by the detector output, there results eight numbers that represent the instantaneous outputs of the present eight lock-in amplifiers (without low pass output filtering). Recall that each Mueller matrix element is proportional to the average level (dc component) of its lock-in output. Electronically that level is determined with a low pass filter smoothing the output over some period of time ($\approx$time constant). We can accomplish the same thing numerically by repeating the measurements just described many times over the same time period and taking averages. Thus, in eight computer memory locations we would accumulate (add) the eight effective lock-in amplifier outputs calculated each time a triplet of data points (ω_1, ω_2, detector) were read in. After enough readings (thousands?) are gathered over a sufficient length of time (.5 sec?), each of the eight accumulated numbers would be divided by the total number of readings and scaled by a fixed factor - previously determined by calibration - to give the value of the corresponding Mueller matrix element.

In the third approach the frequency content of the detected signal isn't considered at all. Instead, our starting point is Equation 12, 14b,16b, or 18b; an exact closed equation relating the detected intensity to the retardation on the two polarization modulators and the Mueller matrix elements. Suppose, as in the last approach (lock-in emulation), we read in a triplet of values representing ω_1, ω_2 and the detector signal. Then, taking $\psi_{11} = 1$, we can evaluate all the quantities of the above equations except the eight Mueller matrix elements, giving one equation with eight unknowns. Reading in seven more triplets of values will yield a total of eight equations in eight unknowns, which can then be solved for the Mueller matrix elements by standard techniques, such as an inverse matrix calculation. This process might be repeated often in a very short time and averages taken to reduce the influence of experimental noise and the occasional (?) ill-conditioned data set.

We wish to investigate soon whether one or more of these three data processing techniques - or perhaps other techniques not yet thought of - can replace the rack of analog electronics now used. All three approaches should be easy to implement on a microvax or PC system, and should be tested using synthetic and/or real data. There are many questions and problems to be considered, such as the density and total number of data points required in each approach, how to reconcile the need for simultaneous data triplets with the sequential nature of multiplexer data acquisition, and the stability of the solutions obtained in the face of experimental noise.

6.2 Rapid Laser Switching Between Resonance-Reference and Resonance-Resonance Beam Wavelengths.

A latter objective of this program is field evaluation in near real time of Mueller matrix elements measured in succession between rapidly switching beams irradiating surfaces down-range to kilometer distances. To accomplish these more distant and more rapid measurements, the ellipsometer system will be expanded to incorporate more powerful lasers and a larger receiver collection aperture.

The transmitter of this future system was eluded to in the previous Figure 4c. Let us return to that same type of configuration, but for the sake of simplicity consider here a three-wavelength switching transmitter system.

The variable beam splitters (VBS1-2) produce full transmission and reflection modulation from $< .05$ to $> .95\ R$ between 9 and 12.5 μm, via a piezo electric interface control module. (For n-laser wavelength pulsing, n-1 VBS optics are required.) The VBS modulators work on a principle of Frustrated Total Internal Reflection. (The technique is proprietary to the optics manufacturer.) Two VBS optical systems are now being custom designed for these ellipsometers by the Kentek Corporation, Laser Tools Division.

Amplitude modulation and triggering of the three incident cw beams is accomplished internally in the laser's power supplies and exciter circuits. Amplitude modulators MOD1-3 designate switching access via TTL logic signals to the power supplies, as shown in Figure 20.

Our concern with this transmitter is the purity of polarization modulation between pulses. The pulsing is adjustable from 10 to 100 milliseconds or greater. Another concern of the frequency agile ellipsometer systems is what tolerance the modulators can withstand on consistency of periodic phase retardation adjustments between pulses of unlike wavelength. Air-cooled ZnSe can apparently operate under a maximum 100 watt beam intensity without significant damage to its anti-reflection coating. However, maintaining a constant retardation (δ_0 in Equation 9) in the PEM's between beam pulses is a stringent constraint placed on the resonant compression and relaxation induced on the ZnSe crystals.

We also have future plans to utilize the dead time between beam pulses in an integrated **pseudo active** emissions fusion sensor concept, where chemical vapor contamination and liquids on a surface are detected spectroscopically in thermoluminescence produced from heating by the beam, and subsequent release of mid IR Planck emissions.[36] The thermoluminescence sensor component of the system would consist of a solid-state interferometer based on the same PEM technology incorporated in the ellipsometer systems. This will be discussed in a later report.

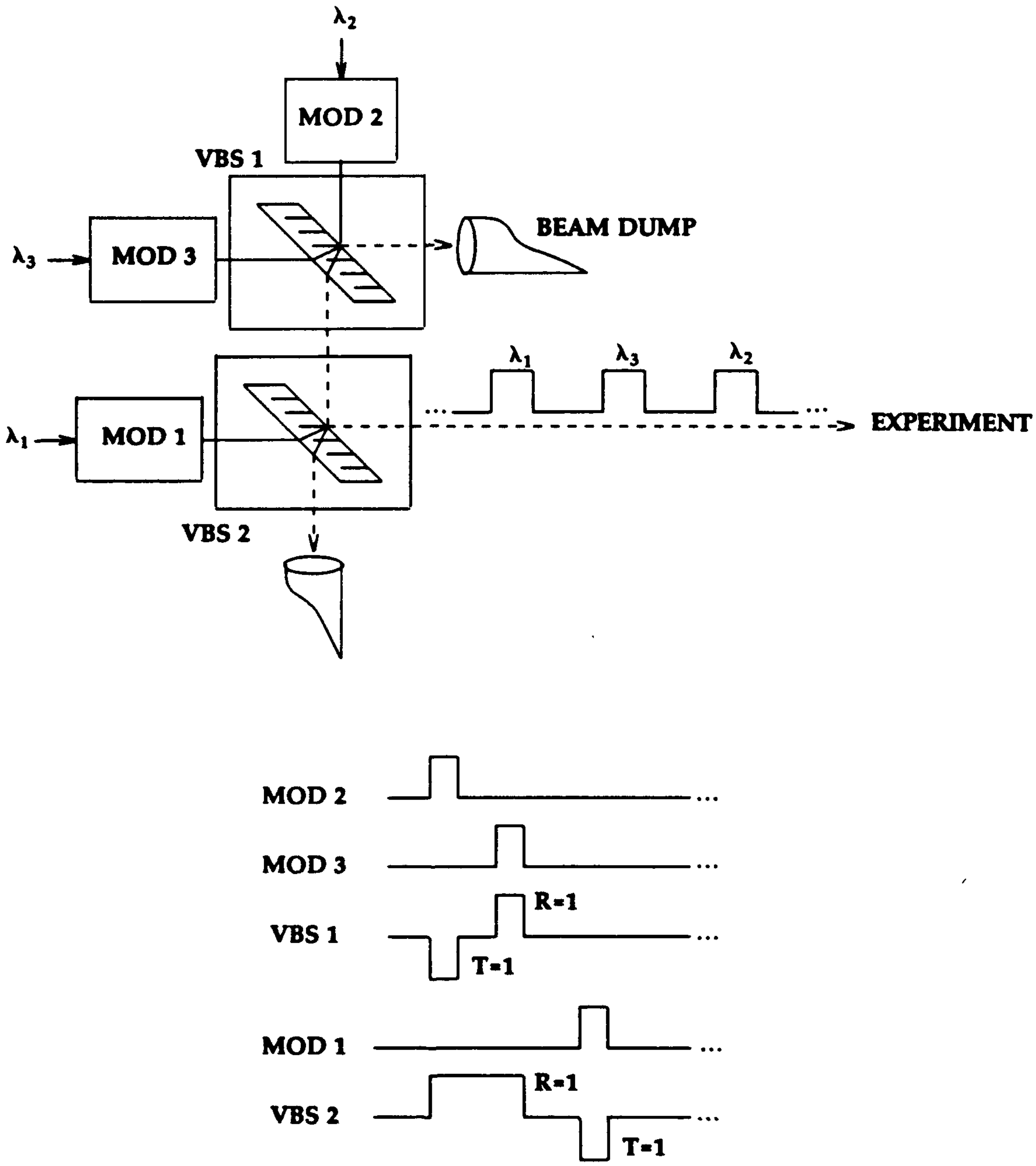


Figure 20. Rapid 3-laser switching based on variable beam splitter (VBS) technology. MOD 1,2 and 3 are amplitude modulators of the four incident beams, and VBS 1,2 are the electronically controlled transmitting/reflecting (T/R) beam splitters. Pulse and triggering sequences generating the train of alternating wavelengths λ_2:λ_3:λ_1 are shown in the bottom half of the figure. Pulse gating and beam durations can be varied by interface to the piezo electric circuitry.

6.3 Neural Network Computing of the Mueller Elements For Standoff Analyte Detections.

Work has begun on applying a neural network to the analog APSD detector outputs for purposes of contaminant decision making and density mappings. As the name implies, neural network systems intend to emulate the brain's parallel processing ability by activating a set of impulses (in this case, real-time information from 16 independent channels of Mueller elements analog outputs from the sensor), pass it along weighted interconnecting nodes (the neurons, weights via a valid theoretical model) that transform these data to a system of hidden layers and other nodes, where new transforms operate on these impulses to produce an output layer. The pattern of information from the network's final output layer (back- or forward-propagated) has interpretation that may correlate to a detection event or non-event.

All networks we are considering are constructed from interconnected nodes, each of which forms a weighted sum of the Mueller matrix element inputs to the node, and adds a threshold value to the weighted sum. The value of this sum plus the threshold is passed through a nonlinearity, and the value of the non linear 'impulse' function is the output of the node. The inputs to each node are a combination of outputs from other nodes and primary Mueller element inputs to the network. The threshold of each node can be viewed as a unit weight for an input.[37] Generally, the connection weights and thresholds can be adapted using iterative procedures to make the network produce a desired output when a particular input is presented. Many of these network concepts have been demonstrated to work well when the input data is noisy.

The detection network application is schematized in the following Figure 21. We refer to Lippmann's paper[38] and references therein for a description of various neural network architectures.

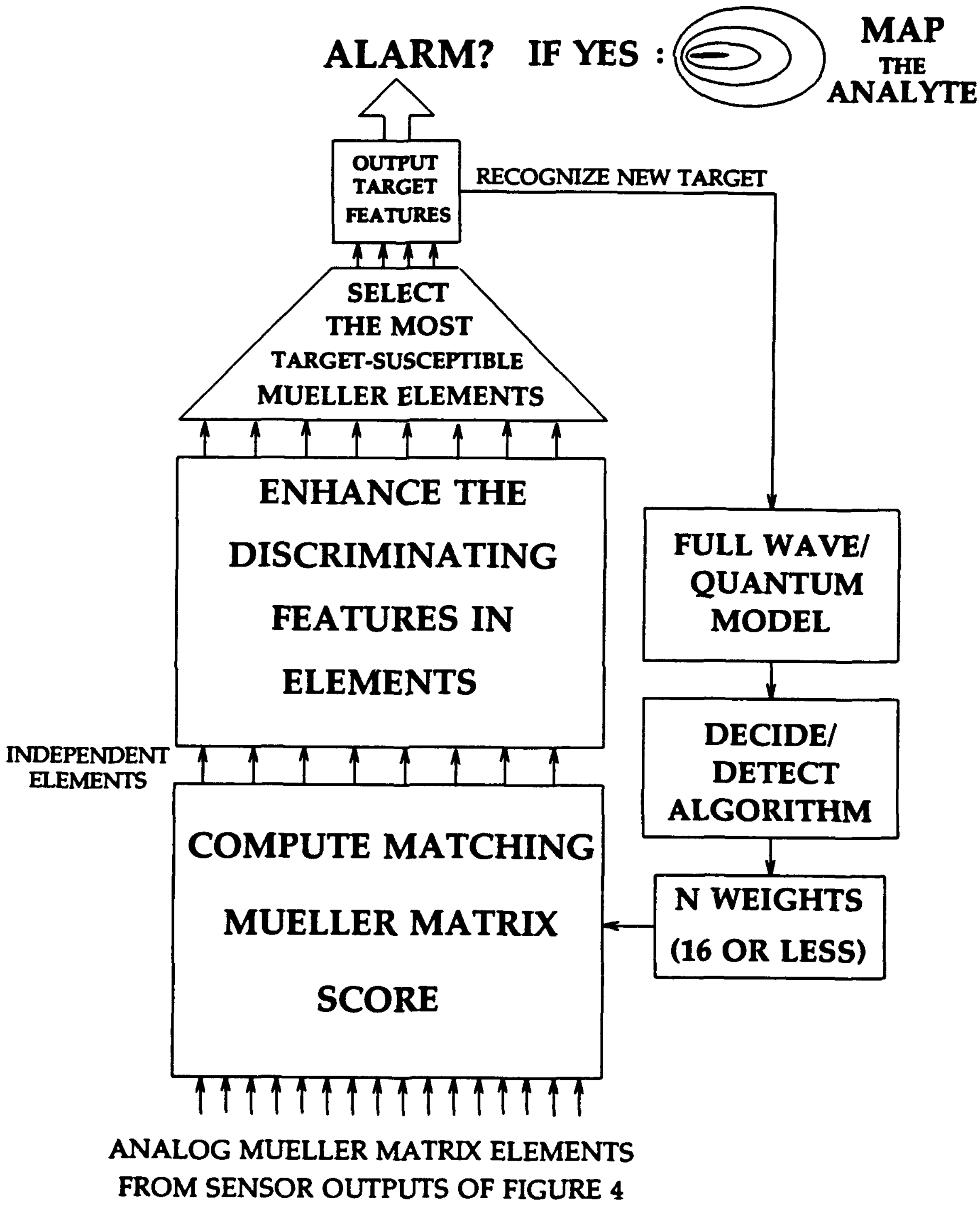


Figure 21. A neural network application for accessing elements of the Mueller matrix in real-time operation of the ellipsometer sensor. Inputs from theoretical and statistical models weigh all discriminating analyte features from the full 16-element matrix field. The successful network architecture will adapt to new analytes (contaminants) from the scattering sample, weighing elements according to their intramolecular phase signatures. The goal of the network is to alarm against the analyte (if present), and map its mass density.

Determining and optimizing a particular architecture that can be best implemented in construction of a Mueller matrix hardware network to serve as a contaminant classifier is a topic of future investigations. Some candidate nets are Hopfield-, Hamming-, Grossberg-, and Kohonen-like architectures. For the APSD sensor-specific network, the best choice of the number of nodes and hidden layers is an important first step in network development.

Neural networks can be used for content-addressable memory, vector quantization, data clustering and pattern recognition. The remote detection application pursued here requires a net that performs the last two of these functions. Two networks that form clusters are the Carpenter/Grossberg classifier and Kohonen self-organizing feature mapper.

The first network architecture we have investigated is a Rumelhart-McClelland single layer perceptron structure using three nodes. The network structure is given as follows.

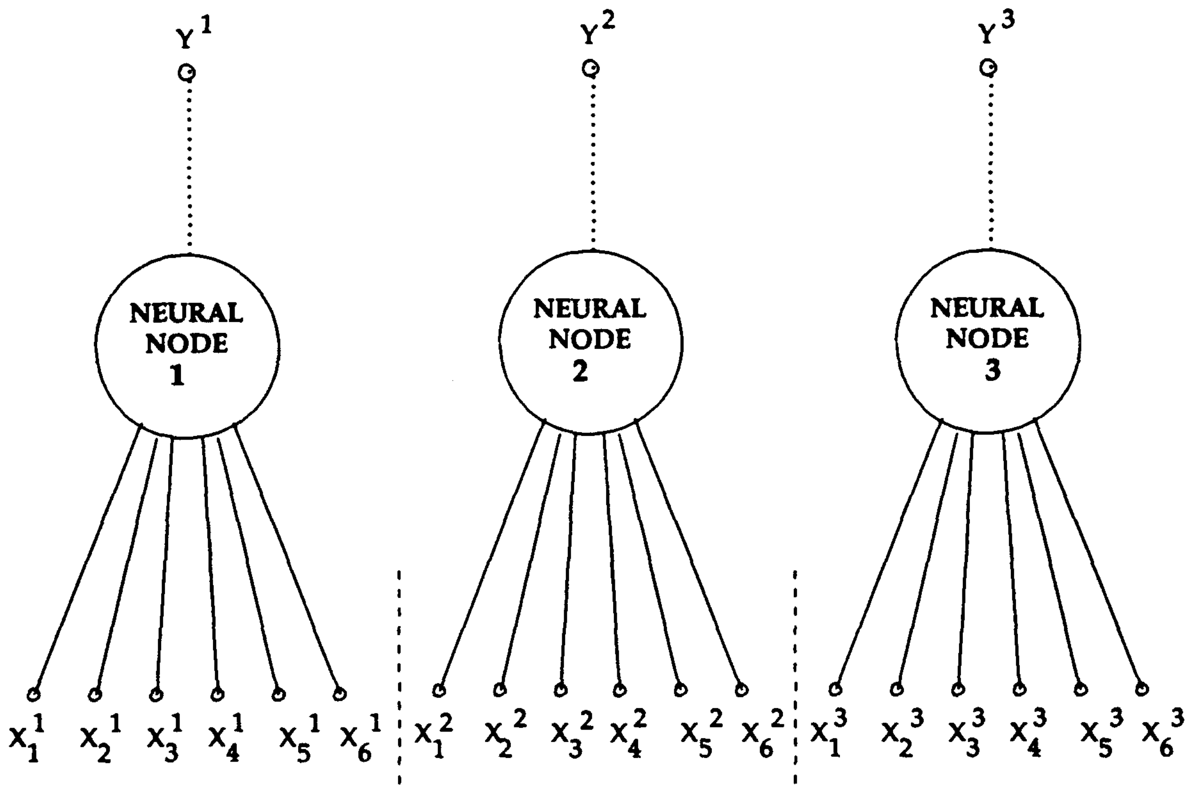


Figure 22. The architecture of a neural network that implements a detection algorithm for the near real-time identification of SF96, DMMP, and/or DIMP.

Each node of the neural network determines on which side of a hyperplane an input Mueller matrix lies. The Mueller matrix data from the subroutine RETRO indicates (with added noise) scattering from various coating materials and unwetted surfaces, separated by a hyperplane. The partitioning of the Mueller elements into classes is done at backscattering angles and wavelength combinations most susceptible to the analyte. Those parameters are obtained from the programs DETECT and DECIDE. Judicious initial choices for the connection weights and thresholds into the network will yield the correct output at the onset of presentation of training data. Several sets (10-30 per backscatter angle-wavelength pair) of noisy data must be taken in order to obtain the standard deviation values for the Mueller elements. These are required for selecting a good initial choice of connection weights and node thresholds.

This neural network model is coded in Fortran 77 and now running on a CRAY2 computer facility. When implemented, it is capable of detecting SF96, DMMP, and DIMP. (Since the three analytes can be identified using three different combinations of backscatter angle and wavelength, the network requires three independent nodes.) The parameters in Figure 22 are defined as follows:

$$Y^i = f_h(\sum_j w_j^i X_j^i + \theta^i) \tag{47}$$

For the output node 1, Y^1 is high (=1) if SF96 is present, and low (=0) otherwise. The output of node 2(3), $Y^2(Y^3)$ is high if DMMP(DMMP) is present, and low otherwise. The values w_j^i are connection weights from inputs X_j^i to node i, θ^i is the threshold at node i, and f_h is a hard limiting nonlinearity (f_h=0 if the argument is negative, and 1 if positive.) The absolute value of the j^{th} Mueller element at the i^{th} angle-wavelength combination is the network input:

$$X_j^i = |\, f_{mn}(\lambda_i, \theta_i)\, | \tag{48}$$

where $j=1 \rightarrow mn=11$, $j=2 \rightarrow mn=12$, $j=3 \rightarrow mn=22$, $j=4 \rightarrow mn=33$, $j=5 \rightarrow mn=34$, and $j=6 \rightarrow mn=44$. To be consistent with previous notations, i=1,2,3 designates detection of SF96, DMMP, and DIMP respectively.

The connection weights w_j^i and thresholds θ^i can be adapted using a perceptron convergence procedure[38]. The iterations required for w and θ to converge can be reduced by using the following initial values:

$$\theta^i = -\tfrac{1}{2}\sum_j \frac{X_{b_j}^i + X_{t_j}^i}{\sigma_{b_j}^i} \tag{49a}$$

$$w_j^i = \frac{1}{\sigma_{b_j}^i} u(X_{t_j}^i - X_{b_j}^i), \tag{49b}$$

where $u(\cdot)$ is a unit step function, $\sigma_{b_j}^i$ is the standard deviation of Mueller element j for the unwetted (bare) surface, and inputs $X_{b(t)j}^i$ are the absolute values of the Mueller elements j for unwetted (wetted) surfaces at the susceptible λ–θ values. Even though these initial w and θ choices near eliminate a need for training sessions, several sets of data are required to obtain accurate values for $\sigma_{b_j}^i$.

6.4 Initial Experimental and Theoretical Mueller Matrix Data Bases.

A thorough data base generated through the laboratory ellipsometer instruments and a successful model, to interpret these data, will determine the sensitivities and limitations of this technology toward solution of a particular detection problem. It will guide us in the selection of Mueller elements (as functions of backscattering angle, wavelength) that can identify an analyte or class of compounds alone, spread across a background surface, or disseminated as an aerosol. What can be done to filter in and enhance (optically or by mathematical algorithms) information by the analyte? That will be answered after a full and reproducible data base is produced, so that we can better understand the principles of polarized IR beam - surface interactions. Once this technology is understood, and if it can be proven feasible, prototype sensors will be designed and applied toward a specific problem, like detecting an agent wetting soil, a biological impurity in a specimen, an aerosol pollutant, an oxide growth on a semiconductor surface, and so on. We present here the plan of an initial data base where the goals are verification of model calculations and rapid detection of classes of analyte compounds in situ.

6.4.1 Metals and Insulators of Known IR Refractive Index and Surface Geometry: Validation of the Full Wave Model.

In Sections 5.2.1-5.2.4 and 6.3 we had discussed the Full Wave electromagnetic scattering theory, and how it can be applied toward: (1) initializing the ellipsometer sensor for maximum probability of a successful analyte detection, (2) simulation of the entire experimental operation, and (3) fabricating a neural network discriminator. The theory, of course, would have to be experimentally verified before these applications can be implemented. We have set out to prove (or disprove) its predictions via a control set of scattering experiments from aluminum, graphite, and other surfaces of known optical constants over the IR, and known geometry (surface slopes and heights). (The surfaces are etched or sand blasted, and surface-profiled in 3-dimensions by an interference-type instrument.) The general experimental procedure will involve scanning the entire range of backscattering angles over many of the laser transitions, comparing results with model data such as those presented in Figures 16-18. The University of Nebraska group (E. Bahar) is now expanding the Full Wave to include scattering from multi-layered structures and non-isotropic scatterers. When the new versions of RETRO code are written, predicted scattering signatures will be compared with data measured by this ellipsometer sensor.

6.4.2 Biological and Controlled Substance Simulants.

The Mueller matrix ellipsometer produces a full optical description of the scatterer by its response to a continuous span of linear and elliptical polarization states over selected frequencies of the irradiating beams. The emphasis here is complete characterization of linear scattering processes. Spectroradiometers that measure absorption bands in collected radiance[36] cannot resolve molecularly similar compounds with overlapping extinction bands – compounds that may have entirely different toxicity!! With phase-sensitive scattering, we would like to test whether isomers (molecules of identical molecular weight but different group symmetry) can be distinguished, for instance, through their dichroism signatures. Many biological compounds contain segments that are chiral, thus the ability to distinguish between chirality has applications of biological and contraband detection. The first sets of experiments to test this uniqueness assertion will be conducted on the biological and controlled substance simulants listed in Table 5. In it are compiled the scatterer's molecular formula, its strongest resonant absorption over the IR, and the nearest ellipsometer laser energy to that absorption center frequency. Typically, three of the four lasers are tuned to a distinct absorption (analytical) band and the fourth is off-tuned to a region of non-resonance in the analyte (reference). The ratios of analytical to reference Mueller elements found most susceptible to the contaminant's optical activity are then sought for making a detection decision.

6.4.3 Chemical Agent Simulants.

The experiments with liquid simulants of chemical agent are trained toward their strong absorption bands, in some agents the P=O, C-O and P-O-C stretching vibrations are most important. In these experiments, Mueller elements of the bare surface are measured. Next, the analyte is ejected (via an aerosol deposition, to simulate and actual agent attack) in low concentration, and Mueller elements remeasured. This continues on to higher concentrations and element re-measurements. A pattern is established in the Mueller elements with concentration of the analyte. Table 5 lists the absorption properties of four common agent simulants: DMMP, DIMP, SF96, and DEP.

6.4.4 Interferents.

Interferents are all scatterers other than the analyte. The analyte scattering signal is usually a small superposition on the interferent scattering signal, and must be observed in the differential resonance/non resonance Mueller elements for successful detections. Fortunately, terrains (a sum of quartz, kaolinite, illite, montmorillonite and other minerals) are broaderband absorbers of IR radiation and the analyte compounds have sharp extinction frequencies. Other interferents such as diesel soots, fog oils,[21] possess their own absorption moieties. Thus, the susceptible Mueller elements can single out the analyte on a surface at the (very narrow) laser line by adjusting the ellipsometer beams to the analyte's center extinction frequencies, ratio these elements to those measured at a reference laser transition, then subtract this result to the bare surface Mueller elements. In Table 5, the common minerals found in soil are listed, all are broadband IR absorbers.

Table 5. Candidates for characterization through select IR Mueller matrix signatures. A data bank of Mueller elements is established per absorption wavelength (plus a minimum of one reference laser wavelength) over all backscattering angles, and organized in a computer file similar to this table's format. The major absorption bands by each material are listed, as are their nearest matching laser line over the ellipsometer's 4-laser bandwidth (see Appendix I).

Scatterer	Formula	Major Vibration (cm^{-1})	Nearest Laser Line to the Scatterer's Vibrational Frequency: Transition	Freq. (cm^{-1})	Band	Type Laser	Comment
							Chiral Organics (Biological Simulant Analytes)
D-(-)-Arabinose	$C_5H_{10}O_5$	842.7	P(28)	842.79	$00^01 \rightarrow 10^00$	$C^{14}O_2^{16}$	
		892.5	P(26)	891.57	$00^01 \rightarrow 10^00$	$C^{13}O_2^{16}$	
			R(36)	892.04	$00^01 \rightarrow 10^00$	$C^{14}O_2^{16}$	
		1000.6	P(64)	1000.82	$00^01 \rightarrow 02^00$	$C^{12}O_2^{16}$	
			P(20)	1000.65	$00^01 \rightarrow 02^00$	$C^{13}O_2^{16}$	
			R(26)	1000.95	$00^01 \rightarrow 02^00$	$C^{14}O_2^{16}$	
		1052.4	P(14)	1052.20	$00^01 \rightarrow 02^00$	$C^{12}O_2^{16}$	
			P(40)	1052.26	$00^01 \rightarrow 02^00$	$C^{12}O_2^{18}$	
DL-Alanine Monohydrate	$C_3H_7NO_2$	852.0	P(16)	853.2	$00^01 \rightarrow 10^00$	$C^{14}O_2^{16}$	
D-Alanine	$C_3H_7NO_2$	850.6	P(18)	851.50	$00^01 \rightarrow 10^00$	$C^{14}O_2^{16}$	
DL-Aspartic Acid	$C_4H_7NO_4$	1073.1	R(12)	1073.28	$00^01 \rightarrow 02^00$	$C^{12}O_2^{16}$	
			P(14)	1073.58	$00^01 \rightarrow 02^00$	$C^{12}O_2^{18}$	
L-Aspartic Acid	$C_4H_7NO_4$	1045.9	P(22)	1045.02	$00^01 \rightarrow 02^00$	$C^{12}O_2^{16}$	
			P(48)	1045.08	$00^01 \rightarrow 02^00$	$C^{12}O_2^{18}$	
(-)-Atropine Sulfate Monohydrate	$C_{17}H_{23}NO_3$	967.3	R(8)	967.71	$00^01 \rightarrow 10^00$	$C^{12}O_2^{16}$	
			P(18)	967.45	$00^01 \rightarrow 02^00$	$C^{14}O_2^{16}$	
		1023.8	P(44)	1023.19	$00^01 \rightarrow 02^00$	$C^{12}O_2^{16}$	
			R(6)	1022.93	$00^01 \rightarrow 02^00$	$C^{13}O_2^{16}$	
		1073.9	R(12)	1073.28	$00^01 \rightarrow 02^00$	$C^{12}O_2^{16}$	
			P(14)	1073.57	$00^01 \rightarrow 02^00$	$C^{12}O_2^{18}$	

Table 5 - continued.

Scatterer	Formula	Major Vibration(s) (cm^{-1})	Nearest Laser Line to the Scatterer's Vibrational Frequency: Transition	Freq. (cm^{-1})	Band	Type Laser	Comment
(1R)-(+)-Camphor	$C_{10}H_{16}O$	853.5	P(16)	853.2	$00^01 \rightarrow 10^00$	$C^{14}O_2^{16}$	
		935.0	P(30)	934.89	$00^01 \rightarrow 10^00$	$C^{12}O_2^{16}$	+ VCD (strong)
			P(38)	935.89	$00^01 \rightarrow 10^00$	$C^{12}O_2^{18}$	
			R(30)	935.14	$00^01 \rightarrow 10^00$	$C^{13}O_2^{16}$	
		1045.3	P(22)	1045.02	$00^01 \rightarrow 02^00$	$C^{12}O_2^{16}$	- VCD(strong)
			P(48)	1045.08	$00^01 \rightarrow 02^00$	$C^{12}O_2^{18}$	
(1S)-(-)-Camphor	$C_{10}H_{16}O$	934.9	P(30)	934.89	$00^01 \rightarrow 10^00$	$C^{12}O_2^{16}$	
			P(38)	935.89	$00^01 \rightarrow 10^00$	$C^{12}O_2^{18}$	
			R(30)	935.14	$00^01 \rightarrow 10^00$	$C^{13}O_2^{16}$	
		1045.3	P(22)	1045.02	$00^01 \rightarrow 02^00$	$C^{12}O_2^{16}$	
			P(48)	1045.08	$00^01 \rightarrow 02^00$	$C^{12}O_2^{18}$	
(±)-Camphor	$C_{10}H_{16}O$	1045.1	P(22)	1045.02	$00^01 \rightarrow 02^00$	$C^{12}O_2^{16}$	
			P(48)	1045.08	$00^01 \rightarrow 02^00$	$C^{12}O_2^{18}$	
L-Cysteine	$C_3H_7NO_2S$	867.2	R(4)	869.96	$00^01 \rightarrow 10^00$	$C^{14}O_2^{16}$	
		1064.8	R(0)	1064.51	$00^01 \rightarrow 02^00$	$C^{12}O_2^{16}$	
			P(26)	1064.13	$00^01 \rightarrow 02^00$	$C^{12}O_2^{18}$	
D-(-)-Fructose	$C_6H_{12}O_6$	978.2	R(24)	978.47	$00^01 \rightarrow 10^00$	$C^{12}O_2^{16}$	
			R(18)	978.89	$00^01 \rightarrow 10^00$	$C^{12}O_2^{18}$	
			P(8)	976.21	$00^01 \rightarrow 02^00$	$C^{14}O_2^{16}$	
		1079.6	R(22)	1079.85	$00^01 \rightarrow 02^00$	$C^{12}O_2^{16}$	
			P(6)	1079.49	$00^01 \rightarrow 02^00$	$C^{12}O_2^{18}$	
D-Glucose Anhydrous	$C_6H_{12}O_6$	838.7	P(32)	839.20	$00^01 \rightarrow 10^00$	$C^{14}O_2^{16}$	
		995.7	R(56)	995.07	$00^01 \rightarrow 10^00$	$C^{12}O_2^{16}$	
			P(26)	994.99	$00^01 \rightarrow 02^00$	$C^{13}O_2^{16}$	
			R(16)	994.82	$00^01 \rightarrow 02^00$	$C^{14}O_2^{16}$	
		1024.1	P(44)	1023.19	$00^01 \rightarrow 02^00$	$C^{12}O_2^{16}$	
			R(8)	1024.37	$00^01 \rightarrow 02^00$	$C^{13}O_2^{16}$	
L-Histidine	$C_6H_7N_3O_2$	924.5	P(40)	924.97	$00^01 \rightarrow 10^00$	$C^{12}O_2^{16}$	
			R(14)	924.53	$00^01 \rightarrow 10^00$	$C^{13}O_2^{16}$	
Glycine	$C_2H_5NO_2$	893	P(26)	891.57	$00^01 \rightarrow 10^00$	$C^{13}O_2^{16}$	
		912	P(54)	907.77	$00^01 \rightarrow 10^00$	$C^{12}O_2^{16}$	
		936	P(28)	936.80	$00^01 \rightarrow 10^00$	$C^{12}O_2^{16}$	
			P(38)	935.89	$00^01 \rightarrow 10^00$	$C^{12}O_2^{18}$	
			R(32)	936.37	$00^01 \rightarrow 10^00$	$C^{13}O_2^{16}$	
		1033	P(34)	1033.48	$00^01 \rightarrow 02^00$	$C^{12}O_2^{16}$	
			P(58)	1035.70	$00^01 \rightarrow 02^00$	$C^{12}O_2^{18}$	

Table 5 - continued.

Scatterer	Formula	Major Vibration(s) (cm^{-1})	Nearest Laser Line to the Scatterer's Vibrational Frequency: Transition	Freq. (cm^{-1})	Band	Type Laser	Comment
S-(-)-Limonene	$C_{10}H_{16}$	887.1	P(30)	887.92	00^01-10^00	$C^{13}O_2^{16}$	
			R(28)	886.93	00^01-10^00	$C^{14}O_2^{16}$	+ VCD (strong)
		914.2	P(50)	914.42	00^01-10^00	$C^{12}O_2^{16}$	+ VCD (weak)
			R(4)	917.25	00^01-10^00	$C^{13}O_2^{16}$	
		1051.4	P(16)	1050.44	00^01-02^00	$C^{12}O_2^{16}$	- VCD (very weak)
			P(42)	1050.49	00^01-02^00	$C^{12}O_2^{18}$	
D-Mannose	$C_6H_{12}O_6$	969.9	R(10)	969.14	00^01-10^00	$C^{12}O_2^{16}$	
			R(4)	970.33	00^01-10^00	$C^{12}O_2^{18}$	
			P(16)	969.26	00^01-02^00	$C^{14}O_2^{16}$	
		1040.0	P(28)	1039.37	00^01-02^00	$C^{12}O_2^{16}$	
			P(54)	1039.50	00^01-02^00	$C^{12}O_2^{18}$	
			R(32)	1039.38	00^01-02^00	$C^{13}O_2^{16}$	
D-(-)-Ribose	$C_5H_{10}O_5$	911.7	P(54)	907.77	00^01-10^00	$C^{12}O_2^{16}$	
		959.0	P(2)	959.39	00^01-10^00	$C^{12}O_2^{16}$	
			P(10)	959.71	00^01-10^00	$C^{12}O_2^{18}$	
			P(26)	959.90	00^01-02^00	$C^{14}O_2^{16}$	
		1037.0	P(30)	1037.43	00^01-02^00	$C^{12}O_2^{16}$	
			P(56)	1037.61	00^01-02^00	$C^{12}O_2^{18}$	
			R(28)	1037.17	00^01-02^00	$C^{13}O_2^{16}$	
L-(-)-Sorbose	$C_6H_{12}O_6$	820.2	P(48)	824.17	00^01-10^00	$C^{14}O_2^{16}$	
		883.1	P(34)	884.18	00^01-10^00	$C^{13}O_2^{16}$	
			R(22)	882.91	00^01-10^00	$C^{14}O_2^{16}$	
		991.8	R(48)	991.57	00^01-10^00	$C^{12}O_2^{16}$	
			R(44)	991.27	00^01-10^00	$C^{13}O_2^{16}$	
			P(30)	991.07	00^01-02^00	$C^{13}O_2^{16}$	
			R(10)	990.79	00^01-02^00	$C^{14}O_2^{16}$	
		1047.9	P(18)	1048.66	00^01-02^00	$C^{12}O_2^{16}$	
			P(44)	1048.71	00^01-02^00	$C^{12}O_2^{18}$	
L-Serine	$C_3H_7NO_3$	1013.1	P(52)	1014.52	00^01-02^00	$C^{12}O_2^{16}$	
L-Threonine	$C_4H_9NO_3$	936.2	P(28)	936.80	00^01-10^00	$C^{12}O_2^{16}$	
			P(38)	935.89	00^01-10^00	$C^{12}O_2^{18}$	
			R(32)	936.37	00^01-10^00	$C^{13}O_2^{16}$	
		1041.1	P(26)	1041.28	00^01-02^00	$C^{12}O_2^{16}$	
			P(52)	1041.38	00^01-02^00	$C^{12}O_2^{18}$	
			R(36)	1041.48	00^01-02^00	$C^{13}O_2^{16}$	

Table 5 - continued.

Scatterer	Formula	Major Vibration(s) (cm^{-1})	Nearest Laser Line to the Scatterer's Vibrational Frequency: Transition	Freq. (cm^{-1})	Band	Type Laser	Comment
L-Tyrosine	$C_9H_{11}NO_3$	841.3	P(30)	841.00	$00^01 \rightarrow 10^00$	$C^{14}O_2^{16}$	
DL-Tartaric Acid	$C_4H_6O_6$	1094.9	R(16)	1094.76	$00^01 \rightarrow 02^00$	$C^{12}O_2^{18}$	
L-Tartaric Acid Hydrate	$C_4H_6O_6$	943.0	P(22)	942.38	$00^01 \rightarrow 10^00$	$C^{12}O_2^{16}$	
			P(30)	943.23	$00^01 \rightarrow 10^00$	$C^{12}O_2^{18}$	
			R(44)	943.34	$00^01 \rightarrow 10^00$	$C^{13}O_2^{16}$	
		1087.8	R(36)	1087.95	$00^01 \rightarrow 02^00$	$C^{12}O_2^{16}$	
			R(4)	1087.11	$00^01 \rightarrow 02^00$	$C^{12}O_2^{18}$	
D-Tartaric Acid	$C_4H_6O_6$	943.3	P(22)	942.38	$00^01 \rightarrow 10^00$	$C^{12}O_2^{16}$	
			P(30)	943.23	$00^01 \rightarrow 10^00$	$C^{12}O_2^{18}$	
			R(44)	943.34	$00^01 \rightarrow 10^00$	$C^{13}O_2^{16}$	
		1087.8	R(36)	1087.95	$00^01 \rightarrow 02^00$	$C^{12}O_2^{16}$	
			R(4)	1087.11	$00^01 \rightarrow 02^00$	$C^{12}O_2^{18}$	
D-(+)-Xylose	$C_5H_{10}O_5$	903.7	P(60)	903.21	$00^01 \rightarrow 10^00$	$C^{12}O_2^{16}$	
		934.2	P(30)	934.89	$00^01 \rightarrow 10^00$	$C^{12}O_2^{16}$	
			P(38)	935.89	$00^01 \rightarrow 10^00$	$C^{12}O_2^{18}$	
			R(30)	935.14	$00^01 \rightarrow 10^00$	$C^{13}O_2^{16}$	
		1039.8	P(28)	1039.37	$00^01 \rightarrow 02^00$	$C^{12}O_2^{16}$	
			P(54)	1039.50	$00^01 \rightarrow 02^00$	$C^{12}O_2^{18}$	
			R(32)	1039.38	$00^01 \rightarrow 02^00$	$C^{13}O_2^{16}$	
D-(-)-Xylose	$C_5H_{10}O_5$	903.6	P(60)	903.21	$00^01 \rightarrow 10^00$	$C^{12}O_2^{16}$	
		934.1	P(30)	934.89	$00^01 \rightarrow 10^00$	$C^{12}O_2^{16}$	
			P(38)	935.89	$00^01 \rightarrow 10^00$	$C^{12}O_2^{18}$	
			R(30)	935.14	$00^01 \rightarrow 10^00$	$C^{13}O_2^{16}$	
		1039.8	P(28)	1039.37	$00^01 \rightarrow 02^00$	$C^{12}O_2^{16}$	
			P(54)	1039.50	$00^01 \rightarrow 02^00$	$C^{12}O_2^{18}$	
			R(32)	1039.38	$00^01 \rightarrow 02^00$	$C^{13}O_2^{16}$	
L-Tryptophan	$C_{11}H_{12}N_2O_2$	1005	P(60)	1005.48	$00^01 \rightarrow 02^00$	$C^{12}O_2^{16}$	
			P(16)	1004.28	$00^01 \rightarrow 02^00$	$C^{13}O_2^{16}$	
			R(34)	1005.31	$00^01 \rightarrow 02^00$	$C^{14}O_2^{16}$	
		918.0	P(46)	918.72	$00^01 \rightarrow 10^00$	$C^{12}O_2^{16}$	
			R(6)	918.74	$00^01 \rightarrow 10^00$	$C^{13}O_2^{16}$	

Table 5 - continued.

Scatterer	Formula	Major Vibration(s) (cm^{-1})	Nearest Laser Line to the Scatterer's Vibrational Frequency: Transition	Freq. (cm^{-1})	Band	Type Laser	Comment
							Chemical Agent Simulant Analytes
Dimethyl-methyl-phosphonate (DMMP)	$CH_3PO(OCH_3)_2$	1049	P(18)	1048.66	00^01-02^00	$C^{12}O_2^{16}$	C-O stretch
			P(44)	1048.71	00^01-02^00	$C^{12}O_2^{18}$	
		1061	P(4)	1060.57	00^01-02^00	$C^{12}O_2^{16}$	C-O stretch
			P(30)	1060.84	00^01-02^00	$C^{12}O_2^{18}$	
		1072	R(10)	1071.88	00^01-02^00	$C^{12}O_2^{16}$	C-O stretch
			P(14)	1073.58	00^01-02^00	$C^{12}O_2^{18}$	
Diisopropyl-methyl-phosphonate (*DIMP*)	$CH_3PO(OCH(CH_3)_2)_2$	995	R(56)	995.07	00^01-10^00	$C^{12}O_2^{16}$	*P-O-C* vib
			P(26)	994.99	00^01-02^00	$C^{13}O_2^{16}$	
			R(16)	994.82	00^01-02^00	$C^{14}O_2^{16}$	
		1014	P(52)	1014.52	00^01-02^00	$C^{12}O_2^{16}$	*(P-O-C)* vib
Polydimethyl-siloxane (*SF96*)	$[-Si(CH_3)_2O-]_n$	1034	P(34)	1033.48	00^01-02^00	$C^{12}O_2^{16}$	*(Si-O-Si)* vib
			P(58)	1035.70	00^01-02^00	$C^{12}O_2^{18}$	
			R(24)	1034.83	00^01-02^00	$C^{13}O_2^{16}$	
		1092	R(46)	1092.96	00^01-02^00	$C^{12}O_2^{16}$	*(Si-O-Si)* vib
			R(12)	1092.29	00^01-02^00	$C^{12}O_2^{18}$	
Diethyl-Phthalate (*DEP*)	$C_{12}H_{14}O_4$	1017.7	P(50)	1016.72	00^01-02^00	$C^{12}O_2^{16}$	
		1073.6	R(12)	1073.28	00^01-02^00	$C^{12}O_2^{16}$	
			P(14)	1073.58	00^01-02^00	$C^{12}O_2^{18}$	
							Controlled Substance Simulants (Analytes)
Methyl-Benzoate	$C_8H_8O_2$	1027.3	P(40)	1027.38	00^01-02^00	$C^{12}O_2^{16}$	
			R(12)	1027.16	00^01-02^00	$C^{13}O_2^{16}$	
Methyl-Acetate	$C_3H_6O_2$	1048.2	P(18)	1048.66	00^01-02^00	$C^{12}O_2^{16}$	
			P(44)	1048.71	00^01-02^00	$C^{12}O_2^{18}$	
Atropine							ibid

Table 5 - continued.

Scatterer	Formula	Major Vibration(s) (cm^{-1})	Nearest Laser Line to the Scatterer's Vibrational Frequency: Transition	Freq. (cm^{-1})	Band	Type Laser	Comment
Scopolamine Hydrobromide hydrate	$C_{17}H_{21}NO_4$	1046.9	P(20) P(46)	1046.85 1046.90	00^01-02^00 00^01-02^00	$C^{12}O_2^{16}$ $C^{12}O_2^{18}$	
Tropine	$C_8H_{15}NO$	1047.4	P(20) P(46)	1046.85 1046.90	00^01-02^00 00^01-02^00	$C^{12}O_2^{16}$ $C^{12}O_2^{18}$	
3-Azabicyclo-[3.2.2] nonane	$C_8H_{15}N$	1079.4	R(20) P(6)	1078.59 1079.49	00^01-02^00 00^01-02^00	$C^{12}O_2^{16}$ $C^{12}O_2^{18}$	
Benztropine Mesylate	$C_{21}H_{25}NO$	1050.1	P(16) P(42)	1050.44 1050.49	00^01-02^00 00^01-02^00	$C^{12}O_2^{16}$ $C^{12}O_2^{18}$	
Nipecotic Acid	$C_6H_{11}NO_2$	1067.1	R(4) P(22)	1067.54 1067.36	00^01-02^00 00^01-02^00	$C^{12}O_2^{16}$ $C^{12}O_2^{18}$	
Piperidine	$C_5H_{11}N$	1051.2	P(16) P(42)	1050.44 1050.49	00^01-02^00 00^01-02^00	$C^{12}O_2^{16}$ $C^{12}O_2^{18}$	
Ethyl-Pipecolinate	$C_8H_{15}NO_2$	1047.3	P(20) P(46)	1046.85 1046.90	00^01-02^00 00^01-02^00	$C^{12}O_2^{16}$ $C^{12}O_2^{18}$	
Ethyl-Isonipecotate	$C_8H_{15}NO_2$	1045.7	P(22) P(48)	1045.02 1045.08	00^01-02^00 00^01-02^00	$C^{12}O_2^{16}$ $C^{12}O_2^{18}$	
33% Mont-morillonite 33% Kaolin 34% Illite	$Al_{1.7}Mg_{0.33}$ $[(OH)_2/Si_4/O_{10}]$ $Na_{0.33}(H_2O)_4$ $Al_4[(OH)_2/AlSi_3O_{10}$ $(K, H_3O)\ Al_2$ $(OH)_2/AlSi_3O_{10}$	1040 (±3)	P(28) P(54) R(32)	1039.37 1039.50 1039.38	00^01-02^00 00^01-02^00 00^01-02^00	$C^{12}O_2^{16}$ $C^{12}O_2^{18}$ $C^{13}O_2^{16}$	*Soil (Interferent)*

7. CONCLUSION

A foundation for applying Amplitude and Phase Sensitive Light Scattering (APSLS) technology toward solution of remote detection problems involving chemical and biological contaminants spread across various surfaces was presented. We are currently proceeding with developing an experimental data base of Mueller matrix elements, and will soon compare these data to theoretical predictions as part of a feasibility study. If these laboratory tests prove that contaminants on a surface can be successfully detected at a distance through selective sets of differential (on-then-off molecular resonance by the analyte) Mueller elements, then development of a prototype sensor can begin. With the data base in hand, optimization of hardware and software components in these ellipsometer systems can lead to a simplified prototype system engineered to a specific class of contaminant compounds.

The potential exists for extending this technology toward solution of other identification and classification problems of interest to the Department of Defense, environmental agencies, academia, and private industry. We will undoubtedly realize and incorporate n ver and better hardware and software modifications into the present instrument designs for these expanded applications.

Work proceeds on analysis of special Mueller elemental features in beam wavelength and angle orientation that can uniquely represent the analyte (**in situ**), and once presence is established, quantify it. All susceptible Mueller elements will be scrutinized from oblique-to-normal backscattering polar angles, and over the instrument's laser bandwidth spanning 9.1 - 12.2 μm: the mid IR 'fingerprint' region of many important biological and chemical compounds.

Methods of data processing will be improved, including a neural network architecture with pattern recognition algorithm that will likely be integrated into the ellipsometer's analog data acquisition unit. We will continue to study and improve on our methods of analysis of the Mueller field of elements, as we become more familiar with the technology. Future reports will provide updated progress as we complete debugging the software and hardware instrument components, as the digital data acquisition comes on-line, and as quantitative data becomes available. Revisions on the Full Wave theory to include scattering from non isotropic, many-layered, rough surfaces will be reported later, as will absorption and VCD predictions of complex analyte molecules via the quantum chemistry models.

The end goal of this program is to collect and analyze a comprehensive data base on CBW simulants on surfaces, from it identify the crucial polarization, angle, and wavelength parameters that will specify a less complex engineered prototype version of the experimental system, then development and engineering of that sensor.

LITERATURE CITED

1. Carrieri, A.H., Ditillo, J.T. and Schlein, M.S., *Depolarized Infrared Reflectance From Dry and Wetted Surfaces*, CRDEC-TR-87084, U.S. Army Chemical Research, Development and Engineering Center, Aberdeen Proving Ground, MD, September 1987, UNCLASSIFIED Report.

2. Renau, J., Cheo, P., and Cooper, H., "Depolarization of Linearly Polarized Waves Backscattered From Rough Metals and Inhomogeneous Dielectrics," *J. Opt. Soc. Am.*, Vol. 57 Num. 4, 1967.

3. Muller, R.H., "Definitions and Conventions in Ellipsometry," *Surf. Sci.*, Vol. 16 Num. 14, 1969.

4. Shurcliff, W.A., *Polarized Light*, Harvard University Press, Cambridge, MA, 1962.

5. Zaininger, K.H., and Revesz, A.G., "Ellipsometry-A Valuable Tool in Surface Research." *RCA Laboratories Review*, pp. 85-115, Princeton, NJ, March 1964.

6. Hauge, P.S., "Conventions and Formulas For Using the Mueller-Stokes Calculus in Ellipsometry," *Surf. Sci.*, Vol. 96 Num. 81, 1980.

7. Simmons, J.W. and Guttmann, M.J., *States, Waves and Photons: A Modern Introduction to Light*, Addison-Wesley Publishing Company, Reading, MA, 1970.

8. van de Hulst, H.C., *Light Scattering by Small Particles*, John Wiley and Sons, Inc., New York, NY, 1957.

9. Bohren, C.F. and Huffman, D.R., *Absorption and Scattering of Light by Small Particles*, John Wiley and Sons, Inc., New York, NY, 1983.

10. Jackson, J.D. *Classical Electrodynamics*, Second Edition, Chapter 7, Section 2, John Wiley and Sons, Inc., New York, NY, 1975.

11. Thompson, R.C., Bottiger, J.R., and Fry E.S., "Measurement of Polarized Light Interactions via the Mueller Matrix," *Appl. Opt.*, Vol. 19 Num. 8, 1980.

12. Williams, M.W., "Depolarization and Cross Polarization in Ellipsometry of Rough Surfaces," *Appl. Opt.*, Vol. 25 Num. 20, pp. 3616-22, 1986.

13. Roseler, A., "Use of the Mueller Matrix Formalism for the Spectroscopical IR Ellipsometry with the Fourier Transform Spectrometer," *Optik*, Vol. 61 Num 2, pp. 177-186, 1982.

14. Vorburger, T.V., and Ludema, K.C., "Ellipsometry of Rough Surfaces," *Appl. Opt.*, Vol. 19 Num. 4, pp.561-573, 1980.

15. Fritz Allen and Carlos Bustamante, Editors, *Applications of Circularly Polarized Radiation Using Synchrotron and Ordinary Sources*, Plenum Press, 1985.

Salzman, G.C., et al., "Multiparameter Light Scattering For Rapid Virus Identification," pp. 103-109,

Newman, C.D., and Allen, F.S., "Measuring the Mueller Matrix by a Multimodulator Scattering Instrument," pp. 59-67,

Bickel, W.S., "The Mueller Scattering Matrix Elements For Rayleigh Spheres," pp. 69-76.

16. Kemp, J.C., "Piezo-Optical Birefringence Modulators: New Use for a Long-Known Effect," *JOSA*, Vol. 59 Num. 950, 1969.

17. Hunt, A.J., and Huffman, D.R., "A New Polarization-Modulated Light Scattering Instrument," *Rev. Sci. Instrum.*, Vol. 44, pp. 1753-1762, 1973.

18. Whitt, M.W., and Ulaby, F.T., "Millimeter-Wave Polarimetric Measurements of Artificial and Natural Targets, "Conference paper, Inter. Geoscience and Remote Sensing Symposium (IGARSS), Ann Arbor, MI, 18-21 May 1987.

19. van Zyl, Jakob J., and Zebker, H. A., "Interpretation of Imaging Radar Polarization Signatures of Rough Surfaces," Conference Paper, Inter. Geoscience and Remote Sensing Symposium (IGARSS), Ann Arbor, MI, 18-21 May 1987.

20. Mendez, E.R., and O'Donnell, K.A., "Observations of Depolarization and Backscattering Enhancement in Light Scattering From Gaussian Random Surfaces," *Opt. Comm.*, Vol. 61, pp. 91-95, 1987.

21. Querry, M., *Optical Constants of Minerals and Other Materials From the Millimeter to the Ultraviolet*, CRDEC-CR-88009, U.S. Army Chemical Research, Development and Engineering Center, Aberdeen Proving Ground, MD, November 1987, UNCALSSIFIED Report.

22. Herzberg, G., *Molecular Spectra and Molecular Structure, Vol. 2, Infrared and Raman Spectra of Polyatomic Molecules*, Van Nostrand, Princeton, NJ, 1945.

23. Tyte, D. C., *Carbon Dioxide Lasers, Advances in Quantum Electronics*, Vol. 1, Goodwin, D. W., Ed., Academic Press, London, 1970.

24. Gradshteyn, I. S., and Ryzhik, I. M., *Table of Integrals, Series, and Products*, Equation 8.511(4), Academic Press, New York, 1980.

25. Frisch, M., *Gaussian90 User's Guide and Programmer's Reference*, Gaussian Inc., 1990.

26. Amos, R.D., *User's Guide to the Cambridge Analytic Derivative Package*, Cambridge University, 1991.

27. Chabolowski, C.F., Jensen, J.O., Yarkony, D.R., and Lengsfield, B.H., "Theoretical Study of the Radiative Lifetime for the Spin-Forbidden Transition $a^3\Sigma_u^+ \text{-} X^1\Sigma_g^+$ in He_2," *J. Chem. Phys.*, Vol. 90 Num. 2504, 1989.

28. Stevens, P.J., "Theory of Vibrational Circular Dichroism," *J. Chem. Phys.* Vol. 89 Num. 661, 1985.

29. Hameka, H.F., "Theoretical Prediction of Structures and Infrared Frequencies," *J. Mol. Struct. (Theochem)* Vol. 226 Num. 241, 1991.

30. Bahar, E., "Review of the Full Wave Solutions for Rough Surface Scattering and Depolarization: Comparisons with Geometric and Physical Optics, Perturbation, and Two-scale Hybrid Solutions," *Journal of Geophysical Research*, Vol. 92 Num. C5, pp. 5209-5224, May 1987.

31. Bahar, E., "Full-Wave Solutions for the Depolarization of the Scattered Radiation Fields by Rough Surfaces of Arbitrary Slope," *IEEE Transactions on Antennas and Propagation*, Vol. AP-29 Num. 3, pp. 443-454, May 1981.

32. Bahar, E. and Fitzwater, M. A., "Scattering Cross Sections for Composite Rough Surfaces Using the Unified Full Wave Approach," *IEEE Transactions on Antennas and Propagation*, Vol. AP-32 Num. 7, pp. 730-734, July 1984.

33. Bahar, E. and Fitzwater, M. A., "Copolarized and Cross-Polarized Incoherent Specific Intensities for Waves at Oblique Incidence Upon a Layer of Finitely Conducting Particles with Rough Surfaces," *Journal of the Optical Society of America A.*, Vol. 4, pp. 41-56, January 1987.

34. Bahar, E., *Scattering and Depolarization of Electromagnetic Waves – Full Wave Solutions*, RADC-TR-83-118, Rome Air Development Center, Air Force Systems Command, Griffiss Air Force Base, NY, November 1983.

35. Morrison, D., *Multivariate Statistical Methods*, Chapters 2,3, and 4, McGraw-Hill, 1967.

36. Carrieri, A.H., "Infrared Detection of Liquids on Terrestrial Surfaces by CO_2 Laser Heating," *Appl. Opt.* Vol. 20 Num. 33, 1990.

37. Rumelhart, McClelland, and the PDP Research Group, *Parallel and Distributed Processing*, MIT Press, pp. 322-330, 1986.

38. Lippmann, R. P., "An Introduction to Computing With Neural Nets," *IEEE ASSP Magazine*, April 1987.

APPENDIX I. EMISSIONS FROM CO_2 PROBE LASERS

APPENDIX I. EMISSIONS FROM CO_2 PROBE LASERS

The following data are reproduced from the **Handbook of Laser Science and Technology**, Marvin J. Weber, CRC Press, Inc. 1982. The laser emissions shown span all allowable wavelength outputs from the ellipsometer's four laser sources. Band nomenclature is of the form (ν_1,ν_2^l,ν_3), where ν_1 and ν_3 are quantum numbers specifying stretching modes, and ν_2 is a quantum number for the bending mode of the linear triatomic CO_2 molecule. The vibrational angular momentum quantum number is specified by l. P(J) and R(J) are vibrational-rotational transitions of the types $(v+1,J-1) \rightarrow (v,J)$ and $(v+1,J+1) \rightarrow (v,J)$, respectively, where J is the rotational quantum number. For further details, see the standard text by **Herzberg.**[20] Each of the ellipsometer's four laser sources can produce between 45 and 75 distinct spectral emissions, most with sufficient power for MCT detection of scattered light from surfaces at ranges to ≈ 100 meters at oblique scattering angles and f/10 receiver optics.

Appendix I

LASER	BAND	TRANSITION	WAVELENGTH μm	FREQUENCY cm^{-1}
$C^{12}O_2^{16}$	$00^01 \rightarrow 02^00$	R(62)	9.09349	1099.6872
		R(60)	9.09976	1098.9301
		R(58)	9.10623	1098.14940
		R(56)	9.11291	1097.344886
		R(54)	9.11979	1096.516356
		R(52)	9.12689	1095.663612
		R(50)	9.13420	1094.786462
		R(48)	9.14173	1093.884721
		R(46)	9.14948	1092.958211
		R(44)	9.15745	1092.006758
		R(42)	9.16565	1091.030196
		R(40)	9.17407	1090.028367
		R(38)	9.18273	1089.001119
		R(36)	9.19161	1087.948306
		R(34)	9.20073	1086.869791
		R(32)	9.21009	1085.765445
		R(30)	9.21969	1084.635145
		R(28)	9.22953	1083.478778
		R(26)	9.23961	1082.296237
		R(24)	9.24995	1081.087426
		R(22)	9.26053	1079.852255
		R(20)	9.27136	1078.590644
		R(18)	9.28244	1077.302520
		R(16)	9.29379	1075.987820
		R(14)	9.30539	1074.646490
		R(12)	9.31725	1073.278484
		R(10)	9.32937	1071.883766
		R(8)	9.34176	1070.462308
		R(6)	9.35441	1069.024093
		R(4)	9.36734	1067.539110
		R(2)	9.38053	1066.037360
		R(0)	9.39400	1064.508853
		P(2)	9.41472	1062.165965
		P(4)	9.42889	1060.570666
		P(6)	9.44333	1058.948714
		P(8)	9.45805	1057.300161
		P(10)	9.47306	1055.625068
		P(12)	9.48835	1053.923503
		P(14)	9.50394	1052.195545
		P(16)	9.51981	1050.441282

Appendix I

LASER	BAND	TRANSITION	WAVELENGTH μm	FREQUENCY cm^{-1}
$C^{12}O_2^{16}$	$00^01 - 02^00$	P(18)	9.53597	1048.660810
		P(20)	9.55243	1046.854234
		P(22)	9.56918	1045.021670
		P(24)	9.58623	1043.163239
		P(26)	9.60357	1041.279074
		P(28)	9.62122	1039.369315
		P(30)	9.63917	1037.434110
		P(32)	9.65742	1035.473616
		P(34)	9.67597	1033.487999
		P(36)	9.69483	1031.477430
		P(38)	9.71400	1029.442092
		P(40)	9.73348	1027.382171
		P(42)	9.75326	1025.297865
		P(44)	9.77336	1023.189375
		P(46)	9.79377	1021.056912
		P(48)	9.81450	1018.900693
		P(50)	9.83554	1016.720942
		P(52)	9.85690	1014.517888
		P(54)	9.87858	1012.291767
		P(56)	9.90057	1010.042823
		P(58)	9.92289	1007.771302
		P(60)	9.94552	1005.47746
		P(62)	9.96849	1003.1615
		P(64)	9.99177	1000.8238
		P(66)	10.01538	998.4646
	$00^01 - 10^00$	R(62)	10.02591	997.41550
		R(62)	10.02591	997.41550
		R(60)	10.03347	996.66441
		R(58)	10.04132	995.884686
		R(56)	10.04948	995.076610
		R(54)	10.05793	994.240442
		R(52)	10.06668	993.376427
		R(50)	10.07572	992.484803
		R(48)	10.08506	991.565748
		R(46)	10.09469	990.619630
		R(44)	10.10462	989.646506
		R(42)	10.11484	988.646626
		R(40)	10.12535	987.620181
		R(38)	10.13616	986.567352
		R(36)	10.14725	985.488312

LASER	BAND	TRANSITION	WAVELENGTH μm	FREQUENCY cm^{-1}
$C^{12}O_2^{16}$	$00^01 \rightarrow 10^00$	R(34)	10.15865	984.383226
		R(32)	10.17033	983.252249
		R(30)	10.18231	982.095531
		R(28)	10.19458	980.913211
		R(26)	10.20715	979.705421
		R(24)	10.22001	978.472286
		R(22)	10.23317	977.213922
		R(20)	10.24663	975.930439
		R(18)	10.26039	974.621939
		R(16)	10.27445	973.288517
		R(14)	10.28880	971.930258
		R(12)	10.30347	970.547244
		R(10)	10.31843	969.139547
		R(8)	10.33370	967.707233
		R(6)	10.34928	966.250361
		R(4)	10.36518	964.768982
		R(2)	10.38138	963.263140
		R(0)	10.39790	961.732874
		P(2)	10.42327	959.391745
		P(4)	10.44059	957.800537
		P(6)	10.45823	956.184982
		P(8)	10.47619	954.545087
		P(10)	10.49449	952.880850
		P(12)	10.51312	951.192264
		P(14)	10.53209	949.479314
		P(16)	10.55140	947.741979
		P(18)	10.57105	945.980230
		P(20)	10.59104	944.194030
		P(22)	10.61139	942.383336
		P(24)	10.63210	940.548098
		P(26)	10.65316	938.688257
		P(28)	10.67459	936.803747
		P(30)	10.69639	934.894496
		P(32)	10.71857	932.960421
		P(34)	10.74112	931.001434
		P(36)	10.76406	929.017437
		P(38)	10.78739	927.008325
		P(40)	10.81111	924.973985
		P(42)	10.83524	922.914294
		P(44)	10.85978	920.829123

LASER	BAND	TRANSITION	WAVELENGTH μm	FREQUENCY cm^{-1}
$C^{12}O_2^{16}$	$00^01 \rightarrow 10^00$	P(46)	10.88473	918.718331
		P(48)	10.91010	916.581770
		P(50)	10.93590	914.419283
		P(52)	10.96214	912.230703
		P(54)	10.98882	910.015853
		P(56)	11.01595	907.774549
		P(58)	11.04354	905.50659
		P(60)	11.07160	903.21177
		P(62)	11.10014	900.88992
		P(64)	11.12915	898.54082
		P(66)	11.15867	896.1643
		P(68)	11.18868	893.7602
$C^{12}O_2^{18}$	$00^01 \rightarrow 02^00$	R(50)	8.98767	1112.635004
		R(48)	8.99495	1112.635004
		R(46)	8.99495	1111.735484
		R(42)	9.00238	1110.817288
		R(44)	9.00998	1109.880340
		R(42)	9.01775	1108.924564
		R(40)	9.02568	1107.949890
		R(38)	9.03378	1106.956250
		R(36)	9.04205	1105.943579
		R(34)	9.05050	1104.911817
		R(32)	9.05911	1103.860906
		R(30)	9.06790	1102.790794
		R(28)	9.07687	1101.701429
		R(26)	9.08601	1100.592768
		R(24)	9.09533	1099.464767
		R(22)	9.10484	1098.317390
		R(20)	9.11452	1097.150603
		R(18)	9.12438	1095.964378
		R(16)	9.13443	1094.758688
		R(14)	9.14467	1093.533515
		R(12)	9.15509	1092.288842
		R(10)	9.16570	1091.024658
		R(8)	9.17649	1089.740957
		R(6)	9.18748	1088.437736
		R(4)	9.19866	1087.114998
		R(2)	9.21003	1085.772750
		P(2)	9.23931	1082.331864

LASER	BAND	TRANSITION	WAVELENGTH μm	FREQUENCY cm^{-1}
$C^{12}O_2^{18}$	$00^01 \rightarrow 02^00$	P(4)	9.25137	1080.921457
		P(6)	9.26362	1079.491631
		P(8)	9.27607	1078.042418
		P(10)	9.28873	1076.573857
		P(12)	9.30158	1075.085991
		P(14)	9.31464	1073.578866
		P(16)	9.32790	1072.052534
		P(18)	9.34137	1070.507051
		P(20)	9.35504	1068.942477
		P(22)	9.36892	1067.358878
		P(24)	9.38301	1065.756323
		P(26)	9.39730	1064.134886
		P(28)	9.41181	1062.494644
		P(30)	9.42653	1060.835680
		P(32)	9.44146	1059.158080
		P(34)	9.45661	1057.461932
		P(36)	9.47196	1055.747333
		P(38)	9.48754	1054.014378
		P(40)	9.50333	1052.263168
		P(42)	9.51933	1050.493809
		P(44)	9.53556	1048.706409
		P(46)	9.55200	1046.901078
		P(48)	9.56866	1045.077932
		P(50)	9.58555	1043.237087
		P(52)	9.60265	1041.378663
		P(54)	9.61998	1039.502785
		P(56)	9.63754	1037.609577
		P(58)	9.65531	1035.699167
	$00^01 \rightarrow 10^00$	R(44)	10.08328	991.274098
		R(42)	10.09604	990.487703
		R(40)	10.10435	989.673232
		R(38)	10.11295	988.830811
		R(36)	10.12186	987.960562
		R(34)	10.13107	987.062600
		R(32)	10.14058	986.137035
		R(30)	10.15039	985.183973
		R(28)	10.16050	984.203513
		R(26)	10.17091	983.195749
		R(24)	10.18163	982.160770
		R(22)	10.19265	981.098661

LASER	BAND	TRANSITION	WAVELENGTH μm	FREQUENCY cm^{-1}
$C^{12}O_2^{18}$	$00^01 - 10^00$	R(20)	10.20398	980.009499
		R(18)	10.21562	978.893358
		R(16)	10.22756	977.750307
		R(14)	10.23981	976.580410
		R(12)	10.25238	975.383724
		R(10)	10.26525	974.160302
		R(8)	10.27844	972.910195
		R(6)	10.29195	971.633444
		R(4)	10.30577	970.330089
		P(6)	10.38759	962.687339
		P(8)	10.40354	961.211656
		P(10)	10.41982	959.709502
		P(12)	10.43644	958.180873
		P(14)	10.45341	956.625761
		P(16)	10.47072	955.044153
		P(18)	10.48838	953.436031
		P(20)	10.50639	951.801372
		P(22)	10.52476	950.140149
		P(24)	10.54349	948.452326
		P(26)	10.56259	946.737867
		P(28)	10.58205	944.996728
		P(30)	10.60188	943.228860
		P(32)	10.62209	941.434209
		P(34)	10.64268	939.612716
		P(36)	10.66366	937.764316
		P(38)	10.68503	935.888939
		P(40)	10.70679	933.986510
		P(42)	10.72896	932.056949
		P(44)	10.75153	930.100167
		P(46)	10.77451	928.116074
		P(48)	10.79792	926.104570
$C^{13}O_2^{16}$	$00^01 - 02^00$	R(36)	9.60169	1041.483334
		R(34)	9.61126	1040.446675
		R(32)	9.62110	1039.381840
		R(30)	9.63123	1038.288700
		R(28)	9.64165	1037.167135
		R(26)	9.65235	1036.017032
		R(24)	9.66335	1034.838287

LASER	BAND	TRANSITION	WAVELENGTH μm	FREQUENCY cm^{-1}
$C^{13}O_2^{16}$	$00^01 \rightarrow 02^00$	R(22)	9.67463	1033.630806
		R(20)	9.68622	1032.394502
		R(18)	9.69810	1031.129298
		R(16)	9.71029	1029.835128
		R(14)	9.72278	1028.511931
		R(12)	9.73558	1027.159658
		R(10)	9.74870	1025.778270
		R(8)	9.76212	1024.367737
		R(6)	9.77586	1022.928037
		R(4)	9.78992	1021.459160
		P(6)	9.87304	1012.859224
		P(8)	9.88923	1011.201098
		P(10)	9.90576	1009.514024
		P(12)	9.92262	1007.798072
		P(14)	9.93983	1006.053323
		P(16)	9.95738	1004.279869
		P(18)	9.97528	1002.477810
		P(20)	9.99353	1000.647256
		P(22)	10.01213	998.788325
		P(24)	10.03108	996.901145
		P(26)	10.05039	994.985854
		P(28)	10.07006	993.042598
		P(30)	10.09009	991.071531
		P(32)	10.11048	989.072816
		P(34)	10.13123	987.046625
		P(36)	10.15235	984.993138
		P(38)	10.17385	982.912542
	$00^01 \rightarrow 10^00$	R(44)	10.60063	943.340303
		R(42)	10.61310	942.231411
		R(40)	10.62585	941.101238
		R(38)	10.63886	939.949924
		R(36)	10.65215	938.777604
		R(34)	10.66571	937.584403
		R(32)	10.67953	936.370443
		R(30)	10.69363	935.135838
		R(28)	10.70801	933.880687
		R(26)	10.72265	932.605121
		R(24)	10.73757	931.309207
		R(22)	10.75277	929.993046

Appendix I

LASER	BAND	TRANSITION	WAVELENGTH μm	FREQUENCY cm^{-1}
$C^{13}O_2^{16}$	$00^01 \rightarrow 10^00$	R(20)	10.76824	928.656723
		R(18)	10.78399	927.300318
		R(16)	10.80002	925.923906
		R(14)	10.81634	924.527554
		R(12)	10.83293	923.111328
		R(10)	10.84981	921.675286
		R(8)	10.86697	920.219482
		R(6)	10.88443	918.743964
		R(4)	10.90217	917.248777
		P(4)	10.98566	910.277955
		P(6)	11.00503	908.675151
		P(8)	11.02472	907.052844
		P(10)	11.04471	905.411040
		P(12)	11.06501	903.749742
		P(14)	11.08563	902.068947
		P(16)	11.10656	900.368647
		P(18)	11.12782	898.648830
		P(20)	11.14940	896.909477
		P(22)	11.17131	895.150565
		P(24)	11.19534	893.372066
		P(26)	11.21612	891.573944
		P(28)	11.23903	889.756160
		P(30)	11.26229	887.918669
		P(32)	11.28590	886.061419
		P(34)	11.30986	884.184353
		P(36)	11.33418	882.287407
		P(38)	11.35885	880.370512
		P(40)	11.38390	878.433591
		P(42)	11.40932	876.476562
		P(44)	11.43511	874.49933
		P(46)	11.46129	872.50181
		P(48)	11.48786	870.48389
$C^{14}O_2^{16}$	$00^01 \rightarrow 02^00$	R(40)	9.91788	1008.280282
		R(38)	9.92733	
		R(36)	9.93709	1006.330912
		R(34)	9.94715	1005.312772
		R(32)	9.95753	1004.265463
		R(30)	9.96821	1003.188845

Appendix I

LASER	BAND	TRANSITION	WAVELENGTH μm	FREQUENCY cm^{-1}
$C^{14}O_2^{16}$	$00^01 \rightarrow 02^00$	R(28)	9.97922	1002.082785
		R(26)	9.99054	1000.947161
		R(24)	10.00218	999.781858
		R(22)	10.01415	998.586771
		R(20)	10.02645	997.361804
		R(18)	10.03908	996.106870
		R(16)	10.05205	994.821893
		R(14)	10.06536	993.506806
		R(12)	10.07900	992.161553
		R(10)	10.09300	990.786087
		R(8)	10.10734	998.380373
		R(6)	10.12203	987.944385
		P(8)	10.24370	976.209763
		P(10)	10.26149	974.516934
		P(12)	10.27967	972.794124
		P(14)	10.29822	971.041421
		P(16)	10.31716	969.258921
		P(18)	10.33649	967.446731
		P(20)	10.35620	965.604971
		P(22)	10.37631	963.733766
		P(24)	10.39681	961.833254
		P(26)	10.41771	959.903583
		P(28)	10.43901	957.944909
		P(30)	10.46072	955.957396
		P(32)	10.48283	953.941220
		P(34)	10.50534	951.896562
		P(36)	10.52827	949.823614
	$00^01 \rightarrow 10^00$	R(50)	11.10699	900.33358
		R(48)	11.12097	899.20226
		R(46)	11.13520	898.05318
		R(44)	11.14968	896.88643
		R(42)	11.16443	895.70211
		R(40)	11.17943	894.50031
		R(38)	11.19468	893.28113
		R(36)	11.21020	892.04463
		R(34)	11.22598	890.79092
		R(32)	11.24202	889.52005
		R(30)	11.25832	888.23212
		R(28)	11.27488	886.92718

Appendix I

LASER	BAND	TRANSITION	WAVELENGTH μm	FREQUENCY cm^{-1}
$C^{14}O_2^{16}$	$00^01 \rightarrow 10^00$	R(26)	11.29171	885.60530
		R(24)	11.30881	884.26654
		R(22)	11.32617	882.91098
		R(18)	11.36170	880.14964
		R(16)	11.37988	878.74397
		R(14)	11.39833	877.32170
		R(12)	11.41705	875.88288
		R(10)	11.43605	874.42754
		R(8)	11.45533	872.95574
		R(6)	11.47490	871.46751
		R(4)	11.49474	869.96288
		P(6)	11.60907	861.39566
		P(8)	11.63081	859.78513
		P(10)	11.65286	858.15839
		P(12)	11.67521	856.51545
		P(14)	11.69787	854.85631
		P(16)	11.72084	853.18100
		P(18)	11.74413	851.48950
		P(20)	11.76773	849.78182
		P(22)	11.79165	848.05797
		P(24)	11.81589	846.31794
		P(26)	11.84046	844.56172
		P(28)	11.86536	842.78930
		P(30)	11.89060	841.00067
		P(32)	11.91617	839.19581
		P(34)	11.94209	837.37471
		P(36)	11.96835	835.53734
		P(38)	11.99496	833.68367
		P(40)	12.02192	831.81368
		P(42)	12.04925	829.92733
		P(44)	12.07694	828.02458
		P(46)	12.10499	826.10540
		P(48)	12.13342	824.16974

APPENDIX II. SYSTEM TO SAMPLE MUELLER MATRIX TRANSFORMATION: 3-MIRROR GONIOMETER TYPE ELLIPSOMETER FOR IN-SITU ANALYSES OF POROUS SURFACES.

APPENDIX II. SYSTEM TO SAMPLE MUELLER MATRIX TRANSFORMATION: 3-MIRROR GONIOMETER TYPE ELLIPSOMETER FOR IN-SITU ANALYSES OF POROUS SURFACES.

The sixteen equations that follow decouple Mueller matrix elements of the scattering sample (f_{ij}) from *system* matrix elements (Equation 6b, ψ_{ij}) generated by the ellipsometer facility of Figure 4a. The total scattering signal includes contributions from the sample and four flat mirrors oriented at 45^0 incidence, and positioned between transmission- and collection-beam PEM modulators.

Elements b_{ij}, c_{ij}, and d_{ij} are experimental calibration data as measured from the configurations schematically drawn in Figure 10, and θ is the angle of backscattering subtended by the goniometer arm. MACSYMA, an interactive symbolic mathematical program, was used in determining the product of the ten 4X4 matrices of Equation 6b, and in producing the Fortran code of the following equations for use in the ellipsometer's data acquisition system.

It suffices, from the experimental complexity and uncertainty of this calibration procedure, that an optical redesign that self-compensates all mirror elements is more practical as a means of accomplishing sample Mueller element measurements from terrestrial surfaces. (See Section 4.6.4.)

Appendix II

$$\begin{pmatrix} \psi_{11} & \psi_{12} & \psi_{13} & \psi_{14} \\ \psi_{21} & \psi_{22} & \psi_{23} & \psi_{24} \\ \psi_{31} & \psi_{32} & \psi_{33} & \psi_{34} \\ \psi_{41} & \psi_{42} & \psi_{43} & \psi_{44} \end{pmatrix} \rightarrow \begin{pmatrix} f_{11} & f_{12} & f_{13} & f_{14} \\ f_{21} & f_{22} & f_{23} & f_{24} \\ f_{31} & f_{32} & f_{33} & f_{34} \\ f_{41} & f_{42} & f_{43} & f_{44} \end{pmatrix}$$

$$f_{44}(\theta) = \frac{\psi_{44}}{b_{33}d_{33}}$$

$$f_{43}(\theta) = -\frac{\psi_{42}\sin(2\theta)-\psi_{43}\cos(2\theta)}{b_{33}d_{33}}$$

$$f_{42}(\theta) = -\frac{b_{11}\psi_{43}\sin(2\theta)+b_{11}\psi_{42}\cos(2\theta)-b_{12}\psi_{41}}{(b_{12}^2-b_{11}^2)d_{33}}$$

$$f_{41}(\theta) = \frac{b_{12}\psi_{43}\sin(2\theta)+b_{12}\psi_{42}\cos(2\theta)-b_{11}\psi_{41}}{(b_{12}^2-b_{11}^2)d_{33}}$$

$$f_{34}(\theta) = [((c_{12}d_{12}-c_{11}d_{11})d_{33}\psi_{24}+(c_{11}d_{12}-c_{12}d_{11})d_{33}\psi_{14})\sin(2\theta)+(c_{33}d_{12}^2-$$

$$\frac{c_{33}d_{11}^2)\psi_{34}\cos(2\theta)]}{(b_{33}c_{33}d_{12}^2-b_{33}c_{33}d_{11}^2)d_{33}}$$

$$f_{33}(\theta) = -[((c_{12}d_{12}-c_{11}d_{11})d_{33}\psi_{22}+(c_{11}d_{12}-c_{12}d_{11})d_{33}\psi_{12})\sin^2(2\theta)+((c_{33}d_{12}^2-$$
$$c_{33}d_{11}^2)\psi_{32}+(c_{11}d_{11}-c_{12}d_{12})d_{33}\psi_{23}+(c_{12}d_{11}-c_{11}d_{12})d_{33}\psi_{13})\cos(2\theta)\sin(2\theta)+$$

$$\frac{(c_{33}d_{11}^2-c_{33}d_{12}^2)\psi_{33}\cos^2(2\theta)]}{(b_{33}c_{33}d_{12}^2-b_{33}c_{33}d_{11}^2)d_{33}}$$

$$f_{32}(\theta) = -\frac{\begin{aligned}&[((b_{11}c_{12}d_{12}-b_{11}c_{11}d_{11})d_{33}\psi_{23}+(b_{11}c_{11}d_{12}-b_{11}c_{12}d_{11})d_{33}\psi_{13})\sin^2(2\theta)+\\&(((b_{11}c_{33}d_{12}^2-b_{11}c_{33}d_{11}^2)\psi_{33}+(b_{11}c_{12}d_{12}-b_{11}c_{11}d_{11})d_{33}\psi_{22}+(b_{11}c_{11}d_{12}-\\&b_{11}c_{12}d_{11})d_{33}\psi_{12})\cos(2\theta)+(b_{12}c_{11}d_{11}-b_{12}c_{12}d_{12})d_{33}\psi_{21}+(b_{12}c_{12}d_{11}-\\&b_{12}c_{11}d_{12})d_{33}\psi_{11})\sin(2\theta)+(b_{11}c_{33}d_{12}^2-b_{11}c_{33}d_{11}^2)\psi_{32}\cos^2(2\theta)+\\&(b_{12}c_{33}d_{11}^2-b_{12}c_{33}d_{12}^2)\psi_{31}\cos(2\theta)]\end{aligned}}{((b_{12}^2-b_{11}^2)c_{33}d_{12}^2+(b_{11}^2-b_{12}^2)c_{33}d_{11}^2)d_{33}}$$

$$f_{31}(\theta) = \frac{\begin{aligned}&[((b_{12}c_{12}d_{12}-b_{12}c_{11}d_{11})d_{33}\psi_{23}+(b_{12}c_{11}d_{12}-b_{12}c_{12}d_{11})d_{33}\psi_{13})\sin^2(2\theta)+\\&(((b_{12}c_{33}d_{12}^2-b_{12}c_{33}d_{11}^2)\psi_{33}+(b_{12}c_{12}d_{12}-b_{12}c_{11}d_{11})d_{33}\psi_{22}+(b_{12}c_{11}d_{12}-\\&b_{12}c_{12}d_{11})d_{33}\psi_{12})\cos(2\theta)+(b_{11}c_{11}d_{11}-b_{11}c_{12}d_{12})d_{33}\psi_{21}+(b_{11}c_{12}d_{11}-\\&b_{11}c_{11}d_{12})d_{33}\psi_{11})\sin(2\theta)+(b_{12}c_{33}d_{12}^2-b_{12}c_{33}d_{11}^2)\psi_{32}\cos^2(2\theta)+(b_{11}c_{33}d_{11}^2-\\&b_{11}c_{33}d_{12}^2)\psi_{31}\cos(2\theta)]\end{aligned}}{((b_{12}^2-b_{11}^2)c_{33}d_{12}^2+(b_{11}^2-b_{12}^2)c_{33}d_{11}^2)d_{33}}$$

$$f_{24}(\theta) = \frac{\begin{aligned}&[(c_{11}c_{33}d_{12}^2-c_{11}c_{33}d_{11}^2)\psi_{34}\sin(2\theta)+((c_{11}^2d_{11}-c_{11}c_{12}d_{12})d_{33}\psi_{24}+\\&(c_{11}c_{12}d_{11}-c_{11}^2d_{12})d_{33}\psi_{14})\cos(2\theta)+(c_{11}c_{12}d_{12}-c_{12}^2d_{11})d_{33}\psi_{24}+\\&(c_{12}^2d_{12}-c_{11}c_{12}d_{11})d_{33}\psi_{14}]\end{aligned}}{((b_{33}c_{12}^2-b_{33}c_{11}^2)d_{12}^2+(b_{33}c_{11}^2-b_{33}c_{12}^2)d_{11}^2)d_{33}}$$

$$f_{23}(\theta) = -\frac{\begin{aligned}&[(c_{11}c_{33}d_{12}^2-c_{11}c_{33}d_{11}^2)\psi_{32}\sin^2(2\theta)+(((c_{11}c_{33}d_{11}^2-\\&c_{11}c_{33}d_{12}^2)\psi_{33}+(c_{11}^2d_{11}-c_{11}c_{12}d_{12})d_{33}\psi_{22}+(c_{11}c_{12}d_{11}-\\&c_{11}^2d_{12})d_{33}\psi_{12})\cos(2\theta)+(c_{11}c_{12}d_{12}-c_{12}^2d_{11})d_{33}\psi_{22}+(c_{12}^2d_{12}-\\&c_{11}c_{12}d_{11})d_{33}\psi_{12})\sin(2\theta)+((c_{11}c_{12}d_{12}-c_{11}^2d_{11})d_{33}\psi_{23}+(c_{11}^2d_{12}-\\&c_{11}c_{12}d_{11})d_{33}\psi_{13})\cos^2(2\theta)+((c_{12}^2d_{11}-c_{11}c_{12}d_{12})d_{33}\psi_{23}+(c_{11}c_{12}d_{11}-\\&c_{12}^2d_{12})d_{33}\psi_{13})\cos(2\theta)]\end{aligned}}{((b_{33}c_{12}^2-b_{33}c_{11}^2)d_{12}^2+(b_{33}c_{11}^2-b_{33}c_{12}^2)d_{11}^2)d_{33}}$$

$$f_{22}(\theta) = -[(b_{11}c_{11}c_{33}d_{12}^2 - b_{11}c_{11}c_{33}d_{11}^2)\psi_{33}\sin^2(2\theta) + (((b_{11}c_{11}c_{33}d_{12}^2 -$$
$$b_{11}c_{11}c_{33}d_{11}^2)\psi_{32} + (b_{11}c_{11}^2 d_{11} - b_{11}c_{11}c_{12}d_{12})d_{33}\psi_{23} + (b_{11}c_{11}c_{12}d_{11} -$$
$$b_{11}c_{11}^2 d_{12})d_{33}\psi_{13})\cos(2\theta) + (b_{12}c_{11}c_{33}d_{11}^2 - b_{12}c_{11}c_{33}d_{12}^2)\psi_{31} +$$
$$(b_{11}c_{11}c_{12}d_{12} - b_{11}c_{12}^2 d_{11})d_{33}\psi_{23} + (b_{11}c_{12}^2 d_{12} - b_{11}c_{11}c_{12}d_{11})d_{33}\psi_{13})\sin(2\theta) +$$
$$((b_{11}c_{11}^2 d_{11} - b_{11}c_{11}c_{12}d_{12})d_{33}\psi_{22} + (b_{11}c_{11}c_{12}d_{11} - b_{11}c_{11}^2 d_{12})d_{33}\psi_{12})\cos^2(2\theta) +$$
$$((b_{11}c_{11}c_{12}d_{12} - b_{11}c_{12}^2 d_{11})d_{33}\psi_{22} + (b_{12}c_{11}c_{12}d_{12} - b_{12}c_{11}^2 d_{11})d_{33}\psi_{21} +$$
$$(b_{11}c_{12}^2 d_{12} - b_{11}c_{11}c_{12}d_{11})d_{33}\psi_{12} + (b_{12}c_{11}^2 d_{12} - b_{12}c_{11}c_{12}d_{11})d_{33}\psi_{11})\cos(2\theta) +$$
$$\frac{(b_{12}c_{12}^2 d_{11} - b_{12}c_{11}c_{12}d_{12})d_{33}\psi_{21} + (b_{12}c_{11}c_{12}d_{11} - b_{12}c_{12}^2 d_{12})d_{33}\psi_{11}]}{(((b_{12}^2 - b_{11}^2)c_{12}^2 + (b_{11}^2 - b_{12}^2)c_{11}^2)d_{12}^2 + ((b_{11}^2 - b_{12}^2)c_{12}^2 + (b_{12}^2 - b_{11}^2)c_{11}^2)d_{11}^2)d_{33}}$$

$$f_{21}(\theta) = [(b_{12}c_{11}c_{33}d_{12}^2 - b_{12}c_{11}c_{33}d_{11}^2)\psi_{33}\sin^2(2\theta) + (((b_{12}c_{11}c_{33}d_{12}^2 -$$
$$b_{12}c_{11}c_{33}d_{11}^2)s_{32} + (b_{12}c_{11}^2 d_{11} - b_{12}c_{11}c_{12}d_{12})d_{33}\psi_{23} + (b_{12}c_{11}c_{12}d_{11} -$$
$$b_{12}c_{11}^2 d_{12})d_{33}\psi_{13})\cos(2\theta) + (b_{11}c_{11}c_{33}d_{11}^2 - b_{11}c_{11}c_{33}d_{12}^2)\psi_{31} +$$
$$(b_{12}c_{11}c_{12}d_{12} - b_{12}c_{12}^2 d_{11})d_{33}\psi_{23} + (b_{12}c_{12}^2 d_{12} - b_{12}c_{11}c_{12}d_{11})d_{33}\psi_{13})\sin(2\theta) +$$
$$((b_{12}c_{11}^2 d_{11} - b_{12}c_{11}c_{12}d_{12})d_{33}\psi_{22} + (b_{12}c_{11}c_{12}d_{11} -$$
$$b_{12}c_{11}^2 d_{12})d_{33}\psi_{12})\cos^2(2\theta) + ((b_{12}c_{11}c_{12}d_{12} - b_{12}c_{12}^2 d_{11})d_{33}\psi_{22} + (b_{11}c_{11}c_{12}d_{12} -$$
$$b_{11}c_{11}^2 d_{11})d_{33}\psi_{21} + (b_{12}c_{12}^2 d_{12} - b_{12}c_{11}c_{12}d_{11})d_{33}\psi_{12} + (b_{11}c_{11}^2 d_{12} -$$
$$b_{11}c_{11}c_{12}d_{11})d_{33}\psi_{11})\cos(2\theta) + (b_{11}c_{12}^2 d_{11} - b_{11}c_{11}c_{12}d_{12})d_{33}\psi_{21} +$$
$$\frac{(b_{11}c_{11}c_{12}d_{11} - b_{11}c_{12}^2 d_{12})d_{33}\psi_{11}]}{(((b_{12}^2 - b_{11}^2)c_{12}^2 + (b_{11}^2 - b_{12}^2)c_{11}^2)d_{12}^2 + ((b_{11}^2 - b_{12}^2)c_{12}^2 + (b_{12}^2 - b_{11}^2)c_{11}^2)d_{11}^2)d_{33}}$$

$$f_{14}(\theta) = -[(c_{12}c_{33}d_{12}^2 - c_{12}c_{33}d_{11}^2)\psi_{34}\sin(2\theta) + ((c_{11}c_{12}d_{11} - c_{12}^2 d_{12})d_{33}\psi_{24} +$$
$$(c_{12}^2 d_{11} - c_{11}c_{12}d_{12})d_{33}\psi_{14})\cos(2\theta) + (c_{11}^2 d_{12} - c_{11}c_{12}d_{11})d_{33}\psi_{24} +$$
$$\frac{(c_{11}c_{12}d_{12} - c_{11}^2 d_{11})d_{33}\psi_{14}]}{((b_{33}c_{12}^2 - b_{33}c_{11}^2)d_{12}^2 + (b_{33}c_{11}^2 - b_{33}c_{12}^2)d_{11}^2)d_{33}}$$

$$
\begin{aligned}
f_{13}(\theta) = [&(c_{12}c_{33}d_{12}^2 - c_{12}c_{33}d_{11}^2)\psi_{32}\sin^2(2\theta) + (((c_{12}c_{33}d_{11}^2 - c_{12}c_{33}d_{12}^2)\psi_{33} + \\
&(c_{11}c_{12}d_{11} - c_{12}^2 d_{12})d_{33}\psi_{22} + (c_{12}^2 d_{11} - c_{11}c_{12}d_{12})d_{33}\psi_{12})\cos(2\theta) + \\
&(c_{11}^2 d_{12} - c_{11}c_{12}d_{11})d_{33}\psi_{22} + (c_{11}c_{12}d_{12} - c_{11}^2 d_{11})d_{33}\psi_{12})\sin(2\theta) + \\
&((c_{12}^2 d_{12} - c_{11}c_{12}d_{11})d_{33}\psi_{23} + (c_{11}c_{12}d_{12} - c_{12}^2 d_{11})d_{33}\psi_{13})\cos^2(2\theta) +
\end{aligned}
$$

$$
\frac{((c_{11}c_{12}d_{11} - c_{11}^2 d_{12})d_{33}\psi_{23} + (c_{11}^2 d_{11} - c_{11}c_{12}d_{12})d_{33}\psi_{13})\cos(2\theta)]}{((b_{33}c_{12}^2 - b_{33}c_{11}^2)d_{12}^2 + (b_{33}c_{11}^2 - b_{33}c_{12}^2)d_{11}^2)d_{33}}
$$

$$
\begin{aligned}
f_{12}(\theta) = [&(b_{11}c_{12}c_{33}d_{12}^2 - b_{11}c_{12}c_{33}d_{11}^2)\psi_{33}\sin^2(2\theta) + (((b_{11}c_{12}c_{33}d_{12}^2 - \\
&b_{11}c_{12}c_{33}d_{11}^2)\psi_{32} + (b_{11}c_{11}c_{12}d_{11} - b_{11}c_{12}^2 d_{12})d_{33}\psi_{23} + (b_{11}c_{12}^2 d_{11} - \\
&b_{11}c_{11}c_{12}d_{12})d_{33}\psi_{13})\cos(2\theta) + (b_{12}c_{12}c_{33}d_{11}^2 - b_{12}c_{12}c_{33}d_{12}^2)\psi_{31} + \\
&(b_{11}c_{11}^2 d_{12} - b_{11}c_{11}c_{12}d_{11})d_{33}\psi_{23} + (b_{11}c_{11}c_{12}d_{12} - \\
&b_{11}c_{11}^2 d_{11})d_{33}\psi_{13})\sin(2\theta) + ((b_{11}c_{11}c_{12}d_{11} - b_{11}c_{12}^2 d_{12})d_{33}\psi_{22} + \\
&(b_{11}c_{12}^2 d_{11} - b_{11}c_{11}c_{12}d_{12})d_{33}\psi_{12})\cos^2(2\theta) + ((b_{11}c_{11}^2 d_{12} - \\
&b_{11}c_{11}c_{12}d_{11})d_{33}\psi_{22} + (b_{12}c_{12}^2 d_{12} - b_{12}c_{11}c_{12}d_{11})d_{33}\psi_{21} + (b_{11}c_{11}c_{12}d_{12} - \\
&b_{11}c_{11}^2 d_{11})d_{33}\psi_{12} + (b_{12}c_{11}c_{12}d_{12} - b_{12}c_{12}^2 d_{11})d_{33}\psi_{11})\cos(2\theta) +
\end{aligned}
$$

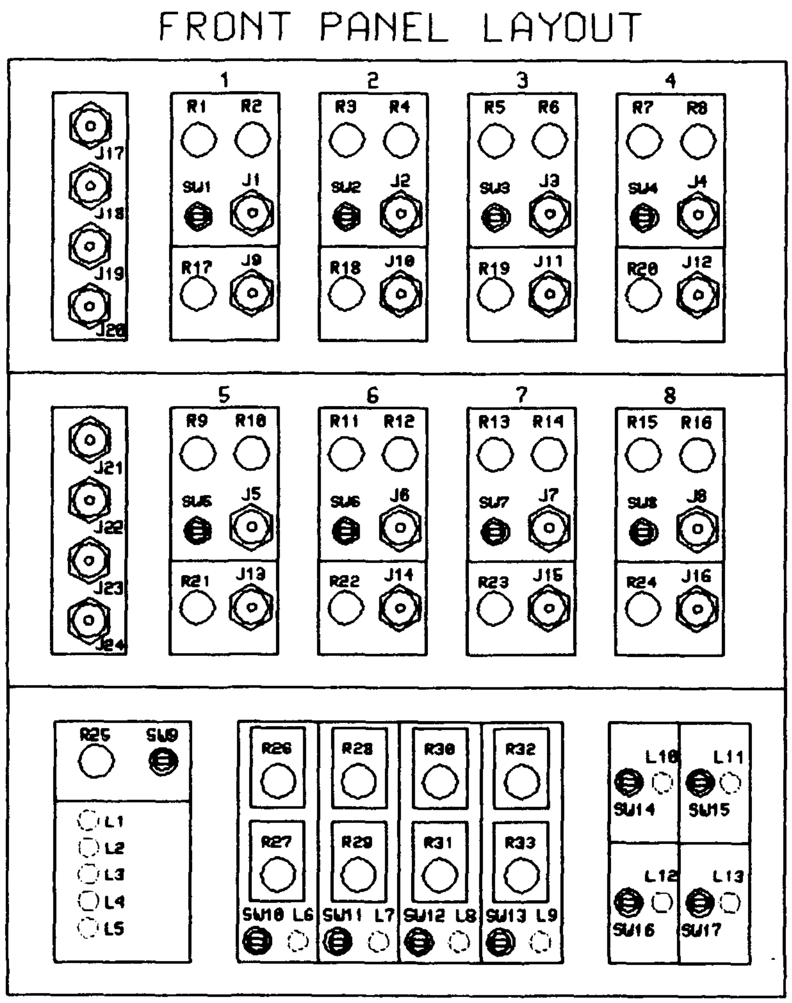
$$
\frac{(b_{12}c_{11}c_{12}d_{11} - b_{12}c_{11}^2 d_{12})d_{33}\psi_{21} + (b_{12}c_{11}^2 d_{11} - b_{12}c_{11}c_{12}d_{12})d_{33}\psi_{11}]}{(((b_{12}^2 - b_{11}^2)c_{12}^2 + (b_{11}^2 - b_{12}^2)c_{11}^2)d_{12}^2 + ((b_{11}^2 - b_{12}^2)c_{12}^2 + (b_{12}^2 - b_{11}^2)c_{11}^2)d_{11}^2)d_{33}}
$$

$$
f_{11}(\theta)=-\frac{\begin{gathered}[(b_{12}c_{12}c_{33}d_{12}^{2}-b_{12}c_{12}c_{33}d_{11}^{2})\psi_{33}\sin^{2}(2\theta)+(((b_{12}c_{12}c_{33}d_{12}^{2}-\\
b_{12}c_{12}c_{33}d_{11}^{2})\psi_{32}+(b_{12}c_{11}c_{12}d_{11}-b_{12}c_{12}^{2}d_{12})d_{33}\psi_{23}+(b_{12}c_{12}^{2}d_{11}-\\
b_{12}c_{11}c_{12}d_{12})d_{33}\psi_{13})\cos(2\theta)+(b_{11}c_{12}c_{33}d_{11}^{2}-b_{11}c_{12}c_{33}d_{12}^{2})\psi_{31}+\\
(b_{12}c_{11}^{2}d_{12}-b_{12}c_{11}c_{12}d_{11})d_{33}\psi_{23}+(b_{12}c_{11}c_{12}d_{12}-\\
b_{12}c_{11}^{2}d_{11})d_{33}\psi_{13})\sin(2\theta)+((b_{12}c_{11}c_{12}d_{11}-b_{12}c_{12}^{2}d_{12})d_{33}\psi_{22}+(b_{12}c_{12}^{2}d_{11}-\\
b_{12}c_{11}c_{12}d_{12})d_{33}\psi_{12})\cos^{2}(2\theta)+((b_{12}c_{11}^{2}d_{12}-b_{12}c_{11}c_{12}d_{11})d_{33}\psi_{22}+(b_{11}c_{12}^{2}d_{12}-\\
b_{11}c_{11}c_{12}d_{11})d_{33}\psi_{21}+(b_{12}c_{11}c_{12}d_{12}-b_{12}c_{11}^{2}d_{11})d_{33}\psi_{12}+(b_{11}c_{11}c_{12}d_{12}-\\
b_{11}c_{12}^{2}d_{11})d_{33}\psi_{11})\cos(2\theta)+(b_{11}c_{11}c_{12}d_{11}-b_{11}c_{11}^{2}d_{12})d_{33}\psi_{21}+(b_{11}c_{11}^{2}d_{11}-\\
b_{11}c_{11}c_{12}d_{12})d_{33}\psi_{11}]\end{gathered}}{(((b_{12}^{2}-b_{11}^{2})c_{12}^{2}+(b_{11}^{2}-b_{12}^{2})c_{11}^{2})d_{12}^{2}+((b_{11}^{2}-b_{12}^{2})c_{12}^{2}+(b_{12}^{2}-b_{11}^{2})c_{11}^{2})d_{11}^{2})d_{33}}
$$

Blank

APPENDIX III: APSD ELECTRONIC CIRCUITS. MODULES I-VIII

The Analog Phase-Sensitive Detector (APSD), designed and fabricated by Dave Owens, is an integrated 8-module electronic system that produces all Mueller elements from the scattergram in a highly automated manner, under the control of a host microVax computer. An experimenter would typically initialize the ellipsometer (select sample, analyte, beam wavelengths, incident backscattering angles) and monitor the automated progress of the APSD by LED readouts on its front control panel. The APSD recognizes the optical configurations (Tables 3-4), acquires the scattergram (Equations 12, 14b, 16b, and 18b), and makes the Mueller element mappings that are digitized, graphed, and stored on disk. We provide in this appendix the major APSD circuits now operating in this ellipsometer system.



CORRESPONDING FREQUENCIES

1: 31.896 khz
2: 33.96 khz
3: 63.79 khz
4: 67.92 khz
5: 4.13 khz
6: 65.86 khz
7: 97.75 khz
8: 99.82 khz

Figure AIII.1. The front panel layout of the Analog Phase-Sensitive Detector (APSD). Eight PSD channels are represented (1-8). Connectors J17-J20 and J21-J24 are inputs/outputs of the primary modulator frequencies $omeg_1$, ω_2, $2\omega_1$, and $2\omega_2$. R1-R16 are the course (odd) and fine (even) reference phase shift controllers, while SW1-SW8 are the phase inverters, per channel. Connectors J1-J8 are reference frequency outputs, while J9-J12 and J13-J16 are the PSD (Mueller elements) output channels. R17-R24 are the PSD amplitude adjustments, while R26-R33 are retardation adjusters to transmitter and receiver PEM's and switches SW10-SW13 allow manual or remote retardation control. LED's L1-L5 are status indicators for the incident beams power regulation circuit, while L10,SW14 - L13,SW17 show operational status of the shutter controllers and allow for manual or remote switching between beams.

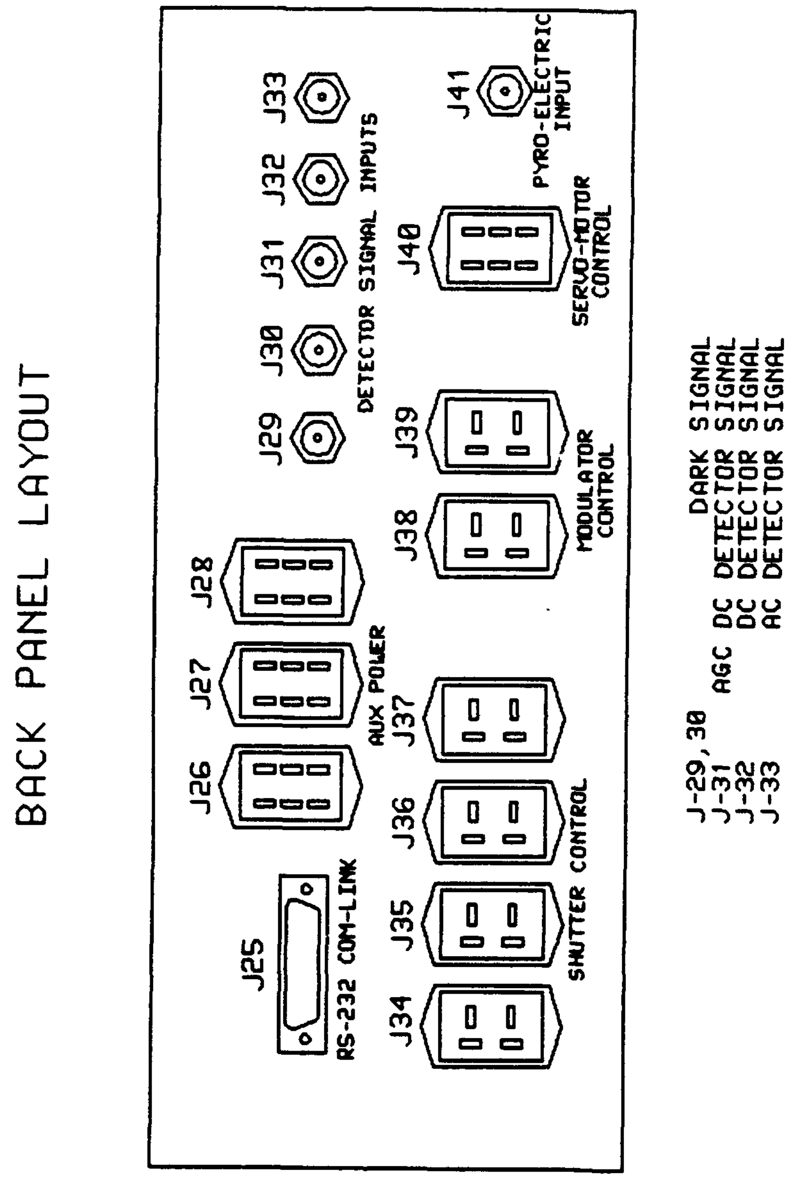


Figure AIII.2. The APSD back panel layout. J25 is the serial communications link to and from the host CPU and stepper motor controllers, A/D converter, modulator and servo-motor controls, shutter and pyro-electric detector devices.

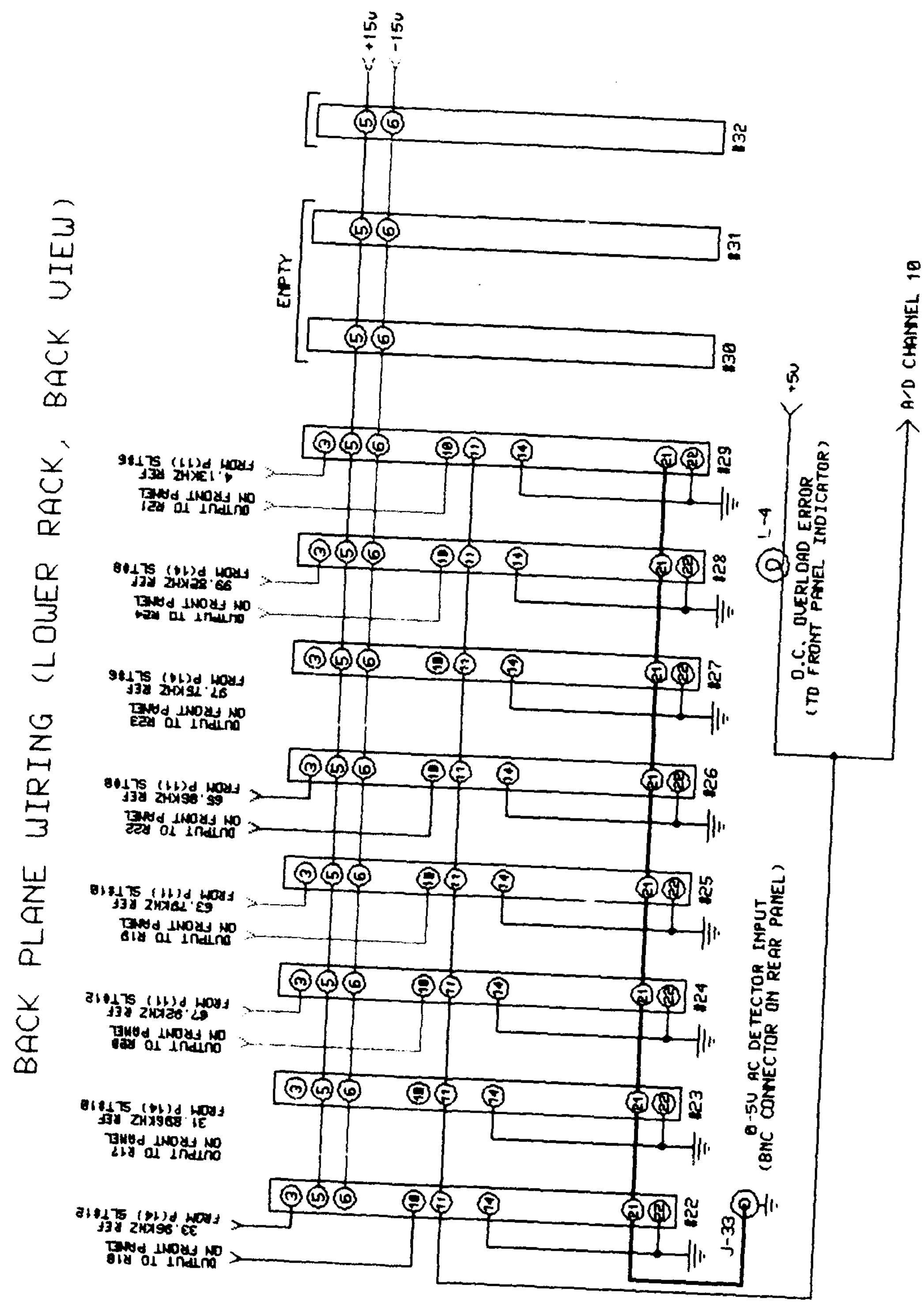


Figure AIII.3. The APSD back panel wiring layout, lower rack.

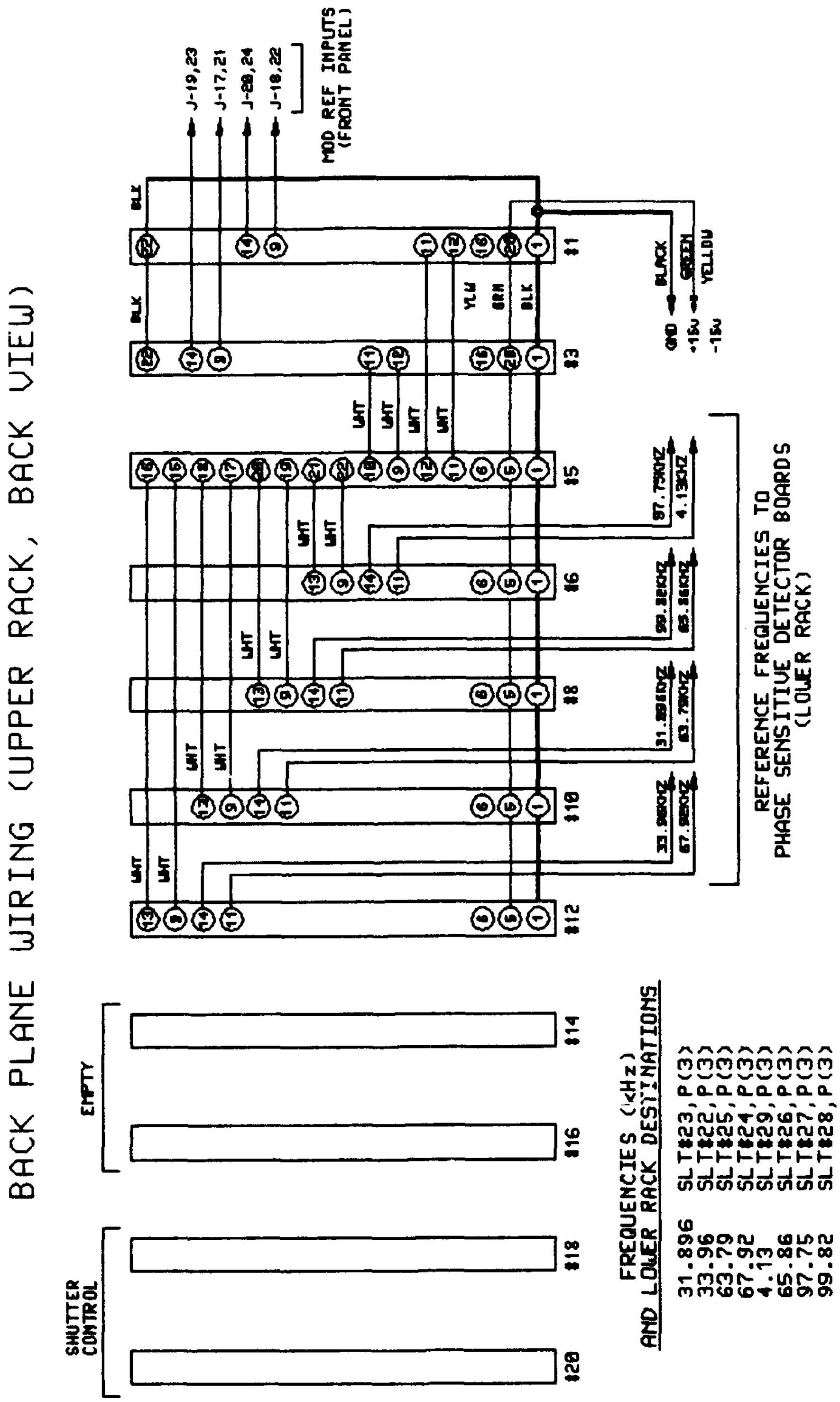


Figure AIII.4. The APSD back panel wiring layout, upper rack.

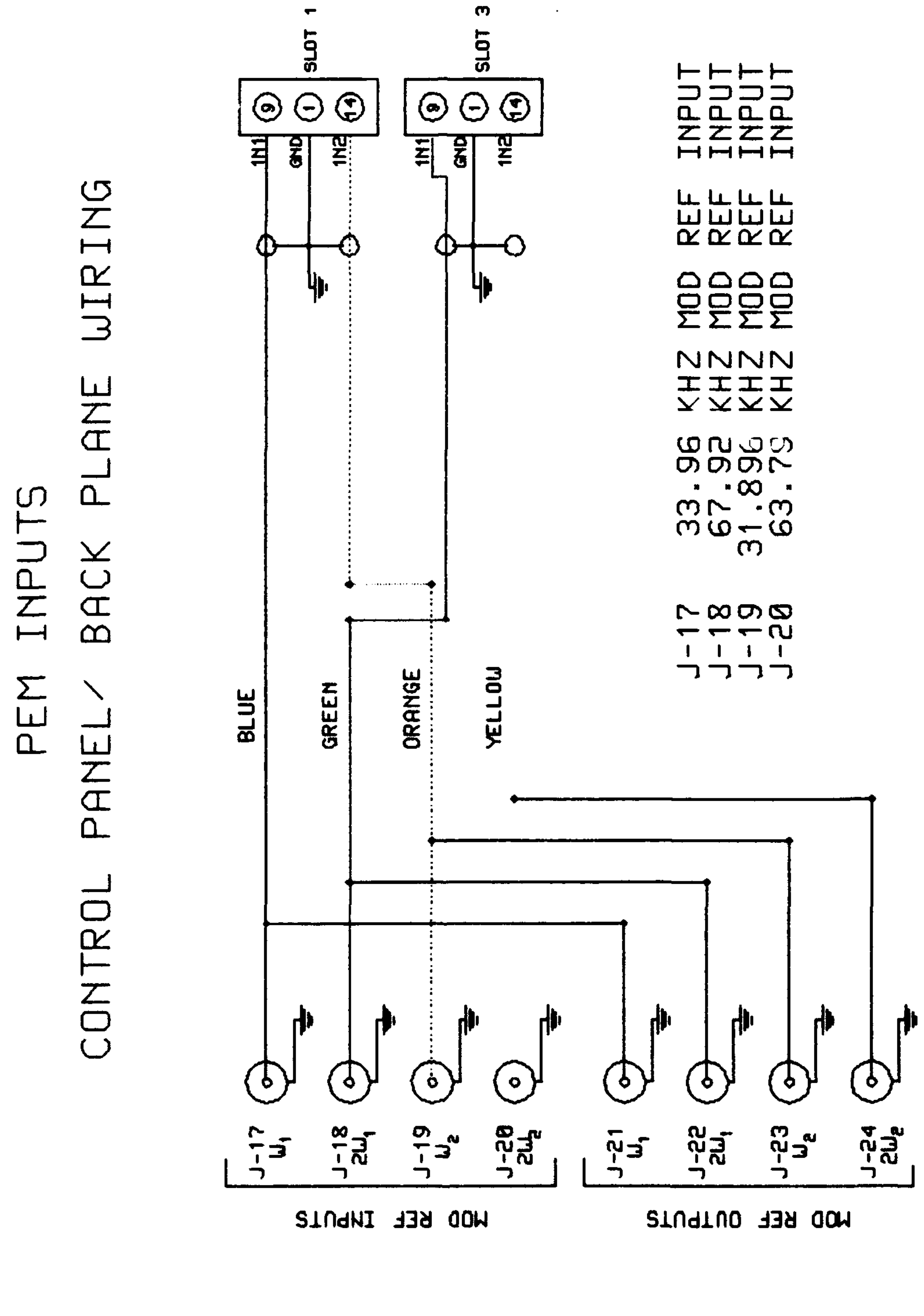


Figure AIII.5. The APSD control to back plane wiring harness. (Reference inputs.)

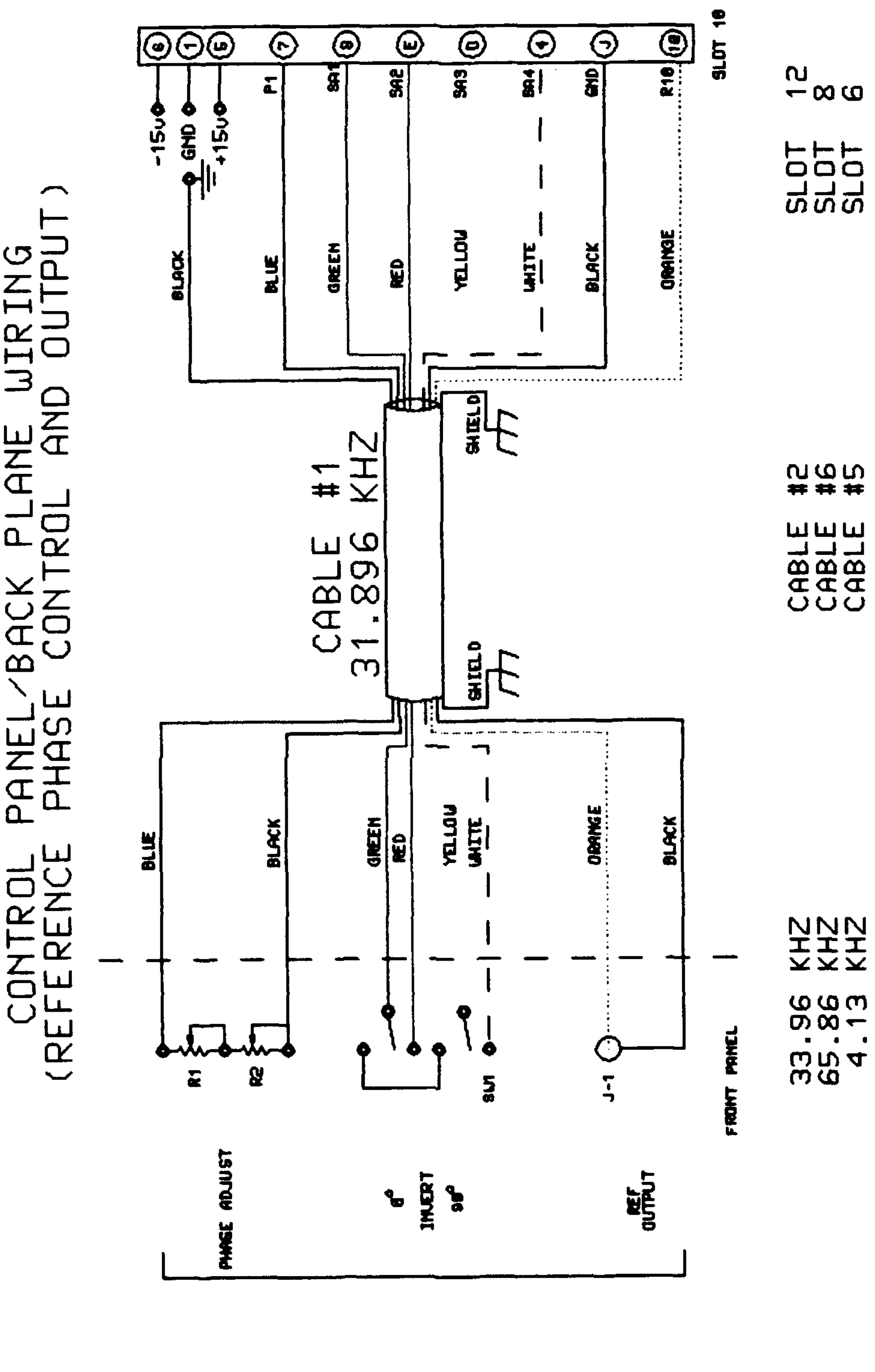


Figure AIII.6a. **The APSD control panel to back plane wiring harness. (Reference phase control and output.)**

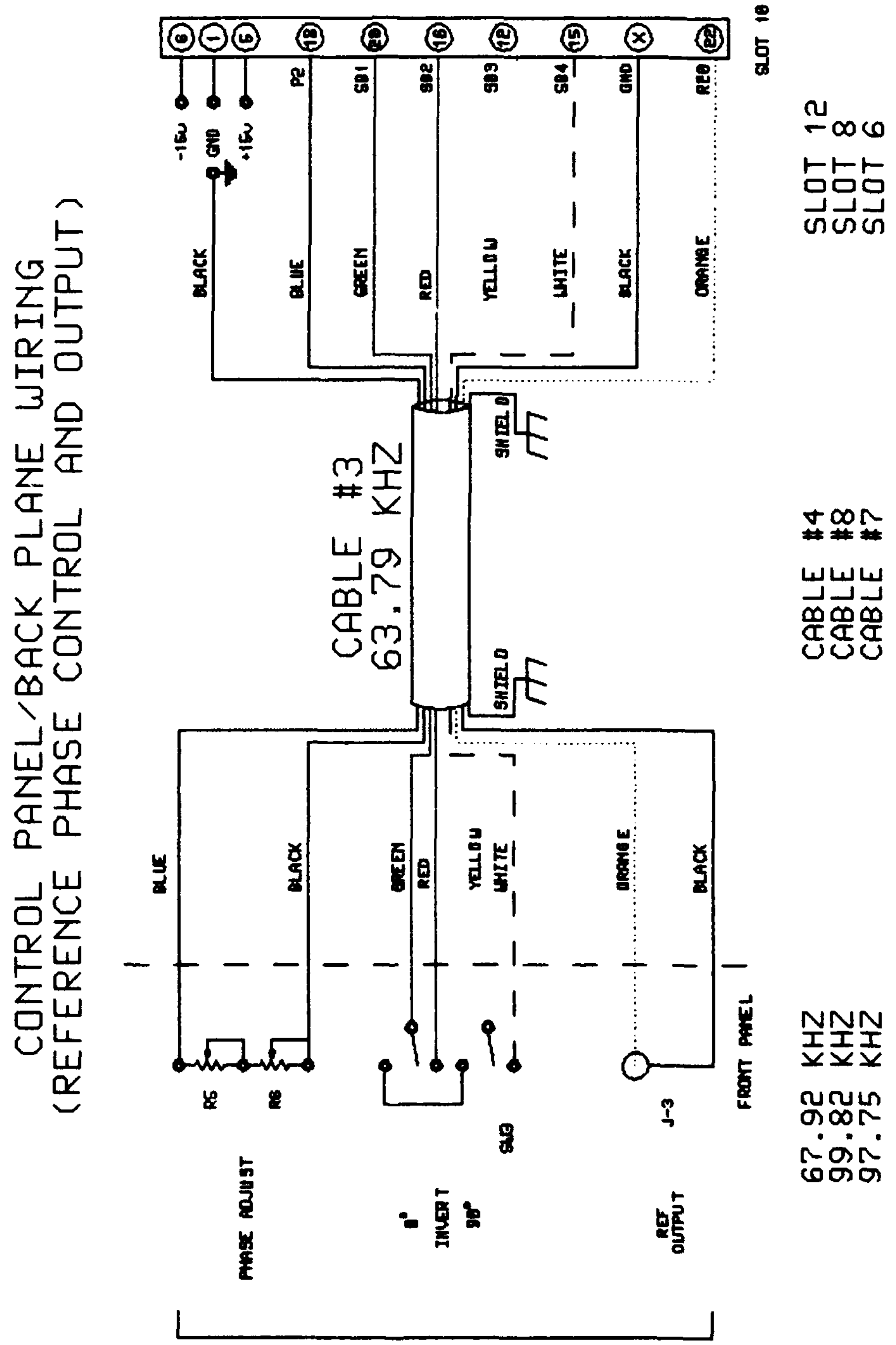


Figure AIII.6b. The APSD control panel to back plane wiring harness. (Reference phase control and output.)

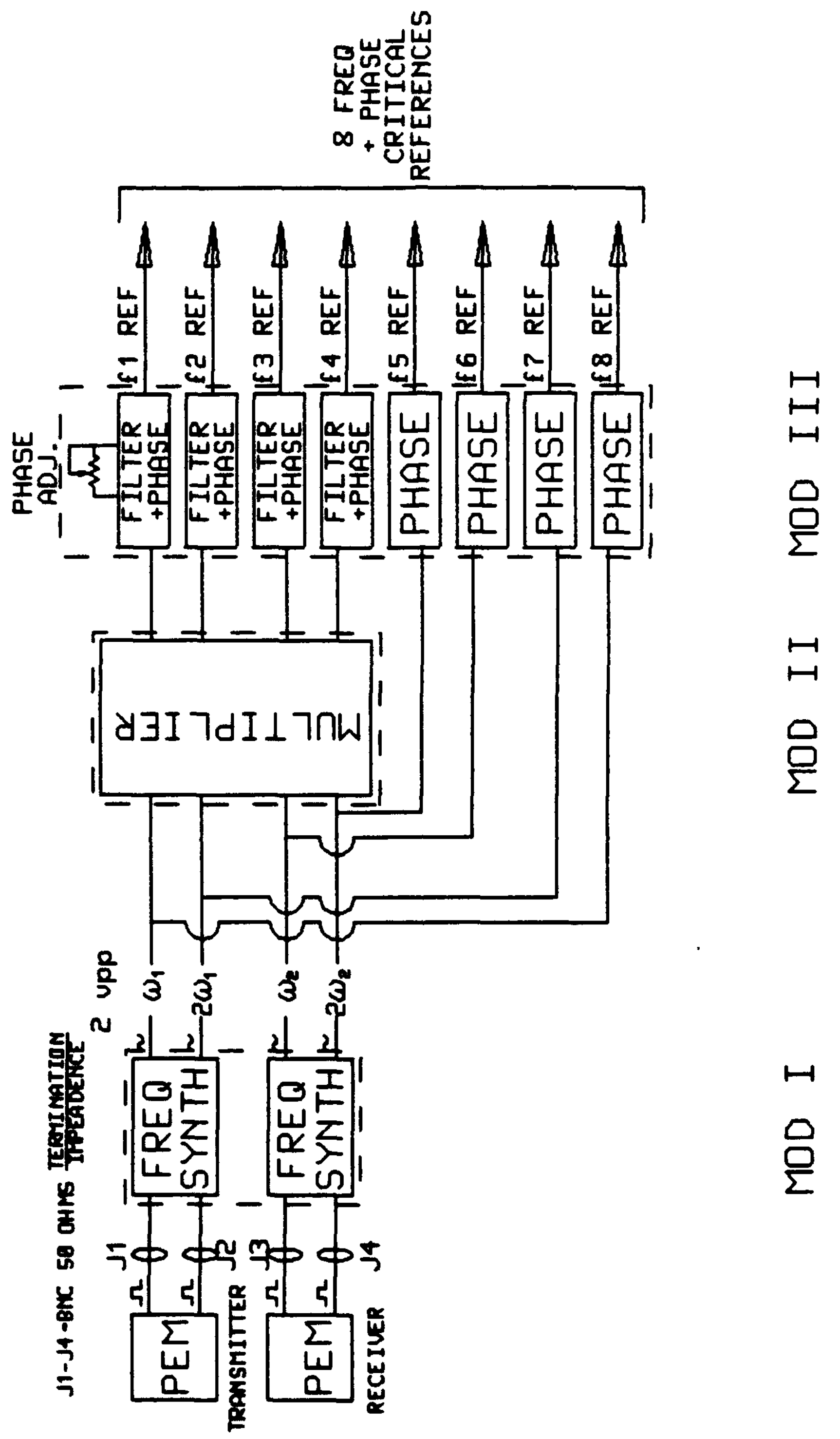


Figure AIII.7a. **Flowchart of PEM modulator, frequency synthesizer and multiplier, and phase adjustment circuits of the APSD (Modules I-III).**

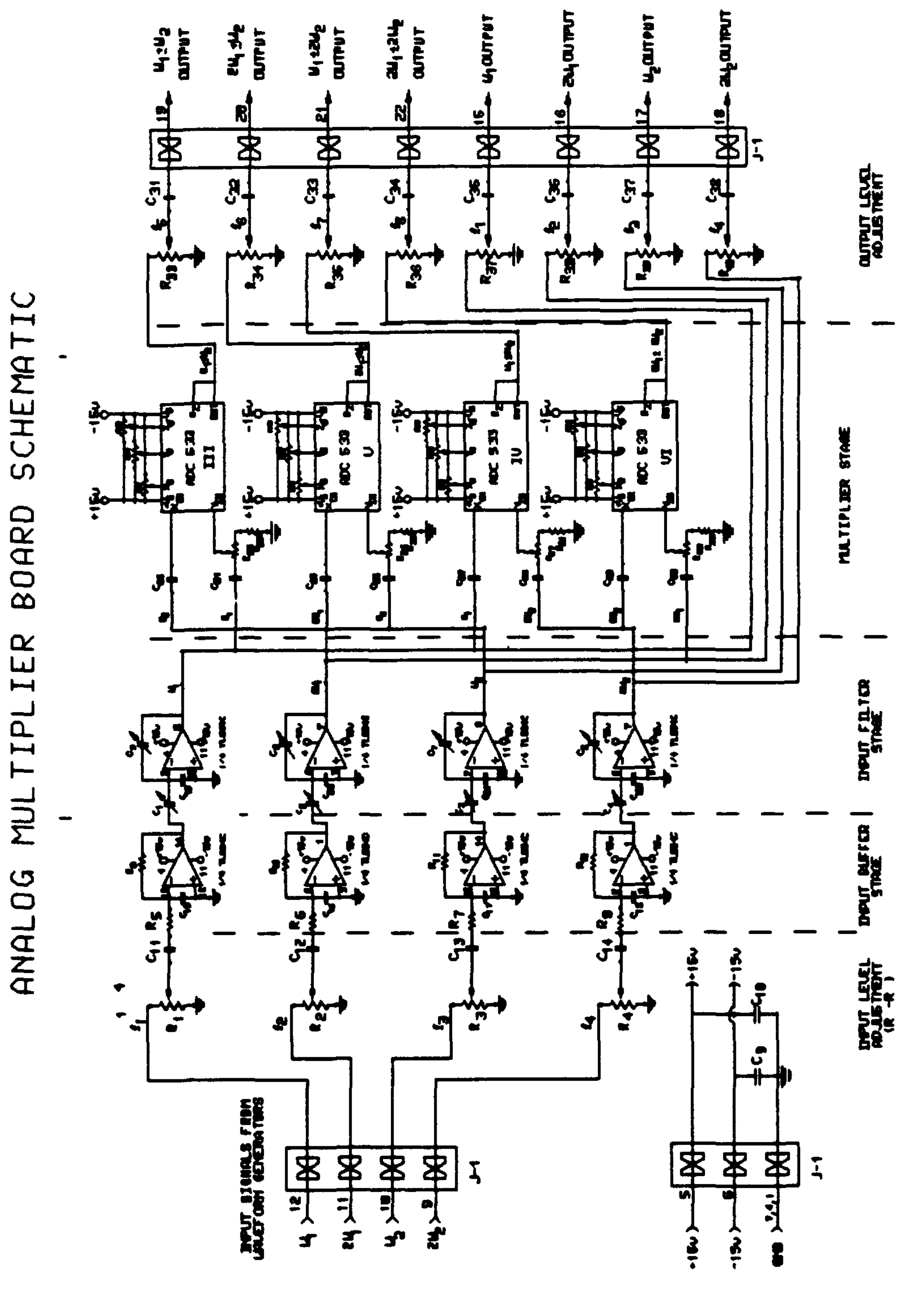


Figure AIII.7b. Schematic of the APSD analog multiplier board (Module II).

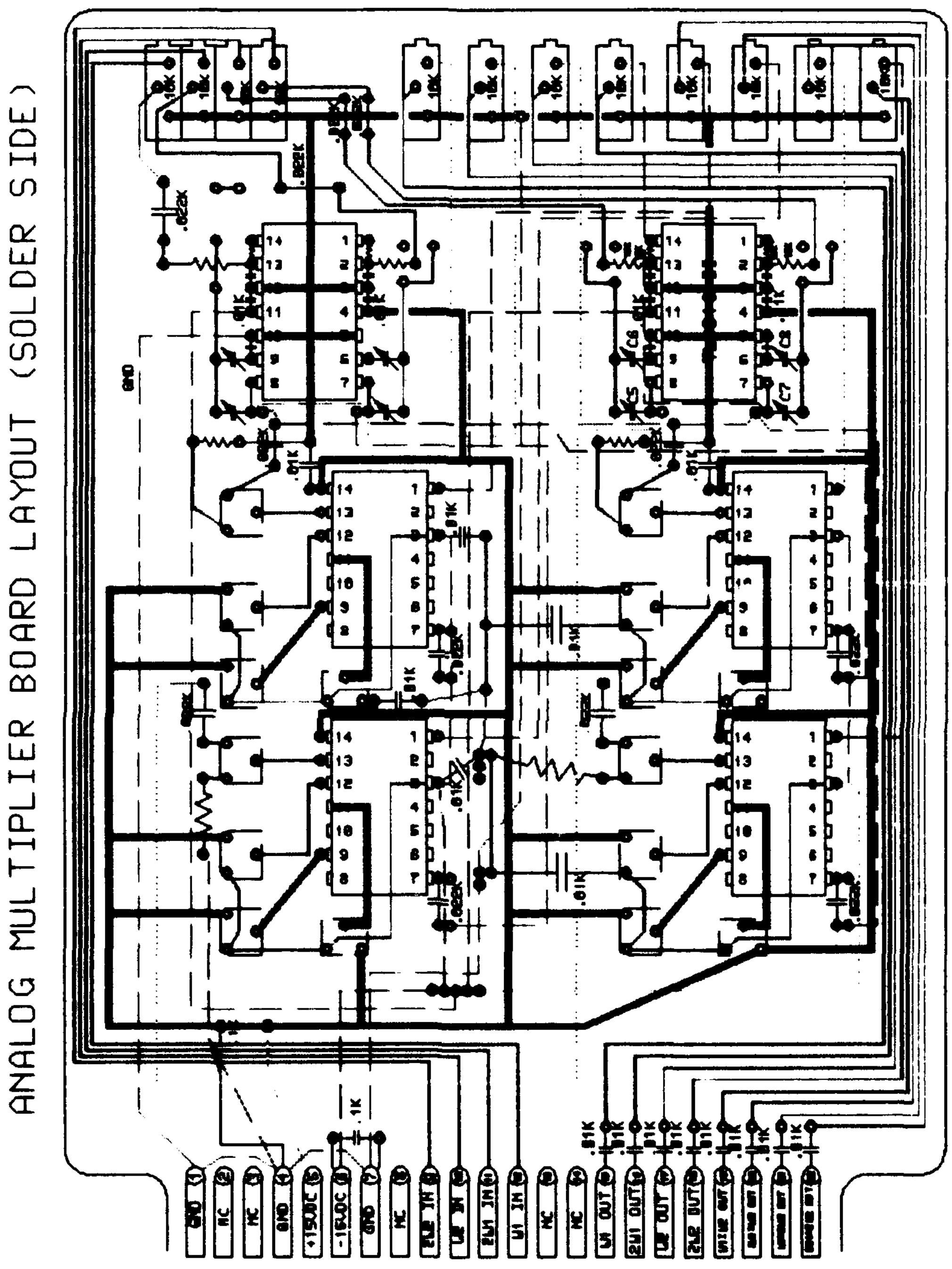


Figure AIII.7b(i). The APSD analog multiplier board layout design (Module II).

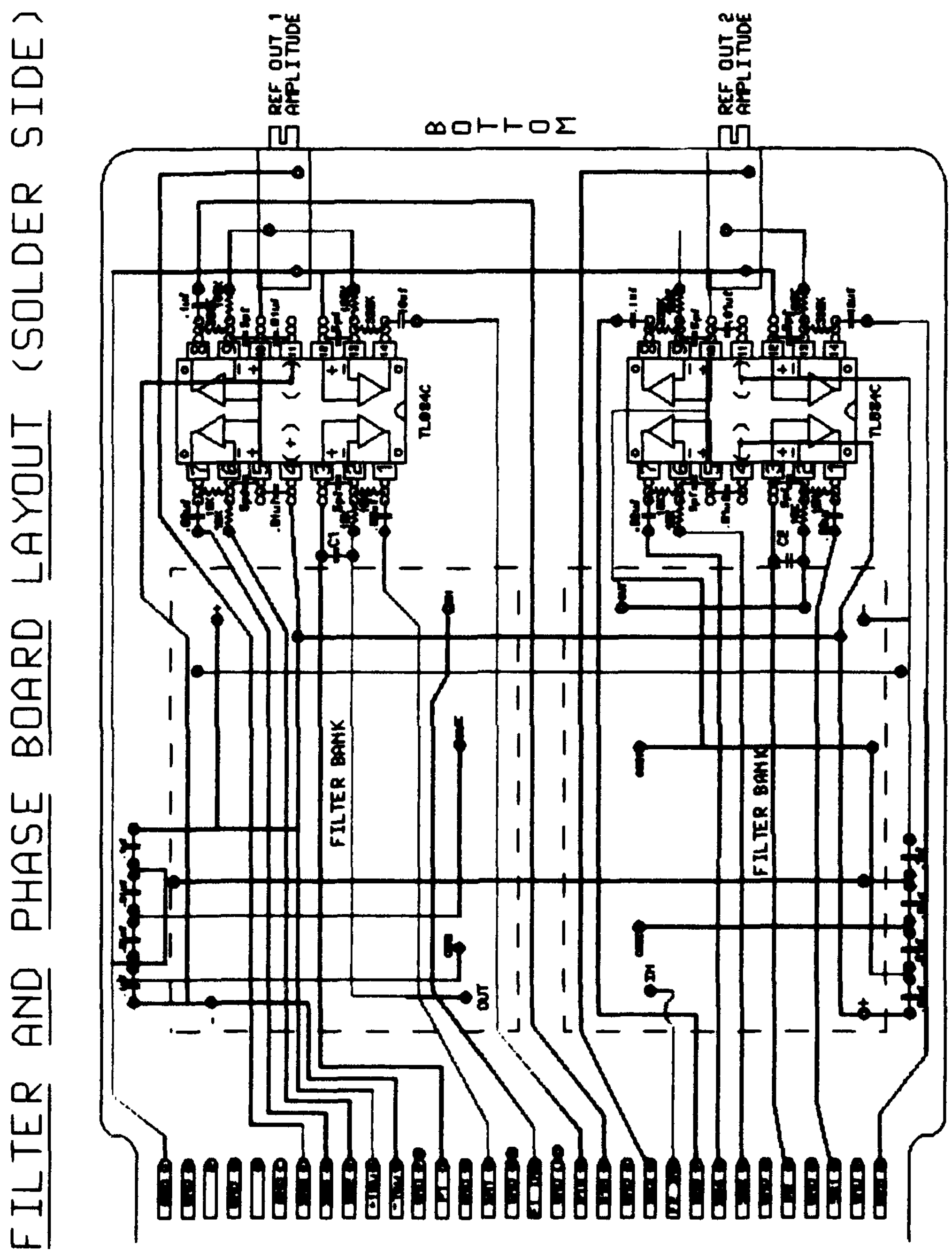


Figure AIII.7c. **The APSD filter and phase board layout design (Module III).**

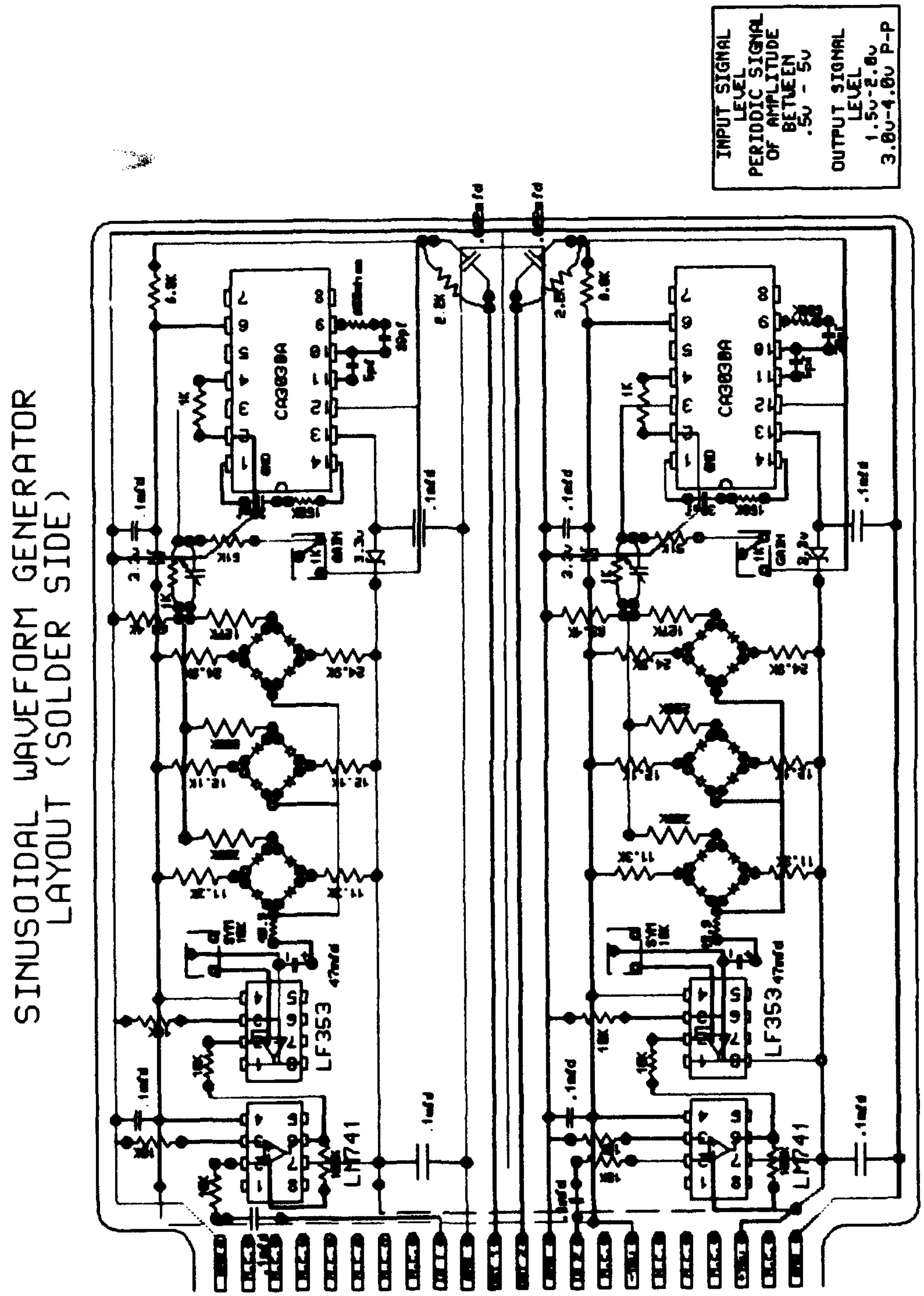


Figure AIII.7d. Schematic of the APSD sinusoid waveform generator (Modules I).

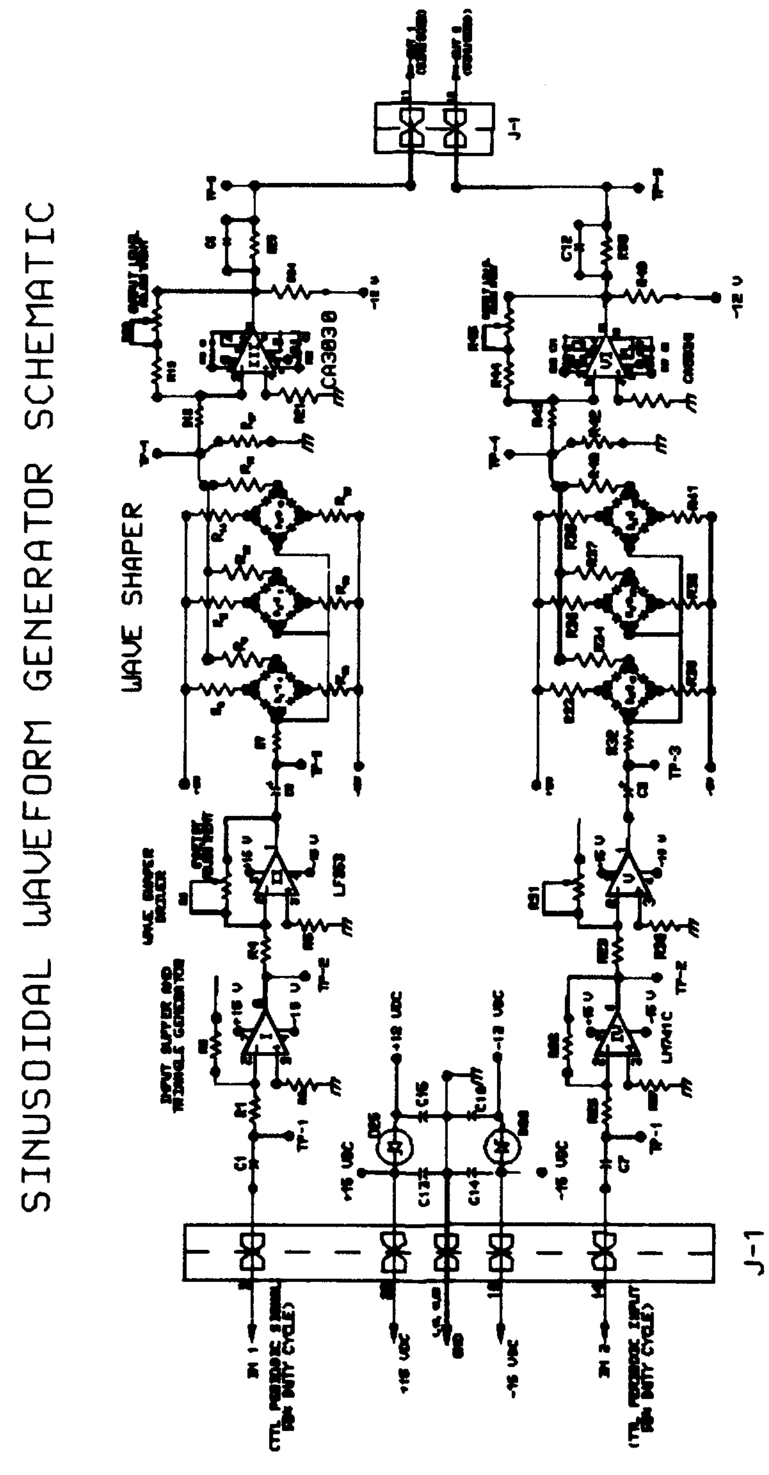


Figure AIII.7e. **The APSD sinusoid waveform generator board layout design (Module I).**

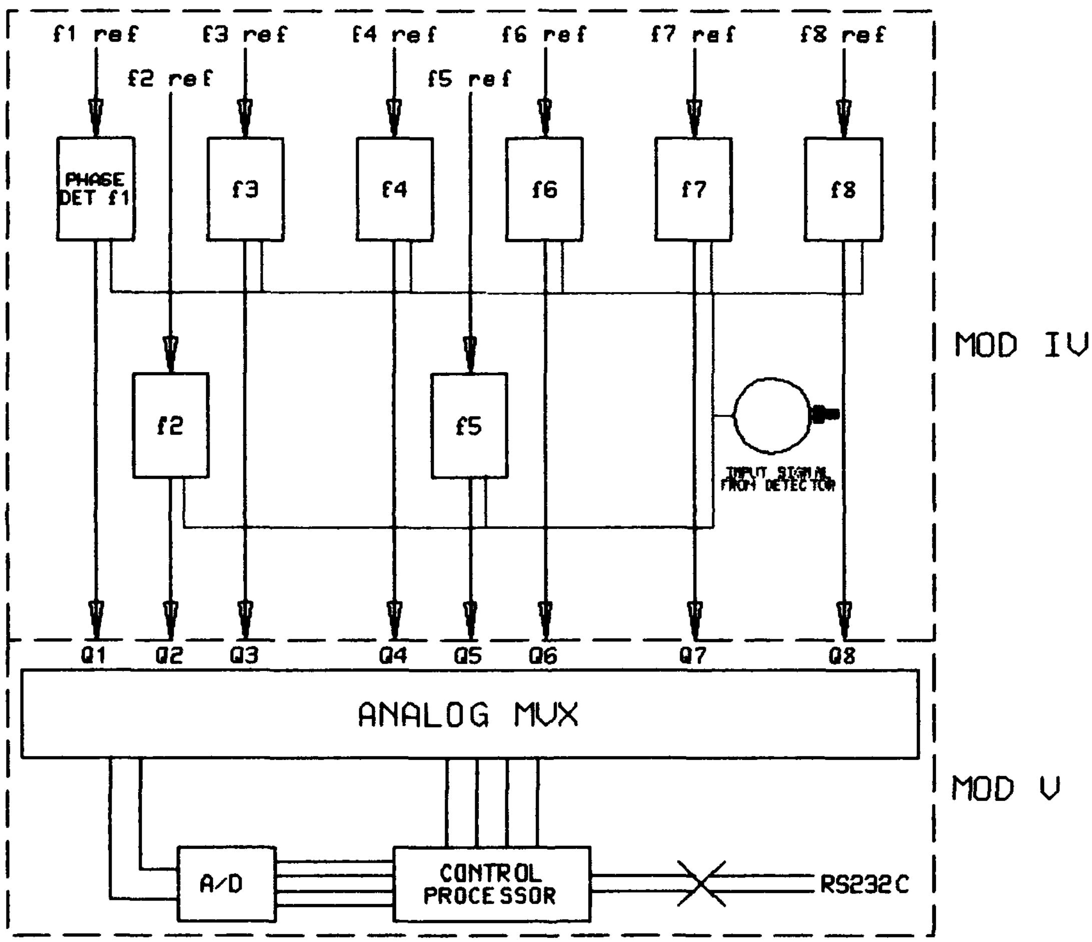


Figure AIII.8. Flowchart of phase references, the Phase Sensitive Detectors, and the DAEDAL ST701 A/D board circuits of the APSD (Modules IV-V).

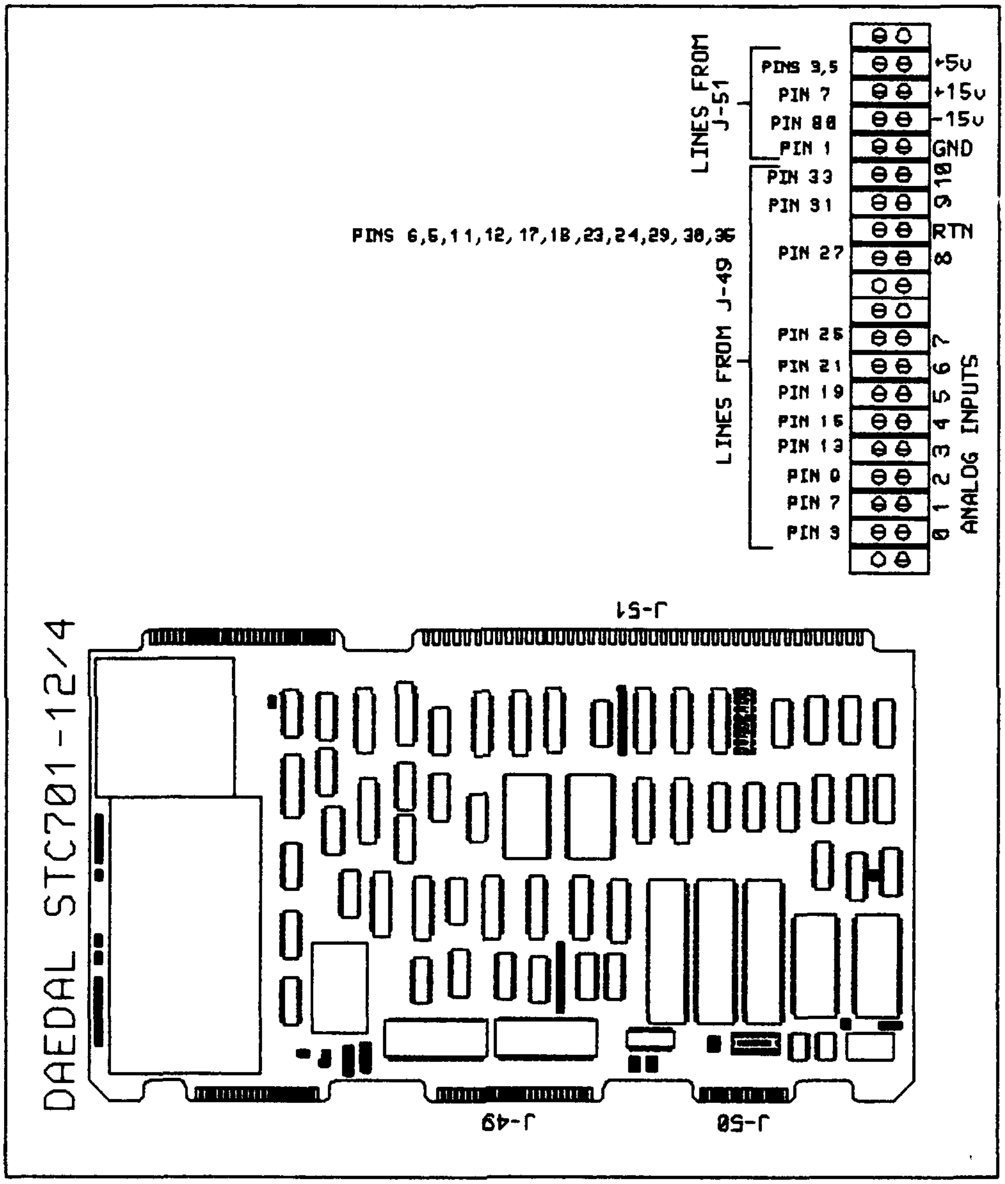


Figure AIII.9a. Module V pin connections to the DAEDAL ST701 A/D microprocessor.

TO TERMINATION BLOCK

TO J-49 (ON A/D)

	J-49 PIN #2
	#1
	#4
CHANNEL 0	#3
RETURN 1	#6
RETURN 0	#5
	#8
CHANNEL 1	#7
	#10
CHANNEL 2	#9
RETURN 3	#12
RETURN 2	#11
	#14
CHANNEL 3	#13
	#16
CHANNEL 4	#15
RETURN 5	#18
RETURN 4	#17
	#20
CHANNEL 5	#19
	#22
CHANNEL 6	#21
RETURN 7	#24
RETURN 6	#23
	#26
CHANNEL 7	#25
	#28
CHANNEL 8	#27
RETURN 9	#30
RETURN 8	#29
	#32
CHANNEL 9	#31
	#34
CHANNEL 10	#33
	#36
RETURN 10	#35
	#38
	#37
	#40
	#39

NOTE: ALL RETURNS ARE TIED TO A COMMON RETURN ON THE TERMINATION BLOCK.

Figure AIII.9b. Module V wiring assignments of J-49 connector to the DAEDAL ST701 A/D microprocessor.

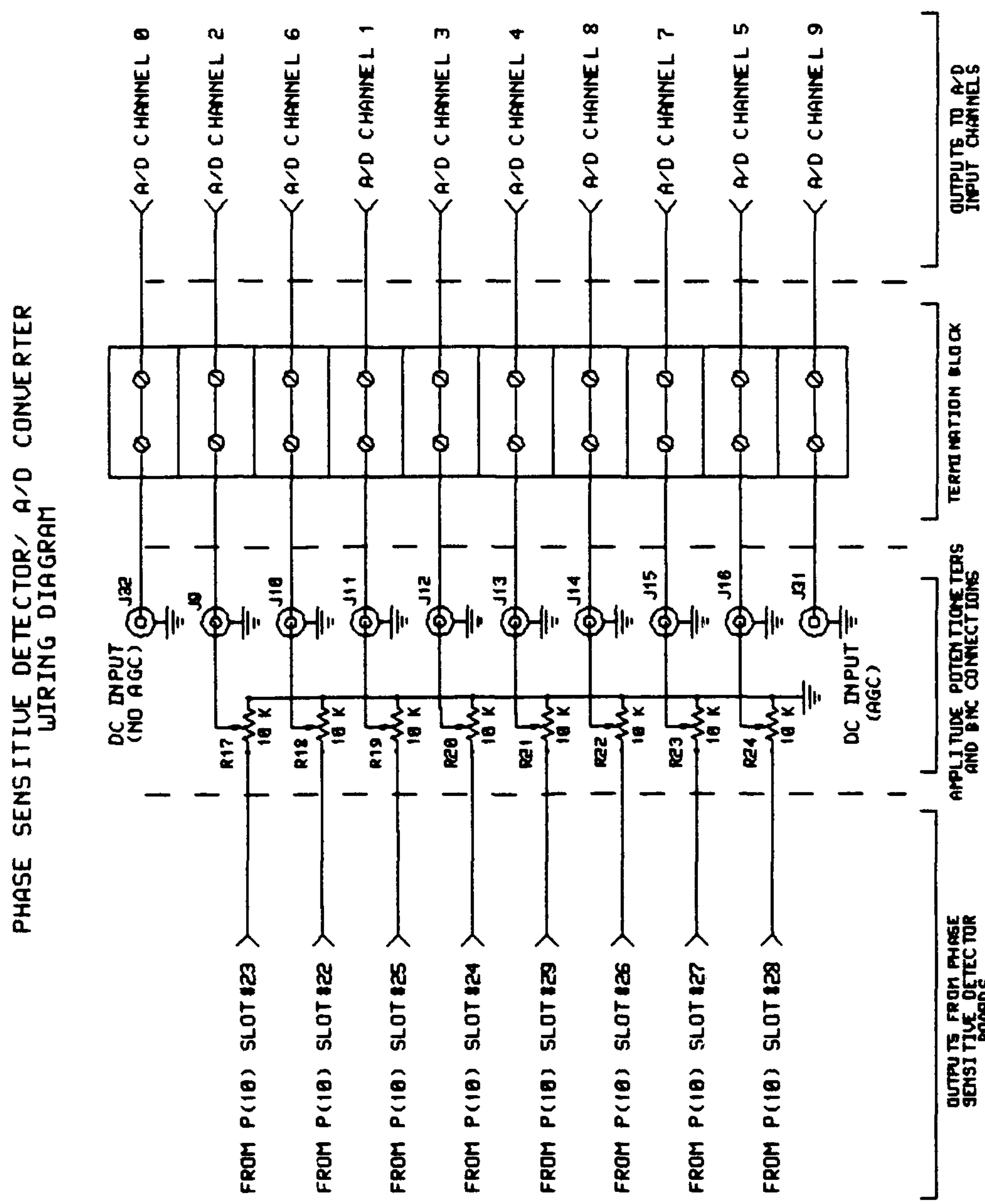


Figure AIII.10. A/D converter (Module V) to Phase Sensitive Detector wiring assignments.

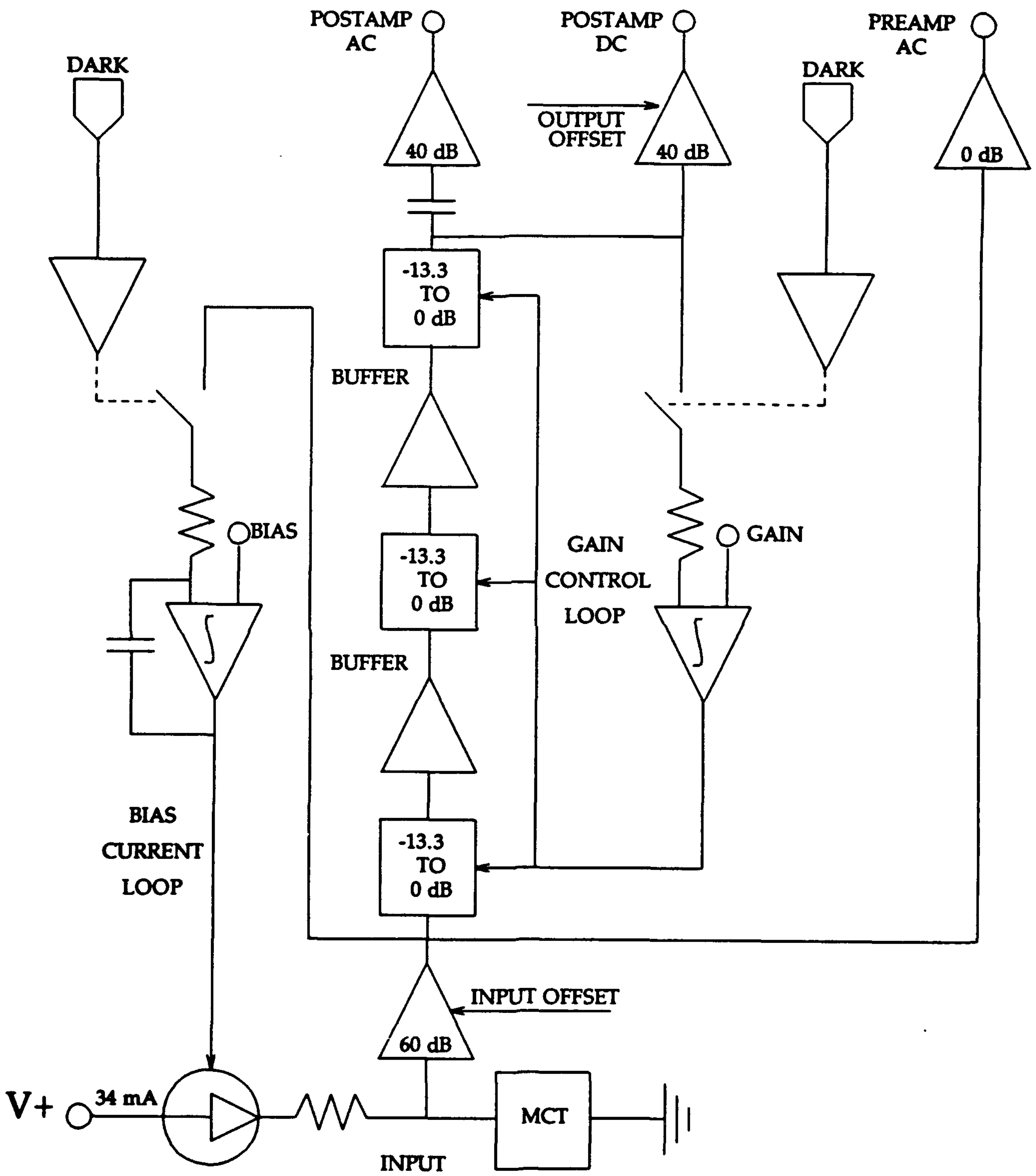


Figure AIII.11. Automatic gain control and amplification circuit (Module VI).

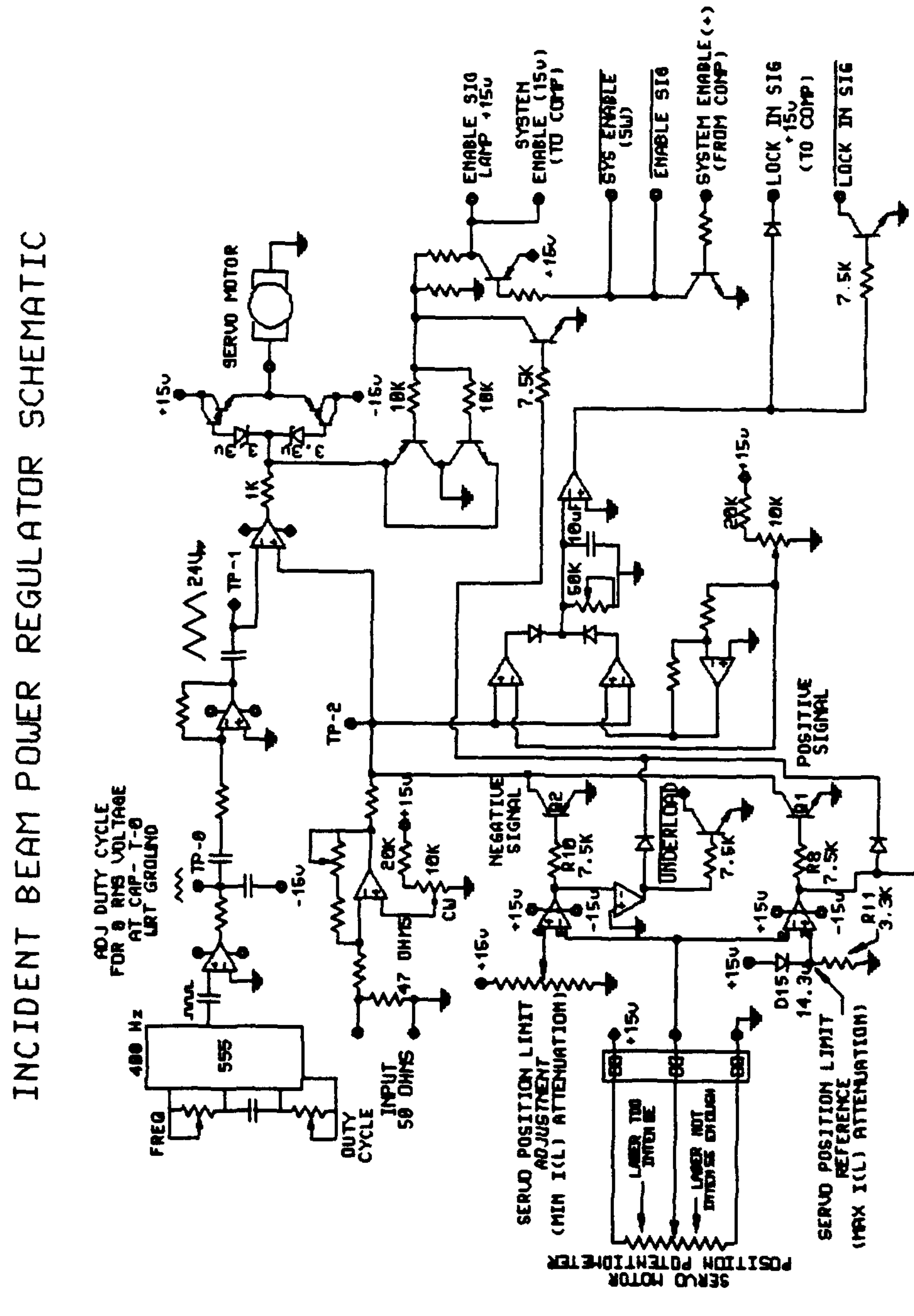


Figure AIII.12a. The incident beam power regulation circuit Module VII. The feedback loop acts between the pyroelectric detector output and the servo motor that adjusts the transmission axis of a polarizing crystal.

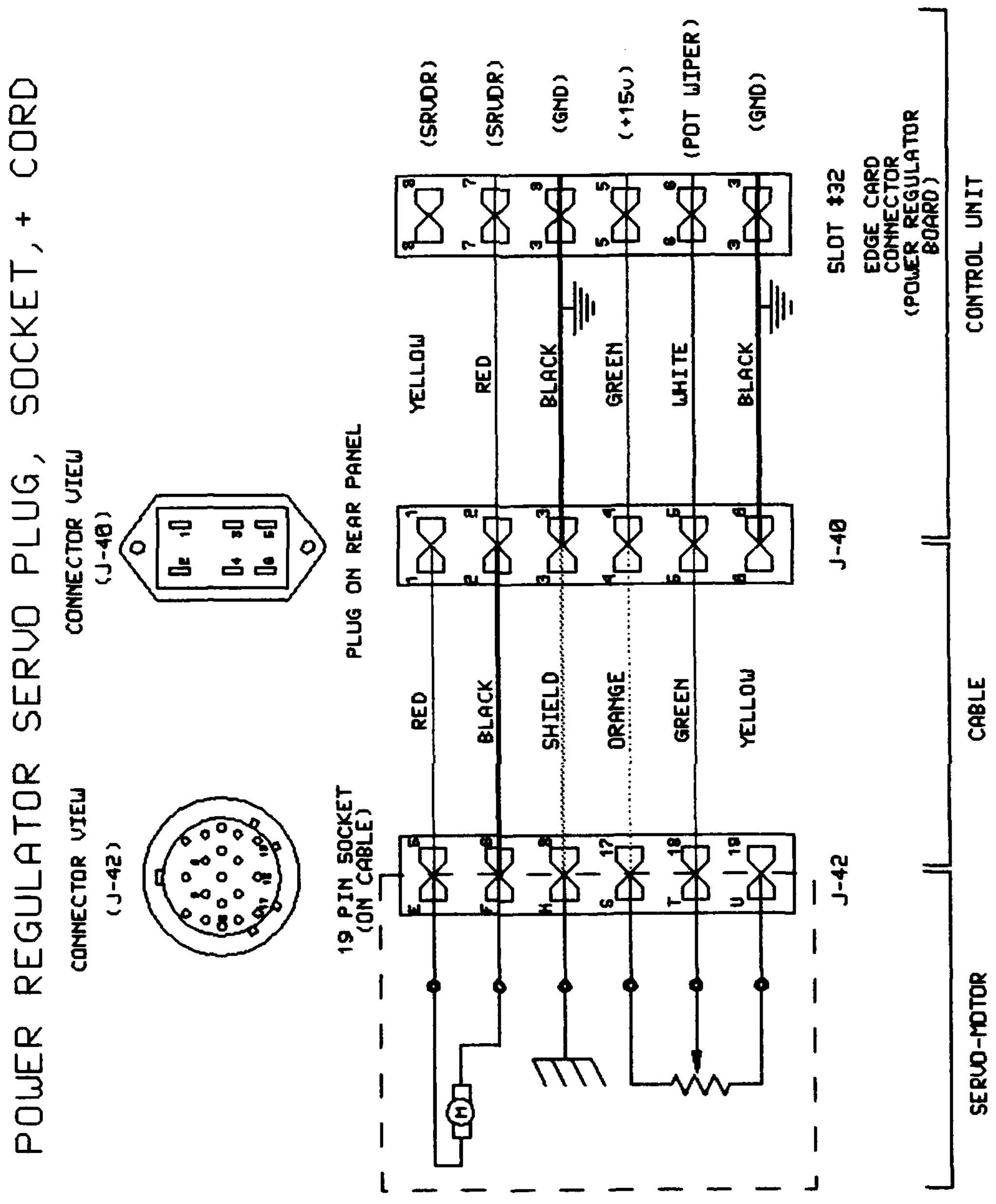


Figure AIII.12b. Layout of the APSD incident beam power regulation servo plug, socket and cord (Module VII).

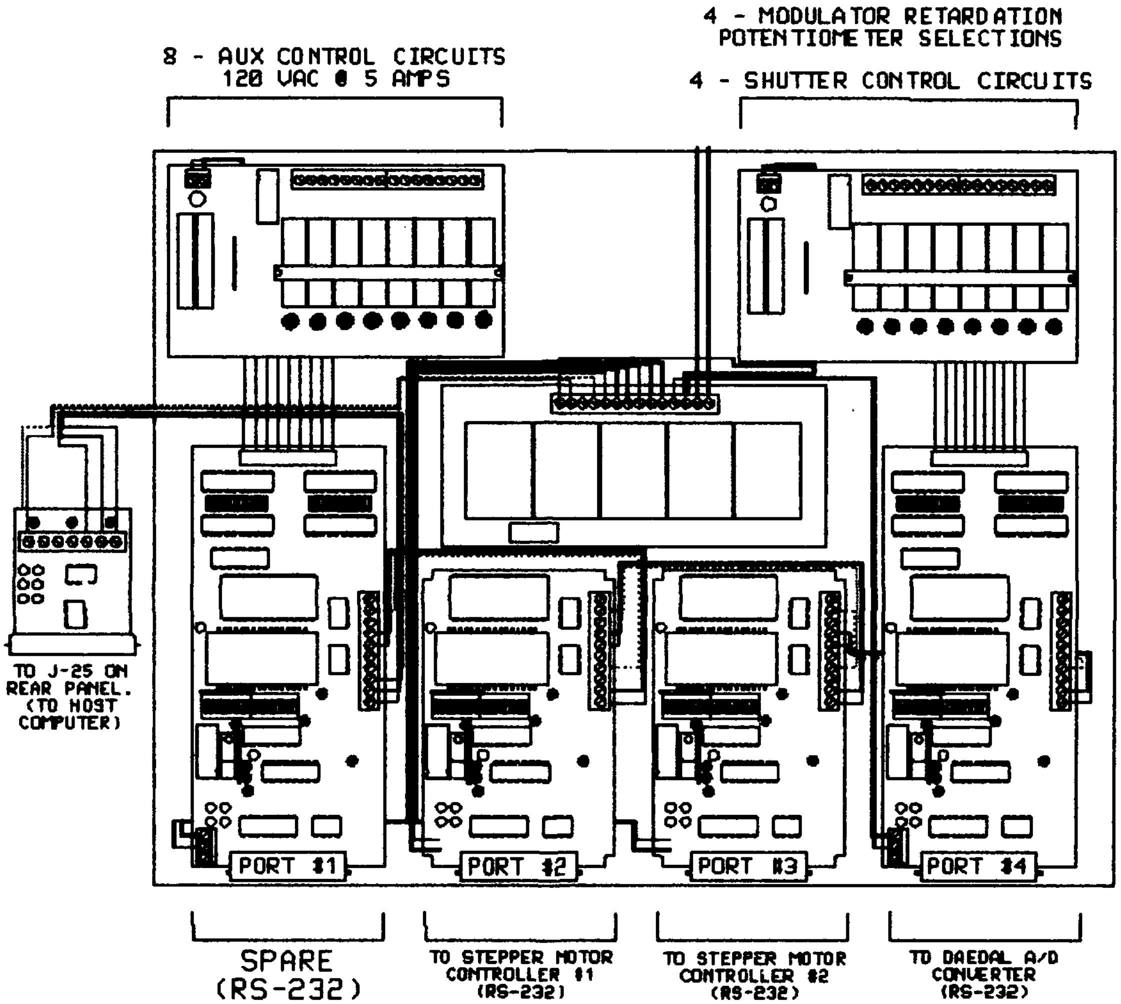


Figure AIII.13a. The APSD Serial Address Gateway (SAG) system (Module VIII).

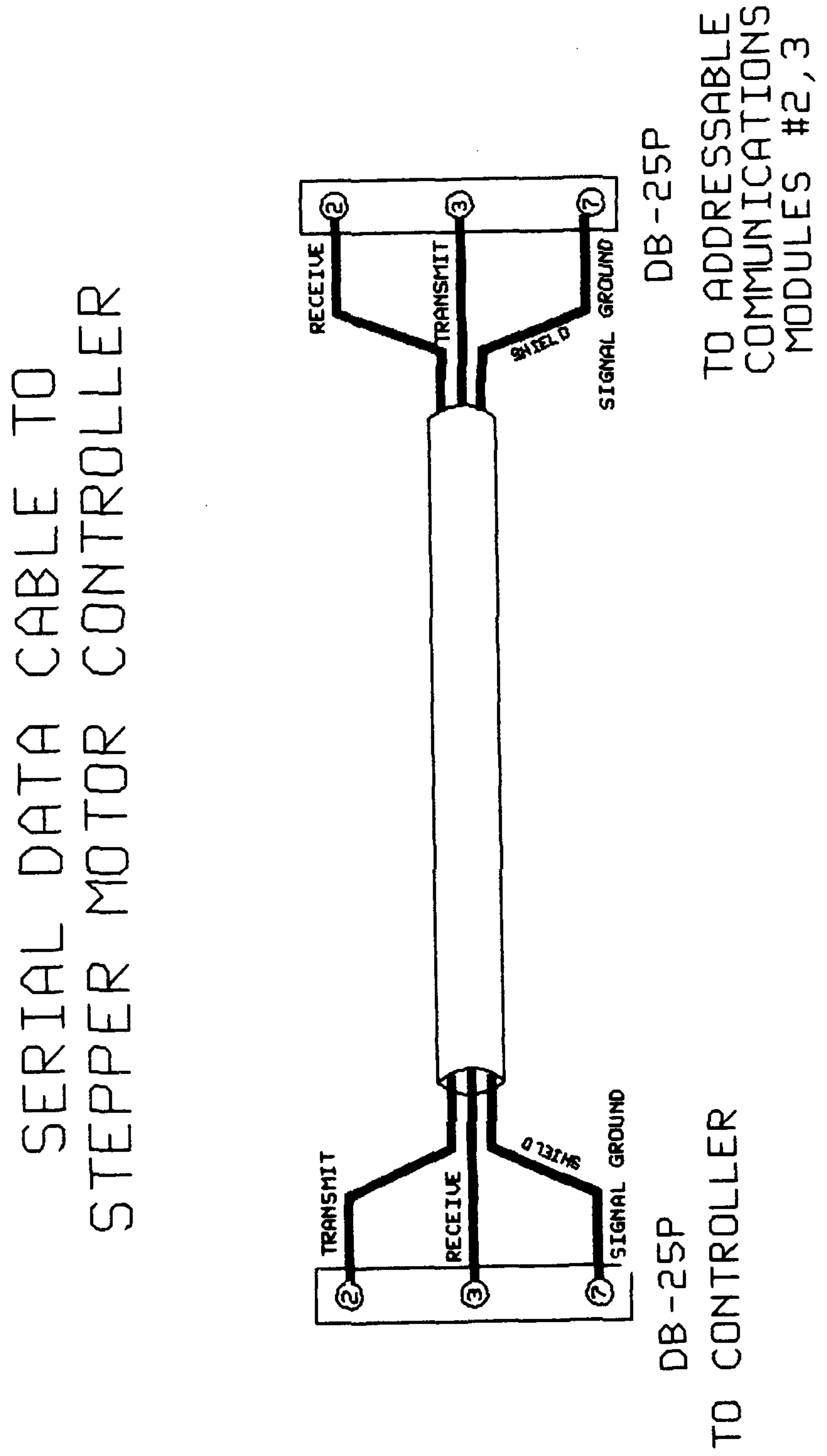


Figure AIII.13b. The serial data cable from the SAG to the stepper motor controller (Module VIII).

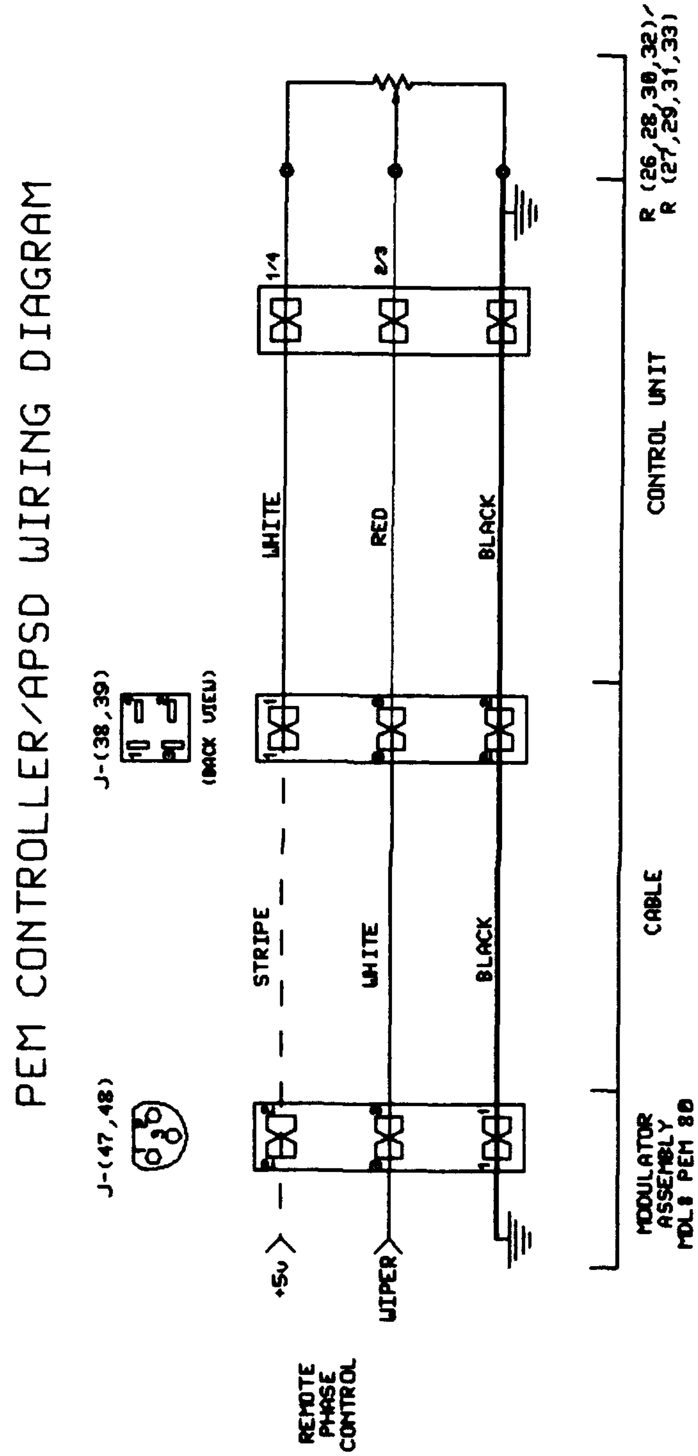


Figure AIII.14. Wiring diagram between transmitter and receiver photoelastic modulators (retardation control) and the SAG.

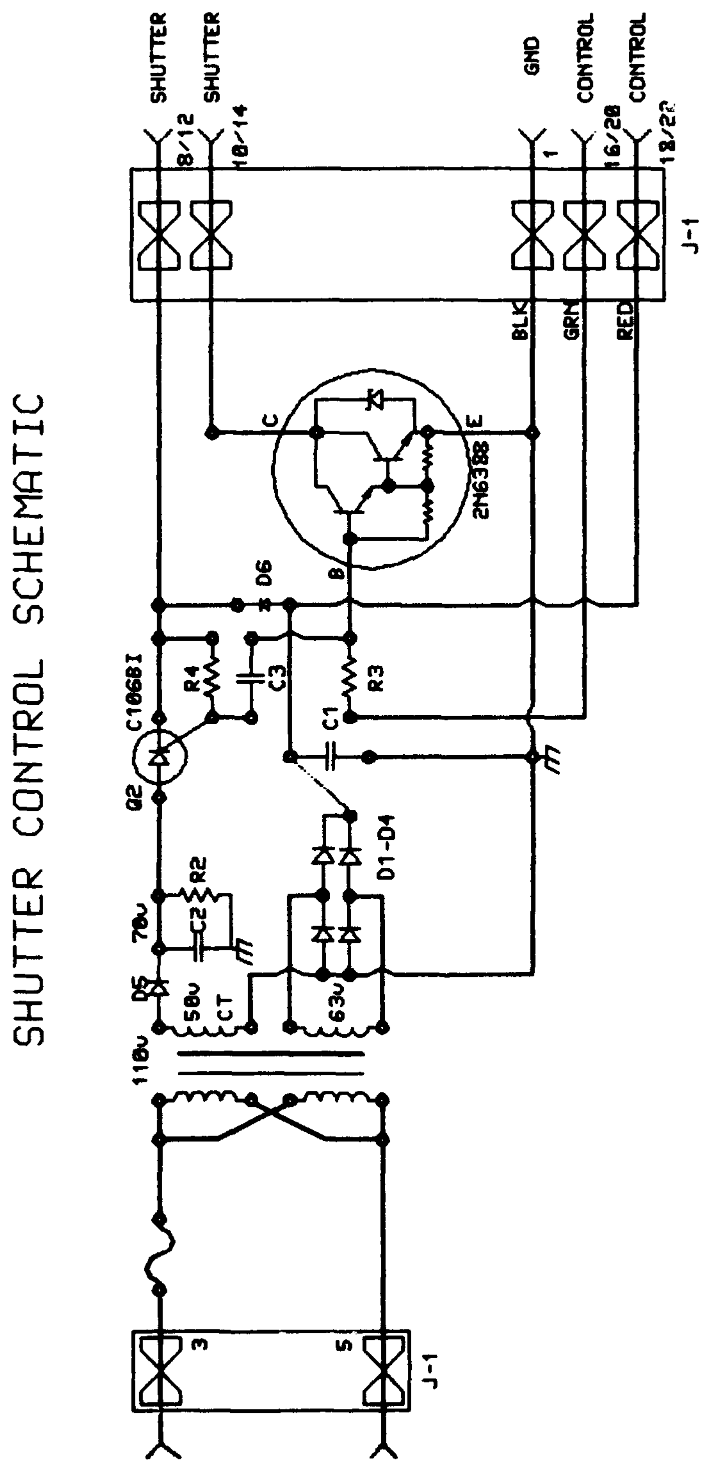


Figure AIII.15a. The APSD shutter control board for switching between incident beams. The shutter drive unit is the model 100-2B UniBlitz design of Vincent Associates, Inc.

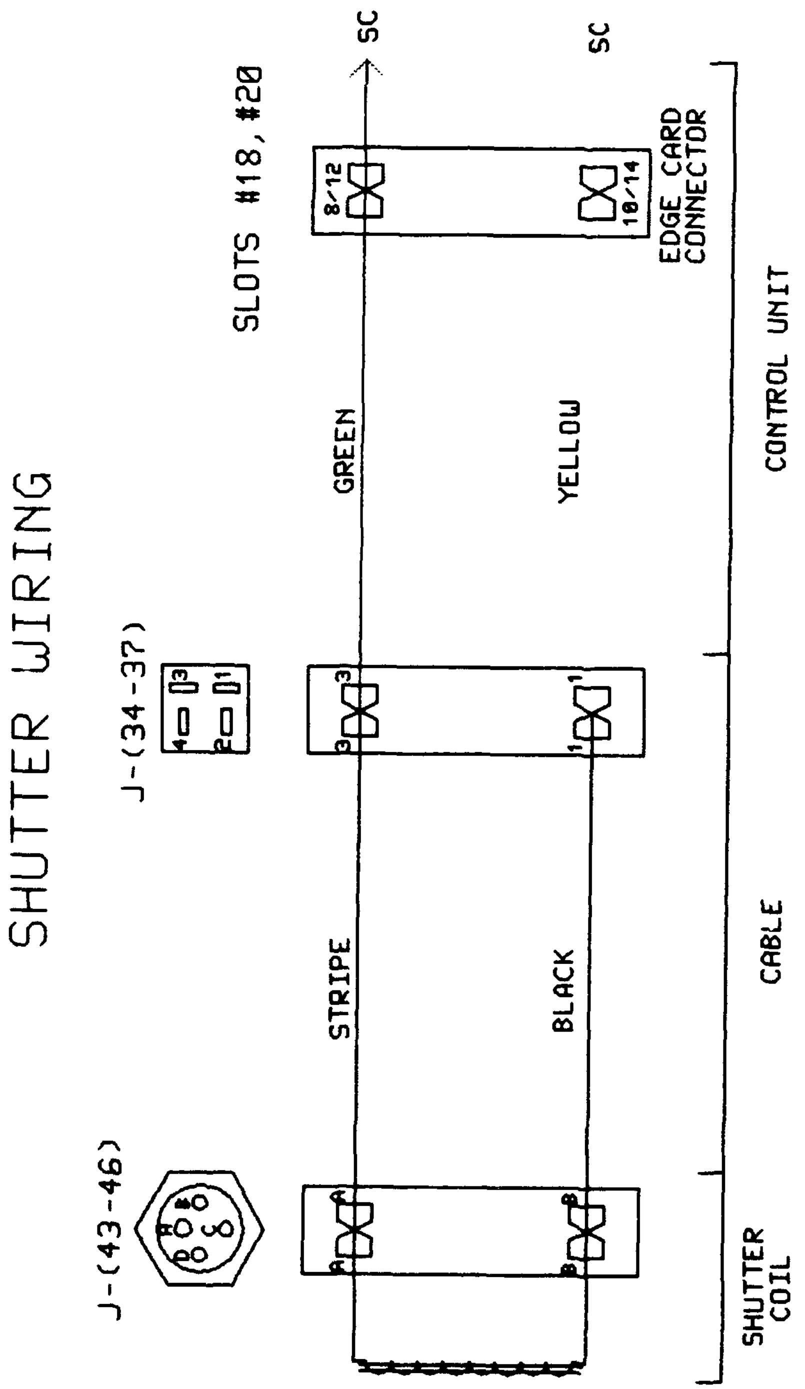


Figure AIII.15b. Wiring diagram between incident beam shutters and the SAG.

APPENDIX IV: SOURCE CODE OF THE ANALOG APSD SOFTWARE MODULES.

APPENDIX IV: SOURCE CODE OF THE ANALOG APSD SOFTWARE MODULES.

Program LISA (LIght Scattering Apparatus), written by Charles Henry, consists of 32 FORTRAN 77 subroutine modules. This appendix includes a flowchart of these modules as integrated into the APSD unit, and their source codes.

When entering "RUN MAIN", LISA prompts for five options: (1) begin a new experiment; (2) review collected data; (3) calibrate optics; (4) calibrate A/D converter; and (5) exit. In Step (1), a header block of information is established before the experiment is executed. That information includes (a) the operator name, (b) the number of scattering samples, (c) the sample name, (d) whether the sample is dry or contaminated, (e) the start and final backscattering angle selections, (f) the resolution of backscattering angle scan between these limits, (g) the number of lasers, and (h) the wavelengths of the laser beams. LISA will then ask which communications port with parity, bits, baudrate, and mode is linked to the Serial Addressable Gateway (SAG). LISA then prompts for whether real-time graphics, real-time A/D channel voltages, or neither are to be presented on the screen. (If 'neither' is selected, the experiment will run faster.) After these data are entered, the experiment proceeds automatically. Steps (2) and (5) are self-explicit. The calibration Steps (3)-(4) will typically be performed before and after a long trial of experiments (see Sections 4.6.1 - 4.6.2).

We now present the system flowchart followed by its source code modules (see also Section 4.7).

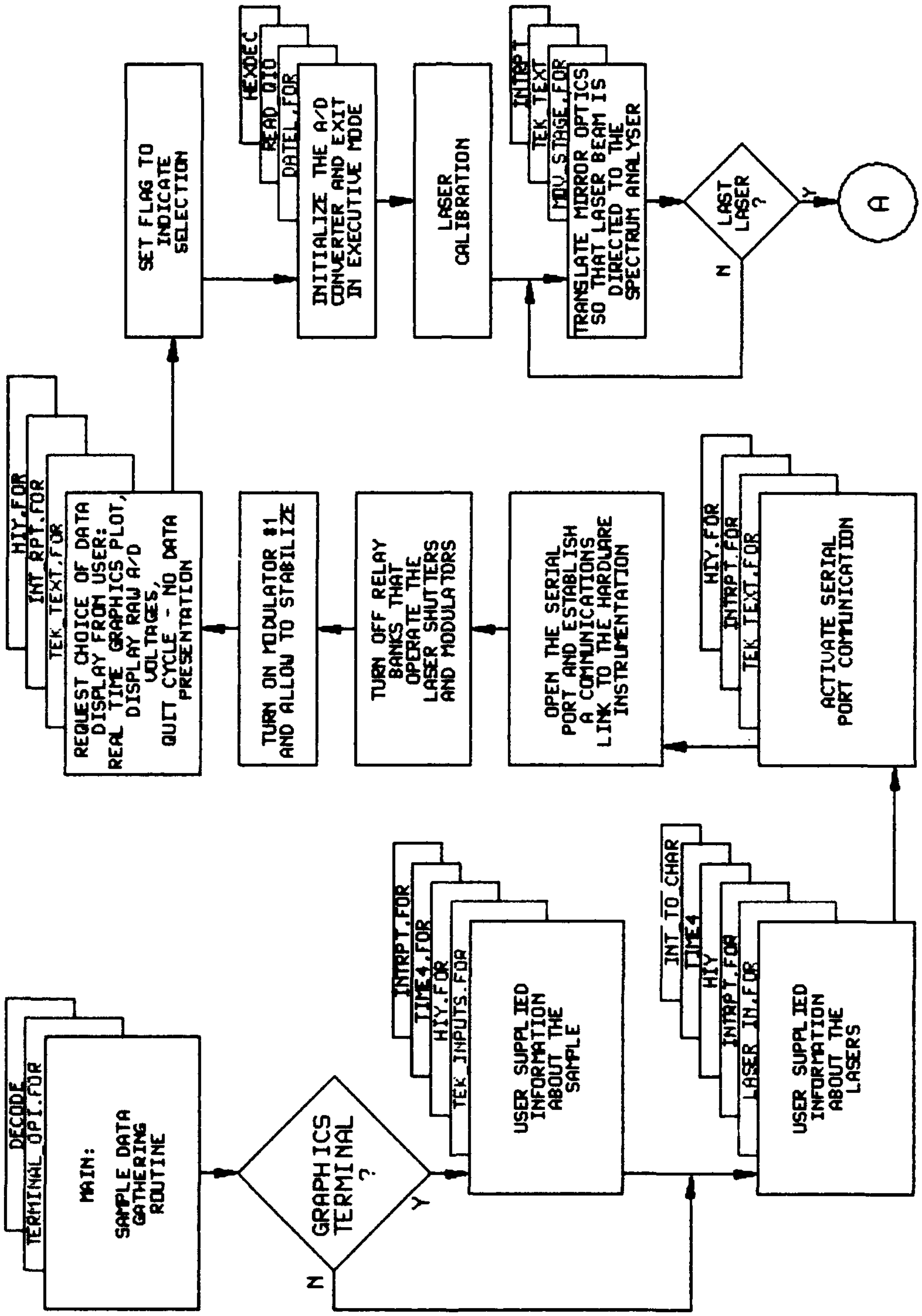


Figure AIV.1 LIght Scattering Apparatus (LISA) flowchart of system software modules.

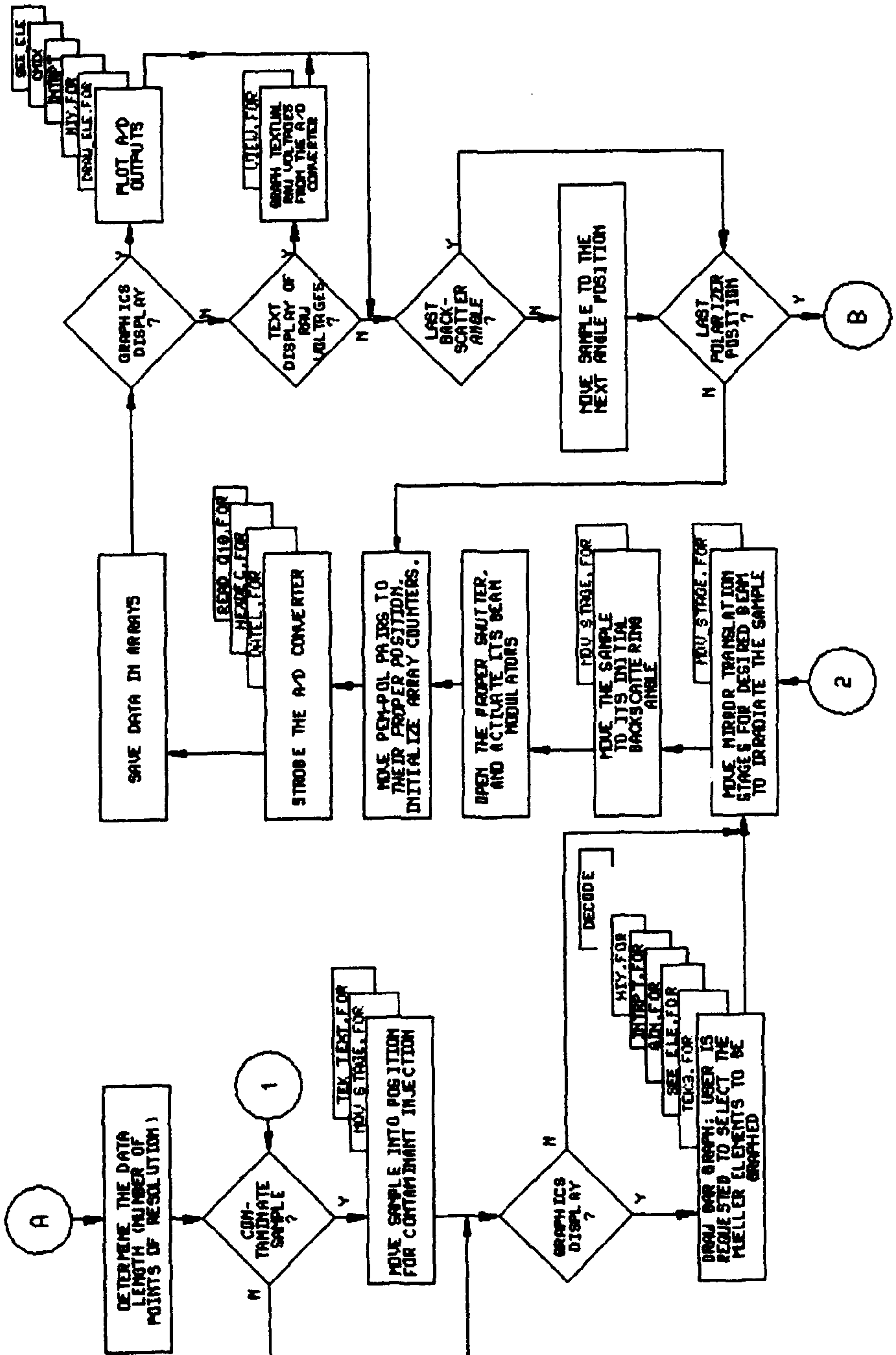


Figure AIV.1 - continued

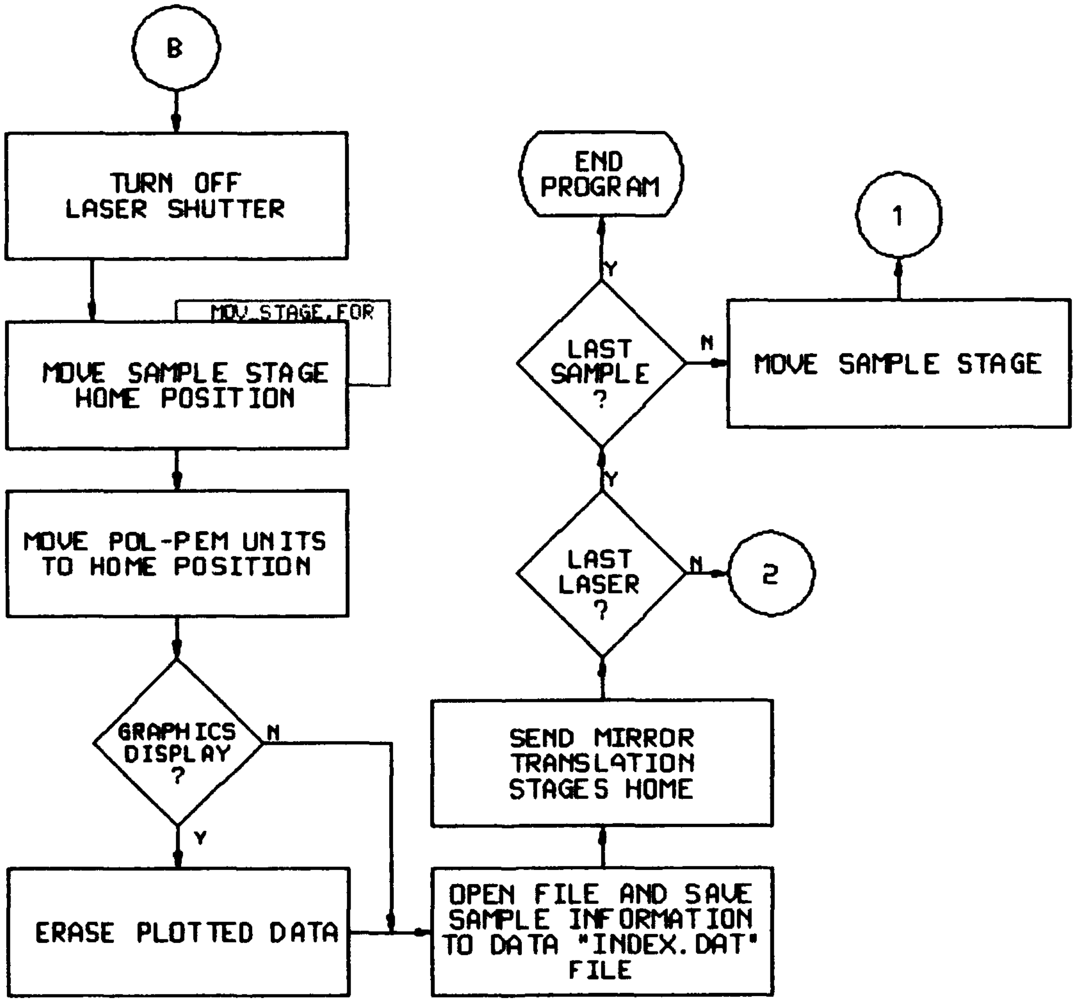


Figure AIV.1 - continued

Appendix IV

AIV.1 Analog APSD Software Modules: BUBBLE_UP Source Code.

```
      SUBROUTINE BUBBLE_UP(ARRAY,SIZE)

C-----------------------------------------------
C     THIS IS A ASCENDING INTEGER BUBBLE SORT ROUTINE.  AN ARRAY IS
C     PASSED IN THAT IS SORTED FROM 1 - 1000.  SIZE IS THE NUMBER
C     OF ITEMS IN THE ARRAY THAT THE USER WANTS SORTED.
C-----------------------------------------------

      INTEGER ARRAY,SIZE,HOLD,BUF
      DIMENSION ARRAY(*),BUF(20)
      L - 0

 10   HOLD - 1000
      J - 0

      DO 20 I - 1,SIZE            ! LOOP THRU THE ARRAY
       IF(ARRAY(I).GE.HOLD)GOTO 20
       HOLD - ARRAY(I)             ! HOLD LOWEST NUMBER IN HOLD
       J - I                      ! KEEP PLACE OF LOWEST NUMBER
 20   CONTINUE                    ! END LOOP

      L - L + 1                   ! INCREMENT ARRAY COUNTER
      BUF(L) - HOLD
      ARRAY(J) - 1000             ! PLACE BIG NUMBER WHERE
      IF(L.EQ.SIZE)GOTO 30        ! SMALLEST WAS.
      GOTO 10

 30   DO 40 I - 1,SIZE
       ARRAY(I) - BUF(I)
 40   CONTINUE

      RETURN
      END
```

AIV.2 Analog APSD Software Modules: CMIX Source Code.

```
      SUBROUTINE CMIX(RNUM,CNUM)

C*****************************************************************
C     THIS ROUTINE IS DESIGNED TO TAKE IN A REAL NUMBER(UNDER 1000)
C     AND CONVERT IT TO A CHARACTER FOR USE WITH A
C     GRAPHICS ROUTINE.
C*****************************************************************

      REAL RNUM R1,RPLC,RBUF
      CHARACTER CNUM*6
      DIMENSION R1(5),RPLC(4),I1(6)
      DATA RPLC/1000.,100.,10.,1./
      CNUM-' '
 1    FORMAT(F6.2)

      RBUF - RNUM

      DO 10 I-1,4
       IF(RNUM.GE.RPLC(I))THEN
        R1(I)-RNUM/RPLC(I)
        I1(I)-IFIX(R1(I))
        RNUM-RNUM-(FLOAT(I1(I))*RPLC(I))

       ELSE
        I1(I)-0
       ENDIF
 10    CONTINUE

      IF(RNUM.GE..1)THEN
      R1(5)-RNUM/.1
      I1(6)-IFIX(R1(5)*10)
      I1(6)-I1(6)/10
      ELSE
      I1(5)-0
      ENDIF

      CNUM(1:1)-CHAR(I1(1)+48)
      CNUM(2:2)-CHAR(I1(2)+48)
      CNUM(3:3)-CHAR(I1(3)+48)
      CNUM(4:4)-CHAR(I1(4)+48)
      CNUM(5:5)-'.'
      CNUM(6:6)-CHAR(I1(6)+48)

      DO 20 I-1,6
       IF(CNUM(I:I).EQ.CHAR(48))THEN
        CNUM(I:I)-' '
       ELSE
        GOTO 30
      ENDIF
 20   CONTINUE

```

```
30  RNUM = RBUF

    RETURN
    END
```

AIV.3 Analog APSD Software Modules: DATEL_1 Source Code.

```
      SUBROUTINE DATEL(TYPE,REPORT,PORT,ICNT,CHAN,RDAT,HEX)


C*******************************************************************
C
C     This mod is designed to initiate the executive level control
C     commands for the datel 701 A/D converter. The first part wakes
C     up the system with a carriage control. The executive is accessed
C     by the turnon command (line 10). The converter will respond with
C     an echo of the mesg and provide a carriage control LP and give an
C     "" as a prompt (character string 42). At this point
C     the system is ready to collect data.
C
C     At this point the converter echos the command and initiates a LP
C     CR which I look for and omit from the data collection. The data
C     values come in a long string seperated by LP CR. I look for these
C     and it is used to shape the data so that individual numbers can
C     be placed in an array called " RDAT ".
C
C
C     TYPE - THE FUNCTION TO BE PERFORMED BY THIS ROUTINE
C          0. - INITIALIZE AND STROBE 9 CHANNELS
C          1. - INITIALIZE THE A/D TO EXECUTIVE MODE ONLY
C          2. - STROBE ONLY ONE CHANNEL PASSED IN BY " CHAN "
C          3. - STROBE CHANNELS 0 - 16 BASED ON ICNT
C
C     REPORT - ERROR FLAG WHEN SET TO 1 INDICATES THAT THE ROUTINE
C          DID NOT FUNCTION CORRECTLY. THIS IS PASSED BACK TO
C          THE CALLER
C
C     PORT  - THIS IS THE SERIAL PORT DEFINED BY THE CALLER THAT
C          WILL BE USED FOR THE A/D COMMUNICATIONS.
C
C     ICNT  - THE NUMBER OF DATA ELEMENTS ( OR CHANNEL NUMBERS )
C          THAT DATA IS BEING COLLECTED FOR. THIS IS REQUIRED
C          BY THE HEXDEC SUBROUTINE.
C
C     CHAN  - THE INDIVIDUAL CHANNEL NUMBER TO BE STROBED. PASSED
C          IN FROM THE CALLER
C
C     RDAT  - ARRAY OF REAL NUMBERS CONVERTED FROM HEXIDECIMAL
C          DEFINING THE RELATIVE VOLTAGES FOR EACH CHANNEL
C          THIS IS PASSED BACK TO THE CALLER.
C*******************************************************************

      CHARACTER MSG,A*80,TDAT*1,ENDDAT*5,TURNON*19,CS,CR,LF
      CHARACTER MSG1*30,PORT*10,MESG*512,CDAT*4,QUIET*3
      CHARACTER STROBE*9,CHAN*2,CNT*9,HEX*4,A2D_INPUT*6,E*1

      INTEGER*4 TIMEOUT,LEN_STRING
      INTEGER ISIZE,IOUT,CHANNEL,TYPE,REPORT
```

```

      REAL RDAT

      DIMENSION RDAT(16),HEX(16),A2D_INPUT(16)


      L     - 0
      E     - CHAR(27)
      GS    - CHAR(29)
      CR    - CHAR(13)
      LF    - CHAR(10)
      ICNT - 10
      IERR  - 0

      TIMEOUT - 1
      IRESCAN - 0
      TURNON  -'ST-701 EXECUTIVE ON'
      LEN_STRING - 512

C      OPEN(4,FILE-'tWA3:',STATUS-'NEW',CARRIAGECONTROL - 'NONE')

C-----------------------------------------

      IF(TYPE.EQ.3.OR.TYPE.EQ.2)GOTO 40     ! STROBING FOR DATA
      IF(TYPE.EQ.1.OR.TYPE.EQ.0)THEN  ! USERS IS INITIALIZING THE BOARD

C-----------------------------------------

 2    FORMAT(A1)

 10   FORMAT (A30)

      WRITE(3,'(A4)')GS//'D0 '      ! CLOSE EXISTING COMMUNICATIONS
      CALL TWAIT(2)                 ! PAUSE .2 SECS
      WRITE(3,'(A4)')GS//'D4'//CR   ! WAKE THE CONTROLLER BOARD

      CALL TWAIT(2)                 ! PAUSE .2 SECONDS

      IERR - 0                      ! INITIALIZE ERROR COUNT

 16   WRITE(3,'(A1)')CR             ! CARRIAGE CONTROL
      CALL TWAIT(2)                 ! PAUSE .2 SECONDS

      CALL READ_QIO (PORT,MESG,LEN_STRING,TIMEOUT,ISIZE,IOUT,
     .CHANNEL,INUSE)



C     WRITE(4,*)ISIZE,' INITIALIZING MESG - ',MESG




      DO 17 I - 1,ISIZE + 5         ! CHECK FOR A PROMPT
```

```
      IF(MESG(I:I).EQ.'-')GOTO 20 ! COMMAND MODE
      IF(MESG(I:I).EQ.'#')GOTO 20 ! COMMAND MODE
      IF(MESG(I:I).EQ.'*')GOTO 30 ! EXECUTIVE MODE
17    CONTINUE

      IERR - IERR + 1          ! INCREMENT ERROR COUNTER

      IF(IERR.EQ.5)THEN        ! NOT COMMUNICATING WITH A/D
       REPORT - 1              ! SET ERROR FLAG
       GOTO 1000               ! RETURN TO CALLER
      ENDIF

      GOTO 16                  ! GO TRY TO WAKE IT AGAIN

C-----------------------------------------
C     THIS WRITES THE MESSAGE "ST701-EXECUTIVE ON" TO THE BOARD TO SET
C     THE MODE TO EXECUTIVE. WHICH THE BOARD WILL ANSWER WITH A * * *
C-----------------------------------------

20    IERR - 0

25    DO 27 I - 1,19
       WRITE(3,'(A)')TURNON(I:I)
27    CONTINUE

      WRITE(3,'(A)')CR

      CALL TWAIT(2)            ! PAUSE 1/2 SECOND

      CALL READ_QIO (PORT,MESG,LEN_STRING,ITIME,ISIZE,IOUT,
     .CHANNEL,INUSE)

      IF(IOUT.EQ.0)GOTO 30   ! THE A/D BOARD ANSWERED MOVE ON

       IERR - IERR + 1         ! INCREMENT ERROR COUNTER

      IF(IERR.EQ.5)THEN        ! NOT COMMUNICATING WITH A/D
       REPORT - 1              ! SET ERROR FLAG
       GOTO 1000               ! RETURN TO CALLER
      ENDIF

      GOTO 25

C-----------------------------------------

30    L - 0
38    IF(TYPE.EQ.1)GOTO 1000   ! INITIALIZATION COMPLETE
                               ! RETURN TO CALLER
      ENDIF

C-----------------------------------------
C     THIS PART ASKS THE A/D TO STROBE ITS CHANNELS
C-----------------------------------------
```

```
40    WRITE(3,'(A4)')GS//'DO '      ! CLOSE EXISTING COMMUNICATIONS

      CALL TWAIT(2)            ! PAUSE .2 SECS

      WRITE(3,'(A4)')GS//'D4'//CR   ! CONTROL IS TO A/D BOARD

      CALL TWAIT(2)            ! PAUSE .2 SECONDS

C------------------------------------------
C     THIS REMOVES THE MOTOR CONTROLS FROM THE READ BUFFER
C------------------------------------------


      CALL READ_QIO (PORT,MESG,LEN_STRING,ITIME,ISIZE,IOUT,
     .CHANNEL,INUSE)

C        WRITE(4,*)'ISIZE - ',ISIZE
C        WRITE(4,*)'MESG - ',MESG     ! TEST DATA

      IERR - 0

42      WRITE(3,'(A9)')'AS 0,9,5'//CR
C       WRITE(4,'(A9)')'AS 0,9,5'//CR

      READ(3,'(A512)',ERR-42)MESG

C------------------------------------------
C     THIS PART LOOKS FOR THE MESSAGE SENT OUT THAT IS ECHOED BACK TO
C     THE SERIAL PORT.  ONCE THIS IS READ THE REST OF THE INPUT STRING
C     IS DATA.
C------------------------------------------
      ISTART - 1
      ICNT - 10                ! STROBE FIRST 9 CHANNELS

45     DO 50 J - 1,80            ! SEARCH FOR PART OF THE STRING

      IF(MESG(J:J + 1).EQ.'AS'.OR.MESG(J:J + 1).EQ.'AK')GOTO 70

50    CONTINUE

C------------------------------------------
C     THIS IS AN ERROR CATCHING PORTION.  IF NO STRING IS FOUND THEN
C     I TRY UP TO 5 TIMES.  IF I STILL GET NOWHERE I SET REPORT  - 1
C     AND RETURN TO THE CALLER.
C------------------------------------------

      IERR - IERR + 1

      IF(IERR.EQ.6) THEN
       REPORT - 1
       GOTO 1000
      ENDIF

      GOTO 42                  ! STROBE THE A/D AGAIN
```

```
C-----------------------------------------

70    DO 80 I - ISTART,ICNT          ! READ IN THE HEX DATA

        READ(3,'(A6)')A2D_INPUT(I)
C        WRITE(4,*)I,'   ',A2D_INPUT(I)//CR


80    CONTINUE

C-----------------------------------------
C     THIS GOES AND SCANS CHANNELS 7 - 10
C-----------------------------------------

      IF(IRESCAN.EQ.0)THEN
        IRESCAN - 1
        ISTART - 11
        ICNT - 11

85      WRITE(3,'(A8)')'AK 10,5'//CR
C       WRITE(4,'(A7)')'AK 10,5'

      CALL TWAIT(2)

      READ(3,'(A255)',ERR-85)MESG

      GOTO 45
      ENDIF

C-----------------------------------------
C     THIS PART ASSUMES THAT THE REST OF THE READ STRING IS HEX DATA
C     SEPARATED BY CARRIAGE RETURNS AND LINE FEEDS.  I USE THESE AS
C     BREAKS BETWEEN THE NUMBERS AND PLACE THE INDIVIDUAL NUMBERS
C     IN AN ARRAY HEX.
C-----------------------------------------

90    L - 0
      K - 0

      DO 110 M - 1,11

C-----------------------------------------
C     THIS DETERMINES IF THE DATA IS INDEED HEX OR JUST GARBAGE.
C     IF ITS GARBAGE I GO READ AGAIN.
C-----------------------------------------

      DO 100 I - 1,6

      IF(A2D_INPUT(M)(I:I).LT.CHAR(48).OR.
     .A2D_INPUT(M)(I:I).GT.CHAR(70))GOTO 100

        L - L + 1                    ! INCREMENT CHARACTER CNT

```

```
            CDAT(L:L) - A2D_INPUT(M)(I:I)     ! PLACE DATA IN ARRAY

        IF(L.EQ.4)THEN                ! CAN ONLY HAVE 4 CHARA'S
          IFLG - 0                  ! INITIALIZE DATA FLAG
          L - 0                    ! RESET CHARACTER CNTER
          HEX(M) - CDAT(1:4)        ! PLACE DATA IN ARRAY
          CDAT - ' '
          GOTO 110
        ENDIF

100   CONTINUE

110   CONTINUE                      ! CONTINUE THRU LOOP

C-----------------------------------------------------
C     THIS REQUESTS THAT THE HEX CHARACTER DATA BE CONVERTED TO REAL
C     DATA TO BE SENT BACK TO THE CALLER.
C-----------------------------------------------------

135   CALL HEXDEC(HEX,RDAT,ICNT)

      WRITE(3,'(A4)')GS//'DO '      ! CLOSE EXISTING COMMUNICATIONS

      CALL TWAIT(2)                ! PAUSE .2 SECS

      WRITE(3,'(A4)')GS//'D3'//CR   ! CONTROL IS TO SAMPLE STAGE

      CALL TWAIT(2)                ! PAUSE .2 SECONDS

1000  RETURN
      END
```

AIV.4 Analog APSD Software Modules: DECODE Source Code.

```
      SUBROUTINE DECODE (CIN,INTOUT)
C*******************************************************************
C     THIS IS DECODE.FOR CALLED BY THE PANEL SUBROUTINE PA.FOR
C     IT IS GIVEN THREE BYTES GENERATED BY THE MOUSE THAT WILL
C     BE DECODED TO RETURN A PICK ID FOR A PARTICULAR PANEL.
C     CIN IS THE 3 CHARACTER INPUT STRING AND INTOUT IS AN
C     INTEGER SENT BACK TO THE CALLER.
C*******************************************************************

      CHARACTER *3 CIN
C     CHARACTER *(*)CIN
      INTEGER*4 HI1,HI2,LO1
C AN INTEGER REPORT IS ALWAYS INCODED AS 3 BYTES.
      I=0
      I=I+1
      HI1=ICHAR(CIN(I:I))-32
      I=I+1
      HI2=ICHAR(CIN(I:I))-32
      I=I+1
      LO1=ICHAR(CIN(I:I))-32
      I=I+1
C     DECODE THE 3 BYTES TO OBTAIN THE INTEGER VALUE.
      INTOUT=HI1*1024+HI2*16+JMOD(LO1,16)
      IF(MOD(LO1/16,2).EQ.0) INTOUT=-INTOUT
      RETURN
      END
```

AIV.5 Analog APSD Software Modules: DRAW_ELE Source Code.

```
      SUBROUTINE DRAW_ELE(ISWEEP,ANGLE,RARM,RINCR,REND,RDAT,
     .INUSE)
C-----------------------------------------
C     THIS MOD DRAWS A GRAPH OF THE USER SELECTED ELEMENTS FOR EACH
C     SWEEP OF THE SAMPLE STAGE.  THIS IS INTENDED TO BE SOMEWHAT REAL
C     TIME SO USER CONTROL OMITTED.
C-----------------------------------------


      CHARACTER TEXT*35,TEXT1*35,TEXT2*35,TEXT3*35,TEXT4*35
      CHARACTER CNUM*10,COL*3,LAST_PT*5,START*6,END*6
      CHARACTER E,A*5,SEG*3,SPACE*40,CT*6,B,A1*5
      CHARACTER CURRENT*6,INCREMENT*6,POINT*5

      INTEGER RECCNT,REC,IFLG,X,Y,BUSY,MATRIX
      INTEGER X2,Y2,X3,Y3

      REAL RARM,RINCR,REND,RDAT,RBUF,ANGLE,R1,R2

      DIMENSION LDAT1(10),LDAT2(10),LDAT3(10),LDAT4(10)

      DIMENSION LAST_PT(9),POINT(9),ICHAN(9)

      DIMENSION IA(9),IB(9),IC(9),ID(9),RDAT(16),IELE(16),RBUF(9)

      COMMON LDAT1,LDAT2,LDAT3,LDAT4

C-----------------------------------------
C     THESE ARE THE MATRIX ELEMENT ARRAY NUMBERS THAT ARE COLLECTED
C     BY EACH PERMUTATION OF THE POLARIZERS.
C-----------------------------------------

      DATA IA / 1,2,4,5,6,8,13,14,16/        ! VERTICAL, VERTICAL
      DATA IB / 1,2,4,9,10,12,13,14,16/      ! VERTICAL, 45 DEG
      DATA IC / 1,3,4,5,7,8,13,15,16/        ! 45 DEG  , VERTICAL
      DATA ID / 1,3,4,9,11,12,13,15,16/      ! 45 DEG  , 45 DEG

 3    FORMAT (40(' '))
      WRITE(SPACE,3)
      E = CHAR(27)

C-----------------------------------------

      WRITE(*,*)E//'LV0'          ! DISABLE DIALOG AREA
      WRITE(*,*)E//'LZ'           ! CLEAR THE DIALOG AREA

C-----------------------------------------
C     THIS SETS UP THE DIALOG AREA SO THE USER HAS AN IDEA WHAT IS BEING
C     DISPLAYED.  THIS REPORTS THE CURRENT POLARIZER POSITION,  THE
C     START, END AND INCREMENT OF THE SAMPLE STAGE ANGLE.  ADDITIONALLY,
C     THE CURRENT ANGLE OF INCIDENCE IS DISPLAYED.
```

```
C——————————————————————————

      IF(INUSE.EQ.2)THEN              ! INITIALIZE COUNTERS
       INUSE - 0
       INUSE1 - 0
      ENDIF

      IF (INUSE.EQ.0)THEN             ! ONLY NEED TO ESTABLISH

       CALL CMIX(RARM,START)           ! TEXT ONCE FOR THE WHOLE

       CALL CMIX(REND,END)             ! ROUTINE.

       CALL CMIX(RINCR,INCREMENT)      ! START, END AND INCREMENT
                                       ! ARE MADE CHARACTERS HERE
       TEXT1 - 'START ANGLE:'//START//'  END ANGLE:'//END
       TEXT2 - 'ANGLE INCREMENT: '//INCREMENT
       TEXT3 - 'CURRENT ANGLE:        Degrees'

C——————————————————————————
C     THIS SAVES THE ELEMENTS THAT WILL BE GRAPHED FOR EACH POLARIZER
C     PERMUTATION TO AN ARRAY.
C——————————————————————————

      IF(ISWEEP.EQ.1)THEN             ! POLARIZERS VERT, VERT

       TEXT - 'POLARIZERS: Vertical, Vertical'
         DO 70 I-1,LDAT1(10)          ! READ THE A/D CHANNELS
          ICHAN(I) - LDAT1(I)         ! INTO A BUFFER ARRAY
 70       CONTINUE

         IEND - LDAT1(10)             ! SAVE COUNT ON ELEMENTS

         DO 75 I - 1,9                ! LOOP THRU MATRIX ELE
          IELE(I) - IA(I)             ! SAVE IN ARRAY
 75       CONTINUE

      ELSEIF(ISWEEP.EQ.2)THEN

       TEXT - 'POLARIZERS: Vertical, 45 Degrees'
         DO 80 I-1,LDAT2(10)          ! READ THE A/D CHANNELS
          ICHAN(I) - LDAT2(I)         ! INTO A BUFFER ARRAY
 80       CONTINUE

         IEND - LDAT2(10)             ! SAVE COUNT ON ELEMENTS

         DO 85 I - 1,9                ! LOOP THRU MATRIX ELE
          IELE(I) - IB(I)             ! SAVE IN ARRAY
 85       CONTINUE

          X - X - INCR

      ELSEIF(ISWEEP.EQ.3)THEN

```

```
          TEXT - 'POLARIZERS:  45 Degrees, Vertical'
          DO 90 I-1,LDAT3(10)         ! READ THE A/D CHANNELS
           ICHAN(I) - LDAT3(I)        ! INTO A BUFFER ARRAY
   90     CONTINUE
          IEND - LDAT3(10)            ! SAVE COUNT ON ELEMENTS

          DO 95 I - 1,9             ! LOOP THRU MATRIX ELE
           IELE(I) - IC(I)         ! SAVE IN ARRAY
   95     CONTINUE
        X - X + INCR

        ELSEIF(ISWEEP.EQ.4)THEN

         TEXT - 'POLARIZERS:  45 Degrees, 45 Degrees'
          DO 100 I-1,LDAT4(10)        ! READ THE A/D CHANNELS
           ICHAN(I) - LDAT4(I)        ! INTO A BUFFER ARRAY
  100     CONTINUE
          IEND - LDAT4(10)            ! SAVE COUNT ON ELEMENTS

          DO 105 I - 1,9            ! LOOP THRU MATRIX ELE
           IELE(I) - ID(I)         ! SAVE IN ARRAY
  105     CONTINUE
        X - X - INCR

        ENDIF

C-------------------------------------------------
C       FLAG THE SEE_ELE ROUTINE TO DISPLAY THE  ELEMENTS THAT ARE
C       POSSIBLE FOR THIS SWEEP
C-------------------------------------------------

        MATRIX - 1
        CALL SEE_ELE(ISWEEP,MATRIX,INU)

C-------------------------------------------------
C       THIS PART DRAWS THE TEXT TO THE DIALOG AREA WINDOW.
C-------------------------------------------------

        ISEG - 800
        CALL INTRPT(ISEG,SEG)
        WRITE(*,*)E//'SK'//SEG
        WRITE(*,*)E//'SE'//SEG
        WRITE(*,*)E//'MT7'
        X2 - 120
        Y2 - 350
        CALL HIY(X2,Y2,A1)
        WRITE(*,*)E//'LP'//A1
        WRITE(*,*)E//'LTB7'//TEXT

        Y2 - Y2 - 100
        CALL HIY(X2,Y2,A1)
        WRITE(*,*)E//'MT1'
        WRITE(*,*)E//'LP'//A1
        WRITE(*,*)E//'LTB8'//TEXT1
```

```
      Y2 - Y2 - 100
      CALL HTY(X2,Y2,A1)
      WRITE(*,*)E//'LP'//A1
      WRITE(*,*)E//'LTB6'//TEXT2

      Y2 - Y2 - 100
      CALL HTY(X2,Y2,A1)
      WRITE(*,*)E//'LP'//A1
      WRITE(*,*)E//'LTB7'//TEXT3
      WRITE(*,*)E//'SC'


      ISEG - 850
      CALL INTRPT(ISEG,SEG)
      WRITE(*,*)E//'SK'//SEG
      WRITE(*,*)E//'KN0'
      CALL TWAIT(10)
      WRITE(*,*)E//'SE'//SEG

     ENDIF

C-----------------------------------------------
C     THIS WRITES THE CURRENT INCIDENT ANGLE TO THE DIALOG AREA
C-----------------------------------------------

     CALL CMIX(ANGLE,CURRENT)

      X2 - 800
      Y2 - 25

      WRITE(*,*)E//'MP3'

      CALL HTY(X2,Y2,A1)
      WRITE(*,*)E//'LP'//A1

      X2 - X2 + 300
      CALL HTY(X2,Y2,A1)
      WRITE(*,*)E//'LG'//A1

      Y2 - Y2 + 105
      CALL HTY(X2,Y2,A1)
      WRITE(*,*)E//'LG'//A1


      X2 - X2 - 300
      CALL HTY(X2,Y2,A1)
      WRITE(*,*)E//'LG'//A1

      Y2 - Y2 - 105
      CALL HTY(X2,Y2,A1)
      WRITE(*,*)E//'LG'//A1
      WRITE(*,*)E//'LE'

```

```
        Y2 - Y2 + 25
      CALL HTY(X2,Y2,A1)

      WRITE(*,*)E//'LF'//A1
      WRITE(*,*)E//'MT7'
      WRITE(*,*)E//'LT6'//CURRENT
      WRITE(*,*)E//'MT1'

C------------------------------------
C     THIS CHECKS TO SEE IF THE USER WANTED TO OMIT DRAWING THE GRAPH
C     FOR THIS SPECIFIC SWEEP.  THE 10TH ELEMENT CONTAINS THE NUMBER
C     OF THE 9 ELEMENTS THAT WILL BE DRAWN.
C------------------------------------


      IF(ISWEEP.EQ.1)THEN
        IF(LDAT1(10).EQ.0)GOTO 1000        ! NO ELEMENTS ARE DRAWN

      ELSEIF(ISWEEP.EQ.2)THEN
        IF(LDAT2(10).EQ.0)GOTO 1000        ! NO ELEMENTS ARE DRAWN

      ELSEIF(ISWEEP.EQ.3)THEN
        IF(LDAT3(10).EQ.0)GOTO 1000        ! NO ELEMENTS ARE DRAWN

      ELSEIF(ISWEEP.EQ.4)THEN
        IF(LDAT4(10).EQ.0)GOTO 1000        ! NO ELEMENTS ARE DRAWN

      ENDIF

C------------------------------------
C------------------------------------
C     THIS MOD DRAWS THE LINE SEGMENTS TO THE SCREEN
C------------------------------------



      DO 200 K - 1,IEND                 ! LOOP THRU THE ELEMENTS

C------------------------------------
C     THIS PART DETERMINES THE COLOR EACH MATRIX LINE ELEMENT WILL BE
C------------------------------------

      M - ICHAN(K)                      ! M - THE CHANNEL READ
      ICOL - IELE(M)                    ! ICOL - THE ELEMENT COLOR

      IF(ICOL.EQ.16)THEN                ! THIS COLOR IS WHITE
       WRITE(*,*)E//'MV1'               ! MAKE LINE DASHED
       WRITE(*,*)E//'ML1'               ! LINE COLOR WHITE
      ELSE
       WRITE(*,*)E//'MV0'               ! MAKE LINE SOLID
       CALL INTRPT(ICOL,COL)              ! INTEGER TO TEK CHARA
       WRITE(*,*)E//'ML'//COL(1:1)      ! WRITE COLOR TO TERMINAL
      ENDIF
C------------------------------------
```

```
C     THE X IS THE BEGINNING ANGLE OF THE SAMPLE FOR WHICH DATA IS
C     RETRIEVED.  THE ZERO POINT IS SCREEN POSITION 245.  EACH DEGREE
C     INCREMENT IS 20 SCREEN UNITS.  THUS THE EQUATION BELOW:
C-----------------------------------------------------------------

      IF(INUSE1.EQ.0)THEN            ! USE ONLY ONCE ON RUN


       X - IFIX(RARM)
       X - 245 + (20 * X)

C-----------------------------------------------------------------
C     THE INCREMENT FOR EACH DATA PT IS IN INCR.  HERE NORMALIZED TO
C     SCREEN UNITS.  1 DEG - 20 SCREEN UNITS.
C-----------------------------------------------------------------

       INCR - IFIX(RINCR)
       INCR - INCR * 20
       INUSE1 - 1
      ENDIF

C-----------------------------------------------------------------

       K1     - 100
       IYPOINT - 0

C-----------------------------------------------------------------
C     THIS MOD RESOLVES THE VOLTAGE INTO A Y COORDINATE ( IYPOINT )
C     THE BEST RESOLUTION IS 1/1000 VOLT.  1/10 VOLT - 100 SCREEN UNITS
C     1/100 VOLT - 10 SCREEN UNITS AND 1/1000 VOLT - 1 SCREEN UNIT.
C-----------------------------------------------------------------

      R2 - RDAT(M)                   ! SAVE DATA IN BUFFER

      DO 180 N - 1,3
       R1 - R2 * 10.0                ! READ PROPER CHANNEL
       INUM - IFIX(R1)
       IYPOINT - INUM * K1 + IYPOINT
       R2 - R1 - (FLOAT(INUM))
       IF(N.EQ.1)K1 - K1 - 90
       IF(N.EQ.2)K1 - K1 - 9
180    CONTINUE

C-----------------------------------------------------------------
C     THIS MOD NORMALIZES THE SCREEN UNIT WITH THE BAR GRAPH AND DRAWS
C     THE LINE FROM ONE POINT TO THE NEXT.  1900 IS THE Y COORDINATE FOR
C     THE 0.0 LINE ON THE Y AXIS.
C-----------------------------------------------------------------

       Y - 1900 + IYPOINT            ! CALCULATE THE Y POINT

       IF(Y.GE.2900)Y - 2900         ! DON'T LET Y LEAVE THE
       IF(Y.LE.900)Y - 900           ! GRAPH
       CALL HTY (X,Y,A)              ! CALC X,Y VECTOR
```

```
          LAST_PT(M)- A                    ! SAVE IN BUFFER ARRAY

        IF(INUSE.EQ.0)THEN                 ! IF THIS IS THE 1ST POINT
          WRITE(*,*)E//'ML2'
          WRITE(*,*)E//'MM2'               ! SET MARKER MODE TO "+"
          WRITE(*,*)E//'LP'//A             ! SET THE ORIGIN
          WRITE(*,*)E//'LH'//A             ! DRAW THE MARKER
        ELSE

C-----------------------------------------
C     THIS PART MUST ESTABLISH AN ORIGIN THAT WAS THE LAST POINT BEFORE
C     EVERY DRAW.
C-----------------------------------------


          WRITE(*,*)E//'LP'//POINT(M)      ! SET THE ORIGIN
          WRITE(*,*)E//'LG'//A             ! DRAW TO NEW POINT

        ENDIF

  200   CONTINUE

C-----------------------------------------
C     THIS INCREMENTS THE COUNTER FOR THE NEXT X
C-----------------------------------------

        IF(ISWEEP.EQ.1.OR.ISWEEP.EQ.3)THEN
          X - X + INCR        ! INCREMENT HHE X POINT
        ELSE
          X - X - INCR        ! DECREMENT THE X VALUE
        ENDIF

C-----------------------------------------
C     THIS SAVES THE VECTORS OF EACH LINE ELEMENT CALCULATED FOR THIS
C     POINT INTO A BUFFER SO THAT THEY CAN BE USED AS ORIGINS FOR THE
C     NEXT GROUP OF POINTS
C-----------------------------------------

        DO 300 I - 1,9
          POINT(I) - LAST_PT(I)
  300   CONTINUE

          WRITE(*,*)E//'MV0'               ! MAKE LINE SOLID

        INUSE - 1                          ! SET INUSE FLAG TO ON

 1000   RETURN
        END
```

AIV.6 Analog APSD Software Modules: GET_ARRAY Source Code.

```
      SUBROUTINE GET_ARRAY(LET,REDRAW)

C-------------------------------------------
C     THIS ROUTINE IS USED TO GET THE  USER TO SELECT AN ARRAY
C     ELEMENT.
C     LET   - THE CHARACTER LETTER A - TO BE SELECTED BY THE USER
C     REDRAW - IS A FLAG FOR THIS ROUTINE TO REDRAW THE INVISIBLE ARRAY
C           BLOCKS.  WHEN SET TO - 1
C-------------------------------------------

      CHARACTER E*1,LET*1,SEG*3,ANS*12

      INTEGER X,Y,REDRAW

      E - CHAR(27)


      IF(REDRAW.EQ.1)THEN          ! FLAG TO TURN ON ALL 5 BLOCKS
       REDRAW - 0                  ! RESET THE FLAG

        IF(ICHAR(LET).GT.64.OR.ICHAR(LET).LT.70)THEN
        J - ICHAR(LET) - 33        ! SEGMENT NUMBER TO START WITH

         DO 10 I - 32,36           ! ONE BLOCK IS ALREADY ON
          IF(I.EQ.J)GOTO 10        ! IF ON SCREEN DON'T DRAW
          CALL INTRPT(I,SEG)       ! CONVERT TO TEK CHARACTER
          WRITE(*,*)E//'SV'//SEG(1:2)//'1'  ! DRAW THE BOX
 10      CONTINUE
        ENDIF
      ENDIF

      X - 1500                     ! PLACE THE MOUSE X,Y
      Y - 550
      IFLAG - 1

      CALL GIN(X,Y,IFLAG,IMODE,ITYPE IGIN,IPORT)

 15   WRITE(*,*)E//'LV0'           ! SET UP THE DIALOG AREA
      WRITE(*,*)E//'LI144'         ! COLOR WHITE ON BLUE
      WRITE(*,*)E//'LZ'            ! CLEAR OF TEXT
      WRITE(*,*)E//'LV1'           ! ENABLE DIALOG AREA


      WRITE(*,*)' SELECT:'
      WRITE(*,*)'      1.  AN ARRAY ELEMENT'
      WRITE(*,*)
      WRITE(*,*)'      3.  EXIT'

 20   READ(*,'(A12)',ERR-15)ANS            ! READ FROM THE MOUSE

      IF(ANS(1:1).EQ.'1'.OR.ANS(1:1).EQ.'3')GOTO 30
```

```
        GOTO 20

C------------------------------------------------
C     THIS CHECKS THE SELECTION TO MAKE SURE AN ARRAY ELEMENT WAS
C     CHOSEN.
C------------------------------------------------

30    SEG = ANS(7:9)              ! GET USER CHARACTER SELECTION
      CALL DECODE(SEG,ISEG)       ! CHANGE SEGMENT NUMBER TO INTEGER

      IF(ISEG + 33.GT.69.OR.ISEG + 33.LT.65)GOTO 20

      WRITE(*,*)E//'ID!'          ! DELETE THE GIN CURSOR
      WRITE(*,*)E//'SKC)1'


      LET = CHAR(ISEG + 33)       ! PLACE THE SELECTED LETTER IN LET

C------------------------------------------------
C     THIS PART ERASES THE REST OF THE BLOCKS THAT ARE NOT IN USE
C     ISEG IS THE SELECTED SEGMENT NUMBER.
C------------------------------------------------


        DO 40 I = 32,36           ! LOOP THRU THE SEGMENT NUMBERS
         IF(I.EQ.ISEG)GOTO 40     ! IF ON SCREEN DON'T DRAW
         CALL INTRPT(I,SEG)       ! CONVERT TO TEK CHARACTER
         WRITE(*,*)E//'SV'//SEG(1:2)//'0'  ! ERASE THE BOX
40       CONTINUE

      RETURN
      END
```

AIV.7 Analog APSD Software Modules: GIN Source Code.

```
      SUBROUTINE GIN(X1,Y1,IFLAG,IMODE,ITYPE,IGIN,IPORT)

C                  CHARLES HENRY JULY 7 1989
C----------------------------------------
C     THIS PROGRAM IS CALLED GIN.FOR.  IT WILL DEFINE A "GIN"( GRAPHICS
C     INPUT, OUTPUT ) CURSORFOR USE IN PROGRAM.
C      THIS WILL EITHER ENABLE THE MOUSE,PUCK OR THUMBWHEELS.
C     THE DEFAULT IS SET FOR THE MOUSE USING THE ARROW CURSOR IN PICK
C     MODE
C
C     X1,Y1 - NEW LOCATION FOR THE CURSOR.  DEFINED BY USER
C
C     IFLAG - FLAG THAT MAKES ALL SEGMENTS VISIBLE. THIS IS USEFUL WHEN
C          THE USER IS BUILDING SOMETHING WITH ALL SEGMENTS TURNED
C          OFF AND THE GIN SUBROUTINE IS CALLED LAST.  THE DEFAULT
C          IS TO ISSUE THE COMMAND EVEN IF NOTHING IS OFF
C
C     IMODE - THIS IS THE GIN MODE.  THERE ARE THREE TYPES
C          EACH TYPE LETS YOU PERFORM SLIGHTLY DIFFERENT TASKS.
C
C          PICK RETURNS BUTTON PRESSED,X/Y LOCATION,VIEW NUMBER AND
C          WHICH SEGMENT WAS PICKED.
C
C          LOCATE RETURNS A SINGLE REPORT CONTAINING THE BUTTON
C          PRESSED,X/Y LOCATION,AND VIEW NUMBER IF SPECIFIED.
C            SINCE THE LOCATE MODE FOR THIS PROGRAMS USE IS NOT MUCH
C          DIFFERENT THEN THE PICK.  I AM USING IT HERE TO DRAW AND
C          USE GIN RUBBERBANDING.
C
C          STROKE RETURNS ONE OR MORE FIRST POINTS WHICH ARE SEPARATED
C          BY DIFFERENT FIRST CHARACTERS,AND A LAST POINT.THE FIRST
C          REPORT WILL INCLUDE THE BUTTON SELECTED THEN THE SUBSEQUENT
C          STROKE POINT CHARACTERS WILL BE A" ] " FOLLOWED BY THE X/Y
C
C          THE LAST POINT WILL BE A " O " FOLLOWED BY THE X/Y.
C
C          0 - PICK
C          1 - LOCATE
C          2 - STROKE
C
C
C     ITYPE - THIS IS THE CURSOR SEGMENT TYPE.  NORMALLY ITS AN ARROW
C          BUT THERE ARE OCCASSIONS WHEN A CROSSHAIR IS USEFUL
C          ADDITIONALLY,  WHEN THE GIN IS ACTIVE AND THE USER IS
C          SELECTING A MENU ITEM THAT REQUIRES NO GIN DEVICE AN
C          ALTERNATE SEGMENT IS PROVIDED THAT IS A DOT.  IF THE
C          USER ALREADY HAS A SEGMENT THAT IS NEEDED FOR THE CURSOR
C          THEN A 3 IS PASSED IN ITYPE AND THE INTEGER VALUE IS
C          PASSED IN IPORT.
C
C          0 - ARROW SEGMENT
```

```
C          1 - CROSSHAIR
C          2 - DOT
C          3 - NUMBER OF SEGMENT TO BE MADE THE CURSOR
C
C     IGIN - THIS IS THE GRAPHICS INPUT DEVICE.  THERE CAN BE FOUR OR
C          FIVE DIFFERENT DEVICES.  THE ONES WE USE ARE LISTED BELOW
C
C          0 - MOUSE
C          1 - THUMBWHEELS
C          2 - PUCK (LARGE TABLET) ..(THIS IS SPECIFIED BY THE PORT)
C
C**************************************************************

      INTEGER X,Y,X1,Y1
      CHARACTER*5 A,A1,A2,A3,A4,A5,A6
      CHARACTER E,AAA*12,G*2,GS,US,SEC*3,A10*6,B*1

      E=CHAR(27)
      GS=CHAR(29)     ! START VECTOR MODE CHAR
      US=CHAR(31)     ! STOP VECTOR MODE CHAR

C------------------------------------------
C     THE FOLLOWING IS THE SETUP FOR THE PARTICULAR GIN DEVICE THE
C     USER IS REQUESTING.  exp...USER MAY WANT THE MOUSE TO MOVE THE
C     CROSSHAIR IN STROKE MODE.
C
C
C          IGIN - 0      MOUSE
C          ITYPE - 1     CROSSHAIR
C          IMODE - 2     STROKE MODE
C
C     BELOW WE START DEFINING WITH THE PICK FUNCTION
C     I HAVE TO DEFINE THE NUMBER OF CHARACTERS IN THE G STRING*** SO
C     THAT THERE WONT BE ANY INNER SPACING IN THE ESCAPE
C     COMMANDS.  THAT WOULD CAUSE A FATAL ERROR.  THIS IS STORED IN "I2"
C
C------------------------------------------

      WRITE(*,*)E//'ID!'          ! DELETE THE GIN SEGMENT
      WRITE(*,*)E//'SKC)1'        ! DELETE THE GIN CURSOR
      WRITE(*,*)E//'LV0'          ! DISABLE DIALOG AREA

      IF(IMODE.EQ.0)THEN             ! USER WANTS PICK FUNCTION
       IF(IGIN.EQ.0)THEN          ! SET UP FOR MOUSE
        G - 'D1'
        I2 - 2                ! NUMBER  CHARA USED IN STRING
       ELSEIF(IGIN.EQ.1)THEN
        G - '1'               ! SET FOR THUMBWHEELS
        I2 - 1                ! NUMBER  CHARA USED IN STRING
       ELSEIF(IGIN.EQ.2)THEN
        IF(IPORT.EQ.0)G - 'A9'     ! TABLET PORT 0
        IF(IPORT.EQ.1)G - 'B1'     ! TABLET PORT 1
        IF(IPORT.EQ.2)G - 'B9'     ! TABLET PORT 2
        I2 - 2                ! NUMBER  CHARA USED IN STRING
```

```
          ENDIF

C-----------------------------------------
C     HERE WE BEGIN WITH THE LOCATE FUNCTION
C-----------------------------------------

        ELSEIF(IMODE.EQ.1)THEN        ! USER WANTS THE LOCATE FUNCTION
         IF(IGIN.EQ.0)THEN          ! SET UP FOR MOUSE
          G = 'D0'
          I2 = 2               ! NUMBER  CHARA USED IN STRING
         ELSEIF(IGIN.EQ.1)THEN
          G = '0'              ! SET FOR THUMBWHEELS
          I2 = 1               ! NUMBER  CHARA USED IN STRING
         ELSEIF(IGIN.EQ.2)THEN
          IF(IPORT.EQ.0)G = 'A8'    ! TABLET PORT 0
          IF(IPORT.EQ.1)G = 'B0'    ! TABLET PORT 1
          IF(IPORT.EQ.2)G = 'B8'    ! TABLET PORT 2
          I2 = 2               ! NUMBER  CHARA USED IN STRING
         ENDIF

C-----------------------------------------
C     THIS PART DEFINES THE STROKE FUNCTIONS
C     THESE ARE ONLY ALLOWED FOR THE TABLET AND THE MOUSE.
C
C-----------------------------------------


        ELSEIF(IMODE.EQ.2)THEN        ! USER WANTS THE STROKE FUNCTION
         IF(IGIN.EQ.0)THEN          ! SET UP FOR MOUSE
          G = 'D2'
          I2 = 2               ! NUMBER  CHARA USED IN STRING
         ELSEIF(IGIN.EQ.1)THEN
          G = ':'              ! SET FOR THUMBWHEELS
          I2 = 1               ! NUMBER  CHARA USED IN STRING
         ELSEIF(IGIN.EQ.2)THEN
          WRITE(*,*)'YOU MUST NOT SPECIFY A PORT FOR STROKE'
          GOTO 1000
         ENDIF
        ENDIF

C-----------------------------------------


        WRITE(*,*)E//'SV'//'0'         ! MAKE CURSOR INVISIBLE

  4     IF(ITYPE.EQ.1.OR.ITYPE.EQ.3)GOTO 30  ! DON'T DRAW A SEGMENT

C-----------------------------------------
C     THE USER IS REQUESTING THAT A DOT BE DRAWN INSTEAD OF THE ARROW
C-----------------------------------------

        IF(ITYPE.EQ.2)THEN          ! THIS IS THE DOT SEGMENT
         WRITE(*,*)E//'ML0'//E//'MT0'//E//'MM0'

```

```
      C--------------------------------------------
     C     THE SEGMENT IS A SMALL INVISIBLE MARK THAT IS USED WHEN THE USER
     C     IS SELECTING MENU ITEMS FROM THE DIALOG AREA.  ALL THAT IS
     C     SEARCHED FOR IS THE BUTTON THAT WAS PRESSED.
     C--------------------------------------------


           WRITE(*,*)E//'SOC]1'  ! BEGIN SEGMENT NO. 4049
           CALL HIY(X1,Y1,A)          ! PLACE THE SEGMENT AT X,Y LOCAL
           WRITE(*,*)E//'LH'//A  ! DRAW THE SEGMENT
           WRITE(*,*)E//'SC'          ! CLOSE THE SEGMENT
           GOTO 30                    ! JUMP OVER ARROW DRAW
          ENDIF

     C--------------------------------------------
     C     THIS BEGINS THE DRAWING OF THE ARROW CURSOR.
     C--------------------------------------------

     5    CONTINUE
          WRITE(*,*)E//'SC'
          X-0
          Y-100
          CALL HIY(X,Y,A)
          WRITE(*,*)E//'SP'//A          ! SET PIVOT POINT FOR CURSOR

          WRITE(*,*)E//'SOC]1'
          WRITE(*,*)E//'MP''//E//'ML1'//E//'MT1'

     C--------------------------------------------
     C     THIS PART DRAWS THE ARROW SEGMENT.  IT IS DRAWN AT THE 0,0 ORIGIN
     C     AND THEN IS MOVED TO THE USER DEFINED LOCATION.
     C--------------------------------------------

          X-0
          Y-100
          CALL HIY(X,Y,A)
          X-80
          Y-70
          CALL HIY(X,Y,A1)
          X-40
          Y-70
          CALL HIY(X,Y,A2)
          X-102
          Y-12
          CALL HIY(X,Y,A3)
          X-95
          Y-0
          CALL HIY(X,Y,A4)
          X-30
          Y-60
          CALL HIY(X,Y,A5)
          X-30
          Y-20
          CALL HIY(X,Y,A6)
```

```
      WRITE(*,*)E//'LP'//A//'1'//GS//A1//A2//A3//A4//A5//A6//
     . US//E//'SC'

   20 CONTINUE

   30 IF(ITYPE.EQ.1)GOTO 35
      WRITE(*,*)E//'SVC}10'          ! MAKE CURSOR INVISIBLE
   35 WRITE(*,*)E//'SV"1'            ! MAKE CURSOR VISIBLE

      IF(IMODE.EQ.1)THEN              ! USER WANTS THE LOCATOR MODE

       IF(ITYPE.EQ.0.OR.ITYPE.EQ.2)THEN

        WRITE(*,*)E//'IT'//G(1:I2)//'1'//
     .  E//'IR'//G(1:I2)//'1'//E//'IC'//G(1:I2)//'C}1'//
     .  E//'IE'//G(1:I2)//'0'
       ELSE
        WRITE(*,*)E//'IT'//G(1:I2)//'1'//
     .  E//'IR'//G(1:I2)//'1'//E//'IE'//G(1:I2)//'0'
       ENDIF


      ELSEIF(IMODE.EQ.2)THEN          ! USER WANTS STROKE MODE

       IF(ITYPE.EQ.0.OR.ITYPE.EQ.2)THEN
        WRITE(*,*)E//'IC'//G(1:I2)//'C}1'//
     .  E//'IP'//G(1:I2)//'0P4'//E//'IE'//G(1:I2)//'0'
       ELSE
        WR  TE(*,*)E//'IP'//G(1:I2)//'P40'//
     .  E//'IE'//G(1:I2)//'0'
       ENDIF

      ELSE                            ! USER WANTS PICK MODE

       IF(ITYPE.EQ.0.OR.ITYPE.EQ.2)THEN
        WRITE(*,*)E//'IC'//G(1:I2)//'C}1'//
     .  E//'IE'//G(1:I2)//'0'

       ELSEIF(ITYPE.EQ.3)THEN         ! USER WANTS A SEGMENT TO BE
        CALL INTRPT(IPORT,SEG)        ! THE GIN CURSOR
        WRITE(*,*)E//'IC'//G(1:I2)//SEG//
     .  E//'IE'//G(1:I2)//'0'//E//'SM'//SEG//'1'

       ELSE

        WRITE(*,*)E//'IE'//G(1:I2)//'0'  ! SET MOUSE FOR PICK
                                         ! FUNCTIONS
       ENDIF
      ENDIF

C-------------------------------------

      CALL HIY(X1,Y1,A)
```

```

      IF(ITYPE.EQ.0)THEN            ! IF ITS SEGMENT
        WRITE(*,*)E//'SXC]1'//A     ! POSITION THE CURSOR
      ELSEIF(ITYPE.EQ.1)THEN        ! IF ITS CROSS HAIR
        WRITE(*,*)E//'SX0'//A       ! POSITION THE CURSOR
      ELSEIF(ITYPE.EQ.3)THEN        ! IF ITS CROSS HAIR
        WRITE(*,*)E//'SX'//SEG//A   ! POSITION THE CURSOR
      ENDIF                         ! THE DOT IS ALREADY
                                    ! PLACED

1000  ICUR=0                        ! RESET THE TOGGLE LOOP
      IMODE = 0                     ! RESET THE GIN TYPE FLAG
      ITYPE = 0                     ! RESET THE CURSOR TYPE TO ARROW

      IF(ITYPE.EQ.1)GOTO 1010
      WRITE(*,*)E//'SVC]11'         ! MAKE CURSOR INVISIBLE
1010  WRITE(*,*)E//'LZ'             ! CLEAR DIALOG AREA
      RETURN
      END
```

AIV.8 Analog APSD Software Modules: HEXDEC Source Code.

```
      SUBROUTINE HEXDEC(HEXIN,RDAT,CHANNELS)

C------------------------------------------
C      THIS PROGRAM WILL RECEIVE INPUT FROM THE A/D CONVERTER
C      CONVERT IT TO BINARY, CHECK TO SEE IF NUMBER IS POSITIVE
C      OR NEGATIVE.  IF POS THE NUMBER WILL BE CONVERTED TO DECIMAL
C      AND SAVED.  IF NEG, A 1 WILL BE SUBTRACTED AND THE BITS
C      REVERSED (2's COMPLIMENT) AND THEN RETURNED TO DECIMAL
C
C      HEXIN - ARRAY OF HEX CHARACTERS PASSED IN FROM THE CALLER
C
C      RDAT  - ARRAY OF DECIMAL REALS PASSED OUT TO THE CALLER
C
C      CHANNELS - THE NUMBER OF ARRAY ELEMENTS PASSED IN FROM THE CALLER
C------------------------------------------

      CHARACTER*4 HEXIN,A,PLACE*1

      INTEGER K,ICNT,BIN,CHANNELS

      REAL RDAT,RT

      DIMENSION BIN(16),PLACE(16),HEXIN(16),IDATA(16),RDAT(16)

      DATA (BIN(I),I=1,16)/32768,16384,8192,4092,2048,
     . 1024,512,256,128,64,32,16,8,4,2,1/



    1 FORMAT (A4)
    2 FORMAT (T10,8(A1,A1))

C---------- TEST DATA ------------------
C
C     I = 1
C
C5    WRITE(*,*)' ENTER UP TO 16 HEX NUMBERS:  ( 4 CHAR MAX )'
C     WRITE(*,*)' ENTER " Z " WHEN DONE'
C     WRITE(*,*)
C     READ(*,'(A4)',ERR = 1000)HEXIN(I)
C
C     IF(HEXIN(I)(1:1).EQ.'Z'.AND.I.EQ.1)GOTO 1000
C     IF(HEXIN(I)(1:1).EQ.'Z')GOTO 10
C
C     CHANNELS = I
C
C     I = I + 1
C     GOTO 5
C
C------------------------------------------
```

```
10    DO 200 KK-1,CHANNELS

      ICNT-0
      J - 0
      K - 0
      IBUFF - 0
      A - HEXIN(KK)
      J - -1

      DO 40 II-4,1,-1
       J - J + 1

        IF(ICHAR(A(II:II)).GT.64) THEN
          K - ICHAR(A(II:II)) - 55
          GOTO 35
        ENDIF

       K - ICHAR(A(II:II)) - 48

35      ICNT - ICNT + K * 16**J

40    CONTINUE

      IBUFF - ICNT

C----------------------------------------------------------
C     THE FOLLOWING MOD CHANGES THE INTEGERS TO BINARY NUMBERS
C     ONLY THE 16TH PLACE HAS TO BE LOOKED AT. IF THIS NUMBER IS
C     '1' THEN WE HAVE A NEGATIVE NUMBER AND WE WILL DO A TOTAL
C     CONVERSION.
C----------------------------------------------------------

      IF(ICNT.LT.BIN(1)) GOTO 100

C----------------------------------------------------------
C     IF THE NUMBER IS NEG ALL THE 1'S AND 0'S ARE REVERSED
C----------------------------------------------------------

      IPUT - 17

      DO 70 II - 1,16

      IPUT - IPUT - 1

      IF(IBUFF.GE.BIN(II)) THEN
        IBUFF - IBUFF - BIN(II)
        PLACE(IPUT) - '0'
      ELSE
        PLACE(IPUT) - '1'
      ENDIF

70    CONTINUE
```

```
C----------------------------------------
C     THIS WRITES THE BINARY NUMBER TO THE SCREEN
C----------------------------------------
C
C     WRITE(*,2)PLACE(16),PLACE(15),PLACE(14),PLACE(13),PLACE(12),
C    1 PLACE(11),PLACE(10),PLACE(9),PLACE(8),PLACE(7),PLACE(6),PLACE(5),
C    1 PLACE(4),PLACE(3),PLACE(2),PLACE(1)
C----------------------------------------
C     THIS PART CONVERTS THE NUMBER BACK TO DECIMAL, CHANGES THE
C     SIGN AND THEN SUBTRACTS 1
C----------------------------------------

      JJ - 1
      ICNT - 0

      DO 90 II-16,1,-1

       IF(PLACE(II).EQ.'1') THEN
        ICNT - ICNT + BIN(JJ)
       ENDIF

       JJ -JJ + 1

90    CONTINUE

      ICNT - ICNT - ( 2 * ICNT )
      ICNT - ICNT - 1
100   IDATA(KK) - ICNT

200   CONTINUE

C----------------------------------------

      DO 300 I - 1,CHANNELS

       IF(IDATA(I).EQ.0.0)THEN
        RDAT(I) - 0.000
        GOTO 300
       ENDIF

      IDATA(I) - IDATA(I) * -1

      IF(IDATA(I).GT.0)THEN
       IDATA(I) - 32676 - IDATA(I)
      ELSE
       IDATA(I) - -(IDATA(I) + 32676)

      ENDIF

C      RDAT(I) - (FLOAT(IDATA(I))/16384.0) + 0.0022  ! NORMALIZE VOLTAGE

       RDAT(I) - (FLOAT(IDATA(I))/8192.0) + 0.0022  ! NORMALIZE VOLTAGE
C      RDAT(I) - (FLOAT(IDATA(I))/4096.0) + 0.0022  ! NORMALIZE VOLTAGE

```

```
  300  CONTINUE


C---------------------------------------------
C    NO NEED TO GO ANY FURTHER IF THE DC ELEMENT = 0
C---------------------------------------------

C    IF(RDAT(1).LE.0.0)GOTO 1000              ! ERROR CONDITION

      RT = 1 / RDAT(1)                ! NORMALIZATION FACTOR

C---------------------------------------------
C    THIS NORMALIZES ALL THE ELEMENTS WITH THE " 11 " ELEMENT
C---------------------------------------------

      DO 400 I = 1, CHANNELS

        RDAT(I) = RDAT(I) * RT

 400  CONTINUE

C---------------------------------------------

1000  RETURN
      END
```

AIV.9 Analog APSD Software Modules: HIY Source Code.

```
      SUBROUTINE HIY(X,Y,A)
C-----------------------------------------
C     THIS ROUTINE IS A UTILITY THAT CONVERTS TWO INTEGER (X,Y) SCREEN
C     COORDINATES INTO TEKTRONICS CHARACTER CODE SCHEME.  THE TEK
C     TERMINALS HAVE THEIR OWN GRAPHICS LANGUAGE.  THIS LANGUAGE REQUIRES
C     THE HOST TO ISSUE EVERY COMMAND AS A CHARACTER STRING.  IN THIS
C     CASE THE TWO VECTORS (X,Y)ARE CONVERTED INTO A FIVE CHARACTER
C     STRING.  THESE REPRESENT BIT POSITIONS FOR THE TERMINAL AS FOLLOWS:
C     HIY, Extra,  LoY,  HiX,  and LoX.
C          THIS UTILITY IS ONE OF THE MOST IMPORTANT UTILITIES FOR
C     ANY HOST TO TEKTRONICS GRAPHICS APPLICATIONS.
C     NOTE:  X,Y REMAIN UNCHANGED WHEN SENT BACK TO CALLER.
C-----------------------------------------

      INTEGER *2 X,X0,X1,X2,Y,Y0,Y1,Y2
      CHARACTER *5 A

      X2-X/128
      Y2-Y/128
      X1-(X-X2*128)/4
      Y1-(Y-Y2*128)/4
      X0-MOD(X,4)
      Y0-MOD(Y,4)

C-----------------------------------------
C     HERE THE INTEGERS ARE CHANGED TO A CHARACTER STRING.
C-----------------------------------------

      A(1:1)-CHAR(32+Y2)
      A(2:2)-CHAR(96+Y0*4+X0)
      A(3:3)-CHAR(96+Y1)
      A(4:4)-CHAR(32+X2)
      A(5:5)-CHAR(64+X1)

      RETURN
      END
```

AIV.10 Analog APSD Software Modules: INT_TO_CHAR Source Code.

```
      SUBROUTINE INT_TO_CHAR(NUMBER,CNUM,LENGTH)

C-------------------------------------------
C     THIS ROUTINE WILL CHANGE INTEGERS FROM +/- 99,999,999 INTO
C     CHARACTERS.
C
C     NUMBER - INTEGER PASSED IN FROM THE CALLER
C
C     CNUM   - CHARACTER REPRESENTATION OF THE ABOVE INTEGER THAT IS
C              RETURNED TO THE CALLER.
C
C     LENGTH - THE NUMBER OF CHARACTERS BEING SENT BACK TO THE CALLER
C-------------------------------------------

      CHARACTER *9 CNUM,CBUF

5     CNUM - ' '                ! INITIALIZE CHARACTER VALUE

C     WRITE(*,*)' ENTER A NUMBER:'
C     WRITE(*,*)
C     READ(*,*)NUMBER

      NUMBUF - NUMBER
      IF(NUMBER.LT.0)THEN          ! ITS A NEGITIVE NUMBER
       NEG - 1                     ! SET A FLAG THAT THE NUMBER IS
       NUMBER - ABS(NUMBER)        ! NEGITIVE.  TAKE ITS ABSOLUTE
      ENDIF

      IF(NUMBER.EQ.0)THEN          ! THE NUMBER IS ZERO
       CNUM - '000000000'          ! THE CHARACTER - ALL 0's
       GOTO 1000                   ! RETURN TO CALLER

      ELSEIF(NUMBER.LT.10)THEN     ! ITS A SINGLE DIGIT NUMBER
       IN - 1                      ! CHARACTER PLACE COUNTER
       CNUM(1:1) - CHAR(NUMBER + 48) ! CONVERT NUMBER TO CHARACTER
       GOTO 110

      ELSEIF(NUMBER.GT.99999999.OR.NUMBER.LT.-99999999)THEN
       CNUM - '000000000'          ! IF THE NUMBER IN IS TO LARGE
       GOTO 1000                   ! CNUM CANT CONVERT THIS NUMBER
      ENDIF

C-------------------------------------------
C     THIS PART SEPARATES EACH INTEGER PLACE VALUE INTO ITS INDIVIDUAL
C     COMPONENTS.  IT THEN CONVERTS EACH COMPONENT INTO A CHARACTER
C     AND PLACES THAT CHARACTER IN THE CNUM STRING.
C-------------------------------------------

      M - 0                        ! PLACE VALUE BUFFER
      J - 1                        ! PLACE VALUE
      IN - 0                       ! CHARACTER PLACE COUNTER
```

```
      DO 10 I = 1,8              ! LOOP THRU THE PLACE VALUES
       IF(NUMBER.LT.J)GOTO 20    ! LOOKING FOR THE PLACE VALUE
       J = J * 10                ! INCREMENT PLACE VALUE
 10   CONTINUE

C-------------------------------------
C     HERE I GET THE ACTUAL INTEGER VALUE THAT RESIDES IN EACH PLACE
C     VALUE. ie...   THOUSANDS PLACE (PLACE VALUE) = 5
C                    HUNDREDS  PLACE (PLACE VALUE) = 3
C                    TENS      PLACE              = 9
C                    ONES      PLACE              = 2
C-------------------------------------

 20   I = I - 2
      J = 10 ** I * 9            ! J GOES TO TOP OF THIS PLACE
      K = 10 ** I                ! K = BOTTOM OF THIS PLACE VALUE

      DO 100 LOOP = 1,I + 1      ! LOOP THRU THE PLACE VALUES

       DO 40 NUM = J,0,-K        ! FIND THE INTEGER FOR THIS PLACE
        L = NUM - NUMBER
        M = M + 1                ! ACTUAL INTEGER VALUE WANTED
        IF(L.LE.0)GOTO 50        ! VALUE IS FOUND
 40    CONTINUE

C-------------------------------------
C     THIS BUILDS THE CHARACTER STRING FROM EACH PLACE VALUE INTEGER
C-------------------------------------

 50   IN = IN + 1                ! CHARACTER PLACE COUNTER
      NUM = 10 - M               ! ACTUAL INTEGER SOUGHT
      CNUM(IN:IN) = CHAR(NUM + 48)  ! CONVERT INTEGER TO CHARACTER

      M = 0                      ! RESET THE INTEGER COUNTER
      NUMBER = NUMBER - ( K * NUM )  ! DECREMENT THE NUMBER
      I = I - 1
      J = 10 ** I * 9            ! DECREMENT TOP VALUE OF PLACE
      K = 10 ** I                ! DECREMENT BOTTOM VALUE OF PLACE

 100  CONTINUE                   ! END OF LOOP

C-------------------------------------
C     IF THE NUMBER WAS NEGITIVE I PLACE A MINUS SIGN IN FRONT OF THE
C     CHARACTER STRING BEFORE I SEND IT OUT.
C-------------------------------------

 110  IF(NEG.EQ.1)THEN           ! THE NUMBER IS NEGITIVE
       NEG = 0                   ! RESET THE NEGITIVE INTEGER FLAG
       CBUF = CNUM               ! MAKE A COPY OF THE CHARACTER STR
       L = 1                     ! INITIALIZE CHARACTER PLACE CNTR

       DO 120 I = 1,IN           ! LOOP THRU THE CHARACTER STRING
        L = L + 1                ! INCREMENT CHARACTER COUNTER
```

```
            CNUM(L:L) - CBUF(I:I)     ! SHIFT THE STRING 1 PLACE RIGHT
   120   CONTINUE

         CNUM(1:1) - '-'          ! ADD THE MINUS SIGN TO FIRST CHR
         IN - IN + 1              ! CORRECT THE CHARACTER COUNTER
      ENDIF

      LENGTH - IN

C      WRITE(*,*)' THE CHARACTER - ',CNUM,'  IN - ',LENGTH
C
C      GOTO 5

1000  NUMBER - NUMBUF
      RETURN
      END
```

AIV.11 Analog APSD Software Modules: INTRPT Source Code.

```
      SUBROUTINE INTRPT(INT,CH)
C-----------------------------------------
C     THIS ROUTINE WILL TAKE AND INTEGER AND TURN IT INTO A
C     CHARACTER STRING EQUIVALENT TO THE INTEGER FOR THE
C     TEK TERMINALS.
C*************************************************************

      CHARACTER CH*6,C
      INTEGER INT,X1,X2
      DIMENSION C(6)
      ICNT-0
      CH-' '
      L-0
      X1-MOD(INT,16)
      X2-INT/16
      IF(INT.GT.0) THEN
       X1-X1+48
      ELSE
       X1--X1+32
      ENDIF
      ICNT-ICNT+1
      C(1)-CHAR(X1)
      IF(INT.LE.0)GOTO 30
      DO 20 I-2,6
       IF (X2.GE.64)THEN
        X1-MOD(X2,64)
        X2-X2/64
        X1-X1+64
        C(I)-CHAR(X1)
        ICNT-ICNT+1
       ELSE
        X1-X2+64
        C(I)-CHAR(X1)
        ICNT-ICNT+1
        GOTO 50
       ENDIF
   20 CONTINUE
      GOTO 50
   30 DO 40 I-2,6
       IF (X2.LE.-64)THEN
        X1-MOD(X2,64)
        X2-X2/64
        X1--X1+64
        C(I)-CHAR(X1)
        ICNT-ICNT+1
       ELSE
        X1--X2+64
        C(I)-CHAR(X1)
        ICNT-ICNT+1
        GOTO 50
       ENDIF
```

```
40    CONTINUE
50    L-7
      IF(INT.LT.16.AND.INT.GT.-16)ICNT-1
      DO 60 J-1,ICNT
       L-L-1
       CH(L:L)-C(J)
60    CONTINUE
      L-0
      DO 70 I-1,6
       IF(CH(I:I).EQ.' ')GOTO 70   ! IS THIS A BLANK SPACE
       L-L+1
       C(L)-CH(I:I)
70    CONTINUE                     ! THESE DO LOOPS PLACE THE
                                   ! CHARACTERS ON THE LEFT
      CH-' '
      DO 80 I-1,L
       CH(I:I)-C(I)
80    CONTINUE
      L-0
      RETURN
      END
```

AIV.12 Analog APSD Software Modules: LASER_IN Source Code.

```
      SUBROUTINE LASER_IN(SAMP,AGENT,CONC,NOSAMP,NAME,LAS,LAMDA,
     .                 LAS_ORD,IEXIT)
C-----------------------------------------
C     THIS MOD IS USED FOR ENTERING SPECIFIC LASER DATA FOR EACH SAMPLE
C     IF THE USER WANTS TEKTRONIX GRAPHICS.  THIS ROUTINE IS RUN JUST
C     AFTER THE SAMPLE INFORMATION GATHERING ROUTINE " TEK_INPUTS "
C-----------------------------------------


      CHARACTER NAME*20,SAMP*20,AGENT*20,CONC*15,LAMDA*15
      CHARACTER DATE*9,TIME*8,TEXT*80,E*1,A*5,SEG*3
      CHARACTER CORRECT*2,SAMPCNT,LASERS,COUNT*3
      CHARACTER SAMPLE*9,ERROR,L_ORDER

      INTEGER CHANGE,ERRMSG,X,Y,X1,Y1,Y2,Y3
      INTEGER LAS,LAS_ORD,ORDER

      REAL REND,RINCR,RARM
      REAL DIST

C-----------------------------------------

      DIMENSION SAMP(10),AGENT(10),CONC(10)
      DIMENSION LAS(10),LAMDA(40),LAS_ORD(40)
      DIMENSION START(10),STOP(10),INC(10),POS(10)

      DIMENSION COUNT(4)
      DATA COUNT /'1st','2nd','3rd','4th'/

    2 FORMAT(A20)
    3 FORMAT(A4)
    4 FORMAT('0')
    5 FORMAT(A2)
    6 FORMAT (A)
    7 FORMAT(I5)
    8 FORMAT (A10)

      E = CHAR(27)
      CHANGE = 0
      ISTART = 1
      ICNT = 1                    ! SAMPLE COUNTER
      ISAMP = 1
      IEXIT = 0

C-----------------------------------------

       CALL INT_TO_CHAR(NOSAMP,SAMPLE,LE)   ! CONVERT TO CHARACTER

C-----------------------------------------
C     THIS PART PLACES A RED PANEL SEGMENT 8000 NO THE ENTIRE SCREEN
C     FOR A BACKGROUND.
```

```
C-----------------------------------------

      WRITE(*,*)E//'RP0'          ! FIXUP LEVEL 0
      WRITE(*,*)E//'SK1'          ! DELETE ALL SEGMENTS
      WRITE(*,*)E//'RP6'          ! FIXUP LEVEL NORMAL

      ISEG - 8000
      CALL INTRPT (ISEG,SEG)
      WRITE(*,*)E//'SE'//SEG       ! BEGIN THE PANEL 8000
      WRITE(*,*)E//'MP-'          ! PANEL COLOR RED
      X - 1
      Y - 1
      CALL HIY(X,Y,A)
      WRITE(*,*)E//'LP'//A        ! SET PANEL ORIGIN

      X - 4095
      CALL HIY(X,Y,A)
      WRITE(*,*)E//'LG'//A        ! DRAW BOTTOM OF PANEL
      Y - 3276
      CALL HIY(X,Y,A)
      WRITE(*,*)E//'LG'//A        ! DRAW LEFT SIDE OF PANEL
      X - X - 4094
      CALL HIY(X,Y,A)
      WRITE(*,*)E//'LG'//A        ! DRAW TOP OF PANEL
      WRITE(*,*)E//'SC'          ! CLOSE  AND FILL PANEL

C-----------------------------------------
C     THIS IS THE FIRST LINE THAT IS REQUIRED BY THE USER.
C     NAME, DATE, TIME AND NUMBER OF SAMPLES.
C-----------------------------------------

      CALL TIME4(DATE,TIME)            ! SYSTEM TIME AND DATE

      ISEG - 1
      CALL INTRPT(ISEG,SEG)



      WRITE(*,*)E//'SE'//SEG       ! BEGIN THE SEGMENT
      WRITE(*,*)E//'MT7'          ! LINE COLOR WHITE
      X - 100
      Y - 3000
      CALL HIY (X,Y,A)
      WRITE(*,*)E//'LP'//A        ! SET TEXT ORIGIN
      WRITE(*,*)E//'LT9'//DATE     ! WRITE THE DATE
      X - 1000
      CALL HIY (X,Y,A)
      WRITE(*,*)E//'LP'//A        ! SET TEXT ORIGIN
      WRITE(*,*)E//'LT8'//TIME     ! WRITE THE TIME

      WRITE(*,*)E//'MT2'          ! LINE COLOR RED
      X - 2000
      CALL HIY (X,Y,A)
      WRITE(*,*)E//'LP'//A        ! SET TEXT ORIGIN
```

```
      TEXT - ' NAME:'
      WRITE(*,*)E//'LTY'//TEXT          ! WRITE THE DATE
      WRITE(*,*)E//'SC'

C------------------------------------------------
C     THIS PART DRAWS  A BOX AT THE BOTTOM OF THE SCREEN FOR THE DIALOG
C     AREA INPUTS.
C------------------------------------------------

      ISEG - ISEG + 1
      CALL INTRPT (ISEG,SEG)
      WRITE(*,*)E//'SE'//SEG            ! BEGIN THE PANEL 8000
      WRITE(*,*)E//'ML1'                ! LINE COLOR WHITE
      WRITE(*,*)E//'MP5'                ! PANEL COLOR BLUE
      X - 1
      Y - 1
      CALL HIY(X,Y,A)
      WRITE(*,*)E//'LP'//A//'1'         ! SET PANEL ORIGIN
      X - 4095
      CALL HIY(X,Y,A)
      WRITE(*,*)E//'LG'//A              ! DRAW BOTTOM OF PANEL
      Y - 450
      CALL HIY(X,Y,A)
      WRITE(*,*)E//'LG'//A              ! DRAW LEFT SIDE OF PANEL
      X - X - 4094
      CALL HIY(X,Y,A)
      WRITE(*,*)E//'LG'//A              ! DRAW TOP OF PANEL
      WRITE(*,*)E//'LE'                 ! FILL THE PANEL

C------------------------------------------------
C     THIS DRAWS A LINE AROUND THE TWO DIALOG AREA LINES:  COLOR RED
C------------------------------------------------

      WRITE(*,*)E//'ML2'                ! PANEL COLOR RED
      X - 100
      Y - 150
      CALL HIY(X,Y,A)
      WRITE(*,*)E//'LP'//A              ! SET PANEL ORIGIN
      X - 3995
      CALL HIY(X,Y,A)
      WRITE(*,*)E//'LG'//A              ! DRAW BOTTOM OF PANEL
      Y - Y + 250
      CALL HIY(X,Y,A)
      WRITE(*,*)E//'LG'//A              ! DRAW LEFT SIDE OF PANEL
      X - 100
      CALL HIY(X,Y,A)
      WRITE(*,*)E//'LG'//A              ! DRAW TOP OF PANEL
      Y - Y - 250
      CALL HIY(X,Y,A)
      WRITE(*,*)E//'LG'//A              ! DRAW LEFT SIDE OF PANEL

C------------------------------------------------

      WRITE(*,*)E//'MT1'                ! TEXT COLOR WHITE
```

```
      TEXT - 'ENTER < E >  TO EXIT'

      X - 1600
      Y - 45
      CALL HIY(X,Y,A)
      WRITE(*,*)E//'LP'//A            ! SET TEXT ORIGIN
      WRITE(*,*)E//'LTA4'//TEXT         ! WRITE THE TEXzz-%y<nJTnu
      WRITE(*,*)E//'SC'              ! CLOSE  AND FILL PANEL

C--------------------------------------------------
C     THIS SETS UP THE DIALOG AREA SO THAT THE TEXT AND DATA ENTRY
C     ARE ALL WITHIN THE DATA INPUT WINDOW.
C--------------------------------------------------

      WRITE(*,*)E//'LV0'             ! DISABLE DIALOG AREA
      WRITE(*,*)E//'LZ'              ! CLEAR DIALOG AREA
      WRITE(*,*)E//'LL2'             ! DIALOG AREA 2 LINES
      WRITE(*,*)E//'LCD1'            ! 65 CHARACTER ALLOWED
      WRITE(*,*)E//'ML1'             ! DIALOG TEXT WHITE
      X - 200
      Y - 160
      CALL HIY(X,Y,A)
      WRITE(*,*)E//'LX'//A           ! SET TEXT DIALOG ORIGIN



C--------------------------------------------------
C     THIS WRITES THE NAME OF THE PERSON RUNNING THE EXPERIMENT
C--------------------------------------------------
C**************

      ISEG - 5
      CALL INTRPT(ISEG,SEG)

      WRITE(*,*)E//'SK'//SEG          ! DELETE THE  SEGMENT

      WRITE(*,*)E//'SE'//SEG          ! BEGIN THE SEGMENT
      WRITE(*,*)E//'MP7'             ! PANEL COLOR GRAY
      WRITE(*,*)E//'MT1'             ! TEXT COLOR WHITE

      X - 2500
      Y - 2975
      CALL HIY(X,Y,A)
      WRITE(*,*)E//'LP'//A           ! SET PANEL ORIGIN
      X - X + 1100
      CALL HIY(X,Y,A)
      WRITE(*,*)E//'LG'//A           ! DRAW BOTTOM OF PANEL
      Y - Y + 100
      CALL HIY(X,Y,A)
      WRITE(*,*)E//'LG'//A           ! DRAW LEFT SIDE OF PANEL
      X - X - 1100
      CALL HIY(X,Y,A)
      WRITE(*,*)E//'LG'//A           ! DRAW TOP OF PANEL
```

```
        WRITE(*,*)E//'LE'              ! FILL THE PANEL

C-----------------------------------
C    THIS PLACES THE TEXT IN THE PANEL
C-----------------------------------

      X - 2550
      Y - 3000
      CALL HTY(X,Y,A)
      WRITE(*,*)E//'LP'//A          ! SET THE ORIGIN
      WRITE(*,*)E//'LTA4'//NAME          ! WRITE THE NAME
      WRITE(*,*)E//'SC'             ! CLOSE THE SEGMENT

C-----------------------------------
C    THIS DRAWS THE SAMPLE DATA HEADER AND NUMBER OF SAMPLES
C-----------------------------------

 40    ISEG - 7
      CALL INTRPT(ISEG,SEG)
      WRITE(*,*)E//'SK'//SEG          ! DELETE THE SEGMENT
      WRITE(*,*)E//'SE'//SEG          ! BEGIN THE SEGMENT

      X - 1
      Y - 2800
      CALL HTY(X,Y,A)
      WRITE(*,*)E//'ML4'            ! LINE COLOR BLUE
      WRITE(*,*)E//'LP'//A          ! SET LINE ORIGIN

      X - 4095
      CALL HTY(X,Y,A)
      WRITE(*,*)E//'LG'//A          ! SET LINE END

      WRITE(*,*)E//'MP7'            ! PANEL COLOR GRAY
      WRITE(*,*)E//'MT1'            ! TEXT COLOR WHITE

      X - 2500
      Y - 2530
      CALL HTY(X,Y,A)
      WRITE(*,*)E//'LP'//A          ! SET PANEL ORIGIN

      X - X + 1100
      CALL HTY(X,Y,A)
      WRITE(*,*)E//'LG'//A          ! DRAW BOTTOM OF BOX

      Y - Y + 100
      CALL HTY(X,Y,A)
      WRITE(*,*)E//'LG'//A          ! RIGHT SIDE OF BOX

      X - X - 1100
      CALL HTY(X,Y,A)
      WRITE(*,*)E//'LG'//A          ! TOP OF BOX
      WRITE(*,*)E//'LE'           ! FILL THE BOX

      WRITE(*,*)E//'MT1'            ! TEXT COLOR WHITE
```

```
      X = X + 100
      Y = Y - 75
      CALL HIY(X,Y,A)
      WRITE(*,*)E//'LP'//A              ! SET TEXT ORIGIN

      CALL INT_TO_CHAR(ICNT,SAMPCNT,LE)     ! CONVERT TO CHARACTER

       CALL INT_TO_CHAR(NOSAMP,SAMPLE,LE)    ! CONVERT TO CHARACTER
                                  ! THE NUMBER OF SAMPLES

      IF(NOSAMP.EQ.1)THEN
       WRITE(*,*)E//'LT?'//'1 OF 1 SAMPLES' ! WRITE THE TEXT
      ELSE

       WRITE(*,*)E//'LTA4'//SAMPCNT//' OF ',SAMPLE(1:LE)//
     .' SAMPLES'

      ENDIF

      WRITE(*,*)E//'SC'                 ! END SEGMENT

C-----------------------------------------
C     THIS WRITE THE COLUMN HEADERS FOR " SAMPLE NAMES ", " AGENT TYPES "
C     AND " CONCENTRATIONS "
C-----------------------------------------

      ISEG = 8
      CALL INTRPT(ISEG,SEG)
      WRITE(*,*)E//'SK'//SEG
      WRITE(*,*)E//'SE'//SEG              ! BEGIN THE SEGMENT

      WRITE(*,*)E//'MT7'                ! TEXT COLOR YELLOW
      X = 500
      Y = 2250
      CALL HIY(X,Y,A)
      WRITE(*,*)E//'LP'//A              ! SET TEXT ORIGIN
      TEXT = 'SAMPLE:'
      WRITE(*,*)E//'LT?'//TEXT            ! WRITE SAMPLE TEXT

      X = 1800
      CALL HIY(X,Y,A)
      WRITE(*,*)E//'LP'//A              ! SET TEXT ORIGIN
      TEXT = 'AGENT:'
      WRITE(*,*)E//'LT>'//TEXT            ! WRITE SAMPLE TEXT

      X = 3000
      CALL HIY(X,Y,A)
      WRITE(*,*)E//'LP'//A              ! SET TEXT ORIGIN
      TEXT = 'CONCENTRATION'
      WRITE(*,*)E//'LTA2'//TEXT           ! WRITE SAMPLE TEXT

      WRITE(*,*)E//'SC'
```

```
C------------------------------------------

50   ISEG = 10
     CALL INTRPT(ISEG,SEG)
     WRITE(*,*)E//'SK'//SEG
     WRITE(*,*)E//'SE'//SEG

     Y = 2100
     X = 200
     CALL HTY(X,Y,A)
     WRITE(*,*)E//'LP'//A              ! SET PANEL ORIGIN

     X = X + 1100
     CALL HTY(X,Y,A)
     WRITE(*,*)E//'LG'//A              ! DRAW BOTTOM OF BOX

     Y = Y + 100
     CALL HTY(X,Y,A)
     WRITE(*,*)E//'LG'//A              ! RIGHT SIDE OF BOX

     X = X - 1100
     CALL HTY(X,Y,A)
     WRITE(*,*)E//'LG'//A              ! TOP OF BOX
     WRITE(*,*)E//'LE'                 ! FILL THE BOX

     WRITE(*,*)E//'MT1'                ! TEXT COLOR WHITE

     X = X + 100
     Y = Y - 75
     CALL HTY(X,Y,A)
     WRITE(*,*)E//'LP'//A              ! SET TEXT ORIGIN

     WRITE(*,*)E//'LT'//SAMP(ICNT)      ! WRITE THE SAMPLE.
     Y = Y - 25

C------------------------------------------

     X = 1500
     CALL HTY(X,Y,A)
     WRITE(*,*)E//'LP'//A              ! SET PANEL ORIGIN

     X = X + 1100
     CALL HTY(X,Y,A)
     WRITE(*,*)E//'LG'//A              ! DRAW BOTTOM OF BOX

     Y = Y + 100
     CALL HTY(X,Y,A)
     WRITE(*,*)E//'LG'//A              ! RIGHT SIDE OF BOX

     X = X - 1100
     CALL HTY(X,Y,A)
     WRITE(*,*)E//'LG'//A              ! TOP OF BOX
     WRITE(*,*)E//'LE'                 ! FILL THE BOX

```

```
        WRITE(*,*)E//'MT1'             ! TEXT COLOR WHITE

      X - X + 100
      Y - Y - 75
      CALL HTY(X,Y,A)
      WRITE(*,*)E//'LP'//A           ! SET TEXT ORIGIN
      WRITE(*,*)E//'LT7'//AGENT(ICNT)
      Y - Y - 25

C------------------------------------

      X - 2800
      CALL HTY(X,Y,A)
      WRITE(*,*)E//'LP'//A           ! SET PANEL ORIGIN

      X - X + 1100
      CALL HTY(X,Y,A)
      WRITE(*,*)E//'LG'//A           ! DRAW BOTTOM OF BOX

      Y - Y + 100
      CALL HTY(X,Y,A)
      WRITE(*,*)E//'LG'//A           ! RIGHT SIDE OF BOX

      X - X - 1100
      CALL HTY(X,Y,A)
      WRITE(*,*)E//'LG'//A           ! TOP OF BOX
      WRITE(*,*)E//'LE'              ! FILL THE BOX

      WRITE(*,*)E//'MT1'             ! TEXT COLOR WHITE

      X - X + 100
      Y - Y - 75
      CALL HTY(X,Y,A)
      WRITE(*,*)E//'LP'//A           ! SET TEXT ORIGIN
      WRITE(*,*)E//'LT7'//CONC(ICNT)
      WRITE(*,*)E//'SC'

C------------------------------------
C     THIS NEXT PART GETS THE NUMBER OF AGENTS...
C------------------------------------

52    FORMAT(5X,'ENTER THE NUMBER OF LASERS ( 1 - 4 ): ',$)

      WRITE(*,*)E//'LV1'             ! ENABLE DIALOG AREA

60    WRITE(*,*)E//'LZ'              ! CLEAR THE DIALOG AREA
      WRITE(*,52)

      READ(*,6)LASERS                ! READ LASER COUNT

        IF(LASERS.EQ.'E'.OR.LASERS.EQ.'e')THEN   ! USER TO EXIT
          IEXIT - 1                  ! SET EXIT FLAG
          GOTO 1000                  ! RETURN TO CALLER
        ENDIF
```

```
      READ(LASERS,'(BN,I2)',ERR-60)LCNT     ! CONVERT TO INTEGER

      IF(LCNT.LT.1.OR.LCNT.GT.4)GOTO 60    ! BAD INPUT DO IT AGAIN

C------------------------------------------------
C     THIS BEGINS THE LASER INPUT SECTION...
C------------------------------------------------

70    ISEG  - 19

      X   - 350
      Y   - 1750

      CALL INTRPT(ISEG,SEG)
      WRITE(*,*)E//'SK'//SEG          ! DELETE THE SEGMENT
      WRITE(*,*)E//'SE'//SEG          ! BEGIN THE SEGMENT

      WRITE(*,*)E//'ML1'              ! LINE COLOR WHITE
      WRITE(*,*)E//'MP0'              ! PANEL COLOR GREEN
      WRITE(*,*)E//'MT7'              ! TEXT COLOR YELLOW


      CALL HIY(X,Y,A)
      WRITE(*,*)E//'LP'//A            ! SET TEXT ORIGIN
      TEXT - '0.'
      WRITE(*,*)E//'LT2'//TEXT         ! WRITE THE TXT

      WRITE(*,*)E//'MT4'              ! TEXT COLOR BLUE
      X - X + 150

      CALL HIY(X,Y,A)
      WRITE(*,*)E//'LP'//A//'1'       ! SET PANEL ORIGIN
      X - X + 700
      CALL HIY(X,Y,A)
      WRITE(*,*)E//'LG'//A            ! DRAW BOTTOM OF BOX

      Y - Y + 100
      CALL HIY(X,Y,A)
      WRITE(*,*)E//'LG'//A            ! RIGHT SIDE OF BOX

      X - X - 700
      CALL HIY(X,Y,A)
      WRITE(*,*)E//'LG'//A            ! TOP OF BOX
      WRITE(*,*)E//'LE'              ! FILL THE BOX

      Y - Y - 75
      X - X + 150
      CALL HIY(X,Y,A)
      WRITE(*,*)E//'LP'//A           ! SET TEXT ORIGIN
      CALL INT_TO_CHAR(LCNT,LASERS,LE)
      WRITE(*,*)E//'LT7'//LASERS//' LASERS'
      WRITE(*,*)E//'SC'              ! CLOSE THE SEGMENT

```

```
C--------------------------------------------
C     THE WRITES THE TEXT HEADER INFORMATION.  THE NUMBER OF THE ITEM
C     FOR THE LASER ORDER AND WAVELENGTH.
C--------------------------------------------

      ISEG - 21
      CALL INTRPT(ISEG,SEG)
      WRITE(*,*)E//'SK'//SEG
      WRITE(*,*)E//'SE'//SEG
      WRITE(*,*)E//'MT7'               ! TEXT COLOR YELLOW
      X - 2350
      Y - 1820
      CALL HIY(X,Y,A)
      WRITE(*,*)E//'LP'//A             ! SET TEXT ORIGIN
      TEXT - 'LASER'
      WRITE(*,*)E//'LT5'//TEXT          ! WRITE THE TXT
      Y - 1750
      CALL HIY(X,Y,A)
      WRITE(*,*)E//'LP'//A             ! SET TEXT ORIGIN
      TEXT - 'ORDER'
      WRITE(*,*)E//'LT5'//TEXT          ! WRITE THE TXT

      X - 3050
      Y - 1750
      CALL HIY(X,Y,A)
      WRITE(*,*)E//'LP'//A             ! SET TEXT ORIGIN
      TEXT - 'WAVELENGTH'
      WRITE(*,*)E//'LT:'//TEXT          ! WRITE THE TXT
      WRITE(*,*)E//'SC'


C--------------------------------------------
C     THIS BEGINS A LOOP THRU THE LASERS FOR EACH SAMPLE. THE LASER
C     NUMBER ( 1 - 4 ) IS TAKEN ALONG WITH THE WAVELENGTH THE LASER WILL
C     BE TUNED TOO.
C--------------------------------------------

      WRITE(*,*)E//'LZ'                ! CLEAR DIALOG AREA
      LAS(ICNT) - LCNT                  ! PLACE NUMBER OF LASERS



      DO 400 I - 1, LCNT               ! LOOP THRU THE LASERS

C--------------------------------------------
C     THIS DRAWS THE PANEL THAT SHOWS WHICH LASER IS TO BE DESCRIBED.
C--------------------------------------------


      Y1  - 1200
      X   - 500
      Y   - Y1

```

```
      ISEG - 20
    CALL INTRPT(ISEG,SEG)
    WRITE(*,*)E//'SK'//SEG            ! DELETE THE SEGMENT
    WRITE(*,*)E//'SE'//SEG            ! BEGIN THE SEGMENT

    CALL HTY(X,Y,A)
    WRITE(*,*)E//'LP'//A//'1'         ! SET PANEL ORIGIN
    X - X + 1000
    CALL HTY(X,Y,A)
    WRITE(*,*)E//'LG'//A              ! DRAW BOTTOM OF BOX

    Y - Y + 100
    CALL HTY(X,Y,A)
    WRITE(*,*)E//'LG'//A              ! RIGHT SIDE OF BOX

    X - X - 1000
    CALL HTY(X,Y,A)
    WRITE(*,*)E//'LG'//A              ! TOP OF BOX
    WRITE(*,*)E//'LE'                 ! FILL THE BOX

    Y - Y - 75
    X - X + 100
    CALL HTY(X,Y,A)
    WRITE(*,*)E//'LP'//A              ! SET TEXT ORIGIN

    WRITE(*,*)E//'MT4'                ! TEXT COLOR BLUE

    IF(LASERS.EQ.'1')THEN
    WRITE(*,*)E//'LT'//'1 OF 1 LASER'    ! WRITE THE TEXT

    ELSE
    CALL INT_TO_CHAR(LCNT,LASERS,LE)
    WRITE(*,*)E//'LT'//COUNT(I)//' OF '//LASERS//' LASERS'

    ENDIF

    WRITE(*,*)E//'SC'                 ! CLOSE THE SEGMENT

C---------------------------------------------

100  FORMAT(5X,'ENTER ',A3,' LASER NUMBER: ',$)

102  FORMAT(5X,'ENTER THE LASER NUMBER: ( 1 - 4 ) ',$)


110  IF(LCNT.EQ.1)THEN                 ! IF ONLY ONE LASER THEN
       WRITE(*,102)                   ! GET THE LASER NUMBER
     ELSE
       WRITE(*,100)COUNT(I)
     ENDIF

    READ (*,6,ERR-110)L_ORDER         ! READ THE LASER ORDER
```

```
          IF(L_ORDER.EQ.' ')GOTO 110        ! LOOK FOR NULL

          IF(L_ORDER.EQ.'E'.OR.L_ORDER.EQ.'e')THEN ! USER TO EXIT
            IEXIT = 1              ! SET EXIT FLAG
            GOTO 1000              ! RETURN TO CALLER
          ENDIF

        READ(L_ORDER,'(BN,I2)',ERR=110)ORDER ! MAKE CHAR AN INTEGER

        IF(ORDER.LT.1.OR.ORDER.GT.4)GOTO 110 ! EVALUATE BOUNDS

        LAS_ORD(I + MCNT) = ORDER          ! PLACE IN ARRAY

C------------------ WRITE(*,*)E//'RP0'          ! FIXUP LEVEL 0
       WRITE(*,*)E//'SK!'              ! DELETE ALL SEGMENTS
       WRITE(*,*)E//'RP6'              ! FIXUP LEVEL NORMAL

       RETURN
       END
```

AIV.13 Analog APSD Software Modules: MENU Source Code.

```
      SUBROUTINE MENU
C-----------------------------------------------
C     THIS IS A MENU FOR THE MULLER MATRIX EXPERIMENT FOR USERS THAT
C     DO NOT HAVE A GRAPHICS TERMINAL
C-----------------------------------------------

      CHARACTER GOOF,DUM2*5
      CHARACTER *40 TITLE,TI1,TI2,TI3,TI4,TI5


      TITLE=' MULLER MATRIX EXPERIMENT'
      TI1='1.  BEGIN A NEW EXPERIMENT'
      TI2='2.  REVIEW COLLECTED DATA'
      TI3='3.  CALIBRATE OPTIC STAGES'
      TI4='4.  CALIBRATE A/D CONVERTER'
      TI5='5.  EXIT'

10    PRINT 1,TITLE               ! THIS PRINTS THE ABOVE MENU
      PRINT 2,TI1
      PRINT 2,TI2
      PRINT 2,TI3
      PRINT 2,TI4
      PRINT 3,TI5

1     FORMAT('1',10(/),T11,59('*')/2(T11,'*',T69,'*'/)
     $ T11,'*',T21,A,T69,'*'/T11,'*',T69,'*'/T11,'*',T69,'*')
2     FORMAT(T11,'*',T21,A,T69,'*'/T11,'*',T69,'*'/T11,'*',T69,'*')
3     FORMAT(T11,'*',T21,A,T69,'*'/T11,'*',T69,'*'/T11,'*',T69,'*'/
     $ T11,59('*')/)
4     FORMAT (A)


C-----------------------------------------------
C     THIS ROUTES THE USER TO THE PROPER SUBROUTINE SELECTION
C-----------------------------------------------

      READ (*,*,ERR=600) I

      IF (I.LT.1.OR.I.GT.5) GOTO 600

      GOTO (100,200,300,400,500) I

100   CALL NEW_TEK(0)             ! BEGIN A NEW EXP
      GOTO 10

200   CALL LOOKUP                 ! LOOK AT OLD DATA
      GOTO 10

300   CALL STAGE_POSITION         ! CALIBRATE OPTIC STAGES
      GOTO 10

```

```
 400   CALL A2D_CAL              ! CALIBRATE DATA COLLECTION
       GOTO 10

 600   WRITE (*,*)'PLEASE KEEP YOUR NUMBERS WITHIN THE LIST'
       WRITE (*,*)'       RETURN TO CONTINUE'
       READ (*,4) GOOF
       GOTO 10



 500   RETURN            ! USER WANTS TO EXIT
       END
```

AIV.14 Analog APSD Software Modules: MIX Source Code.

```
      SUBROUTINE MIX(A,NUMBER)
C-----------------------------------------
C     THIS ROUTINE CHANGES INTEGERS OF 9999 OR LESS INTO CHARACTERS
C     SO THAT THEY CAN BE USED AS NAMES OF FILES FOR THE SAVED DATA
C     STRINGS.
C
C

      CHARACTER NUMBER*5
      INTEGER   A,HTHOU,THOU,HUND,TEN,ONES
C     READ(*,*)A
C
C     THIS WILL SUPPRESS LEADING ZEROS TO THE CHARACTER STRING
C
      IBUFF-A
      IF (A.GE.10000)THEN
        NUMLEN-1
        GOTO 2
      ELSEIF (A.GE.1000)THEN
        NUMLEN-2
        GOTO 5
      ELSE IF (A.GE.100) THEN
        NUMLEN-3
        GOTO 17
      ELSE IF (A.GE.10) THEN
        NUMLEN-4
        GOTO 35
      ELSE
        NUMLEN-5
        GOTO 55
      END IF
C
C     THIS PART SEPARATES THE INTEGER INTO SINGLE PLACE VALUES
C
    2 DO 3 I-90000,0,-10000
      J-I-A
      K-K+1
      IF (J.LE.0) GOTO 4
    3 CONTINUE


    4 HTHOU-(10-K)
C     WRITE(*,*)'HTHOU- ',HTHOU
      A-A-(HTHOU*10000)
      K-0

    5 DO 10 I-9000,0000,-1000
      J-I-A
      K-K+1
      IF (J.LE.0) GOTO 15
   10 CONTINUE
```

```
15    THOU-(10-K)
C     WRITE(*,*)'THOU- ',THOU
      A-A-(THOU*1000)
17    K-0
      DO 20 I-900,0,-100
      J-I-A
      K-K+1
      IF(J.LE.0) GOTO 30
20    CONTINUE
30    HUND-(10-K)
C     WRITE (*,*)'HUND- ',HUND
      A-A-(HUND*100)
35    K-0
      DO 40 I-90,0,-10
      J-I-A
      K-K+1
      IF(J.LE.0) GOTO 50
40    CONTINUE
50    TEN-(10-K)
C     WRITE(*,*)'TEN- ',TEN
      A-A-(TEN*10)
55    K-0
      DO 60 I-9,0,-1
      J-I-A
      K-K+1
      IF (J.LE.0) GOTO 70
60    CONTINUE
70    ONES-(10-K)
C     WRITE(,,*)'ONES- ',ONES
C
C     HERE THE VALUE OF THE INTEGER IS CHANGED TO A CHARACTER
C
      GOTO (90,100,200,300,400) NUMLEN
90    NUMBER-CHAR(HTHOU+48)//CHAR(THOU+48)
     $ //CHAR(HUND+48)//CHAR(TEN+48)//CHAR(ONES+48)
      GOTO 500
100   NUMBER-CHAR(THOU+48)//CHAR(HUND+48)//CHAR(TEN+48)
     $ //CHAR(ONES+48)
      GOTO 500
200   NUMBER-CHAR(HUND+48)//CHAR(TEN+48)//CHAR(ONES+48)
      GOTO 500
300   NUMBER-CHAR(TEN+48)//CHAR(ONES+48)
      GOTO 500
400   NUMBER-CHAR(ONES+48)
```

```
500   A-IBUFF
      K-0
      RETURN
      END
```

AIV.15 Analog APSD Software Modules: MOV_STAGE Source Code.

```
      SUBROUTINE MOV_STAGE(LASER_NUM,TYPE,LAMDA,START_POS,REPORT,
     .                 GRAPHICS)
C
C-----------------------------------------
C  THIS ROUTINE MOVES THE OPTIC STAGES TO THEIR PROPER POSITIONS FOR
C  CHANNELING THE LASER BEAM TO EITHER THE SPECTRUM ANALYSER OR THE
C  SAMPLE.
C
C  LASER_NUM -   THIS IS THE USER DEFINED LASER NUMBER FOR WHICH
C                THE MOVEMENT OF THE OPTIC STAGES IS BEING
C                REQUESTED.
C
C  TYPE      -   1. USER IS REQUESTING THAT THE LASER BEAM BE
C                   DIRECTED TO THE SPECTRUM ANALYZER.
C
C                2. USER IS REQUESTING THAT THE LASER BEAM BE
C                   DIRECTED AT THE SAMPLE.
C
C                3. USER IS REQUESTING THAT THE SAMPLE STAGE BE
C                   SENT TO ITS START POSITION.
C
C  LAMDA     -   WAVELENGTH OF THE LASER BEING TUNED.
C
C  START_POS -   THE DISTANCE IN STEPS FOR AXIS 1 TO MOVE THE
C                THE SAMPLE STAGE TO ITS HOME POSITION.
C
C  RECORD    -   THIS IS AN ERROR FLAG SENT BACK TO THE USER
C                IF ANY ERROR CONDITION OCCURS DURING THIS
C                STAGE MOVEMENT OPERATION.
C
C  GRAPHICS  -   INTEGER FLAG INDICATING THAT A TEKTRONIX TERMINAL
C                IS IN USE AND TO CALL THE GRAPHICS ROUTINE
C                TEK_TEXT.
C
C                     ---- ----
C-----------------------------------------

      CHARACTER*10 VEL,ACC,DIST,DUMMY,START_POS,PORT
      CHARACTER*1 LASER,GS,AXIS,CR,INPUT,ANS
      CHARACTER*20 FILENM
      CHARACTER*80 MSG,MSG1
      CHARACTER*15 LAMDA,ANALYTE,AMOUNT

      INTEGER TYPE,GRAPHICS,TXT_FLG,RESET

      LOGICAL T

      DIMENSION VEL(4),ACC(4),DIST(4),DUMMY(4)

C-----------------------------------------
C  THIS CHECKS TO SEE IT THE TYPE - 3.  THIS IS THE CALLER ASKING
```

```
C     THAT THE SAMPLE STAGE BE MOVED TO ITS USER DEFINED START POSITION.
C-------------------------------------------

      IF(TYPE.EQ.3)THEN
        VEL(1)(1:2) - '10'
        ACC(1)(1:2) - '10'
        DIST(1) - START_POS
        GOTO 50
      ENDIF

C-------------------------------------------

5     FORMAT(12(A10))
      ICNT - 0
      CR - CHAR(13)              ! CARRIAGE RETURN
      GS - CHAR(29)              ! CHARACTER REQUIRED TO ESTABLISH
                              ! A SERIAL THRU LINK TO A DEVICE
      L - LASER_NUM              ! TRANSFER THE LASER NUMBER TO "L"
      REPORT - 0                 ! INITIALIZE ERROR FLAG TO "NONE"

C-------------------------------------------
C     THIS PART ESTABLISHES THE DATA FILE NAME THAT HOLDS THE STAGE
C     MOVEMENT DATA FOR EACH LASER.
C-------------------------------------------

      IF(TYPE.EQ.1)THEN          ! USER WANTS TO CHANNEL BEAM TO
        FILENM - 'SPECTRUM'      ! THE SPECTRUM ANALYZER

      ELSEIF(TYPE.EQ.2)THEN      ! OR ELSE THE USER WANTS TO
        FILENM - 'SAMP'          ! CHANNEL THE BEAM TO THE SAMPLE

      ELSE                       ! OR ELSE INPUT WAS BAD SET
        REPORT - 1               ! THE ERROR FLAG AND RETURN
        GOTO 1000                ! TO THE USER
      ENDIF

C-------------------------------------------
C     THIS INQUIRES IF THE DATA FILE EXISTS PROIR TO OPENING THE FILE
C     IF IT DOES NOT THEN THE REPORT FLAG IS SET THE ROUTINE IS ENDED
C-------------------------------------------

        INQUIRE(FILE-FILENM,EXIST - T) ! ASK IF FILE EXISTS

        IF(.NOT.T)THEN             ! IF THERE IS NO FILE
          REPORT - 2               ! SET THE ERROR FLAG
          GOTO 1000                ! AND RETURN TO THE
        ENDIF                      ! CALLER

C-------------------------------------------
C     HERE THE PROPER FILE IS OPENED.
C-------------------------------------------

      OPEN(2,FILE - FILENM,ACCESS-'DIRECT',FORM-'FORMATTED',
     .RECL-120,STATUS - 'OLD',SHARED,ERR - 20)
```

```
      GOTO 30              ! OPEN WAS SUCCESSFUL MOVE ON

20    REPORT - 3           ! OPEN FAILURE SET THE REPORT FLAG
      GOTO 1000            ! RETURN TO CALLER

C------------------------------------------
C     THIS PART READS THE SPECIFIED RECORD FOR THE LASER PASSED IN
C     AS THE VARIABLE LASER_NUM.  THE NUMBER IS THE FILE RECORD NUMBER.
C------------------------------------------


30    READ(2,REC - L,FMT - 5,ERR - 40)VEL(1),ACC(1),DIST(1),
     .DUMMY(1),VEL(2),ACC(2),DIST(2),DUMMY(2),VEL(3),ACC(3),
     .DIST(3),DUMMY(3)

      GOTO 50              ! THE READ WAS GOOD  MOVE ALONG

40    REPORT - 4           ! THE READ FAILED.  SET THE FLAG AND
      GOTO 1000

C------------------------------------------
C     HERE THE STEPPER MOTORS ARE MOVED FROM THEIR HOME POSITION THE
C     VELOCITY, ACCELERATION AND DISTANCE SPECIFIED BY THE ABOVE INPUT.
C     THIS IS DONE BY USE OF A DO LOOP THAT LOOPS THRU ALL THREE AXIS
C     ON THE CONTROLLER.
C------------------------------------------

50    DO 130 I - 1,3

C------------------------------------------
C     FIRST I MAKE SURE THAT THERE IS DATA PRIOR TO WRITING TO THE PORT
C     NO VELOCITY OR ACCELERATION MEANS NO MOVEMENT.  THE ROUTINE THAT
C     CREATES THESE MOVEMENT FILES UNDERSTANDS THAT NOT ALL AXIS OR
C     STAGES ARE REQUIRED TO MOVE EVERY TIME AND THUS THE RECORDS
C     ARE FILLED WITH ZEROS.
C------------------------------------------

      IF(VEL(I).EQ.'0'.OR.ACC(I).EQ.'0'.OR.DIST(I).EQ.'0')THEN
       ICNT - ICNT + 1
       GOTO 130
      ENDIF

C------------------------------------------
C     THIS TURNS ON THE POWER TO THE STEPPER MOTORS AS THEY ARE NEEDED
C     THE AXIS SPECIFIES THE SPECIFIC MOTOR.
C------------------------------------------

       AXIS - CHAR(48 + I)           ! CHANGE INTEGER TO CHARA

      WRITE(3,'(A5)')AXIS//'ST1 '    ! TURN STEPPER POWER ON

C------------------------------------------
C     THIS CREATES THE COMMAND LINE THAT IS SENT TO THE CONTROLLER
```

```
C    IT IS A CHARACTER STRING THAT MUST BE EXACT IN ITS LENGTH.  THIS
C    IS SO BECAUSE THE CONTROLLER BUFFERS ITS DATA AND DOES NOT LIKE
C    MANY SPACES ON THE END OF THE COMMAND LINE.  AFTER MUCH TRIAL
C    AN ERROR I HAVE FOUND THAT THER IS A POSSIBILITY OF 13 FORMATS
C    OR CHARACTER LENGTHS THAT MAY BE REQUESTED OF THIS ROUTINE. I
C    PROVIDE THEM HERE.
C      THE COMMAND LINE IS ALSO A CHARACTER STRING.
C
C         AXIS  - THE STEPPER MOTOR CONTROLLER AXIS NUMBER
C         MSG1  - THE COMMAND LINE THAT TELLS THE VELOCITY,
C                 ACCELERATION, AND DISTANCE FOR EACH AXIS.
C-------------------------------------------

 450  FORMAT(A11)
 451  FORMAT(A12)
 452  FORMAT(A13)
 453  FORMAT(A14)
 454  FORMAT(A15)
 455  FORMAT(A16)
 456  FORMAT(A17)
 457  FORMAT(A18)
 458  FORMAT(A19)
 459  FORMAT(A20)
 460  FORMAT(A21)
 461  FORMAT(A22)
 462  FORMAT(A23)

C-------------------------------------------


      MSG - AXIS//'V'//VEL(I)          ! BEGIN COMMAND LINE W/VEL
      MSG1 - ''

      M  - 0                           ! M - CHARACTER COUNTER

      DO 70 K - 1,3                    ! LOOP THRU EACH SETTING

       DO 60 J - 1,12                  ! LOOP THRU EACH CHARACTER

        IF(ICHAR(MSG(J:J)).LT.45)GOTO 60
                                       ! DONT COUNT SPACES OR "+"
         M - M + 1                     ! COUNT STRING LENGTH
         MSG1(M:M) - MSG(J:J)          ! BUILD THE STRING
 60    CONTINUE

       M - M + 1                       ! INCREMENT STRING COUNT
       MSG1(M:M) - ''                  ! ADD A SPACE AFTER
                                       ! THE COMMAND.
      IF(K.EQ.1)MSG - AXIS//'A'//ACC(I)   ! ADD ACCELERATION
      IF(K.EQ.2)MSG - AXIS//'D'//DIST(I)  ! ADD TRAVEL DISTANCE

 70   CONTINUE

      M - M + 1
```

```
          MSG1(M:M + 1) = 'G '          ! ADD A GO TO THE END
      M = M + 1
      J = M - 10

      CALL TWAIT(5)

      GOTO(570,580,590,600,610,620,630,640,650,660,670,680,690)J

 570  WRITE(3,450)MSG1(1:M)
      GOTO 100
 580  WRITE(3,451)MSG1(1:M)
      GOTO 100
 590  WRITE(3,452)MSG1(1:M)
      GOTO 100
 600  WRITE(3,453)MSG1(1:M)
      GOTO 100
 610  WRITE(3,454)MSG1(1:M)
      GOTO 100
 620  WRITE(3,455)MSG1(1:M)
      GOTO 100
 630  WRITE(3,456)MSG1(1:M)
      GOTO 100
 640  WRITE(3,457)MSG1(1:M)
      GOTO 100
 650  WRITE(3,458)MSG1(1:M)
      GOTO 100
 660  WRITE(3,459)MSG1(1:M)
      GOTO 100
 670  WRITE(3,460)MSG1(1:M)
      GOTO 100
 680  WRITE(3,461)MSG1(1:M)
      GOTO 100
 690  WRITE(3,462)MSG1(1:M)

 100  IF(TYPE.EQ.3)GOTO 1000       ! EXIT IF MOVING STAGE TO START

 130  CONTINUE

C------------------------------------------
C     AT THIS POINT THE THE ROUTINE MAKES A SWITCH BASED ON THE TYPE OF
C     DATA REQUESTED BY THE CALLER.
C
C         TYPE - 1 THE SPECTRUM ANALYZER IS THE TARGET.  THE STAGES
C                ARE NOW MOVED TO THE CORRECT LOCATON.  THE USER
C                IS REQUESTED TO PRESS RETURN WHEN THE LASER
C                TUNNING IS COMPLETE.  THE STAGES ARE RETURNED TO
C                THEIR HOME POSITIONS AND THE NEXT LASER LOOPS IN
C                TURN.  < IF THERE ARE MORE LASERS >
C
C         TYPE - 2 THE SAMPLE IS THE TARGET.  SINCE THE STAGES ARE
C                NOW IN THE CORRECT POSITION FOR DATA COLLECTION
C                THE ROUTINE IS EXITED.  THE STAGES ARE SENT HOME
C                BY THE CALLER AFTER DATA COLLECTION.
C------------------------------------------
```

```

      IF(TYPE.EQ.1)THEN           ! BEAM IS SPECTRUM ANALYZER
108    FORMAT(24(/))
110    FORMAT(20X,'LASER : ',I4,20X,'WAVELENGTH : ',A15,//)
112    FORMAT(10X,'PRESS RETURN WHEN THE LASER IS CALIBRATED:',//)
114    FORMAT(35X,'WAITING...',//)
116    FORMAT(40X,$)

C------------------------------------------
C     THIS PART CALLS THE GRAPHICS ROUTINE TO DISPLAY THE TEXT ABOVE.
C------------------------------------------

      IF(GRAPHICS.EQ.1)THEN
       RESET = 1                  ! DO NOT REDRAW THE PANEL.
       ANALYTE = LAMDA                 ! PLACE WAVELENGTH AND
       ANS = CHAR(L + 48)            ! LASER No. IN VARIABLES
       TXT_FLG = 4                  ! FLAG FOR ABOVE TEXT
       CALL TEK_TEXT(TXT_FLG,PORT,RESET,ANS,ANALYTE,AMOUNT,IEXIT)

       GOTO 140
      ENDIF
C------------------------------------------

120    WRITE(*,108)
      WRITE(*,110)L,LAMDA
      WRITE(*,112)
      WRITE(*,114)
      WRITE(*,116)
      READ(*,'(A)',ERR=120)INPUT

C------------------------------------------
C     REVERSE DIRECTION AND < G > TELLS IT TO GO THE SAME DISTANCE
C     AND VELOCITY.  ICNT KEPT A COUNT ON THE NUMBER OF STAGES NOT
C     MOVED DURING THIS OPERATION.  IF ITS = 3 THEN THERE IS NO HOME TO
C     GO TOO.
C------------------------------------------

140    IF(ICNT.LT.3)WRITE(3,'(A4)')'H G '     ! SEND ALL THE STAGES HOME


      ENDIF

C------------------------------------------
C     THIS PROVIDES THE USER WITH ERROR INFORMATION IN THE UNLIKELY
C     EVENT THAT A PROBLEM OCCURS WITH THIS ROUTINE.
C     THIS IS BASED ON THE NUMBER OF THE " REPORT "  VARIABLE.
C------------------------------------------

1000   IF(REPORT.GT.0)THEN          ! THERE IS A PROBLEM

1010   FORMAT(10X,'NO FILE TYPE WAS PROVIDED BY THE CALLER',//)
1020   FORMAT(10X,'THE FILE: ',A20,' CANNOT BE FOUND',//)
1030   FORMAT(10X,'THERE WAS AN OPEN ERROR ON THE FILE: ',A20,//)
1040   FORMAT(10X,'THE FILE: ',A20,' FAILED ON THE READ',//)
```

```
      IF(REPORT.EQ.1)WRITE(*,1010)
      IF(REPORT.EQ.2)WRITE(*,1020)FILENM
      IF(REPORT.EQ.3)WRITE(*,1030)FILENM
      IF(REPORT.EQ.4)WRITE(*,1040)FILENM

      ENDIF

      WRITE(3,'(A4)')'ST0 '        ! TURN STEPPER POWER OFF

      RETURN                       ! RETURN TO CALLER
      END
```

AIV.16 Analog APSD Software Modules: NEWTEK Source Code.

```
      SUBROUTINE NEW_TEK(GRAPHICS)
C--------------------------------------------------
C     THIS MOD IS CALLED "NEWEXP" AND IS ACCESSED BY THE MAIN MENU
C     IT WILL SET ALL THE INSTRUMENTATION TO SPECIFIED POSITIONS
C     COLLECT AND STORE ALL DATA. FOR EACH SAMPLE THERE WILL BE
C     SEVERAL DISCRETE FILES EACH TIED TO ONE ANOTHER.
C     1. A HEADER FILE DESCRIBING FACTS ABOUT THE SAMPLE
C     2. ALL SAMPLE DATA FOR THE DIFFERENT WAVELENGTHS
C     3. AN INDEX FILE SO THAT THE DATA CAN BE REFERENCED.
C--------------------------------------------------

      CHARACTER TITLE*48,NAME*20,SAMP*20,AGENT*20,CONC*15
      CHARACTER DATE*9,P1POS*4,P2POS*4,MOD1*2,MOD2*2,ST*3,ERR*4
      CHARACTER OPT_NUM*2,OPT_INFO*3,OPTIC*1,START_POS*10
      CHARACTER ARM*8,INCARM*8,ENDARM*8,RS,ACK,ANS,LAMDA*15
      CHARACTER FNAME*5,FLNM*7,SAMPLE,FILENM*20,PORT*10
      CHARACTER GS,CR,MSG*255,START*6,STOP*6,INC*6,TIME*8
      CHARACTER AMOUNT*15,ANALYTE*20,SEG*3,E*1

      INTEGER CHANGE,ERRMSG,REPORT,TOT_MEM,FREE_MEM
      INTEGER RECNUM,IEXIST,ISHOTS,IN,KK,DUMMY,TP1,STP,TYPE
      INTEGER LASER,T1,LASERS,GRAPHICS
      INTEGER TXT_FLG,RESET
      REAL ARAY,BRAY,CRAY,DRAY,ERAY,REND,RINCR,RARM
      REAL START1,DIST

      LOGICAL T

C--------------------------------------------------
C     ARRAYS ARAY,BRAY,CRAY,DRAY ARE REDUNDANT DATA ARRAYS THAT ARE
C     FOR THE PRESENT USED TO CHECK FOR DISCREPANCIES IS THE DATA.
C     THESE ARE THE TAGGED TO THE POLARIZER POSITIONS.
C
C     VERT/VERT    - ARAY   9 ELEMENTS X 180 degs  - 1620 RECORDS
C     VERT/45 deg  - BRAY
C     45 deg/VERT  - CRAY
C     45 deg/45 deg - DRAY
C     ALL DATA     - ERAY   16 ELEMENTS X 180 degs - 2880 RECORDS
C--------------------------------------------------

      COMMON /MATRIX/ARAY(3600),BRAY(3600),CRAY(3600),DRAY(3600),
     .ERAY(5800)

      DIMENSION SAMP(10),AGENT(10),CONC(10),LAMDA(40),LAS(40)
      DIMENSION RDAT(16),HEX(16),START(10),STOP(10),INC(10)
      DIMENSION LASERS(10)

    1 FORMAT('1',//////T16,A80,//)
    2 FORMAT(A20)
    3 FORMAT(A4)
    4 FORMAT('O')
```

```
5     FORMAT(A2)
6     FORMAT (A)
7     FORMAT(I5)
8     FORMAT (A10)

      RS  - CHAR(30)
      ACK - CHAR(6)
      GS  - CHAR(29)
      CR  - CHAR(13)
      E   - CHAR(27)



      TITLE-'Welcome to the Muller Matrix Ellipsometry Experiment'

      CHANGE - 0

      IF(GRAPHICS.EQ.0)GOTO 20


C-----------------------------------------------
C                ***** TEST DATA *****
      NAME  - 'CHAS'
      DATE  - ' 7-JAN-90'
      TIME  - '17:30:00'
      NOSAMP - 1
      SAMP(1) - 'GOLD'
      AGENT(1)- 'GB'
      CONC(1) - '.123 mu'
      START(1) - '80.'
      STOP(1)  - '90.'
      INC(1)  -  '1.0'

      LAS(1) - 1
      LAS(2) - 2
      LAS(3) - 3
      LAS(4) - 4

      LAMDA(1) - '111.1'
      LAMDA(2) - '222.2'
      LAMDA(3) - '333.3'
      LAMDA(4) - '444.4'

      LASERS(1) - 4

C     IF(I123.EQ.0)GOTO 580

C-----------------------------------------------
C     THIS CHECKS THE TERMINAL TYPE.  IF THE USER IS USING A TEKTRONIX
C     TERMINAL THE ROUTINE WILL USE A COLOR GRAPHICS ROUTINE DESIGNED TO
C     GET ALL THE SAMPLE INFORMATION FOR UP TO 8 SAMPLES.
C-----------------------------------------------
C
C      CALL TERM_INFO(ITERM,NUM_PLANES,TOT_MEM,FREE_MEM,
```

```
C     ,IVERSION,OPT_NUM,OPT_INFO)
C
C     IF(NUM_PLANES.EQ.-1)THEN
C       GOTO 20              ! NOT A TEKTRONIX TERMINAL
C
C------------------------------------------
C    THIS PART CALLS THE GRAPHICS ROUTINE THAT TAKES THE SAMPLE DATA.
C------------------------------------------

C     ELSE

      READ(*,'(I2)')DUMMY        ! THIS IS A DUMMY READ

        CALL TEK_INPUTS(NOSAMP,SAMP,AGENT,CONC,START,STOP,INC,
     .                NAME,DATE,TIME,IEXIT)

C       GRAPHICS = 1           ! FLAG THAT GRAPHICS ARE
C                              ! IN USE.
C------------------------------------------

        IF(IEXIT.EQ.1)GOTO 10000

C------------------------------------------
C    THIS IS THE GRAPHICS ROUTINE THAT TAKES SPECIFIC LASER DATA FOR
C    EACH SAMPLE.  HOW MANY LASERS,  WHAT ORDER, WHAT WAVELENGTHS
C------------------------------------------

        CALL LASER_IN(SAMP,AGENT,CONC,NOSAMP,NAME,LASERS,LAMDA,LAS,
     .            IEXIT)

        IF(IEXIT.EQ.1)GOTO 10000

        GOTO 580                ! GO BEGIN EXPERIMENT


C------------------------------------------
C    THIS GETS THE NAME OF THE PERSON RUNNING THE EXPERIMENT
C------------------------------------------

20    PRINT 1,TITLE
30    FORMAT(T5,' ENTER NAME            : ',$)
40    WRITE(*,30)
      READ(*,2) NAME          ! INPUT NAME
      WRITE(*,4)
      IF (CHANGE.EQ.1) GOTO 500

C------------------------------------------
C    THIS GETS THE DATE THE DATA WAS TAKEN
C------------------------------------------

42    FORMAT(T5,' ENTER DATE            : ',$)
45    WRITE(*,42)
      READ(*,2) DATE          ! INPUT DATE
      WRITE(*,4)
```

Appendix IV

```
         IF (CHANGE.EQ.1) GOTO 500

C--------------------------------------------
C     THIS GETS THE NUMBER OF SAMPLES AND THE NAME OF EACH SAMPLE
C--------------------------------------------

50    FORMAT(T5,' ENTER NUMBER OF SAMPLES     : ',$)
60    WRITE(*,50)
      READ (*,*,ERR-60)NOSAMP            ! INPUT NUMBER OF SAMPLES
      WRITE(*,4)


      WRITE(*,*)'IF ALL SAMPLES ARE THE SAME TYPE < Y >'
      READ(*,6)ANS
      IF (ANS.EQ.'Y') GOTO 110


      DO 90 I-1,NOSAMP

70    FORMAT(T5,' ENTER SAMPLE NAME',I,'     : ',$)
80    WRITE(*,70)
      READ(*,2)SAMP(I)            ! INPUT SAMPLE NAME
      WRITE(*,4)
90    CONTINUE

      IF (CHANGE.EQ.1) GOTO 500
      GOTO 130

100   FORMAT(T5,' ENTER SAMPLE NAME          : ',$)
110   WRITE(*,100)
      READ (*,2)SAMP(1)            ! INPUT SAMPLE NAME
      DO 120 I-2,NOSAMP
      SAMP(I)-SAMP(1)
120   CONTINUE
      WRITE(*,4)
      IF (CHANGE.EQ.1) GOTO 500

C--------------------------------------------
C     THIS GETS THE TYPE OF AGENT THAT WILL BE USED ON THE SAMPLES
C--------------------------------------------

      DO 125 I - 1,NOSAMP
       AGENT(I) - ' '
125   CONTINUE

      WRITE(*,*)

130   WRITE(*,*)'WILL AGENT TYPE BE THE SAME FOR ALL SAMPLES
     1 < Y >'
      READ (*,6,ERR-130)ANS
      IF (ANS.EQ.'Y') GOTO 180

      DO 160 I-1,NOSAMP
140   FORMAT(T5,' ENTER AGENT',I,'      : ',$)
```

```
 150   WRITE(*,140)I
       READ(*,2)AGENT(I)          ! INPUT AGENT NAME
       WRITE(*,4)
 160   CONTINUE

       IF (CHANGE.EQ.1) GOTO 500
       GOTO 200


 170   FORMAT(T5,' ENTER AGENT NAME          : ',$)
 180   WRITE(*,170)

       READ(*,2) AGENT(1)          ! INPUT AGENT NAME

       DO 190,I-2,NOSAMP
       AGENT(I)-AGENT(1)
 190   CONTINUE

       WRITE(*,4)

       IF (CHANGE.EQ.1) GOTO 500

C------------------------------------------------
C      THIS GETS THE AGENT CONCENTRATION OF THE AGENT FOR ALL THHE SAMPLES
C------------------------------------------------

 200   WRITE(*,*)'WILL AGENT CONCENTRATION BE THE SAME FOR ALL SAMPLES'
       WRITE(*,*)'               < Y >'
       READ(*,6,ERR-200)ANS

       IF(ANS.EQ.'Y') GOTO 250


 210   FORMAT(T5,I,' ENTER CONC. OF AGENT(Mg/m3)  : ',$)
 220   DO 230 I-1,NOSAMP
       WRITE(*,210)I
       READ(*,8)CONC(I)
       WRITE(*,4)
 230   CONTINUE
       WRITE(*,4)
       IF (CHANGE.EQ.1) GOTO 500
       GOTO 280

 240   FORMAT(T5,' ENTER CONC. OF AGENT(Mg/m3)    : ',$)
 250   WRITE(*,240)
       READ(*,8)CONC(1)

       DO 260 I-2,NOSAMP
       CONC(I)-CONC(1)
 260   CONTINUE

       WRITE(*,4)

       IF (CHANGE.EQ.1) GOTO 500
```

```
C------------------------------------------
C     THIS PART GETS THE START,STOP AND INCREMENT POSITION OF THE SAMPLE
C     FOR THE ROTATION IN FRONT OF THE LASER.
C------------------------------------------

270   FORMAT(10X,'START POSITION OF ARM <DEGREES>: ',$)

      DO 325 I - NOSAMP           ! LOOP THRU SAMPLES

       WRITE(*,332)
280     WRITE(*,270)

       READ(*,'(A6)',ERR-280)START(I)  ! START POSITION OF GONIOMETER
       WRITE(*,4)

290   FORMAT(10X,'INCREMENT OF ARM <DEGREES>    : ',$)

300     WRITE(*,290)
       READ(*,'(A6)',ERR-300)INC(I)     ! INCREMENT OF GONIOMETER
       WRITE(*,4)

310   FORMAT(10X,'END POSITION OF ARM(DEGREES)  : ',$)

320   WRITE(*,310)
      READ(*,'(A6)',ERR-320)STOP(I)       ! END POSITION OF GONIOMETER
      WRITE(*,4)

      IF (CHANGE.EQ.1) GOTO 500

325   CONTINUE

C------------------------------------------
C     THIS PART GETS THE LASER WAVELENGTH AND NUMBER.  THE NUMBER
C     IS ONE OF THE FOUR LASERS BEING USED FOR THIS PARTICULAR EXP.
C------------------------------------------

330   FORMAT(10X,'ENTER NUMBER OF LASER TO BE USED:',$)
332   FORMAT(10X,'SAMPLE No. ',I2,'  SAMPLE: ',A20)

      LCNT - 0                    ! THIS IS AN ARRAY INCREMENTER

      DO 380 I - 1,NOSAMP         ! LOOP THRU THE SAMPLES

340   WRITE (*,332)I,SAMP(I)
      WRITE (*,330)

      READ(*,*,ERR-340)LASERS(I)      ! HOW MANY LASERS
      WRITE(*,4)

C------------------------------------------
C     THIS PART LOOPS THRU THE LASERS JUST DEFINED AND GETS THE
C     ACTUAL LASER NUMBER 1-4 AND ITS WAVELENGTH.
C------------------------------------------
```

```
 345  FORMAT(5X,'ENTER THE WAVELENGTH FOR THIS LASER: ',$)
 346  FORMAT(5X,I2,' ENTER THE LASER NUMBER (1 - 4) : ',$)

      DO 370 J - 1,LASERS(I)        ! LOOP THRU THE LASERS

 343  WRITE (*,346)J
      READ(*,*,ERR-343)LAS(I + LCNT)        ! GET LASER NUMBER
      WRITE(*,4)

 344  WRITE (*,345)
      READ(*,*,ERR-343)LAMDA(I + LCNT)        ! GET LASER WAVELENGTH
      WRITE(*,4)

 370  CONTINUE                    ! END LASER LOOP

      LCNT - LCNT + 4             ! INCREMENT ARRAY COUNTER

 380  CONTINUE

C-----------------------------------

      IF (CHANGE.EQ.1) GOTO 500

C-----------------------------------
C     THIS PART LETS TH[illegible]EE IF HE WANTS TO MAKE CORRECTIONS
C     AND PROVIDES A MEANS OF DOING SO.
C-----------------------------------

 400  FORMAT (24(/))
 410  FORMAT (T34,'HEADER FILE',/)
 420  FORMAT (10X,'1. OPERATOR: ',A20,10X,'2. DATE: ',A9,' ',A8,/)
 430  FORMAT (T9,'3. SAMPLE',T36,'4. AGENT',T60,'5. CONC',/)
 440  FORMAT (3X,I2,3X,A20,8X,A20,10X,A15)
 450  FORMAT (30X,I2,' LASERS')
 460  FORMAT (10X,'LASER # ',I2,10X,A15)
 470  FORMAT (T30,'- GONIOMETER - < DEGs >')
 480  FORMAT (T11,'7. START: ',T22,A6,T31,'8. STOP: ',T41,A6,
     1 T49,'9. INCREMENT: ',T64,A6,/)


C-----------------------------------
C     THIS SHOWS THE USER THE INPUTS THAT WERE JUST MADE ON THE SAMPLES
C     HERE EACH SAMPLE IS PICTURED INDIVIDUALLY WITH ITS DATA.
C-----------------------------------

      LCNT - 0
      STRT - 1

 500  DO 550 I-STRT,NOSAMP        ! LOOP THRU EACH SAMPLE

      WRITE(*,400)
      WRITE(*,410)
      WRITE(*,4)
```

```
          WRITE(*,420)NAME,DATE,TIME
        WRITE(*,430)
        WRITE(*,440)I,SAMP(I),AGENT(I),CONC(I)
        WRITE(*,4)
        WRITE(*,450)LASERS(I)

         DO 510 J - 1,LASERS(I)
           WRITE(*,460)LAS(I + LCNT),LAMDA(I + LCNT)
  510      CONTINUE

        WRITE(*,4)
        WRITE(*,470)
        WRITE(*,480)START(I),STOP(I),INC(I)
        WRITE(*,4)

C---------------------------------------------


  560   FORMAT(10X,'ANY CHANGES ? ( RETURN FOR NONE
      . OR SELECT NUMBER ) ',$)

  570   WRITE(*,560)

        READ(*,'(A)',ERR-570)ERR

        IF(ERR.EQ.' ')GOTO 550

        READ(ERR,'(BN,I2)')ERRMSG

        IF (ERRMSG.GT.9)GOTO 570
        IF (ERRMSG.GT.0) THEN
          CHANGE - 1
          ISTRT - I
        GOTO (40,45,60,130,200,340,280,320,300) ERRMSG
        ENDIF
        CHANGE - 0
  540   LCNT - LCNT + 1
  550   CONTINUE

C---------------------------------------------
C       AT THIS POINT CONTROL DATA IS SENT TO ADJUST THE INSTRUMENTS
C
C---------------------------------------------
C       1. OPEN THE SERIAL PORT THRU WHICH ALL THE INSTRUMENTATION I/O
C       AND DATA WILL BE CONTROLLED AND COLLECTED.
C---------------------------------------------

  572   FORMAT(////,10X,'THE COMMUNICATIONS PORT MUST BE DEFINED WITH')
  573   FORMAT(10X,'THE FOLLOWING PARAMETERS:',//)
  574   FORMAT(30X,'9600  BAUD        NO  PARITY')
  575   FORMAT(30X,'8    BITS        1 STOP BIT',//)
  576   FORMAT(10X,'THE PORT MUST BE IN A " PASSALL " MODE ',//)
  578   FORMAT(10X,'PRESS RETURN FOR DEFAULT PORT <  TXA2: >,'//)
  579   FORMAT(10X,'ENTER THE SERIAL PORT NAME:  ',$)
```

```
C-----------------------------------------
C     IF THE USER HAS A TEK TERMINAL THEN THE TEK_TEXT ROUTINE IS
C     CALLED TO DISPLAY THE ABOVE TEXT.
C-----------------------------------------

580   IF(GRAPHICS.EQ.1)THEN
      TXT_FLG = 1              ! FLAG FOR PORT SET PARAMETERS
      PORT   = ''              ! CLEAR OUT ANY PORT DATA
      RESET  = 0               ! FLAG TO DRAW A RED PANEL
      IEXIT  = 0               ! INITIALIZE EXIT FLAG

      CALL TEK_TEXT(TXT_FLG,PORT,RESET,ANS,ANALYTE,AMOUNT,IEXIT)

        IF(IEXIT.EQ.1)GOTO 10000   ! USER WANTS TO EXIT

      GOTO 588

      ENDIF
C-----------------------------------------
C     THIS TEXT IS WRITTEN TO A NON TEKTRONIX TERMINAL.
C-----------------------------------------

583   WRITE(*,572)
      WRITE(*,573)
      WRITE(*,574)
      WRITE(*,575)
      WRITE(*,576)
      WRITE(*,578)
      WRITE(*,579)
      READ(*,'(A10)',ERR=583)PORT

      IF(PORT.EQ.' ')PORT = 'TXA2:'

588   OPEN(3,FILE = PORT,STATUS='NEW',CARRIAGECONTROL = 'NONE',
     .ERR = 10000)

C-----------------------------------------
C     THIS INITIALIZES THE RELAY BANKS.
C-----------------------------------------

      WRITE(3,'(A9)')GS//'%WC16,0'//CR
      WRITE(3,'(A9)')GS//'%WC70,0'//CR

C-----------------------------------------
C     INITIALIZE MODULATOR #1.  TO KEEP THE MODULATORS CALM WE
C     DISCOVERED THAT PRIOR TO TURNING ONE OFF ITS GOOD PRACTICE TO
C     TURN ANOTHER ON FIRST.  HERE I GET THE FIRST ONE WARMED UP.
C-----------------------------------------

      WRITE(3,'(A9)')GS//'%WC20,1'//CR      ! MODULATOR #1 ON

C-----------------------------------------
C     2. THIS PART REQUESTS THAT THE USER DECIDE THE METHOD BY WHICH
```

```
C          THE EXPERIMENT WILL BE SEEN.  I HAVE PROVIDED A METHOD
C          BY WHICH REAL TIME GRAPHICS MAY BE GENERATED IF THE
C          EXPERIMENT IS CONDUCTED ON A TEKTRONIX TERMINAL
C          MODELS 4111 OR ABOVE.  ADDITIONALLY, THERE IS A METHOD
C          FOR VIEWING THE A/D VOLTAGES FOR EACH DATA CHANNEL PER
C          DATA COLLECTION CYCLE.  THE USER MAY OPT FOR A "QUIET"
C          CYCLE WHERE NO DATA IS PRESENTED TO SPEED UP THE DATA
C          COLLECTION IF THE RUN TIME CONFIDENCE IS HIGH.
C------------------------------------------------------------

581   FORMAT(/////,5X,'ENTER THE TYPE OF OUTPUT DESIRED:',///)

582   FORMAT(10X,'1.  REAL TIME A/D CHANNEL VOLTAGE OUTPUTS',//)
584   FORMAT(10X,'2.  NO DISPLAY OF DATA',//)
586   FORMAT(10X,'3.  EXIT ROUTINE',//)

C------------------------------------------------------------
C     THIS PART CALLS THE GRAPHICS ROUTINE TO DISPLAY THE TEXT ABOVE
C     AND RETURNS ITS ANSWER IN " ANS ".
C------------------------------------------------------------

      IF(GRAPHICS.EQ.1)THEN
        TXT_FLG - 2              ! FLAG FOR DISPLAY PARAMETERS
        RESET  - 1              ! FLAG NOT TO DRAW THE PANEL
        IEXIT  - 0              ! INITIALIZE EXIT FLAG

      CALL TEK_TEXT(TXT_FLG,PORT,RESET,ANS,ANALYTE,AMOUNT,IEXIT)

        IF(IEXIT.EQ.1)GOTO 10000     ! USER WANTS TO EXIT
        GOTO 589

      ENDIF

C------------------------------------------------------------
C     THIS IS THE NON TEKTRONIX TEXT TO GET THE TYPE OF OUTPUT THE USER
C     IS REQUESTING
C------------------------------------------------------------

593   WRITE(*,581)
      WRITE(*,582)
      WRITE(*,584)
      WRITE(*,586)

      READ(*,'(A)',ERR - 593)ANS

      IF(ANS.EQ.'1'.OR.ANS.EQ.'2'.OR.ANS.EQ.'3')GOTO 589

      GOTO 593

C------------------------------------------------------------


589   IF(GRAPHICS.EQ.1)THEN        ! USER HAS A TEK TERMINAL

```

```
          IF(ANS.EQ.'1')THEN        ! USER WANTS REAL TIME GRAPHICS
           STP = 1              ! SET THE DISPLAY FLAG FOR TEK

          ELSEIF(ANS.EQ.'2')THEN     ' USER WANTS A/D VOLTAGES
           GRAPHICS = 0          ! FLAG THAT NO GRAPHICS REQUESTED
           STP = 2              ! SET DISPLAY FLAG FOR CHART MODE
           RESET  = 2              ! DELETE THE RED PANEL
           CALL TEK_TEXT(TXT_FLG,PORT,RESET,ANS,
     .                ANALYTE,AMOUNT,IEXIT)

          ELSEIF(ANS.EQ.'3')THEN     ! USER DOSENT WANT ANY OUTPUT
           STP = 3              ! SET THE FLAG FOR QUIET MODE
           RESET  = 2              ! DELETE THE RED PANEL
           GRAPHICS = 0          ! FLAG THAT NO GRAPHICS REQUESTED

           CALL TEK_TEXT(TXT_FLG,PORT,RESET,ANS,
     .                ANALYTE,AMOUNT,IEXIT)

          ENDIF

C---------------------------------------------------
C    THE USER DOES NOT HAVE A GRAPHICS TERMINAL
C---------------------------------------------------

         ELSE

          IF(ANS.EQ.'1')THEN  ! USER WANTS A/D VOLTAGES
           STP = 2              ! SET DISPLAY FLAG FOR CHART MODE

          ELSEIF(ANS.EQ.'2')THEN     ! USER DOSENT WANT ANY OUTPUT
           STP = 3              ! SET THE FLAG FOR QUIET MODE

          ELSEIF(ANS.EQ.'3')THEN          ! USER WANTS TO EXIT
           GOTO 10000            ! RETURN TO CALLER

          ENDIF

        ENDIF

C---------------------------------------------------
C    3. THIS PART ESTABLISHES CONTACT WITH THE A/D CONVERTER VIA
C       THE UPLINK CONTROLLER CARD #4.  THIS IS DONE BY SENDING A
C       COMMAND <  CHAR(29) // D4 >.  THIS CHANNELS ALL SERIAL DATA
C       TO THE A/D BOARD.
C         THE A/D IS AWAKENED BY 2 CARRIAGE RETURNS UPON WHICH IT
C         RETURNS  A HYPHEN.  THE COMMAND < ST-701 EXECUTIVE ON >
C         IS ISSUED BY THE HOST COMPUTER TO ENTER  EXECUTIVE MODE
C         AND A STAR " * " PROMT IS ISSUED BY THE A/D, IF ALL IS
C         WELL.  AT THIS POINT THE A/D IS READY TO COLLECT DATA.
C         IF THERE IS A PROBLEM THE REPORT FLAG WILL  =  1.
C---------------------------------------------------

 590    ICNT  = 0              ! INITIALIZE THE REPEAT COUNTER
        REPORT = 0             ! INITIALIZE THE REPORT FLAG
```

```
      T1   - 1

C     WRITE(*,*)T1,REPORT,PORT,IC1,CHAN,RDAT,HEX
      PORT - 'TXA2:'

C     PAUSE' CALLING DATEL TO INITIALIZE IT'


      CALL DATEL(T1,REPORT,PORT,IC1,CHAN,RDAT,HEX)
          ! GO AWAKE THE A/D CONVERTER

      IF(REPORT.EQ.1)THEN          ! THERE IS TROUBLE
       IF(ICNT.EQ.3)GOTO 10000     ! IF OVER 3 TRIES ..QUIT
       WRITE(*,*)
       WRITE(*,*)' THERE IS A PROBLEM WAKING THE A/D CONVERTER'
       WRITE(*,*)
       WRITE(*,*)' CHECK THE CONNECTION AND PRESS RETURN  '
       WRITE(*,*)
       WRITE(*,*)

       READ(*,'(A)',ERR-585)ANS    ! READ THE INPUT

585    ICNT - ICNT + 1             ! COUNT THE TRIES
       GOTO 590                    ! TRY THE A/D AGAIN
      ENDIF

C-------------------------------------------
C     4. THIS BEGINS THE LASER CALIBRATION SEQUENCE.  HERE A PREDEFINED
C          FILE " SPECTRUM.DAT " CONTAINS THE STAGE POSITIONS
C          NECESSARY TO CHANNEL THE LASER BEAM FROM EACH LASER INTO
C          A SPECTRUM ANALYZER.  IF THIS FILE DOES NOT EXIST THEN
C          THE USER IS INSTRUCTED TO GO TO THE CALIBRATION ROUTINE
C          DEFINED IN THE MAIN MENU.
C-------------------------------------------

      REPORT - 0                   ! INITIALIZE ERROR FLAG

      WRITE(3,'(A4)')GS//'D2'//CR    ! INITIALIZE THE OPTIC
C                                  ! STAGE CONTROLLER CARD
      WRITE(3,'(A2)')'E '          ! SEND STARTUP COMMAND
      WRITE(3,'(A2)')'E '          ! SEND STARTUP COMMAND

      WRITE(3,'(A9)')'E MN ST0 '    ! TURN ON CONTROLLER
                                   ! PLACE IN NORMAL MODE
C-------------------------------------------
C     THIS PART MAKES THE USER CALIBRATE THE LASERS THAT WILL BE USED
C     ON THIS SAMPLE.  THIS IS DONE ONCE FOR EACH SAMPLE UNLESS A
C     CALIBRATION IS NOT NEEDED.  THAT IS IF SAMPLE 2+ ON LASER #1 IS
C     THE SAME AS LASER #1 ON SAMPLE 1'
C-------------------------------------------

      DO 595 I - 1,LASERS(1)       ! LOOP THRU LASERS

       OPTIC - CHAR(48 + LAS(I + LCNT))    ! ACTUAL LASER NUMBER
```

```
          LASER - LAS(I + LCNT)          ! HOLD IN BUFFER
        TYPE - 1                    ! FLAG FOR SPECTRUM SETUP


        CALL MOV_STAGE(LASER,TYPE,LAMDA(I),START_POS,REPORT,
     .                GRAPHICS)

595     CONTINUE


C-----------------------------------------------------
C       THIS ERASES THE RED TEXT SCREEN
C-----------------------------------------------------

          RESET   - 2                ! DELETE THE RED PANEL
          CALL TEK_TEXT(TXT_FLG,PORT,RESET,ANS,
     .                ANALYTE,AMOUNT,IEXIT)

C-----------------------------------------------------
C
C       INITIALIZATION IS COMPLETE AND THE INSTRUMENTS ARE READY TO
C       COLLECT DATA.
C       THIS IS THE BEGINNING OF THE MAIN DATA COLLECTING LOOP
C-----------------------------------------------------

      LCNT - 0                  ! LASER ARRAY INCREMENTER

      DO 1070 LOOP - 1,NOSAMP       ! LOOP THRU ALL SAMPLES

C-----------------------------------------------------
C       THIS CONVERTS THE CHARACTER INPUTS FOR START, END AND INCREMENT
C       ANGLES TO REAL NUMBERS.
C-----------------------------------------------------

600     READ(STOP(LOOP),'(BN,F6.2)')REND
      READ(START(LOOP),'(BN,F6.2)')RARM
      READ(INC(LOOP),'(BN,F6.2)')RINCR

      ISHOTS -IFIX((REND - RARM)/RINCR) + 1  ! THIS IS THE TOTAL No

C-----------------------------------------------------
C       THIS CALLS THE GRAPHICS ROUTINE TO DRAW A BAR GRAPH.  THE USER
C       HAS THE OPTION SELECT ALL OR JUST SPECIFIC MATRIX ELEMENTS
C       TO BE GRAPHED FOR THE EXPERIMENT.
C-----------------------------------------------------

      IF(LOOP.EQ.1.AND.K.GT.1)TP1 - 1

      IF(STP.EQ.1)THEN

       CALL TEK3(TP1,SAMP(LOOP),AGENT(LOOP),CONC(LOOP),LAMDA(K))

      ENDIF

```

```
      C-----------------------------------------
      C    THIS PART ESTABLISHES IF THERE IS GOING TO BE ANY CHEMISTRY
      C    APPLIED TO THE SAMPLE.  IF SO THE SAMPLE STAGE MUST BE MOVED TO
      C    THE APPARATUS THAT SUPPLIES THE CHEMICALS.
      C-----------------------------------------

           IF(AGENT(LOOP).EQ.'NONE')THEN
            RESET - 2                    ! FLAG TO CLEAR THE SCREEN
            CALL TEK_TEXT(TXT_FLG,PORT,RESET,ANS,ANALYTE,AMOUNT,IEXIT)
            GOTO 605      ! NO AGENT FOR THIS SAMP
           ENDIF

      C-----------------------------------------
      C    IF THERE IS GOING TO BE AGENT APPLIED THEN WE COME HERE
      C    I TURN OFF THE OPTIC STAGE CONTROLLER.  TURN ON THE SAMPLE STAGE
      C    CONTROLLER AND MOVE THE THE SAMPLE 180000 STEPS  COUNTER CLOCKWISE.
      C    THIS EQUALS A FULL 90 DEGREE ROTATION.
      C-----------------------------------------

           WRITE(3,'(A2)')'F '              ! DISABLE OPTIC CONTROLLER
           CALL TWAIT(3)
           WRITE(3,'(A4)')GS//'DO'//CR         ! INITIALIZE  CONTROLLER
           CALL TWAIT(3)
           WRITE(3,'(A4)')GS//'D3'//CR         ! INITIALIZE THE SAMPLE
                                      !  CONTROLLER CARD
           CALL TWAIT(5)
           WRITE(3,'(A9)')'E MN ST0 '          ! TURN ON CONTROLLER
                                      ! PLACE IN NORMAL MODE
                                      ! AND POWER DOWN MOTORS.
           CALL TWAIT(3)

           IDIST - 180000
           CALL INT_TO_CHAR(IDIST,START_POS,LE) !  CONVERT IT TO CHARACTER

            TYPE - 3                        ! FLAG TO MOVE SAMPLE
                                      ! STAGE ONLY
            CALL MOV_STAGE(LASER,TYPE,LAMDA(I),START_POS,REPORT,
           .              GRAPHICS)

      C-----------------------------------------
      C    SEND A MESSAGE TO THE USER TO APPLY THE CHEMISTRY AT THIS POINT.
      C    THIS IS PRESENTLY DONE MANUALLY.  JUST WAIT FOR A RETURN TO BE
      C    ENTERED.
      C-----------------------------------------

      620  FORMAT(10X,'THE SAMPLE IS NOW READY FOR THE ',A20)
      622  FORMAT(10X,'APPLY ',A10,' (Mg/m3) TO THE SAMPLE'//)
      624  FORMAT(10X,'PRESS TO GO ON.')

      C-----------------------------------------
      C    IF THE USER IS USING GRAPHICS THEN THE ABOVE TEXT IS DISPLAYED
      C    IN GRAPHIC FORM BY CALL THE TEK TEXT ROUTINE
      C-----------------------------------------
```

```
626   IF(GRAPHICS.EQ.1)THEN

c     PAUSE' ABOUT TO MOVE THE SAMPLE'

        IF(LOOP.EQ.1)THEN
         RESET  - 0                ! FLAG TO DRAW A RED PANEL
        ELSE
         RESET  - 1                ! FLAG NOT TO DRAW A RED PANEL
        ENDIF

        IEXIT  - 0                ! INITIALIZE EXIT FLAG

        ANALYTE - AGENT(LOOP)
        AMOUNT  - CONC(LOOP)

        TXT_FLG - 3                ! FLAG FOR SAMPLE PARAMETERS

        CALL TEK_TEXT(TXT_FLG,PORT,RESET,ANS,ANALYTE,AMOUNT,IEXIT)

          IF(IEXIT.EQ.1)GOTO 10000   ! USER WANTS TO EXIT

C-----------------------------------------------
C     THIS CLEARS THE ENTIRE SCREEN PRIOR TO GOING ON WITH THE PROGRAM
C-----------------------------------------------

         RESET - 2                  ! FLAG TO CLEAR THE SCREEN

         CALL TEK_TEXT(TXT_FLG,PORT,RESET,ANS,ANALYTE,AMOUNT,IEXIT)

         GOTO 628

       ENDIF

       WRITE(*,620)AGENT(LOOP)          ! TELL THE USER TO ADD
       WRITE(*,622)CONC(LOOP)           ! THE SPECIFIED CHEMISTRY
       WRITE(*,624)
       MSG(1:1) - ' '               ! CLEAR THE VARIABLE


       CALL TWAIT(50)
C      READ(*,'(A)',ERR-628)MSG          ! READ THE MESSAGE

       IF(MSG(1:1).EQ.' ')GOTO 628        ! GO SEND STAGE HOME

       GOTO 626                   ! INPUT BAD ..DO IT AGAIN

C-----------------------------------------------
C      THIS MOVES THE SAMPLE STAGE BACK TO ITS HOME POSITION.
C      IT ADDITIONALLY DISABLES THE SAMPLE STAGE CONTROLLER.
C      THEN WAKES THE OPTIC CONTROLLER CARD AND PLACES THE OPTIC
C      STAGE CONTROLLER IN NORMAL MODE.
C-----------------------------------------------

628   IDIST - -180000
```

```
      CALL INT_TO_CHAR(IDIST,START_POS,LE) ! CONVERT IT TO CHARACTER

      TYPE = 3
      CALL MOV_STAGE(LASER,TYPE,LAMDA(I),START_POS,REPORT,
     .               GRAPHICS)

      CALL TWAIT(3)
      WRITE(3,'(A2)')'P '           ! DISABLE SAMPLE CONTROL
      CALL TWAIT(3)

605   WRITE(3,'(A4)')GS//'D0'//CR      ! INITIALIZE  CONTROLLER
      CALL TWAIT(3)
      WRITE(3,'(A4)')GS//'D2'//CR      ! INITIALIZE THE OPTIC
C                                  !  STAGE CONTROLLER CARD
      CALL TWAIT(3)
      WRITE(3,'(A9)')'E MN ST0 '       ! TURN ON CONTROLLER
      CALL TWAIT(3)                    ! ALL MOTORS DEENERGIZED


C-----------------------------------------

      L = LOOP                 ! SHORT SAMPLE COUNTER
      K = 1                    ! INITIALIZE LASER COUNTER

C-----------------------------------------
C     K - COUNT ON THE NUMBER OF LASERS USED IN THE EXPERIMENT
C     HERE THE OPTIC POSITIONS ARE DETERMINED BY THE < LAS(K) >
C-----------------------------------------

610   K = K + LCNT             ! INCREMENT LASER COUNTER

      LASER = LAS(K)

C-----------------------------------------
C     THIS SETS THE TRANSLATION STAGES IN THE PROPER POSITIONS TO
C     SEND THE BEAM TO THE SAMPLE.
C-----------------------------------------

615   TYPE = 2                 ! FLAG TO LINE BEAM WITH SAMPLE

      CALL MOV_STAGE(LASER,TYPE,LAMDA(I),START_POS,REPORT,
     .               GRAPHICS)

      IF(REPORT.GE.1)GOTO 10000      ! EXIT PROGRAM ON ERROR

C-----------------------------------------
C     HERE THE SAMPLE STAGE IS SENT TO ITS START POSITION.
C-----------------------------------------

      START1 = 90.0 - RARM      ! DEGREES TO START POSITION

      IDIST = IFIX( START1 * 2000.0 ) ! STEPPER MOTOR INCREMENTS
      IHOME = IDIST

```

```
        CALL INT_TO_CHAR(IDIST,START_POS,LE) ! CONVERT IT TO CHARACTER

      WRITE(3,'(A2)')'F '

      CALL TWAIT(3)
      WRITE(3,'(A4)')GS//'DO'//CR          ! INITIALIZE  CONTROLLER
      CALL TWAIT(3)
      WRITE(3,'(A4)')GS//'D3'//CR          ! INITIALIZE THE SAMPLE
                                           !  CONTROLLER CARD
      CALL TWAIT(5)

      WRITE(3,'(A9)')'E MN ST0 '           ! TURN ON CONTROLLER
                                           ! PLACE IN NORMAL MODE
                                           ! AND POWER DOWN MOTORS.
      CALL TWAIT(3)

       TYPE = 3
       CALL MOV_STAGE(LASER,TYPE,LAMDA(I),START_POS,REPORT,
     .             GRAPHICS)

C--------------------------------------------
C     THIS PART OPENS THE PROPER SHUTTER SO THAT THE PORPER LASER BEAM
C     CAN CHANNEL ITS WAY TO THE SAMPLE.  BITS 16 - 19 REPRESENT
C     CONTACT CLOSURES FOR LASER SHUTTERS 1 - 4 RESPECTIVELY.
C--------------------------------------------

      CALL TWAIT(2)

      IF(LASER.EQ.1)THEN

       WRITE(3,'(A9)')GS//'%WC20,1'//CR    ! MODULATOR #1 ON
       CALL TWAIT(2)
       WRITE(3,'(A9)')GS//'%WC21,0'//CR    ! MODULATOR #2 OFF
       CALL TWAIT(2)
       WRITE(3,'(A9)')GS//'%WC22,0'//CR    ! MODULATOR #3 OFF
       CALL TWAIT(2)
       WRITE(3,'(A9)')GS//'%WC23,0'//CR    ! MODULATOR #4 OFF


       WRITE(3,'(A9)')GS//'%WC16,1'//CR    ! LASER SHUTTER #1 ON
       CALL TWAIT(2)
       WRITE(3,'(A9)')GS//'%WC17,0'//CR    ! LASER SHUTTER #2 OFF
       CALL TWAIT(2)
       WRITE(3,'(A9)')GS//'%WC18,0'//CR    ! LASER SHUTTER #3 OFF
       CALL TWAIT(2)
       WRITE(3,'(A9)')GS//'%WC19,0'//CR    ! LASER SHUTTER #4 OFF


      ELSEIF(LASER.EQ.2)THEN


       WRITE(3,'(A9)')GS//'%WC21,1'//CR    ! MODULATOR #2 ON
       CALL TWAIT(2)
       WRITE(3,'(A9)')GS//'%WC20,0'//CR    ! MODULATOR #1 OFF
```

```
    CALL TWAIT(2)
    WRITE(3,'(A9)')GS//'%WC22,0'//CR     ! MODULATOR #3 OFF
    CALL TWAIT(2)
    WRITE(3,'(A9)')GS//'%WC23,0'//CR     ! MODULATOR #4 OFF
    CALL TWAIT(2)

    WRITE(3,'(A9)')GS//'%WC16,0'//CR     ! LASER SHUTTER #1 OFF
    CALL TWAIT(2)
    WRITE(3,'(A9)')GS//'%WC17,1'//CR     ! LASER SHUTTER #2 ON
    CALL TWAIT(2)
    WRITE(3,'(A9)')GS//'%WC18,0'//CR     ! LASER SHUTTER #3 OFF
    CALL TWAIT(2)
    WRITE(3,'(A9)')GS//'%WC19,0'//CR     ! LASER SHUTTER #4 OFF

  ELSEIF(LASER.EQ.3)THEN

    CALL TWAIT(2)
    WRITE(3,'(A9)')GS//'%WC22,1'//CR     ! MODULATOR #3 ON

    WRITE(3,'(A9)')GS//'%WC20,0'//CR     ! MODULATOR #1 OFF
    CALL TWAIT(2)
    WRITE(3,'(A9)')GS//'%WC21,0'//CR     ! MODULATOR #2 OFF
    CALL TWAIT(2)
    WRITE(3,'(A9)')GS//'%WC23,0'//CR     ! MODULATOR #4 OFF
    CALL TWAIT(2)
    WRITE(3,'(A9)')GS//'%WC16,0'//CR     ! LASER SHUTTER #1 OFF
    CALL TWAIT(2)
    WRITE(3,'(A9)')GS//'%WC17,0'//CR     ! LASER SHUTTER #2 OFF
    CALL TWAIT(2)
    WRITE(3,'(A9)')GS//'%WC18,1'//CR     ! LASER SHUTTER #3 ON
    CALL TWAIT(2)
    WRITE(3,'(A9)')GS//'%WC19,0'//CR     ! LASER SHUTTER #4 OFF



  ELSEIF(LASER.EQ.4)THEN

    WRITE(3,'(A9)')GS//'%WC23,1'//CR     ! MODULATOR #4 ON
    CALL TWAIT(2)
    WRITE(3,'(A9)')GS//'%WC20,0'//CR     ! MODULATOR #1 OFF
    CALL TWAIT(2)
    WRITE(3,'(A9)')GS//'%WC21,0'//CR     ! MODULATOR #2 OFF
    CALL TWAIT(2)
    WRITE(3,'(A9)')GS//'%WC22,0'//CR     ! MODULATOR #3 OFF
    CALL TWAIT(2)
    WRITE(3,'(A9)')GS//'%WC16,0'//CR     ! LASER SHUTTER #1 OFF
    CALL TWAIT(2)
    WRITE(3,'(A9)')GS//'%WC17,0'//CR     ! LASER SHUTTER #2 OFF
    CALL TWAIT(2)
    WRITE(3,'(A9)')GS//'%WC18,0'//CR     ! LASER SHUTTER #3 OFF
    CALL TWAIT(2)
    WRITE(3,'(A9)')GS//'%WC19,1'//CR     ! LASER SHUTTER #4 ON

  ENDIF
```

```
C--------------------------------------------
C     REESTABLISH THE CONNECTION WITH THE #3 SERIAL NODE CONTROLLER
C--------------------------------------------

      CALL TWAIT(3)
      WRITE(3,'(A4)')CS//'D3'//CR          !
      CALL TWAIT(2)


C--------------------------------------------
C     HERE THE DATA COLLECTION BEGINS.  ALWAYS IN THE SAME ORDER
C--------------------------------------------

      INUSE = 0

      CALL VV (ISHOTS,INUSE,ICNT,IS,SAMP,RINCR,RARM,REND,
     .ANGLE,STP,PORT)

C     CALL PRINT_JT

      CALL V45(ISHOTS,INUSE,ICNT,IS,SAMP,RINCR,RARM,REND,
     .ANGLE,STP)
      CALL P45V(ISHOTS,INUSE,ICNT,IS,SAMP,RINCR,RARM,REND,
     .ANGLE,STP)
      CALL P4545(ISHOTS,INUSE,ICNT,IS,SAMP,RINCR,RARM,REND,
     .ANGLE,STP)



C--------------------------------------------
C     THIS PART DELETES THE CHANNEL GRAPHICS THAT WERE CREATED ON THE
C     LAST POLARIZER PERMUTATION.
C--------------------------------------------

      IF(GRAPHICS.EQ.1)THEN
       ISEG = 850                   ! SEGMENT NUMBER
       CALL INTRPT(ISEG,SEG)            ! CONVERT TO TEK CODE
       WRITE(*,*)E//'SK'//SEG          ! DELETE THE SEGMENT
       WRITE(*,*)E//'KN0'              ! RENEW THE VIEW
      ENDIF
C--------------------------------------------
C     THIS PART CLOSES THE LASER SHUTTER THAT IS CURRENTLY OPEN
C--------------------------------------------

      CALL TWAIT(2)
      IF(LASER.EQ.1)THEN

       WRITE(3,'(A9)')CS//'%WC16,0'//CR     ! LASER SHUTTER #1 OFF

      ELSEIF(LASER.EQ.2)THEN
       WRITE(3,'(A9)')CS//'%WC17,0'//CR     ! LASER SHUTTER #2 OFF

      ELSEIF(LASER.EQ.3)THEN
```

```
          WRITE(3,'(A9)')GS//'%WC18,0'//CR      ! LASER SHUTTER #3 OFF


        ELSEIF(LASER.EQ.4)THEN

         WRITE(3,'(A9)')GS//'%WC19,0'//CR      ! LASER SHUTTER #4 OFF

         CALL TWAIT(2)
         WRITE(3,'(A9)')GS//'%WC20,1'//CR      ! MODULATOR #1 ON

         CALL TWAIT(2)

         WRITE(3,'(A9)')GS//'%WC23,0'//CR      ! MODULATOR #4 OFF

        ENDIF
         CALL TWAIT(2)

C-----------------------------------------------
C     REESTABLISH THE CONNECTION WITH THE #3 SERIAL NODE CONTROLLER
C-----------------------------------------------

        CALL TWAIT(3)
        WRITE(3,'(A4)')GS//'D3'//CR          !
        CALL TWAIT(2)

C-----------------------------------------------
C     THIS PART SENDS ALL THE STAGES TO THIER HOME POSITION SO THAT THE
C     NEXT LASER CAN START FRESH.
C-----------------------------------------------

        WRITE(3,'(A7)')'1ST1 C '

        IDIST = 10000
        MSG = ' '
        IHOME = IHOME - ( 2 * IHOME )

        CALL INT_TO_CHAR(IHOME,START_POS,LE) ! CONVERT IT TO CHARACTER

        TYPE = 3
         CALL MOV_STAGE(LASER,TYPE,LAMDA(I),START_POS,REPORT,
     .               GRAPHICS)

        WRITE(3,'(A21)')'2PS 2ST1 2H 2G 2CR C '
        CALL TWAIT(3)

        READ(3,'(A80)',ERR=630)MSG

 630    CALL TWAIT(3)
        MSG = ' '

        WRITE(3,'(A5)')'2ST0 '
        CALL TWAIT(3)

        WRITE(3,'(A21)')'3PS 3ST1 3H 3G 3CR C '
```

```
      CALL TWAIT(3)
     READ(3,'(A80)',ERR-640)MSG

640   WRITE(3,'(A5)')'3ST0 '
     MSG - ' '
     CALL TWAIT(3)

     WRITE(3,'(A2)')'F '          ! TURN OFF THE SAMPLE CONTROLLER
     CALL TWAIT(3)


     IF(GRAPHICS.EQ.1)THEN
       E - CHAR(27)
       ISEG - 850
       CALL INTRPT(ISEG,SEG)
       WRITE(*,*)E//'SK'//SEG
       WRITE(*,*)E//'KN0'
     ENDIF

C-----------------------------------------------
C     THIS PART KEEPS A TOTAL COUNT OF THE DATA RECORDS ON FILE
C-----------------------------------------------

900   INQUIRE(FILE-'INDEX',EXIST-T) !IS THERE A FILE HEADER NUMBER

     IF (.NOT.T)THEN
      RECNUM - 2             ! THE FILE HAS NOT YET GOT DATA

     OPEN (UNIT-1,FILE-'INDEX',ACCESS-'DIRECT',STATUS-'UNKNOWN',
    .FORM-'FORMATTED', RECL-134)

      WRITE(1,REC-1,FMT - 7,ERR-970)RECNUM   ! TOTAL RECORD NUMBER

       WRITE(*,*)'THE CURRENT RECORDS ON FILE ARE ',RECNUM - 1
      GOTO 940
     ENDIF

C-----


     OPEN (1,FILE-'INDEX',ACCESS-'DIRECT',STATUS-'UNKNOWN',
    .FORM-'FORMATTED', RECL-134)

     READ(1,REC-1,FMT-7,ERR-920)RECNUM

920   WRITE(*,*)'THE CURRENT RECORDS ON FILE ARE ',RECNUM

C***************************************************************
C     BELOW THE INDEX FILE IS WRITTEN TO AND WILL CONTAIN
C     THE DATE,SAMPLE,AGENT,AND THE LASER WAVE LENGTHS
C     CONCENTRATION VALUES WILL BE LISTED WITH HEADER DATA
C***************************************************************

956   FORMAT(I5,A20,A9,A8,2(A20),A15,A15,I4,3(F6.2))
```

```
940   WRITE(1,REC-RECNUM,FMT-956,ERR-970)RECNUM,NAME,DATE,TIME,
     .SAMP(L),AGENT(L),CONC(L),LAMDA(K+LCNT),ISHOTS,RARM,REND,RINCR

      WRITE(1,REC-1,FMT-7,ERR-970)RECNUM+1  ! INC & WRITE COUNT TO FILE
970   CLOSE (1)

C*****************************************************************
C
C     NOW THE ACTUAL DATA IS STORED.  IT WILL SAVE AS MANY FILES FOR
C     EACH SAMPLE AS THERE ARE LASERS (DIFFERENT WAVELENGTHS).  EACH
C     TIME THIS MOD IS REACHED ONE SAMPLE HAS BEEN LOOKED AT.
C
C*****************************************************************

995   L1 - 0

      CALL INT_TO_CHAR(RECNUM,FNAME,LE)
      FILENM-FNAME//'DAT'

      DO 1000 J-1,20
        IF (FILENM(J:J).EQ.CHAR(32))GOTO 1000
        L1-L1+1
        FLNM(L1:L1)-FILENM(J:J)        ! SHORTEN THE FILE NAME
1000  CONTINUE

      FILENM - ' '
      FNAME - ' '

1005  FORMAT(F7.5)

      OPEN(1,FILE-FLNM,ACCESS-'DIRECT',RECL-7,FORM-'FORMATTED',
     . STATUS-'NEW',ERR-1090)

      I1 - ISHOTS * 10

1010  DO 1015 KK - 1,I1
       WRITE(1,REC-KK,FMT-1005,ERR-1015)ARAY(KK)
1015  CONTINUE

      J - 0
      I2 - KK

1020  DO 1025 KK - I2,I1 + I2 - 1
       J - J + 1
       WRITE(1,REC-KK,FMT-1005,ERR-1025)BRAY(J)
1025  CONTINUE

      J - 0
      I2 - KK
1030  DO 1035 KK - I2,I1 + I2 - 1
      J - J + 1
       WRITE(1,REC-KK,FMT-1005,ERR-1035)CRAY(J)
1035  CONTINUE
```

```
      J - 0
      I2 - KK

1040  DO 1045 KK - I2,I1 + I2 - 1
      J - J + 1
       WRITE(1,REC-KK,FMT-1005,ERR-1045)DRAY(J)
1045  CONTINUE

      I1 - ISHOTS * 16
      I2 - KK
      J - 0

1050  DO 1055 KK - I2,I1 + I2 - 1
       J - J + 1
       WRITE(1,REC-KK,FMT-1005,ERR-1055)ERAY(J)
1055  CONTINUE

      CLOSE(1)

C-------------------------------------------------------
C     NOW I SWITCH CONTROL TO THE OPTIC STAGE CONTROLLER
C     IF THIS IS LASER #1  NO ACTION NEED BE DONE.
C     FOR LASERS 2 - 4 I SEND THE STEPPER MOTOR #1 OPTIC STAGE HOME
C     IF THIS IS LASER #3 I SEND STAGE #2 -63000 STEPS HOME
C     IF THIS IS LASER #4 I SEND STAGE #2 -126000 STEPS HOME..
C-------------------------------------------------------

      WRITE(3,'(A4)')GS//'D0'//CR          ! INITIALIZE  CONTROLLER
      CALL TWAIT(3)
      WRITE(3,'(A4)')GS//'D2'//CR         ! INITIALIZE THE OPTIC
                                   ! STAGE CONTROLLER CARD
      CALL TWAIT(3)                  ! PAUSE 3/10 SEC
      WRITE(3,'(A9)')'E MN ST0 '          ! TURN ON CONTROLLER
      CALL TWAIT(3)

C-------------------------------------------------------
C     THIS IS THE END OF THE LASER LOOP.  WHATEVER LASER WAS LAST TO
C     TAKE DATA MUST BE SENT HOME BEFORE WE MOVE TO THE NEXT SAMPLE.
C-------------------------------------------------------

      IF (K.GE.LASERS(LOOP))THEN            ! THIS WAS THE LAST LASER
       IF(LAS(K).EQ.1)GOTO 1090          ! IF LASER #1 DO NOTHING
       GOTO 1060                      ! ELSE MOVE ON..
      ENDIF

C-------------------------------------------------------
C     IF THIS IS LASER #1 AND NOT THE LAST LASER THEN WE INCREMENT THE
C     LASER COUNT AND GO GET THE NEXT LASER DATA.  NO MOTORS HAVE
C     TO BE MOVED.
C-------------------------------------------------------

      IF(LAS(K).EQ.1)THEN
       K - K + 1
```

```
        GOTO 610              ! DO NOTHING IF LASER #1
      ENDIF

C------------ HERE I SEND STEPPER #1 TO ITS HOME ----------

1060  WRITE(*,*)'LASER - ',LAS(K)
      WRITE(3,'(A35)')'1PS 1ST1 1V20 1A35 1D52000 G 1X1 C '
      CALL TWAIT(3)                  ! PAUSE 3/10 SEC
      READ(3,'(A255)')MSG
      WRITE(*,*)' MSG - ',MSG
      CALL TWAIT(3)
      WRITE(3,'(A5)')'1ST0 '
      CALL TWAIT(3)                  ! PAUSE 3/10 SEC

C------------------------------------------------
C     IF LASER IS NUMBER 3 I SEND IT HOME
C------------------------------------------------

      IF(LAS(K).EQ.3)THEN
        WRITE(3,'(A36)')'2PS 2ST1 2V20 2A35 2D-63000 G 2X1 C '
        WRITE(*,*)' AT A READ SENDING MOTOR #2 HOME'
        CALL TWAIT(3)
        READ(3,'(A80)')MSG
        WRITE(*,*)' MSG - ',MSG

        WRITE(3,'(A5)')'2ST0 '
        CALL TWAIT(3)
      ENDIF

C------------------------------------------------
C     IF LASER IS NUMBER 4 I SEND IT HOME HERE
C------------------------------------------------

      IF(LAS(K).EQ.4)THEN
        WRITE(3,'(A37)')'2PS 2ST1 2V20 2A35 2D-126000 G 2X1 C '
        WRITE(*,*)' AT A READ SENDING MOTOR #2 HOME'
        CALL TWAIT(3)
        READ(3,'(A80)')MSG

        WRITE(*,*)' MSG - ',MSG

        WRITE(3,'(A5)')'2ST0 '
        CALL TWAIT(3)
      ENDIF

C------------------------------------------------

      IF (K.GE.LASERS(LOOP))THEN    ! THIS WAS THE LAST LASER
        LCNT - LCNT + 4            ! INCREMENT THE ARRAY COUNTER
        GOTO 1090                  ! GOTO NEXT SAMPLE IF ANY
      ENDIF
C------------------------------------------------
C     HERE WE CONTINUE THRU THE LASER LOOP WHERE WE ASK THE NEXT
C     LASER TO SHINE ON THE SAMPLE.
```

```
C--------------------------------------------------

      K - K + 1              ! INCREMENT LASER COUNTER
      GOTO 610               ! GOTO NEXT LASER IF THERE ARE ANY


C--------------------------------------------------
C     THIS MAKES THE USER TUNE THE LASER FOR THE NEXT SAMPLE.
C--------------------------------------------------

1090  DO 1100 IJ - 1,LASERS(I + 1)          ! LOOP THRU LASERS

      OPTIC - CHAR(48 + LAS(IJ + LCNT))  ! ACTUAL LASER NUMBER
      LASER - LAS(IJ + LCNT)          ! HOLD IN BUFFER
      TYPE - 1                        ! FLAG FOR SPECTRUM SETUP


      CALL MOV_STAGE(LASER,TYPE,LAMDA(I),START_POS,REPORT,
     .               GRAPHICS)

1100  CONTINUE


C--------------------------------------------------
C     THIS ERASES THE RED TEXT SCREEN
C--------------------------------------------------

        RESET  - 2                    ! DELETE THE RED PANEL
        CALL TEK_TEXT(TXT_FLG,PORT,RESET,ANS,
     .               ANALYTE,AMOUNT,IEXIT)

C--------------------------------------------------
C     THIS IS THE BOTTOM OF THE SAMPLE LOOP
C--------------------------------------------------

1070  CONTINUE

      WRITE(3,'(A2)')'P '             ! DISABLE CONTROLLER
      WRITE(3,'(A4)')GS//'DO'//CR       ! INITIALIZE  CONTROLLER
                                      ! COMMUNICATIONS

C--------------------------------------------------

      IFLAG-0


10000 CLOSE(3)                        ! CLOSE CONTROLLER COM PORT

      E - CHAR(27)
      WRITE(*,*)E//'RF0'              ! SET FIXUP LEVEL - 0
      WRITE(*,*)E//'LV0'              ! DISABLE THE DIALOG AREA
      WRITE(*,*)E//'SK!'              ! DELETE ALL SEGMENTS
      WRITE(*,*)E//'RF6'              ! RESET THE FIXUP LEVEL
      WRITE(*,*)E//'KN0'              ! RENEW THE VIEW
```

```
10010 RETURN                    ! RETURN TO CALLER

      END
```

AIV.17 Analog APSD Software Modules: P4545 Source Code.

```
      SUBROUTINE P4545(ISHOTS,INUSE,ICNT,IS,SAMP,RINCR,RARM,REND,
     .ANGLE,STP)

C-----------------------------------------------------------
C     THIS MOD CAPTURES THE DATA FOR THE POLARIZERS IN POSITION
C     45 deg/45 deg.  TWO ARRAYS ARE USED TO SAVE THE DATA.
C     DRAY - IS A SEQUENTIAL ARRAY STORING ALL THE DATA RECEIVED FROM
C     THE A/D CONVERTER THRU THE ENTIRE ROTATION OF THE SAMPLE.
C
C     ERAY - IS A REPRESENTATION OF THE ENTIRE MATRIX.  NINE ELEMENTS
C     WILL BE PASSED IN FROM THE A/D CONVERTER OF WHICH ONLY 1 WILL
C     BE NEW. IT IS:  ( 11 ) FROM  THE 1,3,4,9,11,12,13,15,16.
C     THIS WILL COMPLETE THE ERAY DATA COLLECTION.
C-----------------------------------------------------------


      CHARACTER GETDAT*4,SAMP*20,MSG*50,MESG*255,HEX*4
      CHARACTER PORT*10,CHAN*2

      REAL RDAT,ARAY,BRAY,CRAY,DRAY,ERAY,RINCR,REND,ANGLE,RARM

      INTEGER STP,TYPE,REPORT

      DIMENSION GETDAT(16),RDAT(16),HEX(16)

      COMMON /MATRIX/ARAY(3600),BRAY(3600),CRAY(3600),DRAY(3600),
     . ERAY(5800)

      ISWEEP = 4                  ! FLAG THAT THIS IS THE 4TH
                                  ! SWEEP OF THE POLARIZERS
C-----------------------------------------------------------
C     THIS IS THE LAST POLARIZER SETTING AND DATA COLLECTION ROUTINE
C     THE RECEIVER POLARIZER IS SENT TO -45 DEGS.  THE SAMPLE STAGE
C     IS SET IN THE REVERSE DIRECTION TO TRAVEL BACK TO THE START
C     POSITION.
C-----------------------------------------------------------

      WRITE(3,'(A4)')'1X0 '       ! RESET THE CUMM POSITION CNTR
      CALL TWAIT(1)               ! MOVE RECEIVER POLARIZER -45 DEGS
      WRITE(3,'(A38)')'3PS 3ST1 3V10 3A10 3D-225000 3G 3CR C '
      READ(3,'(A50)')MSG(1:50)    ! WAIT FOR CARRIAGE RETURN
      CALL TWAIT(1)
      WRITE(3,'(A5)')'3ST0 '      ! DEENERGIZE AXIS 3 MOTOR
      CALL TWAIT(1)
      WRITE(3,'(A12)')'1PS 1ST1 1H ' ! ENERGIZE AND REVERSE SAMPLE
      CALL TWAIT(1)

      INUSE1 = 0
C-----------------------------------------------------------
C     HERE BEGINS THE LOOP WHERE SAMPLE DATA IS TAKEN AND STORED
C-----------------------------------------------------------
```

```
      DO 200 M - 1,ISHOTS           ! LOOP THRU THE SAMPLE ROTATION

C-----------------------------------------------------------
C     HERE IS WHERE THE A/D CONVERTER IS ASKED FOR THE DATA
C-----------------------------------------------------------

C     CALL TESTDAT(ISWEEP,ANGLE,RDAT)

      IC1 - 11
      TYPE - 3
      CALL DATEL(TYPE,REPORT,PORT,IC1,CHAN,RDAT,HEX)

C-----------------------------------------------------------
C     HERE EACH ARRAY IS SELECTED, WRITTEN TO AND INCREMENTED
C-----------------------------------------------------------

        DO 100 I - 10,1,-1
          DRAY(IS) - RDAT(I)        ! STORE THE 10 ARRAY ELEMENTS
          IS - IS - 1               ! ARAY SEQUENTIAL COUNTER
100       CONTINUE

C-----------------------------------------------------------
C     HERE ONLY ONE ELEMENT OF THE ERAY MATRIX IS NEW.  IT IS
C     ELEMEN" 11. AS STORED IN RDAT IN THE 5TH POSITION.
C-----------------------------------------------------------

          K - 11 + ICNT             ! SELECT CORRECT ARRAY ELEMENT
          ERAY(K) - RDAT(5)         ! PLACE MATRIX DATA IN ERAY

C-----------------------------------------------------------
C     THE STP DETERMINES THE TYPE OF OUTPUT THE USER IS REQUESTING
C-----------------------------------------------------------

        IF(STP.EQ.1)THEN            ! USER WANTS REAL TIME GRAPHICS
          CALL DRAW_ELE(ISWEEP,ANGLE,RARM,RINCR,REND,RDAT,
     .            INUSE1)
          GOTO 150                  ! FLAG TO SKIP THE VIEW
        ENDIF

        IF(STP.EQ.3)GOTO 150        ! NO OUTPUT IS REQUESTED

        CALL VIEW(SAMP,ANGLE,IS,STP,IDIR) ! A/D VOLTAGE OUTPUTS

150     IF(M.EQ.ISHOTS)GOTO 200

        ANGLE - ANGLE - RINCR       ! NEW SAMPLE ANGLE

        ICNT - ICNT - 16            ! DECREMENT THE ARRAY BY 16

      WRITE(3,'(A9)')'1G 1X1 C '    ! MOVE THE SAMPLE
      READ(3,'(A50)',ERR - 200)MESG(1:50)
      CALL TWAIT(1)
      READ(MESG(14:22),'(BN,I9)',ERR - 200)MOTION
```

```
          ICNG - MOTION - MBUP
        MBUP - MOTION
   160  FORMAT(5X,I4,10X,' ACCUMULATED MOTION: ',I9,
       .'  RELATIVE MOTION:  ',I9//)
D       WRITE(*,160)M,MOTION,ICNG
        WRITE(3,'(A4)')'1PS '          ! PAUSE THE STEPPER MOTOR

   200  CONTINUE                       ! LOOP THRU ROTATIONS

C------------------------------------------------------------
C       AT THIS POINT THE DATA COLLECTION FOR POLARIZERS POSITIONED AT
C       45 deg/45 deg IS COMPLETE.  WE NOW RETURN TO THE CALLER
C------------------------------------------------------------

        WRITE(3,'(A5)')'1ST0 '         ! DEENERGIZE THE SAMPLE STAGE

   400  RETURN
        END
```

AIV.18 Analog APSD Software Modules: P45V Source Code.

```
      SUBROUTINE P45V(ISHOTS,INUSE,ICNT,IS,SAMP,RINCR,RARM,REND,
     .ANGLE,STP)

C---------------------------------------------
C     THIS MOD CAPTURES THE DATA FOR THE POLARIZERS IN POSITION
C     45 deg/VERTICLE.  TWO ARRAYS ARE USED TO SAVE THE DATA.
C     CRAY - IS A SEQUENTIAL ARRAY STORING ALL THE DATA RECEIVED FROM
C     THE A/D CONVERTER THRU THE ENTIRE ROTATION OF THE SAMPLE.
C
C     ERAY - IS A REPRESENTATION OF THE ENTIRE MATRIX.  NINE ELEMENTS
C     WILL BE PASSED IN FROM THE A/D CONVERTER OF WHICH ONLY 3 WILL
C     BE NEW. THEY ARE :  (3,7,15) FROM  THE 1,3,4,5,7,8,13,15,16.
C     ONLY ONE GAP WILL BE LEFT IN THE ARRAY AT THE 11 POSITION WHICH
C     WILL BE FILLED IN BY THE LAST POLARIZER SETTING.
C---------------------------------------------


      CHARACTER GETDAT*4,SAMP*20,MSG*50,MESG*255,HEX*4,PORT*10
      CHARACTER CHAN*2

      REAL RDAT,ARAY,BRAY,CRAY,DRAY,ERAY,RARM,RINCR,ANGLE,REND

      INTEGER POS,STP,REPORT,TYPE

      DIMENSION RDAT(16),J1(3),HEX(16)

      COMMON /MATRIX/ARAY(3600),BRAY(3600),CRAY(3600),DRAY(3600),
     . ERAY(5800)

      DATA J1 /3,7,15/           ! NEW MATRIX ARRAY ELEMENTS

      ISWEEP = 3                 ! FLAG THAT THIS IS 3RD SWEEP
                                 ! OF THE POLARIZERS
C---------------------------------------------
C     THIS ADJUSTS THE INSTRUMENTS:
C
C     THE CUMM POSITION BUFFER IS INITIALIZED
C     THE TXMITER POLARIZER IS SENT TO -45 DEGS.
C     THE RECEIVER IS SENT TO HOME OR VERTICAL
C     THE SAMPLE STAGE IS SET TO REVERSE THE DIRECTION
C---------------------------------------------

      WRITE(3,'(A4)')'1X0 '       ! RESET CUMM POSITION COUNTER
      CALL TWAIT(1)
      WRITE(3,'(A21)')'3PS 3ST1 3H 3G 3CR C ' ! SEND RECEIVER HOME
      READ(3,'(A50)')MSG(1:50)          ! WAIT FOR CARRIAGE CNTRL
      CALL TWAIT(1)
      WRITE(3,'(A5)')'3ST0 '            ! DEENERGIZE AXIS #3
      CALL TWAIT(1)                     ! SEND XTMR -45 DEGS
      WRITE(3,'(A37)')'2PS 2ST1 2V10 2A10 2D225000 2G 2CR C '
      READ(3,'(A50)')MSG(1:50)          ! WAIT FOR CARRIAGE CNTRL
```

```
      CALL TWAIT(1)
    WRITE(3,'(A5)')'2ST0 '          ! DEENERGIZE AXIS #2
    CALL TWAIT(1)
    WRITE(3,'(A13)')'1PS 1ST1 1H '       ! REVERSE SAMPLE STAGE
    INUSE1 - 0
C-----------------------------------
C    HERE BEGINS THE LOOP WHERE SAMPLE DATA IS TAKEN AND STORED
C-----------------------------------
    DO 200 M - 1,ISHOTS          ! LOOP THRU THE SAMPLE ROTATION
C-----------------------------------
C    HERE IS WHERE THE A/D CONVERTER IS ASKED FOR THE DATA
C-----------------------------------
C    CALL TESTDAT(ISWEEP,ANGLE,RDAT)
    IC1 - 11
    TYPE - 3
    CALL DATEL(TYPE,REPORT,PORT,IC1,CHAN,RDAT,HEX)
C-----------------------------------
C    HERE EACH ARRAY IS SELECTED, WRITTEN TO AND INCREMENTED
C-----------------------------------
      DO 100 I-1,10
       IS - IS + 1          ! ARAY SEQUENTIAL COUNTER
       CRAY(IS) - RDAT(I)      ! STORE THE 10 ARRAY ELEMENTS
100    CONTINUE
C-----------------------------------
C    HERE ONLY THREE ELEMENTS OF THE ERAY MATRIX ARE NEW.  THEY ARE
C    ELEMENTS 3,7,15. THEY ARE STORED IN THE RDAT ARRAY IN POSITIONS
C    2,5,8
C-----------------------------------
       K - J1(1) + ICNT      ! SELECT CORRECT ARRAY ELEMENT
       ERAY(K) - RDAT(2)       ! PLACE MATRIX DATA IN ERAY
       K - J1(2) + ICNT      ! SELECT CORRECT ARRAY ELEMENT
       ERAY(K) - RDAT(5)       ! PLACE MATRIX DATA IN ERAY
       K - J1(3) + ICNT      ! SELECT CORRECT ARRAY ELEMENT
       ERAY(K) - RDAT(8)       ! PLACE MATRIX DATA IN ERAY
C-----------------------------------
    IF(STP.EQ.1)THEN          ! USER WANTS REAL TIME GRAPHICS
     CALL DRAW_ELE(ISWEEP,ANGLE,RARM,RINCR,REND,RDAT,INUSE1)
     GOTO 150
    ENDIF
```

```
      IF(STP.EQ.3.)GOTO 150

      CALL VIEW(SAMP,ANGLE,IS,STP,IDIR)

150   IF(M.EQ.ISHOTS)GOTO 200       ! DO NOT ROTATE SAMPLE ON LAST

      ANGLE = ANGLE + RINCR         ! NEW SAMPLE ANGLE

      ICNT=ICNT+16                  ! INCREMENT THE ARRAY BY 16


C-----------------------------------------

      WRITE(3,'(A9)')'1G 1X1 C '    ! MOVE THE SAMPLE STAGE
      READ(3,'(A50)',ERR = 200)MESG(1:50)
      CALL TWAIT(1)
      READ(MESG(14:22),'(BN,I9)',ERR = 200)MOTION
      ICNG = MOTION - MBUF
      MBUF = MOTION
170   FORMAT(5X,I4,10X,' ACCUMULATED MOTION: ',I9,
     .' RELATIVE MOTION: ',I9//)
D     WRITE(*,170)M,MOTION,ICNG
      WRITE(3,'(A4)')'1PS '
      CALL TWAIT(1)
200   CONTINUE                      ! LOOP THRU ROTATIONS

C-----------------------------------------
C     AT THIS POINT THE DATA COLLECTION FOR POLARIZERS POSITIONED AT
C     45 deg/VERTICLE IS COMPLETE. WE NOW RETURN TO THE CALLER
C     WHERE THE NEXT POLARIZER SETTING WILL BE MADE AND THE SAMPLE WILL
C     BE ROTATED BACKWARDS.
C-----------------------------------------

      WRITE(3,'(A5)')'1ST0 '        ! DEENERGIZE SAMPLE STAGE

400   RETURN
      END
```

AIV.19 Analog APSD Software Modules: READ_Q10_1 Source Code.

```
      SUBROUTINE READ_QIO(DEVICE,MESC,LENGTH,TIMEOUT,ISIZE,
     .                 IOUT,CHANNEL,INUSE)

C-----------------------------------------------------------
C     THIS IS A ROUTINE USED FOR VAX VMS HARDWARE TO ALLOW FOR A QUEUED
C     I/O READ.  IN THIS CASE I AM INTERRESTED IN READING A SERIAL PORT
C     WITH A TIMEOUT SPECIFIED IN SECONDS ( FROM THE USER ).
C
C     DEVICE - THE SERIAL PORT THAT IS READ FROM        ( IN )
C
C     METSTR - THE STRING OF DATA THAT IS READ          ( OUT )
C
C     LENGTH - THE MAXIMUM LENGTH OF THE STRING THATS READ ( IN )
C
C     TIMEOUT - THE TIME IN SECONDS TO WAIT AT THE PORT FOR DATA ( IN )
C
C     ISIZE  - THE LENGTH OF THE STRING THAT IS READ    ( OUT )
C
C     IOUT   - FLAG THAT HAS 2 OPTIONS:             ( IN/OUT )
C
C          1. IOUT IS SET IN THE NORMAL MODE TO FLAG WHETHER
C             ANY DATA WAS READ FROM THE PORT PRIOR TO TIMING
C             OUT.   IOUT - 1   NO DATA READ AND PORT TIMED OUT
C                    IOUT - 0   DATA WAS READ     ( OUT )
C
C          2. IOUT IS ALSO USED AS A FLAG FROM THE USER WHEN
C             THE READ ROUTINE IS COMPLETE.  THIS IS DONE SO THAT
C             THE ASSIGNED CHANNEL FOR THE SERIAL PORT CAN BE
C             DEASSIGNED.  IOUT - 99            ( IN )
C-----------------------------------------------------------

      IMPLICIT INTEGER*4 (A-Z)
      INCLUDE '($IODEF)'          ! EXTERNAL VMS VAX DEFINITIONS
      INCLUDE '($SSDEF)'          ! EXTERNAL VMS VAX DEFINITIONS

      CHARACTER*512   METSTR           ! STRING RETURNED TO CALL
      CHARACTER*255   MESC             ! INPUT STRING
      CHARACTER*10    DEVICE           ! THIS IS THE PORT
      INTEGER*2       CHANNEL          ! CHANNEL ASSIGNED BY
                                  ! THE SYSTEM
      INTEGER*2       ISIZE            ! LENGTH OF INPUT STRING
      INTEGER*2       IOUT             ! FLAG FOR TIMEOUT
      INTEGER*4       LENGTH           ! LENGTH OF RECEIVED STR
      INTEGER*4       TIMEOUT          ! PORT TIMEOUT IN SECS
      INTEGER*4       MASK(2)          ! QUAD WORD ARRAY THAT
                                  ! SETS PORT TERMINATOR
                                  ! CHARACTER.
      INTEGER*4       FUNCTION         ! FUNCTION FOR QIO READ

C-----------------------------------------------------------
C     HERE I CREATE A STRUCTURED BLOCK THAT IS USED FOR THE IOSB.
```

```
      C    THIS INCLUDES THE STATUS TO THE QUEUED OPERATION.
C     THIS IS NECESSARY BECAUSE THE QIOW MIGHT RETURN A 1 AS STAUS
C     THERE WILL BE 4 DIFFERENT TYPES OF STATUS INFORMATION.
C     1. IOSTAT      - RETURNS THE STATUS OF THE READ  1 - GOOD
C     2. TERM_OFFSET - CHARACTERS READ AT THE PORT
C     3. TERMINATOR  - ASCI VALUE OF TERMINATOR OR 1ST CHARACTER OF TERM
C     4. TERM_SIZE   - LENGTH OF THE TERMINATOR STRING
C-----------------------------------------
      STRUCTURE /IOSTAT_BLK/
      INTEGER*2     IOSTAT,
     .              TERM_OFFSET,
     .              TERMINATOR,
     .              TERM_SIZE
      END STRUCTURE
      RECORD /IOSTAT_BLK/ IOSB
C-----------------------------------------
C THIS STRUCTURE BLOCK IS USED TO ALLOW ANY ASCII CHARACTER UP TO 7 BITS
C TO BE USED AS A TERMINATION CHARACTER IN A QIO READ OPERATION.
C-----------------------------------------
      STRUCTURE /TERM_BLK/          ! CHAR0_15 IS THE NAME FOR THE
      INTEGER*2    CHAR0_15,       ! TWO BYTES OF THE STRUCTURE.  THIS
     .             CHAR16_31,      ! STRUCTURE IS 16 CONTIGUOUS BYTES
     .             CHAR32_47,      ! IN MEMORY WITH EACH BIT IN THE
     .             CHAR48_63,      ! STRUCTURE CORRESPONDING TO THE
     .             CHAR64_79,      ! ASCII CHARACTER WITH THAT VALUE.
     .             CHAR80_95,      ! IF WE WANT TO SET THE CHAR "]" TO
     .             CHAR96_111,     ! BE A TERMINATION CHAR FOR A QIO
     .             CHAR112_127     ! READ THEN WE SET THE 93RD BIT IN
      END STRUCTURE                ! THE STRUCTURE BY :
                           ! TERM_CHAR.CHAR80_95 - 2**13
      RECORD /TERM_BLK/ TERM_CHAR
C-----------------------------------------
      STRUCTURE /PARAMETERA/
      INTEGER*2    MASK_SIZE,     ! # BYTES IN CHAR0 - CHAR127 FIELDS
     .             DUMMY          ! NOT USED
      INTEGER*4    TERM_LOC       ! ADDRESS OF TERM_CHAR STRUCTURE
      END STRUCTURE
      RECORD /PARAMETERA/ MASK_P4
      MASK_P4.TERM_LOC - %LOC(TERM_CHAR)
      MASK_P4.MASK_SIZE - 16       ! # OF BYTES IN CHAR STRUCTURES
      TERM_CHAR.CHAR32_47 - 2**10   ! STAR IS TERMINATOR
                           ! SET - 1 IF TIMED OUT
```

```
      C------------------------------------
C     HERE I CHECK THE FLAG "IOUT" FOR A CHANGE IN MODE. THAT IS A CALL
C     TO DEASSIGN THE CURRENT CHANNEL ASSIGNED TO THE PORT.
C     THIS IS DONE USUALLY AT THE END OF THE CALLING ROUTINE.
C------------------------------------

      IF(IOUT.EQ.99)THEN
       IOUT - 0                ! INITIALIZE TIMEOUT FLAG
       STATUS - SYS$DASSGN(%VAL(CHANNEL))  ! SYSTEM ASSIGNED CHANNEL
       IF(.NOT. STATUS) CALL LIB$STOP (%VAL(STATUS))
       GOTO 1000                   ! RETURN TO CALLER
      ENDIF

C------------------------------------
C     THIS IS THE CHANNEL ASSIGNMENT FUNCTION. THIS CHANNEL IS ASSIGNED
C     TO THE DEVICE THAT WILL BE FUNCTIONING WITH THE QUEUED I/O.
C     HERE THE DEVICE IS INPUT BY THE USER AND THE SYSTEM ASSIGNS THAT
C     DEVICE A CHANNEL.
C------------------------------------



      IF(INUSE.EQ.0)THEN              ! ASSIGN CHANNEL ONLY
       INUSE - 1                   ! ONCE.
       STATUS - SYS$ASSIGN(DEVICE,       ! USER READ DEVICE
     .                 CHANNEL,         ! SYSTEM ASSIGNED CHANNEL
     .                 ,             ! PRIVILEGED ACCESS MODE
     .                 ,)            ! LOGICAL NAME OF MAILBOX

       IF(.NOT. STATUS) CALL LIB$STOP (%VAL(STATUS))

      ENDIF

C------------------------------------
C     THIS IS THE QUEUED INPUT/OUTPUT REQUEST.
C------------------------------------

      IOUT - 0                ! INITIALIZE TIMEOUT FLAG
      MESG - ''               ! CLEAR OUT THE MESSAGE STRING
      METSTR - ''              ! CLEAR OUT THE SEND BACK STRING

                           ! ASSIGN THE FUNCTION OF THE QIOW
      FUNCTION - IO$_READVBLK.OR.IO$M_TIMED

                           ! TO BE A READ
     STATUS - SYS$QIOW(,          ! QIO COMPLETION EVENT FLAG
     .           %VAL(CHANNEL),  ! ASSIGN THE CHANNEL FOR QIOW
     .           %VAL(FUNCTION), ! TYPE OF QIOW REQUESTED BY USER
     .           IOSB,        ! I/O STATUS BLOCK TO CHECK I/O
     .           ,            ! AST ROUTINE TO BE EXECUTED
     .           ,            ! AST PARAMETERS TO ABOVE ROUTINES
     .           %REF(MESG),    ! P1 - INPUT DATA FROM DEVICE
     .           %VAL(LENGTH),  ! P2 - LENGTH OF THE INPUT DATA
     .           %VAL(TIMEOUT), ! P3 - PORT TIMEOUT ON THE READ
```

```
      .          MASK_P4,      ! P4 - SET TERMINATOR CHARACTER
      .          ,             ! P5 - CHARACTER FOR READ PROMPT
      .          )             ! P6 - SIZE OF THE PROMPT


C-------------------------------------------------
C     THIS IS THE DIAGNOSTICS SECTION OF THE ROUTINE.
C     IOUT = 1 MEANS THAT THE PORT HAS TIMED OUT ON THE READ.
C     ISIZE =  THE LENGTH OF THE STRING
C     METSTR = THE READ DATA SENT BACK TO THE CALLER
C     I INCREMENT THE ISIZE VARIABLE TO INCLUDE THE TERMINATOR CHARACTER
C-------------------------------------------------

       IF(.NOT. STATUS) CALL LIB$STOP (%VAL(STATUS))

      IF(.NOT.IOSB.IOSTAT)THEN

      IF(IOSB.IOSTAT.EQ.SS$_TIMEOUT)THEN
       IOUT = 1               ! FLAG THAT PORT TIMED OUT
       GOTO 1000
      ENDIF
      ENDIF

C-------------------------------------------------

      ISIZE = IOSB.TERM_OFFSET
      ISIZE = ISIZE + 1           ! LET ISIZE INCLUDE THE TERMINATOR
      METSTR = MESG(1:ISIZE)        ! CHARACTER

1000  RETURN
      END
```

AIV.20 Analog APSD Software Modules: REAL_to_CHAR Source Code.

```
      SUBROUTINE REAL_TO_CHAR (RNUM,ICNT,CNUM)

C*************************************************************
C     THIS IS DESIGNED TO TAKE IN A REAL NUMBER(UNDER 10.0)
C     AND CONVERT THEM TO A CHARACTER FOR USE WITH A
C     GRAPHICS ROUTINE.  I TAKE THEM OUT TO THE THOUSANDS PLACE
C
C     RNUM - ARRAY OF UP TO 16 REAL NUMBERS  ( PASSED IN BY CALLER )
C
C     ICNT - THE NUMBER OF REALS TO BE CONVERTED  (  PASSED IN )
C
C     CNUM - ARRAY OF THE 6 CHARACTER REPRESENTATION OF THE REALS
C             ( PASSED BACK TO THE CALLER )
C*************************************************************

      REAL RNUM,RBUF

      CHARACTER CNUM*6

      DIMENSION RNUM(16),CNUM(16)

C--------------------------------------

      DO 50 J - 1,ICNT                    ! LOOP THRU REALS

      RBUF - RNUM(J)

      IF(RBUF.LT.0.0)CNUM(J)(1:1) - '-'      ! IF NUMBER IS NEGATIVE
                                      ! WILL REQUIRE A SIGN

      IF(RBUF.GE.10.0)GOTO 50              ! CANT HANDLE NUMBERS
                                      ! GREATER THAN 10.0

      RBUF - ABS(RBUF)                    ! TAKE ABSOLUTE VALUE

C--------------------------------------
C     THIS PART CHECKS TO SEE IF THE REAL HAS A WHOLE NUMBER PART TO IT.
C     IF SO I MAKE THE NUMBER THE SECOND VALUE AND PLACE A DECIMAL IN
C     THE THIRD VALUE.  THE FIRST CHARACTER IS RESERVED FOR SIGN (+/-)
C     IF THERE IS NO WHOLE NUMBER PORTION I PLACE A ZERO IN PLACE 2
C     AND DECIMAL IN PLACE 3.
C--------------------------------------


      IF(RBUF.GT.0.0)THEN                  ! IF > 1 MAKE IT THE
                                      ! FIRST CHARACTER.
       INUM - IFIX(RBUF)                   ! TAKE THE WHOLE NUMBER
       CNUM(J)(2:3) - CHAR(INUM + 48)//'.'   ! CONVERT TO CHARACTER
                                      !
       ELSE                             ! OR ELSE

```

```
         CNUM(J)(2:3) = '0.'              ! 1ST CHAR IS ZERO
      ENDIF

      RBUF = RBUF - FLOAT(INUM)           ! SUBTRACT THE WHOLE NUM

C-------------------------------------------------
C     THIS PART DEALS WITH THE NUMBERS LESS THAN ONE.
C     I LOOP THRU EACH PLACE VALUE BEGINNING WITH 1/10 MULTIPLY THE
C     NUMBER BY 10.0 SO I HAVE A WHOLE NUMBER VALUE 0 - 9 THEN
C     CONVERT IT TO INTEGER AND SAVE IT AS A CHARACTER IN THE
C     CNUM STRING.
C-------------------------------------------------

      M = 4                   ! CHARACTER PLACE COUNTER

      DO 30 K = 1,3           ! LOOP THRU PLACE VALUES

       RBUF = RBUF * 10.0     ! SHIFT NUMBER ONE PLACE LEFT.

      IF(RBUF.GE.1.0)THEN     ! CHECK FOR DATA IN PLACE

        INUM = IFIX(RBUF)     ! SAVE NUMBER AS INTEGER

        CNUM(J)(M:M) = CHAR(INUM + 48) ! CONVERT IT TO CHARACTER

        RBUF = RBUF - FLOAT(INUM) ! SUBTRACT NUMBER FROM TOTAL

      ELSE                    ! OR ELSE

       CNUM(J)(M:M) = '0'     ! PLACE IS A ZERO.

      ENDIF

      M = M + 1               ! INCREMENT PLACE COUNTER

30    CONTINUE

C-------------------------------------------------
C     THIS LOOKS AT THE 10,000th PLACE TO SEE IF 1/1000th SHOULD BE
C     INCREMENTED UP OR LEFT ALONE.  THIS IS NOT A ROUNDING UP
C     PROCEDURE.  BECAUSE OF THE WAY COMPUTERS STORE REALS THE LAST
C     NUMBER MUST BE EVALUATED TO CORRECT FOR THE ORIGINAL INPUT VALUE.
C-------------------------------------------------

      M = M - 1               ! PUT PLACE POINTER TO 1/1000
      RBUF = RBUF * 10.0      ! SHIFT NUMBER ONE PLACE LEFT.
      INUM = IFIX(RBUF)       ! SAVE NUMBER AS INTEGER


      IF(INUM.GE.5)THEN       ! CHECK FOR DATA IN PLACE VALUE
                      !       IF ABOVE 4 THEN
       CNUM(J)(M:M) = CHAR(ICHAR(CNUM(J)(M:M)) + 1) ! INCREMENT BY 1
                              ! CHECK THE CHARA
       IF(CNUM(J)(M:M).EQ.':')CNUM(J)(M:M) = '9'   ! VALUE. IF 9 IS
```

```
                                ! INCREMENTED ...KEEP IT 9
      ENDIF

C-----------------------------------------------

 50   CONTINUE


      RETURN
      END
```

AIV.21 Analog APSD Software Modules: SEE_ELE Source Code.

```
      SUBROUTINE SEE_ELE(ISWEEP,MATRIX,INU)

C------------------------------------------
C      THIS MOD LETS THE USER SELECT THE ELEMENTS OF CHOICE FOR EACH
C      SWEEP OF THE SAMPLE STAGE.  SINCE 9 BITS OF DATA ARE SENT TO
C      THE DRAWING ROUTINE EVERY TIME A SAMPLE IS GATHERED THE SCREEN
C      CAN BECOME SOMEWHAT CLUTTERED... HERE THE USER HAS THE OPTION
C      TO SELECT SPECIFIC MATRIX ELEMENTS OF CHOICE.  THE SELECTION
C      OF THE DIFFERENT MATRIX ELEMENTS ARE PROVIDED FOR EACH SWEEP
C      OF THE SAMPLE STAGE PRIOR TO THE BEGINNING OF THE EXPERIMENT.
C------------------------------------------


      CHARACTER TEXT1*35,TEXT2*35,TEXT3*35
      CHARACTER E,A*5,ANS*12,SEG*3

      INTEGER X,Y,MATRIX

      DIMENSION LDAT1(10),LDAT2(10),LDAT3(10),LDAT4(10)
      DIMENSION IA(9),IB(9),IC(9),ID(9)

      DATA IA / 1,2,4,5,6,8,13,14,16/
      DATA IB / 1,2,4,9,10,12,13,14,16/
      DATA IC / 1,3,4,5,7,8,13,15,16/
      DATA ID / 1,3,4,9,11,12,13,15,16/

      COMMON LDAT1,LDAT2,LDAT3,LDAT4

      E - CHAR(27)

C------------------------------------------

      WRITE(*,*)E//'LV0'          ! DISABLE DIALOG AREA
      WRITE(*,*)E//'LZ'           ! CLEAR THE DIALOG AREA

      IF(INU.EQ.1)THEN            ! INUSE BEING RESET FROM OUTSIDE
       INUSE - 0
       INU  - 0
      ENDIF
C------------------------------------------
C      THIS PART IS JUST AS ADD ON TO THE ROUTINE.  IT IS USED BY
C      THE ROUTINE DRAW_ELE TO SHOW THE USER WHICH ELEMENTS THAT
C      ARE BEING VIEWED.  THE FLAG MATRIX WHEN SET - 1 JUMPS THE
C      USER TO LINE 13 CHECKS THE SWEEP OF THE POLARIZERS, TURNS OFF THE
C      SEGMENTS THAT ARE NOT IN USE THEN THEN RETURNS TO THE CALLER
C------------------------------------------

      IF(MATRIX.EQ.1)THEN
       WRITE(*,*)E//'SV!1'        ! TURN ON ALL SEGMENTS
       GOTO 13
      ENDIF
```

```
C------------------------------------------

10   IF(INUSE.EQ.0)THEN          ! IF THIS IS THE FIRST TIME IN
       ISWEEP = 1               ! SET SWEEP FLAG TO 1 INDICATING
       INUSE = 1                ! SET INUSE FLAG TO ON POSITION
      ELSE                      ! A VERTICAL, VERTICAL POSITION
       ISWEEP = ISWEEP + 1       ! OF THE POLARIZERS.
       IEXIT = 0                ! INITIALIZE THE EXIT FLAG
       WRITE(*,*)E//'SV11'       ! TURN ALL SEGMENTS ON
       ISEG = 800               ! DEFINE TEXT SEGMENT AT 800
       CALL INTRPT(ISEG,SEG)     ! CONVERT INTEGER TO TEK CHARACTER
       WRITE(*,*)E//'SK'//SEG    ! DELETE THE SEGMENT
       WRITE(*,*)E//'KN0'        ! RENEBW THE VIEW
      ENDIF

      J = 1                     ! INITIALIZE ARRAY COUNTER
      IEXIT = 0                 ! INITIALIZE EXIT FLAG
      IALL  = 0                 ! INITIALIZE SELECT ALL FLAG

C------------------------------------------
C    THIS PART MAKES THE MATRIX ELEMENTS THAT WILL BE PROVIDING DATA
C    FOR A PARTICULAR POLARIZER COMBINATION VISIBLE TO THE USER.
C    THE FOUR POSSIBLE POSITIONS ARE AS FOLLOWS:
C
C    ISWEEP = 1                 ! VERTICAL, VERTICAL
C    ISWEEP = 2                 ! VERTICAL, 45 Degrees
C    ISWEEP = 3                 ! 45 Degrees, VERTICAL
C    ISWEEP = 4                 ! 45 Degrees, 45 Degrees.
C------------------------------------------

13   IF(ISWEEP.EQ.1)THEN

       J = 0                    ! INITIALIZE COUNTER

       DO 20 I = 1,9            ! LOOP THRU THE ARRAY ELEMENTS

15       J = J + 1              ! INCREMENT THE COUNTER

        IF(J.EQ.17)GOTO 20       ! KEEP UNDER 17

        IF(IA(I).EQ.J)GOTO 20    ! JUMP OUT ON A MATCH

        ISEG = J + 9            ! CALCULATE THE SEGMENT NUMBER

        CALL INTRPT(ISEG,SEG)    ! CONVERT IT TO A TEK CHARACTER

        IF(ISEG.LT.16)THEN       ! IF ONLY ONE TEK CHARACTER
         WRITE(*,*)E//'SV'//SEG(1:1)//'0' ! ERASE IT
        ELSE                    ! ELSE IF 2 CHARACTERS
         WRITE(*,*)E//'SV'//SEG(1:2)//'0' ! ERASE IT
        ENDIF

        GO TO 15                ! GO INCREMENT COUNTER
```

```
20      CONTINUE              ! END LOOP

     TEXT1 - 'POLARIZERS:  VERTICAL,VERTICAL'

C------------------------------------------
C     ISWEEP - 2    ( VERTICAL,  45 DEG  )
C------------------------------------------

      ELSEIF(ISWEEP.EQ.2)THEN

        J - 0                 ! INITIALIZE COUNTER

        DO 30 I - 1,9          ! LOOP THRU THE ARRAY ELEMENTS

25        J - J + 1            ! INCREMENT THE COUNTER

          IF(J.EQ.17)GOTO 30     ! KEEP UNDER 17

          IF(IB(I).EQ.J)GOTO 30  ! JUMP OUT ON A MATCH

          ISEG - J + 9          ! CALCULATE THE SEGMENT NUMBER

          CALL INTRPT(ISEG,SEG)   ! CONVERT IT TO A TEK CHARACTER

          IF(ISEG.LT.16)THEN     ! IF ONLY ONE TEK CHARACTER
            WRITE(*,*)E//'SV'//SEG(1:1)//'0'  ! ERASE IT
          ELSE                   ! ELSE IF 2 CHARACTERS
            WRITE(*,*)E//'SV'//SEG(1:2)//'0'  ! ERASE IT
          ENDIF

          GO TO 25               ! GO INCREMENT COUNTER

30      CONTINUE

     TEXT1 - 'POLARIZERS:  VERTICAL, 45 Degrees'

C------------------------------------------
C     ISWEEP - 3    ( 45 DEG, VERTICAL  )
C------------------------------------------


      ELSEIF(ISWEEP.EQ.3)THEN

        J - 0

        DO 40 I - 1,9          ! LOOP THRU THE ARRAY ELEMENTS

35        J - J + 1            ! INCREMENT THE COUNTER

          IF(J.EQ.17)GOTO 40     ! KEEP UNDER 17

          IF(IC(I).EQ.J)GOTO 40  ! JUMP OUT ON A MATCH

          ISEG - J + 9          ! CALCULATE THE SEGMENT NUMBER
```

```
          CALL INTRPT(ISEG,SEG)    ! CONVERT IT TO A TEK CHARACTER

          IF(ISEG.LT.16)THEN       ! IF ONLY ONE TEK CHARACTER
            WRITE(*,*)E//'SV'//SEG(1:1)//'0'  ! ERASE IT
          ELSE                     ! ELSE IF 2 CHARACTERS
            WRITE(*,*)E//'SV'//SEG(1:2)//'0'  ! ERASE IT
          ENDIF

          GO TO 35           ! GO INCREMENT COUNTER

  40      CONTINUE

      TEXT1 - 'POLARIZERS:  45 Degrees, VERTICAL'

C-----------------------------------------
C     ISWEEP - 4    ( 45 DEG,  45 DEG  )
C-----------------------------------------


       ELSEIF(ISWEEP.EQ.4)THEN

      J - 0

        DO 50 I - 1,9            ! LOOP THRU THE ARRAY ELEMENTS

  45      J - J + 1              ! INCREMENT THE COUNTER

          IF(J.EQ.17)GOTO 50     ! KEEP UNDER 17

          IF(ID(I).EQ.J)GOTO 50  ! JUMP OUT ON A MATCH

          ISEG - J + 9           ! CALCULATE THE SEGMENT NUMBER

          CALL INTRPT(ISEG,SEG)    ! CONVERT IT TO A TEK CHARACTER

          IF(ISEG.LT.16)THEN       ! IF ONLY ONE TEK CHARACTER
            WRITE(*,*)E//'SV'//SEG(1:1)//'0'  ! ERASE IT
          ELSE                     ! ELSE IF 2 CHARACTERS
            WRITE(*,*)E//'SV'//SEG(1:2)//'0'  ! ERASE IT
          ENDIF

          GO TO 45           ! GO INCREMENT COUNTER

  50      CONTINUE

      TEXT1 - 'POLARIZERS:  45 Degrees, 45 Degrees'

      ENDIF



      IF(MATRIX.EQ.1)GOTO 1000      ! RETURN TO CALLER JOB DONE

```

```
C-----------------------------------------
C     THIS PART ASSEMBLES THE CORRECT TEXT TO SHOW THE USER THE POSSIBLE
C     SELECTIONS FOR THE CURRENT ARRAY.
C-----------------------------------------

      TEXT2 - '1. SELECT ELEMENTS TO VIEW'
      TEXT3 - '3. SELECTION COMPLETE'

C-----------------------------------------
C     THIS PART DRAWS THE TEXT TO THE DIALOG AREA WINDOW.
C-----------------------------------------

      ISEG - 800                  ! BEGIN SEGMENT AT 800
      CALL INTRPT(ISEG,SEG)       ! CONVERT INTEGER TO TEK CHARACTER
      WRITE(*,*)E//'SE'//SEG      ! BEGIN THE SEGMENT
      WRITE(*,*)E//'MT1'          ! TEXT COLOR WHITE
      X - 120                     ! X POSITION OF THE ORIGIN
      Y - 350                     ! Y POSITION OF THE ORIGIN
      CALL HIY(X,Y,A)             ! CONVERT TO TEK CHARACTER
      WRITE(*,*)E//'LF'//A        ! SET THE ORIGIN
      WRITE(*,*)E//'LTB7'//TEXT1  ! WRITE 1ST LINE OF TEXT

      Y - Y - 200                 ! DECREMENT Y BY 100 SCREEN UNITS
      CALL HIY(X,Y,A)             ! INTEGER TO TEK CHARACTER
      WRITE(*,*)E//'LF'//A        ! SET THE ORIGIN
      WRITE(*,*)E//'LTB7'//TEXT2  ! WRITE THE TEXT.

      Y - Y - 100                 ! DECREMENT Y BY 100 SCREEN UNITS
      CALL HIY(X,Y,A)             ! INTEGER TO TEK CHARACTER
      WRITE(*,*)E//'LF'//A        ! SET THE ORIGIN
      WRITE(*,*)E//'LTB7'//TEXT3  ! WRITE THE TEXT.
      WRITE(*,*)E//'SC'           ! CLOSE THE SEGMENT


      J - 1                       ! INITIALIZE PLACE COUNTER

C-----------------------------------------
C     THIS PART CHECKS TO MAKE SURE THE USER HAS SELECTED A CORRECT
C     ARRAY ELEMENT, PLACES THE DATA IN THE CORRECT ARRAY ASSOCIATED
C     WITH THE SWEEP NUMBER
C-----------------------------------------

60    READ(*,'(A12)',ERR-60)ANS          ! READ THE SELECTION

      IF(ANS(1:1).EQ.'1'.OR.ANS(1:1).EQ.'3')GOTO 65
      GOTO 60

C-----------------------------------------
C     HERE THE USER HAS ELECTED TO END THE ELEMENT SELECTION.
C     IF THIS IS NOT THE 4TH SWEEP THEN WE GO SELECT AGAIN,
C     OTHERWISE THIS ROUTINE IS COMPLETE.
C-----------------------------------------

```

```
 65   IF(ANS(1:1).EQ.'3')THEN           ! END THE SELECTION
        IF(ISWEEP.EQ.4)GOTO 200          ! IF LAST SWEEP THEN EXIT
        GOTO 10                          ! GO TO NEXT SWEEP
      ENDIF

C-----------------------------------------
C     THIS MOD FILLS THE CORRECT ARRAY ( LDAT ) WITH THE CHOSEN
C     ELEMENTS. J = THE LDAT# ARRAY PLACEMENT. THE LAST ARRAY ELEMENT
C     CONTAINS THE NUMBER OF USER ELEMENTS SELECTED.
C-----------------------------------------


      SEG = ANS(7:9)                    ! THIS IS THE SEGMENT SELECTED

      CALL DECODE(SEG,ISEG)             ! CONVERT IT TO AN INTEGER

      IF(ISEG.EQ.40)THEN                ! IF ITS THE " EXIT " BUTTON
       IEXIT = 1                        ! SET THE EXIT FLAG = 1
       GOTO 70                          ! PLACE THIS INFO IN THE ARRAY
      ENDIF                             ! NO ELEMENTS ARE TO BE VIEWED

      IF(ISEG.EQ.41)THEN                ! THIS IS THE " ALL " BUTTON
       IALL = 1                         ! SET THE SHOW ALL ELEMENTS FLAG
       GOTO 70
      ENDIF

      IF(ISEG.LT.10.OR.ISEG.GT.25)GOTO 60 ! CHECK FOR CORRECT SEGMENTS

 70   IPICK = ISEG - 9                  ! CONVERT THEM TO ELEMENT NUMBERS

      WRITE(*,*)'IPICK =',IPICK
      WRITE(*,*)'ISEG = ',ISEG

C-----------------------------------------
C     THIS PART PLACES THE THE SELECTED A/D CHANNEL NUMBERS IN THE
C     CORRECT LDAT FILE. THE LAST ELEMENT IS THE COUNT OF DATA ELEMENTS
C     THE USER SELECTED. AS THE USER SELECTS THE ELEMENTS I ERASE THE
C     GRAPHIC SEGMENT ASSOCIATED WITH IT.
C-----------------------------------------

      CALL INTRPT(ISEG,SEG)             ! CONVERT SEGMENT INTEGER TO TEK

      IF(ISEG.EQ.40.OR.ISEG.EQ.41)GOTO 80 ! EXIT,ALL -DO NOT ERASE THEM

      IF(ISEG.LT.16)THEN                ! IF ONLY 1 CHARACTER THEN
      WRITE(*,*)'E/'SV'//SEG(1:1)//'0'  ! ERASE IT.
      ELSE                              ! OR ELSE IF IT IS 2 CHARACTERS
      WRITE(*,*)'E/'SV'//SEG(1:2)//'0'  ! ERASE IT ALSO.
      ENDIF

 80   IF(ISWEEP.EQ.1)THEN               ! IF SWEEP #1

      IF(IEXIT.EQ.1)THEN                ! USER WANTED TO EXIT
       LDAT1(10) = 0                    ! PLACE A O IN NUMBER OF
```

```
       GO TO 10               ! ELEMENTS TO VIEW
      ENDIF                  ! GO TO NEXT SWEEP INPUTS

      IF(IALL.EQ.1)THEN        ! USER WANTS TO SEE ALL 9
       LDAT1(10) - 9           ! ELEMENTS.  PLACE THE NUMBER
      DO 90 I - 1 , 9          ! IN PLACE 10,  PUT THE 9
       LDAT1(I) - I            ! CHANNELS IN ORDER IN THE
90    CONTINUE                 ! ARRAY.
       IALL - 0                ! INITIALIZE THE ALL ELEMENTS
       GOTO 10                 ! FLAG AND GO TO THE NEXT
      ENDIF                    ! SWEEP

      DO 100 I - 1,9           ! LOOP THRU ARRAY ELEMENTS
       IF(IA(I).EQ.IPICK)LDAT1(J) - I ! PUT A/D CHANNEL NUMBER IN
100   CONTINUE                 ! THE ARRAY

       LDAT1(10) - J           ! PUT CHANNEL COUNT IN LAST HOLE
       J - J + 1               ! INCREMENT ARRAY PLACE COUNTER
       IF(J.EQ.10)GOTO 10      ! CHECK TO SEE IF ITS FULL
       GOTO 60                 ! GO BACK FOR MORE DATA

C-----------------------------------

      ELSEIF(ISWEEP.EQ.2)THEN

      IF(IEXIT.EQ.1)THEN       ! USER WANTED TO EXIT
       LDAT2(10) - 0           ! PLACE A O IN NUMBER OF
       GO TO 10                ! ELEMENTS TO VIEW
      ENDIF                    ! GO TO NEXT SWEEP INPUTS

      IF(IALL.EQ.1)THEN        ! USER WANTS TO SEE ALL 9
       LDAT2(10) - 9           ! ELEMENTS.  PLACE THE NUMBER
       DO 110 I - 1 , 9        ! IN PLACE 10,  PUT THE 9
        LDAT2(I) - I           ! CHANNELS IN ORDER IN THE
110     CONTINUE               ! ARRAY.
       IALL - 0                ! INITIALIZE THE ALL ELEMENTS
       GOTO 10                 ! FLAG AND GO TO THE NEXT
      ENDIF                    ! SWEEP

      DO 120 I - 1,9           ! LOOP THRU ARRAY ELEMENTS
       IF(IB(I).EQ.IPICK)LDAT2(J) - I ! PUT A/D CHANNEL NUMBER IN
120   CONTINUE                 ! THE ARRAY

      LDAT2(10) - J
      J - J + 1
      IF(J.EQ.10)GOTO 10
      GOTO 60

C-----------------------------------

      ELSEIF(ISWEEP.EQ.3)THEN

      IF(IEXIT.EQ.1)THEN       ! USER WANTED TO EXIT
       LDAT3(10) - 0           ! PLACE A O IN NUMBER OF
```

```
          GO TO 10                  ! ELEMENTS TO VIEW
        ENDIF                      ! GO TO NEXT SWEEP INPUTS

        IF(IALL.EQ.1)THEN            ! USER WANTS TO SEE ALL 9
         LDAT3(10) = 9              ! ELEMENTS.  PLACE THE NUMBER
         DO 130 I = 1 , 9           ! IN PLACE 10,  PUT THE 9
           LDAT3(I) = I             ! CHANNELS IN ORDER IN THE
 130     CONTINUE                    ! ARRAY.
         IALL = 0                  ! INITIALIZE THE ALL ELEMENTS
         GOTO 10                    ! FLAG AND GO TO THE NEXT
        ENDIF                      ! SWEEP

        DO 140 I = 1,9              ! LOOP THRU ARRAY ELEMENTS
         IF(IC(I).EQ.IPICK)LDAT3(I) = I ! PUT A/D CHANNEL NUMBER IN
 140    CONTINUE                      ! THE ARRAY

        LDAT3(10) = J
        J = J + 1
        IF(J.EQ.10)GOTO 10

        GOTO 60

C------------------------------------

       ELSEIF(ISWEEP.EQ.4)THEN

        IF(IEXIT.EQ.1)THEN           ! USER WANTED TO EXIT
         LDAT4(10) = 0              ! PLACE A 0 IN NUMBER OF
         GO TO 200                  ! ELEMENTS TO VIEW
        ENDIF                      ! GO TO NEXT SWEEP INPUTS


        IF(IALL.EQ.1)THEN            ! USER WANTS TO SEE ALL 9
         LDAT4(10) = 9              ! ELEMENTS.  PLACE THE NUMBER
         DO 150 I = 1 , 9           ! IN PLACE 10,  PUT THE 9
           LDAT4(I) = I             ! CHANNELS IN ORDER IN THE
 150     CONTINUE                    ! ARRAY.
         IALL = 0                  ! INITIALIZE THE ALL ELEMENTS
         GOTO 200                   ! FLAG AND GO TO THE NEXT
        ENDIF                      ! SWEEP

        DO 160 I = 1,9              ! LOOP THRU ARRAY ELEMENTS
         IF(ID(I).EQ.IPICK)LDAT4(I) = I ! PUT A/D CHANNEL NUMBER IN
 160    CONTINUE                      ! THE ARRAY

        LDAT4(10) = J
        J = J + 1
        IF(J.EQ.10)GOTO 200
        GOTO 60
       ENDIF

C------------------------------------
C      THIS PART SORTS EACH ARRAY IS ASCENDING ORDER IF THE LAST
C      ELEMENT IS NOT A ZERO.
```

```
      C-------------------------------------

  200 IF(LDAT1(10).GT.0)THEN
        ISIZE - LDAT1(10)
        CALL BUBBLE_UP(LDAT1,ISIZE)
      ENDIF

      IF(LDAT2(10).GT.0)THEN
        ISIZE - LDAT2(10)
        CALL BUBBLE_UP(LDAT2,ISIZE)
      ENDIF

      IF(LDAT3(10).GT.0)THEN
        ISIZE - LDAT3(10)
        CALL BUBBLE_UP(LDAT3,ISIZE)
      ENDIF

      IF(LDAT4(10).GT.0)THEN
        ISIZE - LDAT4(10)
        CALL BUBBLE_UP(LDAT4,ISIZE)
      ENDIF

 1000 MATRIX - 0
      RETURN
      END
```

AIV.22 Analog APSD Software Modules: STAGE_POSITION Source Code.

```
      SUBROUTINE STAGE_POSITION(IUT)

C--------------------------------------
C     THIS ROUTINE ALLOWS THE USER TO SET THE SPEED ,DIRECTION AND
C     DISTANCE OF THE STEPPER MOTORS FOR EACH AXIS ON THE CONTROLLER
C     FOR EACH LASER IN THE MATRIX EXPERIMENT.
C--------------------------------------

      CHARACTER*10 VEL,ACC,DIST,DUMMY,FILENM*10

      CHARACTER PORT*10,PORT1*10,LASER*1,YES*1,AXIS*1,MSG*12
      CHARACTER MSG1*80
      LOGICAL T

      DIMENSION VEL(4),ACC(4),DIST(4),DUMMY(4)


      IUT = 99
      CALL GETPORT(PORT,I)
      OPEN(IUT,FILE=PORT,STATUS='NEW')

C--------------------------------------

 5    FORMAT(12(A10))
 10   FORMAT(A80)
 20   FORMAT(80('-'))
 30   FORMAT(80(' '))
 31   FORMAT('      THE FOLLOWING SETTINGS WILL DEFINE STAGE
     .POSITIONS FOR EACH LASER')
 32   FORMAT(10X,'1.  POSITION STAGES FOR SAMPLE DATA COLLECTION')
 34   FORMAT(10X,'2.  POSITION STAGES FOR THE SPECTRUM ANALYSER')
 36   FORMAT(10X,'3.  EXIT')


 40   FORMAT(10X,'ENTER LASER NUMBER.. ( 1 - 4 ):  ',$)
 50   FORMAT(10x,'LASER #',I2,15X,'AXIS #',I2,15X,'CONTROLLER #1')
 60   FORMAT(10X,'FOR AXIS ',I2,20X,'LASER #',I2)
 70   FORMAT(10X,'1.  ENTER THE VELOCITY:          ',$)
 80   FORMAT(//,10X,'2.  ENTER THE ACCELERATION:      ',$)
 90   FORMAT(//,10X,'3.  ENTER THE DISTANCE ( + OR - ): ',$)
 100  FORMAT(10X,I2,'          VELOCITY: = ',A10)
 110  FORMAT(10X,I2,'      ACCELERATION: = ',A10)
 120  FORMAT(10X,I2,'  DISTANCE IN STEPS: = ',A10)
 130  FORMAT(24X,'ENTER AXIS NUMBER FOR CORRECTION:')
 140  FORMAT(24X,'     ENTER RETURN TO CONTINUE')
 150  FORMAT(20X,'  < A > TO APPLY   OR  < E >  TO EXIT')
 160  FORMAT(30X,$)
 170  FORMAT(24X,'    ENTER RETURN FOR NEXT LASER')
 180  FORMAT(10X,'  WHICH SERIAL PORT IS ATTACHED TO THE CONTROLLER: ',
     . $)
 190  FORMAT(10X,'ENTER AXIS NUMBER YOU WANT TO TRY: ',$)
```

```
    200  FORMAT(20X,'PRESS ENTER WHEN YOUR READY OR < 99 > TO EXIT')
210  FORMAT(10X,'1.  THE MOTION IS CORRECT ')
220  FORMAT(10X,'2.  CHANGE THE DATA FOR THIS AXIS')
230  FORMAT(10X,'3.  EXIT')
240  FORMAT(23X,'DOES THE LASER NEED OPTICS MOVED IN')
250  FORMAT(23X,'ORDER TO BE CALIBRATED < Y or N >')
260  FORMAT(10X,'THE PRESENT PORT DEFINED FOR OUTPUT IS:',A10)
270  FORMAT(29X,'PRESS RETURN IF CORRECT')
280  FORMAT(30X,'PRESS < 1 > TO CHANGE')

C------------------------------------------------
C    HERE THE USER IS ASKED TO SELECT THE TYPE OF STAGE SETTING FOR
C    EACH LASER THAT IS TO BE SAVED.
C    EACH LASER REQUIRES THAT THE OPTICS CHANNEL LIGHT TO THE
C    MODULATORS FOR CALIBRATION AND TO THE SAMPLE FOR DATA
C    COLLECTION.  THESE SETTINGS WILL OFTEN BE DIFFERENT.  TO MAKE
C    THINGS EASY I AM SAVING THE TWO TYPES OF SETTINGS IN DIFFERENT
C    FILES.  SAMPLE.DAT FOR LIGHT TO THE SAMPLE.  MOD.DAT FOR LIGHT
C    TO THE MODULATORS..
C------------------------------------------------


290   WRITE(IUT,30)
     WRITE(IUT,31)
     WRITE(IUT,30)
     WRITE(IUT,32)
     WRITE(IUT,34)
     WRITE(IUT,36)
     WRITE(IUT,30)
     READ(IUT,'(A)',ERR=300)LASER    ! READ TYPE OF STAGE SETTINGS

     IF(LASER.EQ.'1')FILENM = 'SAMP'
     IF(LASER.EQ.'2')FILENM = 'SPECTRUM'
     IF(LASER.EQ.'3')GOTO 1000      ! USER WANTS TO EXIT

300   WRITE(IUT,30)
     WRITE(IUT,30)
     WRITE(IUT,140)
     WRITE(IUT,40)        ! ENTER LASER NUMBER

     READ(IUT,'(A)',ERR=300)LASER    ! MAXIMUM OF 4 LASERS NOW..

     IF(LASER.EQ.' ')GOTO 1000      ! USER WANTS TO EXIT

     IF(LASER.GT.CHAR(48).AND.LASER.LT.CHAR(53))GOTO 310
     GOTO 300

310   READ(LASER,'(BN,I4)',ERR=300)IL   ! LASER INTEGER VALUE

     IFIX = 0                ! INITIALIZE DATA TYPE FLAGS

C------------------------------------------------
C    THIS CHECKS TO SEE IF THE FILE EXISTS, SETS A FLAG THEN OPENS IT
C------------------------------------------------
```

```
      IF(IUSED.EQ.0)THEN
       IUSED - 1              ! SET FLAG NOT TO COME BACK HERE
       INQUIRE(FILE-FILENM,EXIST - T) ! ASK IF FILE EXISTS

      OPEN(2,FILE - FILENM,ACCESS-'DIRECT',FORM-'FORMATTED',
     .RECL-120,STATUS - 'UNKNOWN',SHARED,ERR -1000)

      READ(2,REC - 9,FMT - '(A10)',ERR - 315)PORT1
      ENDIF

C------------------------------------------------
C     HERE I WRITE BACK TO THE USER THE LAST ENTRY FOR THE AXIS FOR
C     THIS LASER. IF THE FILE EXISTS I READ IT FROM THERE.
C------------------------------------------------

 315   IF(IJUMP.EQ.1)GOTO 370 ! DONT READ

      IF(.NOT.T)GOTO 370      ! DONT READ IF NO FILE

      READ(2,REC - IL,FMT - 5,ERR - 370)VEL(1),ACC(1),DIST(1),
     .DUMMY(1),VEL(2),ACC(2),DIST(2),DUMMY(2),VEL(3),ACC(3),
     .DIST(3),DUMMY(3)

C      IF(VEL(1).EQ.' '.OR.ACC(1).EQ.' '.OR.DIST(1).EQ.' '.OR.
C     .ICHAR(VEL(1)).EQ.0.OR.ICHAR(ACC(1)).EQ.0)THEN
C        IJUMP - 1
C        GOTO 370
C      ENDIF

      GOTO 370

 370   K - 0
      IJUMP - 0            ! RESET THE JUMP FLAG

      DO 390 I - 1,3       ! LOOP THRU THE INPUTS AND DISPLAY
                           ! THE FILE SETTINGS FOR THIS LASER
       WRITE(IUT,60)I,IL
       WRITE(IUT,30)
       K - K + 1
       WRITE(IUT,100)K,VEL(I)      ! VELOCITY
       K - K + 1
       WRITE(IUT,110)K,ACC(I)      ! ACCELERATION
       K - K + 1
       WRITE(IUT,120)K,DIST(I)     ! DISTANCE OF TRAVEL
       WRITE(IUT,20)

 390   CONTINUE


      IF (KUM_BACK.EQ.1)GOTO 510

C------------------------------------------------
C     HERE THE USER IS ASKED IF THE DATA IS CORRECT. SHOULD THERE BE
```

```
C     CHANGES, DOES THE USER WISH TO APPLY THE DATA TO THE STEPPER
C     MOTOR CONTROLLERS OR JUST EXIT.
C------------------------------------------

400   WRITE(IUT,30)
      WRITE(IUT,130)
      WRITE(IUT,140)
      WRITE(IUT,150)
      WRITE(IUT,160)

      READ(IUT,'(A1)',ERR - 400)YES

      IF(YES.EQ.' ')THEN
                        ! IF THIS IS CALIBRATION DATA THEN RECORD
        IREC - IL       ! OR ELSE ITS JUST THE LASER NUMBER


      IF(IFIX.EQ.1)THEN     ! SAVE THE NEW DATA IF FLAG IS SET - 1

      WRITE(2,REC - IREC,FMT - 5)VEL(1),ACC(1),DIST(1),
     .DUMMY(1),VEL(2),ACC(2),DIST(2),DUMMY(2),VEL(3),ACC(3),
     .DIST(3),DUMMY(3)

      IFIX - 0              ! RESET THE SAVE DATA FLAG
      ENDIF
        GOTO 300            ! JUMP TO TOP OF ROUTINE

       ENDIF


      IF(YES.EQ.'A'.OR.YES.EQ.'a')GOTO 500
      IF(YES.EQ.'E'.OR.YES.EQ.'e')GOTO 1000

C------------------------------------------
C     HERE THE USER HAS SELECTED ONE OF THE LASER TO GIVE NEW
C     SETTINGS FOR.  I LOOP THRU THE INPUTS THEN SHOW THEM BACK
C------------------------------------------

      IF(YES.EQ.'1'.OR.YES.EQ.'2'.OR.YES.EQ.'3')THEN

       READ(YES,'(BN,I4)',ERR-1000)I

       IFIX - 1             ! FLAG THAT A CHACE IS TO BE MADE


      WRITE(IUT,30)
      WRITE(IUT,30)
      WRITE(IUT,50)IL,I
      WRITE(IUT,30)
      WRITE(IUT,70)
      READ(IUT,'(A10)',ERR - 370)VEL(I)
      WRITE(IUT,30)
      WRITE(IUT,80)
      READ(IUT,'(A10)',ERR - 370)ACC(I)
```

```
        WRITE(IUT,30)
      WRITE(IUT,90)
      READ(IUT,'(A10)',ERR = 370)DIST(I)
      WRITE(IUT,30)
      WRITE(IUT,20)

      GOTO 370          ! DISPLAY INPUTS BACK TO THE USER
       ENDIF

       GOTO 400         ! INPUT WAS BAD DO AGAIN

C-----------------------------------------
C    THIS PART ALLOWS THE USER TO APPLY THE INPUT BEFORE IT IS SAVED
C    TO FILE.  HERE I GET THE SERIAL PORT NUMBER FROM THE USER
C    THEN INSTRUCT THE USER TO PRESS ENTER TO HAVE THE DATA SENT
C    TO THE SPECIFIC PORT.
C-----------------------------------------

500    WRITE(IUT,30)

      IF(INUSE.EQ.0)THEN
       INUSE = 1
       IF(PORT1.EQ.' '.OR.ICHAR(PORT1(1:1)).EQ.0)THEN
505     WRITE(IUT,180)
       READ(IUT,'(A10)',ERR=500)PORT1
      ELSE
       WRITE(IUT,30)
       WRITE(IUT,260)PORT1
       WRITE(IUT,270)
       WRITE(IUT,280)
       WRITE(IUT,30)

       READ(IUT,'(A1)',ERR = 400)YES

       IF(YES.EQ.'1')GOTO 505
       IF(YES.NE.' ')GOTO 505
      ENDIF

      WRITE(*,*)' PORT1 = ',PORT1

       OPEN(3,FILE=PORT1,CARRIAGECONTROL = 'NONE',STATUS='NEW')
       WRITE(3,'(A9)')'E MN ST0 '        ! WAKE UP CONTROLLER

      WRITE(2,REC = 9,FMT = '(A10)',ERR = 510)PORT1

      ENDIF


C-----------------------------------------
C    HERE THE AXIS NUMBER OR STAGE IS REQUESTED FROM THE USER
C    AN INPUT OF NOTHING JUST REDISPLAYS THE CURRENT SETTINGS
C-----------------------------------------

510    WRITE(IUT,30)
```

```
      WRITE(IUT,140)
      WRITE(IUT,190)

      READ(IUT,'(A1)',ERR-510)AXIS

      IF(AXIS.EQ.'1'.OR.AXIS.EQ.'2'.OR.
     .AXIS.EQ.' '.OR.AXIS.EQ.'3')GOTO 520
      GOTO 510

520   KUM_BACK - 0

C--------------------------------------------------------
C     HERE THE STEPPER MOTORS ARE MOVED FROM THEIR HOME POSITION THE
C     VELOCITY, ACCELERATION AND DISTANCE SPECIFIED BY THE ABOVE INPUT.
C--------------------------------------------------------

      READ(AXIS,'(BN,I4)')IX        ! CONVERT CHARACTER TO INTEGER

C--------------------------------------------------------

      IF(IX.EQ.0)GOTO 370           ! IF THE USER PRESSED RETURN W/OUT
                                    ! AN INPUT I GO REDISPLAY THE
                                    ! CURRENT SETTINGS
C--------------------------------------------------------
C     HERE I MAKE SURE THAT THERE IS DATA PRIOR TO WRITING TO THE PORT
C     IF NOT I NOTIFY THE USER AND TRY AND GO GET SOME...
C--------------------------------------------------------

      IF(VEL(IX).EQ.' '.OR.DIST(IX).EQ.' ')THEN
        WRITE(IUT,*)' SORRY THERE IS NO DATA DEFINED FOR THIS AXIS...'
        GOTO 370
      ENDIF


      WRITE(3,'(A5)')AXIS//'ST1 '
      WRITE(*,*)'TURN ON THE MOTOR  ',AXIS//'ST1 '

C--------------------------------------------------------
C     THIS IS A DUMMY THAT LETS THE USER GET HARDWARE READY PRIOR TO
C     MOVING THE ACTUAL STAGES.
C--------------------------------------------------------

450   FORMAT(A11)
451   FORMAT(A12)
452   FORMAT(A13)
453   FORMAT(A14)
454   FORMAT(A15)
455   FORMAT(A16)
456   FORMAT(A17)
457   FORMAT(A18)
458   FORMAT(A19)
459   FORMAT(A20)
460   FORMAT(A21)
461   FORMAT(A22)
```

```
462   FORMAT(A23)



      WRITE(IUT,30)
      WRITE(IUT,200)

      READ(IUT,'(A1)')YES

      L - 0

      IF(YES.EQ.' ')THEN

       MSG - AXIS//'V'//VEL(IX)

      DO 560 I - 1,3

530     DO 540 J - 1,12
        IF(MSG(J:J).EQ.' '.OR.MSG(J:J).EQ.'+')GOTO 540
         L - L + 1
         MSG1(L:L) - MSG(J:J)
540     CONTINUE

        L - L + 1
        MSG1(L:L) - ' '
      IF(I.EQ.1)MSG - AXIS//'A'//ACC(IX)
      IF(I.EQ.2)MSG - AXIS//'D'//DIST(IX)

560   CONTINUE

      L - L + 1
      MSG1(L:L + 1) - 'G '
      L - L + 1

      WRITE(*,*)' MSG1 - ',MSG1(1:L),'  I - ',I


      J - L - 10
      WRITE(*,*)' J - ',J
      PAUSE
      GOTO(570,580,590,600,610,620,630,640,650,660,670,680,690)J


570   WRITE(3,450)MSG1(1:L)
      GOTO 700
580   WRITE(3,451)MSG1(1:L)
      GOTO 700
590   WRITE(3,452)MSG1(1:L)
      GOTO 700
600   WRITE(3,453)MSG1(1:L)
      GOTO 700
610   WRITE(3,454)MSG1(1:L)
      GOTO 700
620   WRITE(3,455)MSG1(1:L)
```

```
        GOTO 700
630   WRITE(3,456)MSG1(1:L)
      GOTO 700
640   WRITE(3,457)MSG1(1:L)
      GOTO 700
650   WRITE(3,458)MSG1(1:L)
      GOTO 700
660   WRITE(3,459)MSG1(1:L)
      GOTO 700
670   WRITE(3,460)MSG1(1:L)
      GOTO 700
680   WRITE(3,461)MSG1(1:L)
      GOTO 700
690   WRITE(3,462)MSG1(1:L)

      ENDIF

C-----------------------------------
C     HERE THE USER CAN CHECK TO SEE IF THE MOVEMENT WAS CORRECT.
C     IF NOT THE DATA CAN BE CHANGED.
C-----------------------------------

700     WRITE(IUT,30)
      WRITE(IUT,210)          ! STAGE MOVEMENT CORRECT
      WRITE(IUT,220)          ! CHANGE THE MOTION
      WRITE(IUT,230)          ! EXIT
      WRITE(IUT,30)

      READ(IUT,'(A1)',ERR-700)YES

C-----------------------------------
C     HERE I SEND THE STAGE BACK TO ITS HOME POSITION  < H > MEANS
C     REVERSE DIRECTION AND < G > TELLS IT TO GO THE SAME DISTANCE
C     AND VELOCITY.
C-----------------------------------

      WRITE(3, '(A4)')'H G '
      WRITE(3,'(A5)')AXIS//'STO '

C-----------------------------------

      IF(YES.EQ.'1')THEN     ! THE DATA IS CORRECT WRITE IT TO FILE

         IREC - IL       ! OR ELSE ITS JUST THE LASER NUMBER

      WRITE(2,REC - IREC,FMT - 5)VEL(1),ACC(1),DIST(1),
     .DUMMY(1),VEL(2),ACC(2),DIST(2),DUMMY(2),VEL(3),ACC(3),
     .DIST(3),DUMMY(3)

      KUM_BACK - 1
      GOTO 370          ! GOTO GET MORE DATA FOR THESE AXIS

C-----------------------------------
C     HERE THE USER DOES NOT LIKE THE STAGE MOVEMENT AND WOULD LIKE TO
```

```
 C     MAKE AN ADJUSTMENT.  I SEND THE USER BACK UP IN THE ROUTINE WHERE
C     THE CORRECTIONS ARE MADE.
C--------------------------------------------

      ELSEIF(YES.EQ.'2')THEN

       I = IX           ! PLACE THE AXIS NUMBER TO CORRECT IN I

       GOTO 370          ! GOTO BACK THRU THE ROUTINE.

C--------------------------------------------
C     HERE THE USER WANTS TO EXIT.
C--------------------------------------------

      ELSEIF(YES.EQ.'3')THEN
       GOTO 300          ! GO TO THE EXIT CLOSE THE FILES...

      ELSE

       GOTO 700          ! BAD INPUT GO BACK TO THE READ...

      ENDIF

1000   WRITE(3,'(A2)')'F '      ! TAKE CONTROLLER OFF LINE.
       CLOSE(3)
       CLOSE(2)

      RETURN
      END
```

AIV.23 Analog APSD Software Modules: TEK3 Source Code.

```
      SUBROUTINE TEK3(TYPE,SAMP,AGENT,CONC,LAMDA)

C-----------------------------------------------------
C     THIS ROUTINE WILL TAKE REAL TIME MULLER MATRIX DATA AND PRESENT IT
C     IN GRAPHIC FORM ON A TEKTRONIX COMPUTER.
C-----------------------------------------------------

      CHARACTER E,PORT*5,A*5,A1*5,SEGCNT*3,TEXT*30,TIC1*4,COL*2,LET*1
      CHARACTER SAMP*20,AGENT*20,CONC*15,LAMDA*15,TIC*3,NUM*2,CP*2
      CHARACTER TEXT1*6,SCNT*1,ANS*12,SEG*3,POINT*6,KEY,FILENM*20

      REAL RARM,REND,RINCR,RDAT

      INTEGER X,Y,MATRIX,TYPE

      DIMENSION TIC(18),TIC1(21),NUM(16),COL(16),CP(16),TEXT1(2)
      DIMENSION LDAT1(18),LDAT2(18),LDAT3(18),LDAT4(1)



      DATA TIC /'10','20','30','40','50','60','70','80','90','100','110',
     .'120','130','140','150','160','170','180'/

      DATA TIC1 /'+1.0','+ .9','+ .8','+ .7','+ .6','+ .5','+ .4',
     .'+ .3','+ .2','+ .1',' 0.0','- .1','- .2','- .3','- .4','- .5',
     .'- .6','- .7','- .8','- .9','-1.0'/

      DATA NUM/'11','12','13','14','21','22','23','24',
     .'31','32','33','34','41','42','43','44'/
      DATA COL /'2','1','4','1','2','1','2','1','4','4','1','1','1',
     .'1','1','2'/

      DATA CP /'!','"','#','$','%','&','Q','(',')','*','+',',','-',
     .'.','/','?'/

      DATA TEXT1 /' EXIT','  ALL'/


      E = CHAR(27)
      IFILECNT = 0
      IFLG = 0
      ITOGH = 0
      ITOGV = 0

C-----------------------------------------------------
C              ***** TEST DATA *****
C     SAMP = 'GREEN PAINT'
C     AGENT= 'GB'
C     CONC = '.003 Mg/m3'
C     LAMDA= '.094 um'
C     WRITE(*,*)E//'%!0'
```

```
C------------------------------------

      WRITE(*,*)E//'LV0'
      WRITE(*,*)E//'LZ'

1     FORMAT (A12)





C------------------------------------
C     THIS PART DRAWS THE GRAPH BOX.  COLOR RED.
C------------------------------------

      ISEG - 1

      IF(TYPE.EQ.1)GOTO 5          ! GRAPH ALREADY EXISTS

      CALL INTRPT(ISEG,SEGCNT)

      WRITE(*,*)E//'SE'//SEGCNT
      WRITE(*,*)E//'ML2'
      X - 145
      Y - 3100
      CALL HIY(X,Y,A)
      WRITE(*,*)E//'LP'//A
      X - X + 3800
      CALL HIY(X,Y,A)
      WRITE(*,*)E//'LG'//A
      Y - Y - 2400
      CALL HIY(X,Y,A)
      WRITE(*,*)E//'LG'//A
      X - X - 3800
      CALL HIY(X,Y,A)
      WRITE(*,*)E//'LG'//A
      Y - Y + 2400
      CALL HIY(X,Y,A)
      WRITE(*,*)E//'LG'//A
      WRITE(*,*)E//'SC'

C------------------------------------
C     THIS DRAWS THE TEXT IN THE BOX.  SAMPLE,AGENT, CONC, WAVELENGTH
C     START,END AND INCREMENT POINTS
C     THIS FIRST ONE WRITES THE SAMPLE
C------------------------------------

5     ISEG - ISEG + 1

      CALL INTRPT(ISEG,SEGCNT)
      WRITE(*,*)E//'SK'//SEGCNT
      WRITE(*,*)E//'SE'//SEGCNT

      WRITE(*,*)E//'MT7'
```

```
       X - 600
      Y - 2950
      CALL HTY(X,Y,A)
      WRITE(*,*)E//'LP'//A
      TEXT - 'SAMPLE: '//SAMP
      WRITE(*,*)E//'LTA;'//TEXT

C------------------------------------------------
C     THIS WRITES THE AGENT TYPE
C------------------------------------------------

      X - 2300
      CALL HTY(X,Y,A)
      WRITE(*,*)E//'LP'//A
      TEXT - 'AGENT: '//AGENT
      WRITE(*,*)E//'LTA;'//TEXT

C------------------------------------------------
C     THIS WRITE THE AGENT CONCENTRATION
C------------------------------------------------

      X - 600
      Y - 750
      CALL HTY(X,Y,A)
      WRITE(*,*)E//'LP'//A
      TEXT - 'CONC: '//CONC
      WRITE(*,*)E//'LTA;'//TEXT

C------------------------------------------------
C     THIS WRITES THE LASER WAVELENGTH
C------------------------------------------------

      X - 2300
      Y - 750
      CALL HTY(X,Y,A)
      WRITE(*,*)E//'LP'//A
      TEXT - 'WAVELENGTH: '//LAMDA
      WRITE(*,*)E//'LTB:'//TEXT
      WRITE(*,*)E//'SC'


      IF(TYPE.EQ.1)GOTO 1010

C------------------------------------------------
C     THIS WRITES THE BAR GRAPH IN THE CENTER .
C------------------------------------------------


       ISEG - 1000

      CALL INTRPT(ISEG,SEGCNT)
      WRITE(*,*)E//'SK'//SEGCNT
      WRITE(*,*)E//'SE'//SEGCNT

```

```
        WRITE(*,*)E//'ML15'
       X - 245
       Y - 1900

       CALL HIY(X,Y,A)
       A1 - A
       WRITE(*,*)E//'LP'//A
       X - 3845
       CALL HIY(X,Y,A)
       WRITE(*,*)E//'LG'//A

C------------------------------------------

C      NOW THE TICS ARE ADDED TO THE LINE
C------------------------------------------

       X - 245
       Y - Y - 25
       CALL HIY(X,Y,A)
       WRITE(*,*)E//'LP'//A
       Y-Y+50
       CALL HIY(X,Y,A)
       WRITE(*,*)E//'LG'//A          ! THE FIRST TIC 50 PIXELS

       X - X - 5
       Y - Y - 90
       CALL HIY(X,Y,A)
       WRITE(*,*)E//'LP'//A
       WRITE(*,*)E//'MC:A45'         ! CHANGE SIZE OF TEXT
       WRITE(*,*)E//'LT:0'           ! PLACE 0 UNDER FIRST TIC

       X - 245
       Y - Y + 77

C------------------------------------------

       DO 40 I-1,18

       DO 20 II-1,9

       X - X + 20
       Y - Y - 25
       CALL HIY(X,Y,A)
       WRITE(*,*)E//'LP'//A
       Y - Y + 25
       CALL HIY(X,Y,A)
       WRITE(*,*)E//'LG'//A

   20  CONTINUE

C------------------------------------------
C      THIS PART PLACE THE LARGER TIC MARKS AND LABELS THEM
C------------------------------------------
```

```
      X = X + 20
      Y = Y - 37
      CALL HIY(X,Y,A)
      WRITE(*,*)E//'LP'//A
      Y = Y + 50
      CALL HIY(X,Y,A)
      WRITE(*,*)E//'LG'//A          ! THE TIC MARKS AT 10 DEG INTERVALS

      X = X - 5
      Y = Y - 90
      CALL HIY(X,Y,A)
      WRITE(*,*)E//'LP'//A
      WRITE(*,*)E//'LT3'//TIC(I)
      Y = Y + 77
      X = X + 5

40    CONTINUE
      WRITE(*,*)E//'SC'

C-----------------------------------------
C     THIS PLACES THE VERTICAL TICS AND TEXT OF THE GRAPH
C-----------------------------------------

      ISEG = ISEG + 1
      X = 150
      Y = 2890

      CALL INTRPT(ISEG,SEGCNT)

      WRITE(*,*)E//'SE'//SEGCNT

      DO 60 I = 1, 21

      WRITE(*,*)E//'ML7'
      CALL HIY(X,Y,A)
      WRITE(*,*)E//'LP'//A
      WRITE(*,*)E//'LT4'//TIC1(I)

      WRITE(*,*)E//'ML15'
      X = X + 65
      Y = Y + 10
      CALL HIY(X,Y,A)
      WRITE(*,*)E//'LP'//A

      X = X + 50
      CALL HIY(X,Y,A)
      WRITE(*,*)E//'LG'//A

      X = X - 50
      Y = Y - 10

C-----------------------------------------
C     YOU DONT NEED TIC MARKS AFTER THE LAST NUMBER
C-----------------------------------------
```

```
      IF(I.EQ.21)GOTO 60

C--------------------------------------
C     THIS DRAWS 9 TIC MARKS IN BETWEEN THE LARG ONES
C--------------------------------------

      DO 50 J - 1,9

      CALL HIY(X,Y,A)
      WRITE(*,*)E//'LP'//A

      X - X + 25
      CALL HIY(X,Y,A)
      WRITE(*,*)E//'LG'//A
      Y - Y - 10
      X - X - 25
50    CONTINUE
      Y - Y - 10
      X - X - 65

60    CONTINUE

      WRITE(*,*)E//'SC'

C--------------------------------------
C     AT THIS POINT WE MAKE 16 BOXES OF DIFFERENT COLORS THAT REPRESENT
C     THE MULLER MATRIX.  TWO OF THE BOXES WILL BE THE SAME COLOR WHITE
C     BECAUSE BLACK IS COLOR 16 AND THAT IS THE LIMIT OF A 4111.  BUT
C     SINCE THESE ARE LINES I WILL JUST CHANGE THE STYLE OF LINE 16.
C
C     THE MATRIX ELEMENTS ARE DEFINED AS SEGMENT NUMBERS 10 - 25
C--------------------------------------

65    L - 0
      K - 0
      X - 3000
      Y - 600

      ISEG - 9

      DO 80 I - 1,4              ! ROW LOOP
       DO 70 J - 1,4             ! COLUMN LOOP
          L - L + 1              ! COLOR INDEX COUNTER
          ISEG - ISEG + 1        ! SEGMENT COUNTER
          CALL INTRPT(ISEG,SEGCNT)      ! CONVERT INTEGER TO TEK

          WRITE(*,*)E//'SE'//SEGCNT     ! BEGIN THE SEGMENT

C--------------------------------------
C     THIS SETS THE BLOCK COLOR
C--------------------------------------

      IF(I.EQ.4.AND.J.EQ.4)WRITE(*,*)E//'MV1' ! DASHED LINE ON LAST BOX
```

```

      IF(CP(L).EQ.'Q')CP(L) - CHAR(39)  ! DEFINE THE COLOR

      WRITE(*,*)E//'MP'//CP(L)        ! WRITE THE COLOR TO TERM

C-----------------------------------------

      CALL HIY(X,Y,A)                 ! CONVERT ORIGIN VECTOR

      WRITE(*,*)E//'LP'//A//'1'       ! AND WRITE IT TO TERMINAL


      X - X + 150                     ! INCREMENT THE X BY 150
      CALL HIY(X,Y,A)                 ! CONVERT IT TO TEK CHARA
      WRITE(*,*)E//'LG'//A            ! DRAW FROM LAST VECTOR

      Y - Y - 100                     ! DECREMENT THE Y BY 100
      CALL HIY(X,Y,A)                 ! CONVERT TO TEK CHARACTER
      WRITE(*,*)E//'LG'//A            ! DRAW FROM LAST VECTOR

      X - X - 150                     ! DECREMENT X BY 150
      CALL HIY(X,Y,A)                 ! CONVERT TO TEK CHARACTER
      WRITE(*,*)E//'LG'//A            ! DRAW FROM LAST VECTOR
      WRITE(*,*)E//'LE'               ! FILL THE PANEL

C-----------------------------------------
C     THIS PUTS THE TEXT NUMBER IN THE BOX
C-----------------------------------------

      K - K + 1                       ! ELEMENT COUNTER USED FOR
                                      ! PLACING THE TEXT.  IF
C     IF(K.LT.10)THEN                 ! THE TEXT IS 1 CHARACTER
C       X - X + 60                    ! THEN X IS INCREMENTED BY
C       Y - Y + 25                    ! 60 AND Y BY 25.  IF THE
C     ELSE                            ! TEXT IS 2 CHARACTERS
        X - X + 30                    ! X IS INCREMENTED BY 30
        Y - Y + 25                    ! SO THAT IT IS CENTERED.
C     ENDIF

      WRITE(*,*)E//'MC84C4:'          ! THIS IS THE TEXT SIZE

      CALL HIY(X,Y,A)                 ! CONVERT X,Y TO TEK CHARA
      WRITE(*,*)E//'LF'//A            ! SET THE ORIGIN
      WRITE(*,*)E//'MT'//COL(K)       ! SET THE BOX COLOR
      WRITE(*,*)E//'LT2'//NUM(K)      ! WRITE THE TEXT

      WRITE(*,*)E//'SC'               ! CLOSE THE SEGMENT

C     IF(K.LT.10)THEN                 ! INCREMENT X AND Y FOR
C       X - X + 140                   ! THE NEXT BOX IN THE
C       Y - Y + 75                    ! ROW.
C     ELSE
      Y - Y + 75
      X - X + 155
```

```
C     ENDIF


 70   CONTINUE                      ! END COLUMN LOOP

      X - 3000                    ! GO TO START POSITION
      Y - Y - 150                 ! FOR THE NEXT ROW

 80   CONTINUE                      ! END ROW LOOP

C----------------------------------------------
C     HERE I AM PROVIDING TWO BUTTONS FOR THE USER.  THE FIRST IS "EXIT"
C     AND THE OTHER IS " ALL ".  WHICH IF SELECTED WILL INDICATE TO THE
C     USER THAT ALL INCOMING VECTOR INFORMATION IS TO BE DISPLAYED.
C     THIS MEANS THAT IN EACH SWEEP OF THE SAMPLE STAGE 9 ELEMENTS WILL
C     BE SIMULTANEOUSLY DRAWN TO THE BAR GRAPH WHICH WILL DIRECTLY
C     REPRESENT 9 MATRIX ELEMENTS PROVIDING DATA FOR THAT PARTICULAR
C     POLARIZING POSITION.
C
C     THESE SEGMENTS WILL BE 40 AND 41 RESPECTIVELY.
C----------------------------------------------

      ISEG - 39
      X - 2000                    ! X ORIGIN OF EXIT BOX
      Y - 650                     ! Y ORIGIN OF EXIT BOX
      K - 0
      WRITE(*,*)E//'MV0'             ! SOLID LINES

       DO 100 J - 1,2

      ISEG - ISEG + 1               ! INCREMENT SEGMENT CNTR
      CALL INTRPT(ISEG,SEGCNT)          ! CONVERT INTEGER TO TEK

      WRITE(*,*)E//'SO'//SEGCNT          ! BEGIN THE SEGMENT

C----------------------------------------------
C     THIS SETS THE BLOCK COLOR
C----------------------------------------------

      IF(J.EQ.2)THEN                ! IF " ALL " BOX THEN
        WRITE(*,*)E//'MP5'           ! COLOR IT BLUE
      ELSE                        ! OR ELSE IF ITS THE
        WRITE(*,*)E//'MP-'           ! " EXIT " BOX THEN COLOR
      ENDIF                       ! IT RED.


        CALL HIY(X,Y,A)             ! CONVERT VECTOR TO TEK
        WRITE(*,*)E//'LP'//A//'1'     ! SET THE ORIGIN

        X - X + 400               ! INCREMENT THE X BY 400
        CALL HIY(X,Y,A)             ! CONVERT TO TEK CHARACTER
        WRITE(*,*)E//'LG'//A         ! DRAW FROM LAST VECTOR

        Y - Y - 100               ! DECREMENT THE Y BY 100
```

```
        CALL HIY(X,Y,A)             ! CONVERT TO TEK CHARACTER
        WRITE(*,*)E//'LG'//A        ! DRAW FROM LAST VECTOR

        X - X - 400                 ! DECREMENT X BY 400
        CALL HIY(X,Y,A)             ! CONVERT TO TEK CHARACTER
        WRITE(*,*)E//'LG'//A        ! DRAW FROM LAST VECTOR
        WRITE(*,*)E//'LE'           ! FILL THE BOX WITH COLOR

C------------------------------------------------
C     THIS PUTS THE TEXT NUMBER IN THE BOX
C------------------------------------------------

        X - X + 50                  ! X,Y ARE SET TO PLACE
        Y - Y + 25                  ! THE TEXT ORIGIN
        K - K + 1                   ! ARRAY COUNTER

      WRITE(*,*)E//'MCB4C4:'        ! THIS IS TEXT SIZE

      CALL HIY(X,Y,A)               ! ORIGIN INTEGER TO TEK
      WRITE(*,*)E//'LP'//A          ! SET THE ORIGIN
      WRITE(*,*)E//'MT1'            ! TEXT COLOR WHITE
      WRITE(*,*)E//'LT6'// TEXT1(K) ! WRITE THE TEXT

      WRITE(*,*)E//'SC'             ! CLOSE THE SEGMENT

      X - X + 400                   ! SET UP FOR NEXT BOX
      Y - Y + 75
100   CONTINUE                      ! LOOP FOR NEXT BOX

C------------------------------------------------
C     THIS PART ACTIVATES A SUBROUTINE TO PLACE A GIN DEVICE FOR THE
C     MOUSE.
C------------------------------------------------

      X - 2350
      Y - 600
      IFLAG - 1

      CALL GIN(X,Y,IFLAG,IMODE,ITYPE IGIN,IPORT)

C------------------------------------------------
C     THIS DRAWS A BLUE PANEL IN THE LOWER LEFT PART OF THE SCREEN
C     WITH A RED BORDER.  THIS WILL BE USED AS THE BACKGROUND FOR THE
C     DIALOG AREA WHICH WILL SIT ON TOP OF THIS PANEL.
C------------------------------------------------


      ISEG - 1010
      CALL INTRPT(ISEG,SEGCNT)
      WRITE(*,*)E//'SE'//SEGCNT
      WRITE(*,*)E//'MP3'
      WRITE(*,*)E//'ML2'
      X - 0
      Y - 0
```

```
      CALL HIY(X,Y,A)
      WRITE(*,*)E//'LP'//A//'1'
      X - 0
      Y - 500
      CALL HIY(X,Y,A)
      WRITE(*,*)E//'LG'//A
      X - 1950
      Y - 500
      CALL HIY(X,Y,A)
      WRITE(*,*)E//'LG'//A
      X - 1950
      Y - 0
      CALL HIY(X,Y,A)
      WRITE(*,*)E//'LG'//A
      WRITE(*,*)E//'SC'

C-----------------------------------------------------------
C     THIS PART ESTABLISHES THE DIALOG AREA TO BE 40 CHARACTERS IN
C     LENGTH, 5 LINES WIDE, WITH ORIGIN IN THE LOWER LEFT.
C-----------------------------------------------------------

112   WRITE(*,*)E//'LV0'
      WRITE(*,*)E//'LX00'
      WRITE(*,*)E//'LL5'

115   WRITE(*,*)E//'LZ'
      WRITE(*,*)E//'LCB:'
      WRITE(*,*)E//'LI144'
      WRITE(*,*)E//'LV1'

      INU - 1
      CALL SEE_ELE(ISWEEP,MATRIX,INU)

1000  WRITE(*,*)E//'LV0'
      WRITE(*,*)E//'TD'
      WRITE(*,*)E//'LZ'
      WRITE(*,*)E//'LV1'

1010  RETURN

      END
```

AIV.24 Analog APSD Software Modules: TEK_INPUTS Source Code.

```
      SUBROUTINE TEK_INPUTS(NOSAMP,SAMP,AGENT,CONC,START,STOP,
     .               INC,NAME,DATE,TIME,IEXIT)

C-----------------------------------
C     THIS MOD IS USED FOR ENTERING DATA IF THE USER WANTS TEKTRONIX
C     GRAPHICS.  IT ALLOWS THE USER TO ENTER ALL SAMPLE AND LASER DATA
C     FOR THE MULLER MATRIX EXPERIMENT.
C-----------------------------------


      CHARACTER TITLE*48,NAME*20,SAMP*20,AGENT*20,CONC*15
      CHARACTER SAMPLE,CORRECT*2
      CHARACTER DATE*9,TIME*8,TEXT*80,E*1,A*5,SEG*3
      CHARACTER SAME*1,START*6,STOP*6,INC*6,POS*6

      INTEGER CHANGE,X,Y,X1,Y1,Y2,Y3

C-----------------------------------

      DIMENSION SAMP(10),AGENT(10),CONC(10)
      DIMENSION START(10),STOP(10),INC(10),POS(10)

    1 FORMAT('1',//////T16,A,//)
    2 FORMAT(A20)
    3 FORMAT(A4)
    4 FORMAT('O')
    5 FORMAT(A2)
    6 FORMAT (A)
    7 FORMAT(I5)
    8 FORMAT (A10)


      E    - CHAR(27)
      CHANGE - 0
      ISTART - 1
      IEXIT  - 0              ! USER WANTS TO QUIT ...RETURN

C-----------------------------------
C     THIS PART PLACES A GREEN PANEL SEGMET 8000 NO THE ENTIRE SCREEN
C     FOR A BACKGROUND.
C-----------------------------------

      WRITE(*,*)E//'%!0'           ! PLACE IN TEK MODE

      ISEG - 8000
      CALL INTRPT (ISEG,SEG)
      WRITE(*,*)E//'SE'//SEG         ! BEGIN THE PANEL 8000
      WRITE(*,*)E//'MP#'             ! PANEL COLOR GREEN
      X - 1
      Y - 1
      CALL HIY(X,Y,A)
```

```
          WRITE(*,*)E//'LP'//A              ! SET PANEL ORIGIN

        X - 4095
        CALL HTY(X,Y,A)
        WRITE(*,*)E//'LG'//A              ! DRAW BOTTOM OF PANEL
c       Y - 3150
        Y - 3276
        CALL HTY(X,Y,A)
        WRITE(*,*)E//'LG'//A              ! DRAW LEFT SIDE OF PANEL
        X - X - 4094
        CALL HTY(X,Y,A)
        WRITE(*,*)E//'LG'//A              ! DRAW TOP OF PANEL
        WRITE(*,*)E//'SC'                 ! CLOSE  AND FILL PANEL

C--------------------------------------------------
C       THIS IS THE FIRST LINE THAT IS REQUIRED BY THE USER.
C       NAME, DATE, TIME AND NUMBER OF SAMPLES.
C--------------------------------------------------


        CALL TIME4(DATE,TIME)               ! SYSTEM TIME AND DATE
        TEXT - '1.  NAME:'
        ISEG - 1
        CALL INTRPT(ISEG,SEG)

        WRITE(*,*)E//'SE'//SEG            ! BEGIN THE SEGMENT
        WRITE(*,*)E//'MT4'                ! LINE COLOR WHITE
        X - 100
        Y - 3100
        CALL HTY (X,Y,A)
        WRITE(*,*)E//'LP'//A              ! SET TEXT ORIGIN
        WRITE(*,*)E//'LT9'//DATE            ! WRITE THE DATE
        X - 1000
        CALL HTY (X,Y,A)
        WRITE(*,*)E//'LP'//A              ! SET TEXT ORIGIN
        WRITE(*,*)E//'LT8'//TIME            ! WRITE THE TIME

        WRITE(*,*)E//'MT2'                ! LINE COLOR RED
        X - 2000
        CALL HTY (X,Y,A)
        WRITE(*,*)E//'LP'//A              ! SET TEXT ORIGIN
        WRITE(*,*)E//'LT9'//TEXT            ! WRITE THE DATE
        WRITE(*,*)E//'SC'

C--------------------------------------------------
C       THIS PART DRAWS  A BOX AT THE BOTTOM OF THE SCREEN FOR THE DIALOG
C       AREA INPUTS.
C--------------------------------------------------

        ISEG - ISEG + 1
        CALL INTRPT (ISEG,SEG)
        WRITE(*,*)E//'SE'//SEG            ! BEGIN THE PANEL 8000
        WRITE(*,*)E//'ML1'                ! LINE COLOR WHITE
        WRITE(*,*)E//'MP5'                ! PANEL COLOR BLUE
```

```
      X - 1
      Y - 1
      CALL HTY(X,Y,A)
      WRITE(*,*)E//'LP'//A//'1'        ! SET PANEL ORIGIN
      X - 4095
      CALL HTY(X,Y,A)
      WRITE(*,*)E//'LG'//A             ! DRAW BOTTOM OF PANEL
      Y - 450
      CALL HTY(X,Y,A)
      WRITE(*,*)E//'LG'//A             ! DRAW LEFT SIDE OF PANEL
      X - X - 4094
      CALL HTY(X,Y,A)
      WRITE(*,*)E//'LG'//A             ! DRAW TOP OF PANEL
      WRITE(*,*)E//'LE'                ! FILL THE PANEL

C-----------------------------------------------
C     THIS DRAWS A LINE AROUND THE TWO DIALOG AREA LINES:  COLOR RED
C-----------------------------------------------

      WRITE(*,*)E//'ML2'               ! PANEL COLOR RED
      X - 100
      Y - 150
      CALL HTY(X,Y,A)
      WRITE(*,*)E//'LP'//A             ! SET PANEL ORIGIN
      X - 3995
      CALL HTY(X,Y,A)
      WRITE(*,*)E//'LG'//A             ! DRAW BOTTOM OF PANEL
      Y - Y + 250
      CALL HTY(X,Y,A)
      WRITE(*,*)E//'LG'//A             ! DRAW LEFT SIDE OF PANEL
      X - 100
      CALL HTY(X,Y,A)
      WRITE(*,*)E//'LG'//A             ! DRAW TOP OF PANEL
      Y - Y - 250
      CALL HTY(X,Y,A)
      WRITE(*,*)E//'LG'//A             ! DRAW LEFT SIDE OF PANEL

C-----------------------------------------------

      WRITE(*,*)E//'MT1'               ! TEXT COLOR WHITE

      TEXT - 'ENTER < E >  TO EXIT'

      X - 1600
      Y - 45
      CALL HTY(X,Y,A)
      WRITE(*,*)E//'LP'//A             ! SET TEXT ORIGIN
      WRITE(*,*)E//'LTA4'//TEXT          ! WRITE THE TEXT

      WRITE(*,*)E//'SC'                ! CLOSE  AND FILL PANEL

C-----------------------------------------------
C     THIS SETS UP THE DIALOG AREA SO THAT THE TEXT AND DATA ENTRY
C     ARE ALL WITHIN THE DATA INPUT WINDOW.
```

```
      C-----------------------------------------------

            WRITE(*,*)E//'LV0'          ! DISABLE DIALOG AREA
            WRITE(*,*)E//'LZ'           ! CLEAR DIALOG AREA
            WRITE(*,*)E//'LL2'          ! DIALOG AREA 2 LINES
            WRITE(*,*)E//'LCD1'         ! 65 CHARACTER ALLOWED
            WRITE(*,*)E//'ML1'          ! DIALOG TEXT WHITE
            X = 200
            Y = 160
            CALL HIY(X,Y,A)
            WRITE(*,*)E//'LX'//A        ! SET TEXT DIALOG ORIGIN

            WRITE(*,*)E//'LV1'          ! ENABLE DIALOG AREA


      C-----------------------------------------------
      C     THIS GETS THE NAME OF THE PERSON RUNNING THE EXPERIMENT
      C-----------------------------------------------
      C**************

      30    FORMAT(5X,'ENTER YOUR NAME:  ',$)      ! GET USERS NAME

      40    WRITE(*,30,ERR=40)
           READ(*,2,ERR=40) NAME             ! READ EXPERIMENTER NAME

           IF(NAME.EQ.'E'.OR.NAME.EQ.'e')THEN    ! USER WANTS TO EXIT
            IEXIT = 1                     ! SET EXIT FLAG
            GOTO 1000                     ! RETURN TO CALLER
           ENDIF

      C-----------------------------------------------
      C     THIS WRITES THE USER INPUTS TO THE SCREEN PANEL AS A  GRAPHIC
      C-----------------------------------------------

           ISEG = 5
           CALL INTRPT(ISEG,SEG)

           WRITE(*,*)E//'SK'//SEG         ! DELETE THE  SEGMENT

           WRITE(*,*)E//'SE'//SEG         ! BEGIN THE SEGMENT
           WRITE(*,*)E//'MP7'             ! PANEL COLOR GRAY
           WRITE(*,*)E//'MT1'             ! TEXT COLOR WHITE

           X = 2500
           Y = 3075
           CALL HIY(X,Y,A)
           WRITE(*,*)E//'LP'//A           ! SET PANEL ORIGIN
           X = X + 1100
           CALL HIY(X,Y,A)
           WRITE(*,*)E//'LG'//A           ! DRAW BOTTOM OF PANEL
           Y = Y + 100
           CALL HIY(X,Y,A)
           WRITE(*,*)E//'LG'//A           ! DRAW LEFT SIDE OF PANEL
           X = X - 1100
```

```
      CALL HTY(X,Y,A)
      WRITE(*,*)E//'LG'//A           ! DRAW TOP OF PANEL
      WRITE(*,*)E//'LE'              ! FILL THE PANEL

C-----------------------------------------------
C     THIS PLACES THE TEXT IN THE PANEL
C-----------------------------------------------

      X = 2550
      Y = 3100
      CALL HTY(X,Y,A)
      WRITE(*,*)E//'LP'//A           ! SET THE ORIGIN
      WRITE(*,*)E//'LTA4'//NAME         ! WRITE THE NAME
      WRITE(*,*)E//'SC'              ! CLOSE THE SEGMENT

C-----------------------------------------------

      IF (CHANGE.EQ.1) GOTO 500

C****************
C-----------------------------------------------
C     THIS GETS THE NUMBER OF SAMPLES AND THE NAME OF EACH SAMPLE
C-----------------------------------------------
C****************

50    FORMAT(5X,' ENTER NUMBER OF SAMPLES < 8 MAX > : ',$)
60    WRITE(*,50)
      READ (*,'(A1)',ERR=60)SAMPLE         ! INPUT NUMBER OF SAMPLES

      IF(SAMPLE.EQ.'E'.OR.SAMPLE.EQ.'e')THEN  ! USER WANTS TO EXIT
       IEXIT = 1                     ! SET EXIT FLAG
       GOTO 1000                      ! RETURN TO CALLER
      ENDIF

      IF(ICHAR(SAMPLE).GT.48.AND.ICHAR(SAMPLE).LT.57)GOTO 65
      GOTO 60                        ! INPUT BAD DO IT AGAIN

65    READ(SAMPLE,'(I4)',ERR = 60)NOSAMP    ! CONVERT TO INTEGER

C-----------------------------------------------
C     THIS DRAWS THE SAMPLE DATA HEADER AND NUMBER OF SAMPLES
C-----------------------------------------------

      ISEG = 7
      CALL INTRPT(ISEG,SEG)
      WRITE(*,*)E//'SK'//SEG           ! DELETE THE SEGMENT
      WRITE(*,*)E//'SE'//SEG           ! BEGIN THE SEGMENT

      X = 1
      Y = 3025
      CALL HTY(X,Y,A)
      WRITE(*,*)E//'MLA'              ! LINE COLOR BLUE
      WRITE(*,*)E//'LP'//A             ! SET LINE ORIGIN
```

```
      X - 4095
     CALL HTY(X,Y,A)
     WRITE(*,*)E//'LG'//A          ! SET LINE END

     WRITE(*,*)E//'MT2'            ! TEXT COLOR RED

     X - 2000
     Y - 2895
     CALL HTY(X,Y,A)
     WRITE(*,*)E//'LF'//A          ! SET TEXT ORIGIN
     TEXT - '2. '
     WRITE(*,*)E//'LT4'//TEXT       ! WRITE THE TEXT

     WRITE(*,*)E//'MP7'            ! PANEL COLOR GRAY
     WRITE(*,*)E//'MT1'            ! TEXT COLOR WHITE

     X - 2500
     Y - 2870
     CALL HTY(X,Y,A)
     WRITE(*,*)E//'LP'//A          ! SET PANEL ORIGIN

     X - X + 1100
     CALL HTY(X,Y,A)
     WRITE(*,*)E//'LG'//A          ! DRAW BOTTOM OF BOX

     Y - Y + 100
     CALL HTY(X,Y,A)
     WRITE(*,*)E//'LG'//A          ! RIGHT SIDE OF BOX

     X - X - 1100
     CALL HTY(X,Y,A)
     WRITE(*,*)E//'LG'//A          ! TOP OF BOX
     WRITE(*,*)E//'LE'            ! FILL THE BOX

     WRITE(*,*)E//'MT1'            ! TEXT COLOR WHITE

     X - X + 100
     Y - Y - 75
     CALL HTY(X,Y,A)
     WRITE(*,*)E//'LF'//A          ! SET TEXT ORIGIN

     IF(NOSAMP.EQ.1)THEN
      WRITE(*,*)E//'LT8'//SAMPLE//' SAMPLE' ! WRITE THE TEXT
     ELSE
      WRITE(*,*)E//'LT9'//SAMPLE//' SAMPLES'     ! WRITE THE TEXT
     ENDIF

     WRITE(*,*)E//'SC'            ! END SEGMENT

     IF (CHANGE.EQ.1) GOTO 500

C------------------------------------------
C    THIS WRITE THE COLUMN HEADERS FOR " SAMPLE NAMES ", " AGENT TYPES "
C    AND " CONCENTRATIONS "
```

```
C---------------------------------------------

      ISEG - 8
      CALL INTRPT(ISEG,SEG)
      WRITE(*,*)E//'SE'//SEG            ! BEGIN THE SEGMENT

      WRITE(*,*)E//'MT4'                ! TEXT COLOR BLUE
      X - 420
      Y - 2750
      CALL HTY(X,Y,A)
      WRITE(*,*)E//'LP'//A              ! SET TEXT ORIGIN
      TEXT - '3. SAMPLE TYPE'
      WRITE(*,*)E//'LT?'//TEXT          ! WRITE SAMPLE TEXT

      X - 1790
      CALL HTY(X,Y,A)
      WRITE(*,*)E//'LP'//A              ! SET TEXT ORIGIN
      TEXT - '4. AGENT TYPE'
      WRITE(*,*)E//'LT>'//TEXT          ! WRITE SAMPLE TEXT

      X - 3000
      CALL HTY(X,Y,A)
      WRITE(*,*)E//'LP'//A              ! SET TEXT ORIGIN
      TEXT - '5. CONCENTRATION'
      WRITE(*,*)E//'LTA2'//TEXT         ! WRITE SAMPLE TEXT
      WRITE(*,*)E//'SC'                 ! CLOSE THE SEGMENT

C---------------------------------------------
C     THIS NEXT PART GETS THE SAMPLE NAME/S
C---------------------------------------------


67    FORMAT(5X,'IF ALL SAMPLES ARE THE SAME TYPE < Y >  ',$)

68    ISAME - 0
      INUSE - 0
      WRITE(*,*)E//'LZ'                 ! CLEAR THE DIALOG AREA
      WRITE(*,67)
      READ(*,6)SAME

      IF(SAME.EQ.'Y'.OR.SAME.EQ.'y')ISAME - 1

C---------------------------------------------

66    ISEG  - 9
      LETTER - 65                       ! ITEM BEGINS WITH " A "
      Y1    - 2600

69    X  - 70

70    FORMAT(5X,'ENTER SAMPLE NAME:  ',$)
72    FORMAT(5X,'J',A1,'  ENTER SAMPLE NAME:  ',$)

      DO 90 I - ISTART , NOSAMP
```

```
C-----------------------------------------------
C      THIS WILL ONLY REQUIRE ONLY ONE INPUT IF THE SAMPLES ARE ALL THE
C      SAME.
C-----------------------------------------------

      IF(ISAME.EQ.1)THEN              ! ALL SAMPLES ARE THE SAME

       IF(INUSE.EQ.0)THEN             ! ONLY READ ONCE
      WRITE(*,*)E//'LZ'              ! CLEAR THE DIALOG AREA
 75      WRITE(*,70)
        READ(*,2,ERR=75)SAMP(I)

      IF(SAMP(I).EQ.'E'.OR.SAMP(I).EQ.'e')THEN ! USER WANTS TO EXIT
          IEXIT = 1                ! SET EXIT FLAG
          GOTO 1000                ! RETURN TO CALLER
         ENDIF

      IF(SAMP(I).EQ.' ')GOTO 75       ! DO AGAIN FIELD WAS NULL

        INUSE = 1                  ! SET FLAG DONT COME BACK
        GOTO 85
       ELSE

        SAMP(I) = SAMP(I - 1)         ! EQUATE THE SAMPLE TYPES

        GOTO 85                    ! WRITE THE SAMPLE NAME
       ENDIF
      ENDIF


 80   WRITE(*,*)E//'LZ'              ! CLEAR THE DIALOG AREA
      WRITE(*,72)CHAR(LETTER)         ! DESIGNATOR

      READ(*,2,ERR=80)SAMP(I)         ! INPUT SAMPLE NAME

      IF(SAMP(I).EQ.'E'.OR.SAMP(I).EQ.'e')THEN ! USER WANTS TO EXIT
       IEXIT = 1                   ! SET EXIT FLAG
       GOTO 1000                   ! RETURN TO CALLER
      ENDIF

      IF(SAMP(I).EQ.' ')GOTO 80       ! DO AGAIN FIELD WAS NULL

 85   ISEG = ISEG + 1

      CALL INTRPT(ISEG,SEG)
      WRITE(*,*)E//'SK'//SEG         ! DELETE THE SEGMENT
      WRITE(*,*)E//'SE'//SEG         ! BEGIN THE SEGMENT

      WRITE(*,*)E//'MT2'             ! TEXT COLOR RED

      CALL HTY(X,Y1,A)
      WRITE(*,*)E//'LF'//A           ! SET TEXT ORIGIN
      TEXT = '3'//CHAR(LETTER)        ! ITEM DESIGNATOR
```

```
        WRITE(*,*)E//'LT2'//TEXT            ! WRITE THE TEXT

      WRITE(*,*)E//'MP7'              ! PANEL COLOR GRAY
      WRITE(*,*)E//'MT1'              ! TEXT COLOR WHITE

      Y - Y1 - 25

      X - X + 130
      CALL HTY(X,Y,A)
      WRITE(*,*)E//'LP'//A            ! SET PANEL ORIGIN

      X - X + 1100
      CALL HTY(X,Y,A)
      WRITE(*,*)E//'LG'//A            ! DRAW BOTTOM OF BOX

      Y - Y + 100
      CALL HTY(X,Y,A)
      WRITE(*,*)E//'LG'//A            ! RIGHT SIDE OF BOX

      X - X - 1100
      CALL HTY(X,Y,A)
      WRITE(*,*)E//'LG'//A            ! TOP OF BOX
      WRITE(*,*)E//'LE'             ! FILL THE BOX

      Y - Y - 75
      X - X + 50

      WRITE(*,*)E//'MT1'              ! TEXT COLOR WHITE

      CALL HTY(X,Y,A)
      WRITE(*,*)E//'LP'//A            ! SET TEXT ORIGIN
      WRITE(*,*)E//'LTA4'//SAMP(I)        ! WRITE THE TEXT

      WRITE(*,*)E//'SC'             ! CLOSE THE SEGMENT

      IF(ITEM_CNG.EQ.1)GOTO 500           ! JUST CHANGING ONE ITEM

      LETTER - LETTER + 1             ! INCREMENT LETTER VALUE
      Y1 - Y1 - 125
      X - 70

90    CONTINUE

C--------------------------------------------
C
      IF (CHANGE.EQ.1) GOTO 500

C--------------------------------------------
C     THIS ASKS IF THERE IS GOING TO BE ANY CHEMICALS ADDED TO THE
C     SAMPLES.  IF NONE ARE USED THEN I SKIP RIGHT TO THE LASER INFO.
C--------------------------------------------

100   FORMAT(5X,'WILL THERE BE AGENT ON THE SAMPLES: < Y > ',$)
```

```
110   WRITE(*,100)
      READ (*,6,ERR-110)ANS

      WRITE(*,*)E/'LZ'              ! CLEAR THE DIALOG AREA

      IF (ANS.EQ.'Y'.OR.ANS.EQ.'y') GOTO 130
      AGENT(1) - 'NONE'
      GOTO 263                     !GOTO REVIEW SCREEN

C----------------------------------------
C     THIS GETS THE TYPE OF AGENT THAT WILL BE USED ON THE SAMPLES
C----------------------------------------

120   FORMAT(5X,'WILL AGENT TYPE BE THE SAME FOR ALL SAMPLES < Y > ',$)

130   INUSE - 0
      ISAME - 0
      SAME  - ''

      WRITE(*,120)

132   READ(*,6,ERR-132)SAME

      IF(SAME.EQ.'Y'.OR.SAME.EQ.'y')ISAME - 1

C----------------------------------------

      WRITE(*,*)E/'LZ'              ! CLEAR THE DIALOG AREA
      ISEG - 19
      X - 1500
      Y1 - 2600
      LETTER - 65
135   FORMAT(5X,'ENTER AGENT NAME: ',$)
137   FORMAT(5X,'4',A1,' ENTER AGENT NAME: ',$)

138   DO 190 I - ISTART , NOSAMP

C----------------------------------------
C     THIS WILL ONLY REQUIRE ONLY ONE INPUT IF THE SAMPLES ARE ALL THE
C     SAME.
C----------------------------------------

      IF(ISAME.EQ.1)THEN             ! ALL AGENTS ARE THE SAME

       IF(INUSE.EQ.0)THEN            ! ONLY READ ONCE

140     WRITE(*,135)
        READ(*,2,ERR-140)AGENT(I)

      IF(AGENT(I).EQ.'E'.OR.AGENT(I).EQ.'e')THEN   ! USER WANTS TO EXIT
          IEXIT - 1                 ! SET EXIT FLAG
          GOTO 1000                 ! RETURN TO CALLER
         ENDIF

```

```
          IF(AGENT(I).EQ.' ')GOTO 140        ! DO AGAIN FIELD WAS NULL

         INUSE = 1                  ! SET FLAG DONT COME BACK

         GOTO 160
        ELSE

         AGENT(I) = AGENT(I - 1)        ! EQUATE THE AGENT TYPES
         GOTO 160                   ! WRITE THE AGENT NAME
        ENDIF
       ENDIF

 150   WRITE(*,137)CHAR(LETTER)           ! LIST DESIGNATOR

       READ(*,2,ERR=150)AGENT(I)           ! INPUT AGENT NAME

       IF(AGENT(I).EQ.'E'.OR.AGENT(I).EQ.'e')THEN ! USER WANTS TO EXIT
        IEXIT = 1                    ! SET EXIT FLAG
        GOTO 1000                    ! RETURN TO CALLER
       ENDIF

       IF(AGENT(I).EQ.' ')GOTO 150        ! DO AGAIN FIELD WAS NULL

 160   ISEG = ISEG + 1
       CALL INTRPT(ISEG,SEG)
       WRITE(*,*)E//'SK'//SEG             ! DELETE THE SEGMENT
       WRITE(*,*)E//'SE'//SEG             ! BEGIN THE SEGMENT

       WRITE(*,*)E//'MT2'                 ! TEXT COLOR RED

       CALL HTY(X,Y1,A)
       WRITE(*,*)E//'LP'//A               ! SET TEXT ORIGIN
       TEXT = '4'//CHAR(LETTER)             ! ITEM DESIGNATOR
       WRITE(*,*)E//'LT2'//TEXT            ! WRITE THE TEXT

       WRITE(*,*)E//'MP7'                 ! PANEL COLOR GRAY
       WRITE(*,*)E//'MT1'                 ! TEXT COLOR WHITE

       Y = Y1 - 25

       X = X + 130
       CALL HTY(X,Y,A)
       WRITE(*,*)E//'LP'//A               ! SET PANEL ORIGIN

       X = X + 1100
       CALL HTY(X,Y,A)
       WRITE(*,*)E//'LG'//A               ! DRAW BOTTOM OF BOX

       Y = Y + 100
       CALL HTY(X,Y,A)
       WRITE(*,*)E//'LG'//A               ! RIGHT SIDE OF BOX

       X = X - 1100
       CALL HTY(X,Y,A)
```

```
      WRITE(*,*)E//'LG'//A              ! TOP OF BOX
      WRITE(*,*)E//'LE'                 ! FILL THE BOX

      Y - Y - 75
      X - X + 50

      WRITE(*,*)E//'MT1'                ! TEXT COLOR WHITE

      CALL HIY(X,Y,A)
      WRITE(*,*)E//'LP'//A              ! SET TEXT ORIGIN
      WRITE(*,*)E//'LTA4'//AGENT(I)     ! WRITE THE TEXT

      WRITE(*,*)E//'SC'                 ! CLOSE THE SEGMENT

      IF(ITEM_CNG.EQ.1)GOTO 500         ! CHANGING ONE ITEM

      LETTER - LETTER + 1               ! INCREMENT LETTER VALUE
      Y1 - Y1 - 125
      X - 1500

190   CONTINUE

      IF (CHANGE.EQ.1) GOTO 500

C--------------------------------------------
C     THIS GETS THE AGENT CONCENTRATION OF THE AGENT FOR ALL THHE SAMPLES
C--------------------------------------------

195   FORMAT(12X,'WILL AGENT CONCENTRATION BE THE SAME FOR ALL SAMPLES')
198   FORMAT(35X,'< Y > ',$)

199   INUSE - 0
      ISAME - 0
      SAME - ' '

      WRITE(*,195)
      WRITE(*,198)

200   READ(*,6,ERR-200)SAME
      IF(SAME.EQ.'Y'.OR.SAME.EQ.'y')ISAME - 1

C--------------------------------------------

      WRITE(*,*)E//'LZ'                 ! CLEAR THE DIALOG AREA
      ISEG - 29
      X - 2950
      Y1 - 2600
      LETTER - 65
210   FORMAT(5X,'ENTER CONCENTRATION: ',$)
212   FORMAT(5X,'5',A1,' ENTER CONCENTRATION: ',$)

215   DO 260 I - ISTART , NOSAMP

C--------------------------------------------
```

```
C    THIS WILL ONLY REQUIRE ONLY ONE INPUT IF THE SAMPLES ARE ALL THE
C    SAME.
C-----------------------------------------------------------

      IF(ISAME.EQ.1)THEN                ! ALL AGENTS ARE THE SAME

       IF(INUSE.EQ.0)THEN               ! ONLY READ ONCE

 220     WRITE(*,210)

        READ(*,2,ERR-220)CONC(I)

         IF(CONC(I).EQ.'E'.OR.CONC(I).EQ.'e')THEN  ! USER WANTS TO EXIT
          IEXIT - 1                   ! SET EXIT FLAG
          GOTO 1000                   ! RETURN TO CALLER
         ENDIF

      IF(CONC(I).EQ.' ')GOTO 220        ! DO AGAIN FIELD WAS NULL

        INUSE - 1                     ! SET FLAG DONT COME BACK
        GOTO 250
       ELSE

        CONC(I) - CONC(I - 1)           ! EQUATE THE AGENT TYPES

        GOTO 250                      ! WRITE THE AGENT NAME
       ENDIF
      ENDIF

      WRITE(*,212)CHAR(LETTER)

 240  READ(*,2,ERR-240)CONC(I)          ! INPUT AGENT NAME

     IF(CONC(I).EQ.'E'.OR.CONC(I).EQ.'e')THEN  ! USER WANTS TO EXIT
       IEXIT - 1                    ! SET EXIT FLAG
       GOTO 1000                    ! RETURN TO CALLER
      ENDIF

      IF(CONC(I).EQ.' ')GOTO 240        ! DO AGAIN FIELD WAS NULL

 250  ISEG - ISEG + 1
      CALL INTRPT(ISEG,SEG)
      WRITE(*,*)E//'SK'//SEG          ! DELETE THE SEGMENT
      WRITE(*,*)E//'SE'//SEG          ! BEGIN THE SEGMENT

      WRITE(*,*)E//'MT2'              ! TEXT COLOR RED

      CALL HTY(X,Y1,A)
      WRITE(*,*)E//'LP'//A            ! SET TEXT ORIGIN
      TEXT - '3'//CHAR(LETTER)          ! ITEM DESIGNATOR
      WRITE(*,*)E//'LT2'//TEXT        ! WRITE THE TEXT

      WRITE(*,*)E//'MP/'              ! PANEL COLOR GRAY
      WRITE(*,*)E//'MT1'              ! TEXT COLOR WHITE
```

```
      Y - Y1 - 25

      X - X + 130
      CALL HTY(X,Y,A)
      WRITE(*,*)E//'LP'//A              ! SET PANEL ORIGIN

      X - X + 750
      CALL HTY(X,Y,A)
      WRITE(*,*)E//'LG'//A              ! DRAW BOTTOM OF BOX

      Y - Y + 100
      CALL HTY(X,Y,A)
      WRITE(*,*)E//'LG'//A              ! RIGHT SIDE OF BOX

      X - X - 750
      CALL HTY(X,Y,A)
      WRITE(*,*)E//'LG'//A              ! TOP OF BOX
      WRITE(*,*)E//'LE'                 ! FILL THE BOX

      Y - Y - 75
      X - X + 50

      WRITE(*,*)E//'MT1'                ! TEXT COLOR WHITE

      CALL HTY(X,Y,A)
      WRITE(*,*)E//'LP'//A              ! SET TEXT ORIGIN
      WRITE(*,*)E//'LT7'//CONC(I)          ! WRITE THE TEXT

      IF(ITEM_CNG.EQ.1)GOTO 500

      WRITE(*,*)E//'SC'                 ! CLOSE THE SEGMENT

      LETTER - LETTER + 1                  ! INCREMENT LETTER VALUE
      Y1 - Y1 - 125
      X - 2950

  260 CONTINUE

      IF (CHANGE.EQ.1) GOTO 500

C--------------------------------------------
C     THIS DRAWS A LINE IN BLUE ACROSS THE BOTTOM OF THE SAMPLE BOXES
C--------------------------------------------

  263 ISEG - 39
      CALL INTRPT(ISEG,SEG)
      WRITE(*,*)E//'SE'//SEG
      X - 1
      Y - Y1
      ILINE - Y                         ! BUP THE Y FOR THE LINE

      CALL HTY(X,Y,A)
      WRITE(*,*)E//'ML4'                ! LINE  COLOR BLUE
```

```
        WRITE(*,*)E//'LP'//A              ! SET LINE ORIGIN

        X - 4095
        CALL HTY(X,Y,A)
        WRITE(*,*)E//'LG'//A              ! SET LINE END
        WRITE(*,*)E//'MT4'                ! TEXT COLOR RED

        X - 70
        Y - Y - 100
        Y2 - Y
        CALL HTY(X,Y,A)
        WRITE(*,*)E//'LP'//A              ! SET TEXT ORIGIN
        TEXT - '6. START'
        WRITE(*,*)E//'LT9'//TEXT          ! WRITE THE TEXT
        WRITE(*,*)E//'SC'                 ! CLOSE THE SEGMENT

C-----------------------------------------
C     THIS PART GETS THE START, STOP, AND INCREMENT OF THE SAMPLE
C     STAGE FOR EACH SAMPLE.  IN THIS CASE I ALLOW 1/4 DEGREE
C     RESOLUTION ( 500 STEPS ).
C-----------------------------------------

265   FORMAT(5X,'WILL ALL START POSITIONS BE THE SAME: < Y > ',$)
270   FORMAT(5X,'START POSITION OF ARM <DEGREES>: ',$)
272   FORMAT(5X,A2,' START POSITION OF ARM <DEGREES>: ',$)

        ISEG  - 39                        ! SEG NUMBER BEGINS AT 40
        Y3    - Y1
        X     - 70                        ! START POSITION OF X
        NUM   - 54                        ! ASCII VALUE OF 6
        IPOS  - 0                         ! INITIALIZE TYPE OF DAT

275     INUSE - 0

        X1    - X                         ! SAVED VALUE OF START
        Y1    - Y1 - 200                  ! START POSITION OF Y
        LETTER - 65                       ! ASCII VALUE OF A

        WRITE(*,*)E//'LZ'                 ! CLEAR THE DIALOG AREA

        ISAME - 0                         ! INITIALIZE DIFFERENCE

        IF(IPOS.EQ.0)THEN                 ! IF THIS IS START DATA
          WRITE(*,265)                    ! IN DATA FLAG
        ELSEIF(IPOS.EQ.1)THEN             ! THIS IS END ANGLE DATA
          WRITE(*,312)
        ELSEIF(IPOS.EQ.2)THEN             ! THIS IS INCR ANGLE DATA
          WRITE(*,332)
        ENDIF

        READ(*,2,ERR-280)SAME
        IF(SAME.EQ.'Y'.OR.SAME.EQ.'y')ISAME - 1

278     WRITE(*,*)E//'LZ'                 ! CLEAR THE DIALOG AREA
```

```
      DO 310 I = ISTART , NOSAMP

C-----------------------------------------------
C     THIS WILL ONLY REQUIRE ONLY ONE INPUT IF THE SAMPLES ARE ALL THE
C     SAME.
C-----------------------------------------------

      IF(ISAME.EQ.1)THEN                 ! ALL AGENTS ARE THE SAME

       IF(INUSE.EQ.0)THEN               ! ONLY READ ONCE
280    WRITE(*,*)E//'LZ'               ! CLEAR DIALOG AREA

      IF(IPOS.EQ.0)THEN                 ! IF THIS IS START DATA
        WRITE(*,270)
      ELSEIF(IPOS.EQ.1)THEN              ! THIS IS END ANGLE DATA
        WRITE(*,314)
      ELSEIF(IPOS.EQ.2)THEN              ! THIS IS INCR ANGLE DATA
        WRITE(*,334)
      ENDIF


        READ(*,'(A5)',ERR=280)POS(I)

C-----------------------------------------------
C     HERE I CHECK IF THE USER ADDED A DECIMAL POINT TO THE INPUT.
C     THIS WILL BE REQUIRED FOR THE REAL INPUT.  IF IT WAS NOT
C     PROVIDED I ADD ONE.  ANY VALUE IN THE 1/10th PLACE WILL BE MADE
C     A " .5 " AS I AM LIMITING THE RESOLUTION TO 1/2 A DEGREE.
C-----------------------------------------------

      IDEC = 0
      K    = 0

      DO 282 J = 5,1,-1                  ! LOOP THRU INPUT

       IF(POS(I)(J:J).EQ.' ')GOTO 282      ! LOOK FOR A SPACE

       IF(POS(I)(J:J).EQ.'.')THEN       ! LOOK FOR A DECIMAL
        IDEC = IDEC + 1
        IF(IDEC.EQ.2)GOTO 280            ! 2 DECIMALS

       IF(POS(I)(J+1:J+1).GT.CHAR(48).AND.   ! CHANGE ANY INPUT = .5
     .  POS(I)(J+1:J+1).LT.CHAR(58))THEN    ! IF NOT = .0
        POS(I)(J+1:J+1) = '5'
       ENDIF

        GOTO 282                      ! GO CONVERT TO REAL

       ENDIF

       IF(POS(I)(J:J).LT.CHAR(48).OR.POS(I)(J:J).GT.
     .    CHAR(57))GOTO 280

```

```
        IF(POS(I)(J:J).GT.CHAR(47).AND.POS(I)(J:J).LT.
     .   CHAR(58))THEN

       K - K + 1                  ! COUNT NUMBERS

       IF(K.EQ.4)GOTO 280            ! INPUT TOO LARGE
      ENDIF                       ! CAN ONLY BE 3 CHARS
 282  CONTINUE

     IF(K.GT.0)THEN
      K - K + 1
      POS(I)(K:K) - '.'            ! PLACE THE DECIMAL
     ENDIF


     READ(POS(I),'(F5.2)',ERR-280)RCHECK


        WRITE(*,*)E//'LZ'           ! CLEAR THE DIALOG AREA

        IF(POS(I).EQ.'E'.OR.POS(I).EQ.'e')THEN ! USER WANTS TO EXIT
         IEXIT - 1                ! SET EXIT FLAG
         GOTO 1000                 ! RETURN TO CALLER
        ENDIF

     IF(POS(I).EQ.' ')GOTO 280           ! DO AGAIN FIELD WAS NULL
       INUSE - 1                   ! SET FLAG DONT COME BACK
       GOTO 300
      ELSE

       POS(I) - POS(I - 1)       ! EQUATE THE AGENT TYPES

       GOTO 300                    ! WRITE THE AGENT NAME
      ENDIF
     ENDIF


 285  IF(IPOS.EQ.0)THEN                ! IF THIS IS START DATA
       WRITE(*,272)CHAR(NUM)//CHAR(LETTER)
     ELSEIF(IPOS.EQ.1)THEN              ! THIS IS END ANGLE DATA
       WRITE(*,318)CHAR(NUM)//CHAR(LETTER)
     ELSEIF(IPOS.EQ.2)THEN              ! THIS IS INCR ANGLE DATA
       WRITE(*,336)CHAR(NUM)//CHAR(LETTER)
     ENDIF

     POS(I) - ' '
     K - 0

 290  READ(*,'(A5)',ERR-290)POS(I)             ! INPUT AGENT NAME

C-----------------------------------------------
C    HERE I CHECK IF THE USER ADDED A DECIMAL POINT TO THE INPUT.
C    THIS WILL BE REQUIRED FOR THE REAL INPUT.  IF IT WAS NOT
C    PROVIDED I ADD ONE.  ANY VALUE IN THE 1/10th PLACE WILL BE MADE
```

```
C     A * .5 * AS I AM LIMITING THE RESOLUTION TO 1/2 A DEGREE.
C-----------------------------------------

      IDEC - 0
      K   - 0

      DO 292 J - 5,1,-1                        ! LOOP THRU INPUT

       IF(POS(I)(J:J).EQ.' ')GOTO 282        ! LOOK FOR A SPACE

       IF(POS(I)(J:J).EQ.'.')THEN            ! LOOK FOR A DECIMAL
        IDEC - IDEC + 1
        IF(IDEC.EQ.2)GOTO 285                  ! 2 DECIMALS

       IF(POS(I)(J+1:J+1).GT.CHAR(48).AND.   ! CHANGE ANY INPUT - .5
     .  POS(I)(J+1:J+1).LT.CHAR(58))THEN     ! IF NOT - .0
        POS(I)(J+1:J+1) - '5'
       ENDIF

        GOTO 297                            ! GO CONVERT TO REAL
       ENDIF

       IF(POS(I)(J:J).GT.CHAR(47).AND.POS(I)(J:J).LT.
     .   CHAR(58))THEN

        K - K + 1                           ! COUNT NUMBERS

        IF(K.EQ.4)GOTO 285                    ! INPUT TOO LARGE
       ENDIF                                ! CAN ONLY BE 3 CHARS

  292 CONTINUE

      IF(K.GT.0)THEN
       K - K + 1
       POS(I)(K:K) - '.'                    ! PLACE THE DECIMAL
      ENDIF

      READ(POS(I),'(F5.2)',ERR-295)RCHECK
      GOTO 297

C-----------------------------------------
C     THE START,STOP,AND INCREMENT OF STAGES MUST BE GOOD NUMBERS OR THE
C     EXPERIMENT WONT WORK...
C-----------------------------------------

  295 WRITE(*,*)'E//LZ'                     ! CLEAR DIALOG AREA
      GOTO 285

C-----------------------------------------


  297 IF(POS(I).EQ.'E'.OR.POS(I).EQ.'e')THEN  ! USER WANTS TO EXIT
       IEXIT - 1                            ! SET EXIT FLAG
       GOTO 1000                             ! RETURN TO CALLER
```

```
        ENDIF

      IF(POS(I).EQ.' ')GOTO 285          ! DO AGAIN FIELD WAS NULL

 300  ISEG - ISEG + 1
      CALL INTRPT(ISEG,SEG)
      WRITE(*,*)E//'SK'//SEG             ! DELETE THE SEGMENT
      WRITE(*,*)E//'SE'//SEG             ! BEGIN THE SEGMENT
      CALL HTY(X,Y1,A)
      WRITE(*,*)E//'LP'//A
      WRITE(*,*)E//'MT2'                 ! TEXT COLOR RED

      TEXT - CHAR(NUM)//CHAR(LETTER)
      WRITE(*,*)E//'LT2'//TEXT           ! WRITE THE TEXT

      WRITE(*,*)E//'MP/'                 ! PANEL COLOR GRAY
      WRITE(*,*)E//'MT1'                 ! TEXT COLOR WHITE

      Y - Y1 - 25

      X - X + 130
      CALL HTY(X,Y,A)
      WRITE(*,*)E//'LP'//A               ! SET PANEL ORIGIN

      X - X + 300
      CALL HTY(X,Y,A)
      WRITE(*,*)E//'LG'//A               ! DRAW BOTTOM OF BOX

      Y - Y + 100
      CALL HTY(X,Y,A)
      WRITE(*,*)E//'LG'//A               ! RIGHT SIDE OF BOX

      X - X - 300
      CALL HTY(X,Y,A)
      WRITE(*,*)E//'LG'//A               ! TOP OF BOX
      WRITE(*,*)E//'LE'                  ! FILL THE BOX

      Y - Y - 75
      X - X + 50

      WRITE(*,*)E//'MT1'                 ! TEXT COLOR WHITE

      CALL HTY(X,Y,A)
      WRITE(*,*)E//'LP'//A               ! SET TEXT ORIGIN
      WRITE(*,*)E//'LTS'//POS(I)         ! WRITE THE TEXT

      WRITE(*,*)E//'SC'                  ! CLOSE THE SEGMENT

      IF(ITEM_CNG.EQ.1)GOTO 500          ! CHANGING ONE ITEM

      LETTER - LETTER + 1                ! INCREMENT LETTER VALUE
      Y1 - Y1 - 125
      X - X1                             ! GIVE X ITS ORIGINAL
                                         ! VALUE
```

```
 310    CONTINUE

        IF(CHANGE.EQ.1)GOTO 500

C------------------------------------------------

        IF(IPOS.EQ.0)THEN                ! THIS IS START POSITION
        DO 315 I = 1,NOSAMP              ! DATA.
          START(I) = POS(I)              ! EXCHANGE DATA
 315      CONTINUE

        ELSEIF(IPOS.EQ.1)THEN            ! THIS IS STOP ANGLE
        DO 316 I = 1,NOSAMP              ! DATA.
          STOP(I) = POS(I)               ! EXCHANGE DATA
 316      CONTINUE

        ELSEIF(IPOS.EQ.2)THEN            ! THIS IS INCR ANGLE
        DO 317 I = 1,NOSAMP              ! DATA.
          INC(I) = POS(I)                ! EXCHANGE DATA
 317      CONTINUE

        ENDIF

C------------------------------------------------
C       THIS NEXT PART USES THE ABOVE SECTION OF THE ROUTINE AGAIN
C       BY CHANGING SOME OF THE VARIABLES, THESE ARE THE X,Y POSITIONS
C       OF THE SEGMENTS AD THEIR ASSIGNED NUMERIC VALUES.
C       THE END POSITIONS WILL BEGIN AT 50 AND THE INCREMENT SEGMENTS
C       WILL BEGIN AT 60.
C------------------------------------------------

 312    FORMAT(5X,'WILL THE END ANGLE BE THE SAME FOR ALL SAMPLES:
      . < Y > ',$)
 314    FORMAT(5X,'END POSITION OF ARM < DEGREES >  : ',$)
 318    FORMAT(5X,A2,' END POSITION OF ARM < DEGREES >  : ',$)

 320    IPOS = IPOS + 1                  ! POSITION TYPE COUNTER
                                         ! 1 - END POSITION AND
                                         ! 2 - INCREMENT OF ANGLE
C------------------------------------------------
C       THIS IS DATA FOR THE STOP ANGLE POSITION FOR THE SAMPLES.
C------------------------------------------------

        IF(IPOS.EQ.1)THEN                ! THIS IS FOR END POSITION DATA
        ISEG = 49
        CALL INTRPT(ISEG,SEG)
        WRITE(*,*)E//'SE'//SEG
        WRITE(*,*)E//'MT4'               ! TEXT COLOR BLUE
        X = 700
        Y = Y2
        CALL HTY(X,Y,A)
        WRITE(*,*)E//'LP'//A             ! SET TEXT ORIGIN
        TEXT = '7. STOP'
        WRITE(*,*)E//'LT'//TEXT          ! WRITE THE TEXT
```

```
      WRITE(*,*)E//'SC'             ! CLOSE THE SEGMENT

      NUM   = NUM + 1         ! INCREMENT NUMERIC VALUE
      X1    = X
      Y1    = Y - 100
      LETTER = 65              ! ASCII VALUE OF A
      ISAME  = 0
      INUSE  = 0

330   WRITE(*,*)E//'LZ'              ! CLEAR THE DIALOG AREA

      ISAME  = 0                 ! INITIALIZE DIFFERENCE
      WRITE(*,312)               ! IN DATA FLAG

      READ(*,2,ERR=330)SAME
      IF(SAME.EQ.'Y'.OR.SAME.EQ.'y')ISAME = 1
      GOTO 278
      ENDIF

C-----------------------------------------
C     THIS IS DATA FOR THE INCREMENT OF THE SAMPLES ANGLES.
C-----------------------------------------

      IF(IPOS.EQ.2)THEN           ! THIS IS FOR END POSITION DATA
332   FORMAT(5X,'WILL THE INCREMENT BE THE SAME FOR ALL SAMPLES:
     . < Y > ',$)
334    FORMAT(5X,'  INCREMENT OF SAMPLE STAGE <DEGREES>    : ',$)
336    FORMAT(5X,A2,'  INCREMENT OF SAMPLE STAGE <DEGREES> : ',$)
      ISEG = 59
      CALL INTRPT(ISEG,SEG)
      WRITE(*,*)E//'SE'//SEG
      WRITE(*,*)E//'MT4'              ! TEXT COLOR BLUE
      X = 1400
      Y = Y2
      CALL HTY(X,Y,A)
      WRITE(*,*)E//'LP'//A           ! SET TEXT ORIGIN
      TEXT = '8.  INCR'
      WRITE(*,*)E//'LT8'//TEXT          ! WRITE THE TEXT
      WRITE(*,*)E//'SC'             ! CLOSE THE SEGMENT


      NUM   = NUM + 1         ! INCREMENT NUMERIC VALUE
      X1    = X
      Y1    = Y - 100
      LETTER = 65              ! ASCII VALUE OF A
      ISAME  = 0
      INUSE  = 0

340   WRITE(*,*)E//'LZ'              ! CLEAR THE DIALOG AREA

      ISAME  = 0                 ! INITIALIZE DIFFERENCE
      WRITE(*,332)               ! IN DATA FLAG

      READ(*,2,ERR=340)SAME
```

```
       IF(SAME.EQ.'Y'.OR.SAME.EQ.'y')ISAME = 1
      GOTO 278
      ENDIF

C-----------------------------------------------------
C     THIS DRAWS A BLUE LINE FROM THE TEXT INPUT SCREEN TO THE BLUE LINE
C     UNDER THE SAMPLE, AGENT AND CONCENTRATION SECTION.  THIS AREA
C     IS USED AS A TEXT INSTRUCTION AS TO HOW THE USER MIGHT MAKE
C     A CORRECTION.
C-----------------------------------------------------

      ISEG = 79
      CALL INTRPT(ISEG,SEG)
      WRITE(*,*)E//'SK'//SEG
      WRITE(*,*)E//'SE'//SEG
      X = 2000
      Y = ILINE
      CALL HIY(X,Y,A)
      WRITE(*,*)E//'ML4'              ! LINE  COLOR BLUE
      WRITE(*,*)E//'LP'//A            ! SET LINE ORIGIN
      Y = 450
      CALL HIY(X,Y,A)
      WRITE(*,*)E//'LG'//A            ! SET LINE END
      X = X + 20
      Y = ILINE - 20
      CALL HIY(X,Y,A)
      WRITE(*,*)E//'LP'//A            ! SET LINE ORIGIN
      Y = 450
      CALL HIY(X,Y,A)
      WRITE(*,*)E//'LG'//A            ! SET LINE END
      Y = ILINE - 20
      CALL HIY(X,Y,A)
      WRITE(*,*)E//'LP'//A            ! SET LINE ORIGIN
      X = 4075
      CALL HIY(X,Y,A)
      WRITE(*,*)E//'LG'//A            ! SET LINE END
      X = 4095
      Y = ILINE
      CALL HIY(X,Y,A)
      WRITE(*,*)E//'LP'//A            ! SET LINE ORIGIN
      Y = 450
      CALL HIY(X,Y,A)
      WRITE(*,*)E//'LG'//A            ! SET LINE END
      Y = ILINE - 20
      X = X - 20
      CALL HIY(X,Y,A)
      WRITE(*,*)E//'LP'//A            ! SET LINE ORIGIN
      Y = 450
      CALL HIY(X,Y,A)
      WRITE(*,*)E//'LG'//A            ! SET LINE END

C-----------------------------------------------------
C     THIS PLACES TEXT IN THE PANEL THAT WAS JUST OUTLINED ABOVE.
C     TEXT COLOR IS RED.
```

```
C-----------------------------------------

      TEXT = '***** CORRECTIONS *****'

      X = 2350
      Y = ILINE - 200
      CALL HTY(X,Y,A)
      WRITE(*,*)'E//LP'//A
      WRITE(*,*)'E//MT2'
      WRITE(*,*)'E//LTA<'//TEXT

      TEXT = 'Enter a single number to'
      Y = Y - 200
      X1 = X - 150

      CALL HTY(X,Y,A)
      WRITE(*,*)'E//LP'//A
      WRITE(*,*)'E//MT4'
      WRITE(*,*)'E//LTA<'//TEXT

      TEXT = 'change a whole group.'
      Y = Y - 100
      CALL HTY(X,Y,A)
      WRITE(*,*)'E//LP'//A
      WRITE(*,*)'E//LTA<'//TEXT


      TEXT = 'Enter a single number with'
      Y = Y - 200
      CALL HTY(X,Y,A)
      WRITE(*,*)'E//LP'//A
      WRITE(*,*)'E//LTA<'//TEXT


      TEXT = 'a letter to change a specific'
      Y = Y - 100
      CALL HTY(X,Y,A)
      WRITE(*,*)'E//LP'//A
      WRITE(*,*)'E//LTA<'//TEXT

      TEXT = 'entry.'
      Y = Y - 100
      CALL HTY(X,Y,A)
      WRITE(*,*)'E//LP'//A
      WRITE(*,*)'E//LTA<'//TEXT


      TEXT = '**** PRESS RETURN TO CONTINUE ****'
      Y = Y - 200
      X = X1
      CALL HTY(X,Y,A)
      WRITE(*,*)'E//LP'//A
      WRITE(*,*)'E//MT2'
      WRITE(*,*)'E//LTB2'//TEXT
```

```

      WRITE(*,*)E//SC'                ! END THE SEGMENT

C------------------------------------------------
C     THIS PART ALLOWS THE USER TO MAKE CORRECTIONS TO ANY DATA ITEM
C     BEFORE MOVING FORWARD.
C------------------------------------------------

500   CHANGE   - 0              ! INITIALIZE CORRECTION FLAGS
      ITEM_CNG - 0
      ISTART   - 1

345   WRITE(*,*)E//LZ'          ! CLEAR THE SCREEN

      CORRECT - ' '             ! INITIALIZE THE ERROR INPUT


350   FORMAT(5X,'ENTER A NUMBER FOR CORRECTIONS: ',$)


      WRITE(*,350)
      READ(*,'(A2)',ERR-345)CORRECT


      IF(CORRECT.EQ.' ')GOTO 1000          ! USER WANTS TO GO ON

C------------------------------------------------
C     THIS EVALUATES THE FIRST CHARACTER TO MAKE  SURE IT IS A NUMBER
C------------------------------------------------

      IF(ICHAR(CORRECT(1:1)).GT.48.AND.
     .     ICHAR(CORRECT(1:1)).LT.58)GOTO 360
      GOTO 345            ! FIRST CHARACTER WAS NOT A NUMBER

C------------------------------------------------
C     THIS CHECKS THE SECOND CHARACTER TO SEE IF IT IS THERE OR THAT
C     IT IS A CHARACTER FROM A - H DEPENDING ON THE NUMBER OF SAMPLES.
C------------------------------------------------

360   IF(CORRECT(2:2).EQ.' ')THEN          ! USER WANTS TO CHANGE AN
                                           ! ENTIRE GROUP
       READ(CORRECT(1:1),'(BN,I2)',ERR-345)NEW_INPUT

       IF(NEW_INPUT.LT.1.OR.NEW_INPUT.GT.8)GOTO 345

       CHANGE - 1
       IF(NEW_INPUT.LT.6)THEN              ! GO CHANGE SAMPLE DATA
        GOTO(40,60,68,110,199)NEW_INPUT    ! ITEMS 1 - 5

       ELSE                        ! STAGE MOVEMENT DATA

          IF(NEW_INPUT.EQ.6)THEN
           IPOS - 0
           ISEC - 39
```

```
         X   - 70              ! START POSITION OF X
         NUM  - 54              ! ASCII VALUE OF 6

        ELSEIF(NEW_INPUT.EQ.7)THEN
         IPOS - 1
         ISEG - 49
         X   - 700             ! START POSITION OF X
         NUM  - 55              ! ASCII VALUE OF 7


        ELSEIF(NEW_INPUT.EQ.8)THEN
         IPOS - 2
         ISEG - 59
         X   - 1400            ! START POSITION OF X
         NUM  - 56              ! ASCII VALUE OF 8

        ENDIF
      Y1 - Y3

      GOTO 275

      ENDIF
     ENDIF

C--------------------------------------------------
C    THIS PART IS ENTERED IF THE USER IS SELECTING A SPECIFIC ITEM
C    ON THE MENU TO CHANGE.  FIRST I GET THE INTEGER AND COMPARE IT
C    TO THE NUMBER OF SAMPLES.  A BAD ENTRY MEANS DO IT AGAIN.
C     THEN I TAKE THE SECOND ENTRY AND MAKE SURE ITS CORRECT AND
C    WITHIN BOUNDS.
C--------------------------------------------------

      READ(CORRECT(1:1),'(BN,I2)',ERR-345)NEW_INPUT   ! CNG TO INTEG

     IF(NEW_INPUT.LT.1.OR.NEW_INPUT.GT.8)GOTO 345 ! BAD INPUT

     LET - ICHAR(CORRECT(2:2))              ! CNG TO INTEGER

C--------------------------------------------------
C    THIS CHECKS FOR CASE. IF INPUT WAS LOWER CASE I MAKE IT UPPER
C    CASE HERE.   a - h is changed to upper case
C--------------------------------------------------

    IF(LET.GT.96.AND.LET.LT.105)THEN           ! ITS LOWER CASE
     LET - LET - 32                       ! MAKE LOWER CASE
    ENDIF

    IF(LET.LT.65.OR.LET.GT.64 + NOSAMP)GOTO 345    ! BAD INPUT

C--------------------------------------------------
C    HERE I SET GET THE SPECIFIC SEGMENT NUMBER I NEED AND
C    THEN I SET ISTART FOR THE CORRECT ARRAY VALUE.  I ARM A SINGLE
C    EVENT DATA FLAG " ITEM_CNG - 1 " AND I GO GET THE DATA.
C--------------------------------------------------
```

```
      ISTART  - LET - 64          ! GET ITEM NUMBER
      ISAME   - 0               ! FLAG FOR SINGLE INPUT
      ITEM_CNG - 1              ! FLAG TO JUMP OUT
      LETTER  - LET
     IF(NEW_INPUT.EQ.3)THEN            ! CORRECTING A SAMPLE ITEM
      ISEG    - 8 + ISTART          ! THIS IS THE SEGMENT No.
      Y1 - 2600 - (125 * (ISTART - 1))   ! Y VECTOR OF SEGMENT
      GOTO 69
C-----------------------------------------
C     AGENT DATA
C-----------------------------------------
     ELSEIF(NEW_INPUT.EQ.4)THEN          ! CORRECTING AN AGENT ITEM
      IF(AGENT(1).EQ.'NONE')GOTO 345     ! CANT PICK SPECIFIC
                                    ! NOTHING.
      ISEG    - 18 + ISTART          ! THIS IS THE SEGMENT No.
      Y1 - 2600 - (125 * (ISTART - 1))   ! Y VECTOR OF SEGMENT
      X - 1500
      GOTO 138
C-----------------------------------------
C     CONCENTRATION DATA
C-----------------------------------------
     ELSEIF(NEW_INPUT.EQ.5)THEN          ! CORRECTING AN AGENT ITEM
      IF(AGENT(1).EQ.'NONE')GOTO 345     ! CANT PICK SPECIFIC
                                    ! NOTHING.
      ISEG    - 28 + ISTART          ! THIS IS THE SEGMENT No.
      Y1 - 2600 - (125 * (ISTART - 1))   ! Y VECTOR OF SEGMENT
      X - 2950
      GOTO 215
C-----------------------------------------
C     SAMPLE STAGE START POSITION
C-----------------------------------------
     ELSEIF(NEW_INPUT.EQ.6)THEN          ! CORRECTING START POS
      IPOS - 0
      ISEG    - 38 + ISTART          ! THIS IS THE SEGMENT No.
      Y1 - Y3 - (125 * (ISTART - 1))    ! Y VECTOR OF SEGMENT
      Y1 - Y1 - 200
      X - 70
      NUM   - 54                ! ASCII VALUE OF 6
     GOTO 278
```

```
C————————————————————————————————
C     SAMPLE STAGE STOP POSITION
C————————————————————————————————

      ELSEIF(NEW_INPUT.EQ.7)THEN        ! CORRECTING STOP POS
        IPOS - 1
        ISEG   - 48 + ISTART            ! THIS IS THE SEGMENT No.
        Y1 - Y3 - (125 * (ISTART - 1))  ! Y VECTOR OF SEGMENT
        X - 700
        Y1 - Y1 - 200
        NUM   - 55                      ! ASCII VALUE OF 7
      GOTO 278

C————————————————————————————————
C     SAMPLE STAGE INCREMENT
C————————————————————————————————

      ELSEIF(NEW_INPUT.EQ.8)THEN        ! CORRECTING INCREMENT
        IPOS   - 2
        ISEG   - 58 + ISTART            ! THIS IS THE SEGMENT No.
        Y1     - Y3 - (125 * (ISTART - 1))  ! Y VECTOR OF SEGMENT
        X      - 1400
        Y1     - Y1 - 200
        NUM   - 56                      ! ASCII VALUE OF 8
      GOTO 278


      ENDIF

1000  RETURN
      END
```

AIV.25 Analog APSD Software Modules: TEK_TEXT Source Code.

```
      SUBROUTINE TEK_TEXT(TXT_FLG,PORT,RESET,ANS,ANALYTE,
     .      AMOUNT,IEXIT)

C------------------------------------------
C   THIS ROUTINE IS CALLED JUST TO WRITE GRAPHIC TEXT FOR THE USER
C
C   TXT_FLG - 1
C    FIRST THE PORT SETTINGS ARE DEFINED AND THE USER IS REQUESTED
C    TO ENTER THE PORT UPON WHICH ALL COMMUNICATIONS WILL TAKE PLACE.
C
C   TXT_FLG - 2
C    HERE THE USER IS ASKED IF THE REAL TIME GRAPHICS WILL BEUSED
C    OR A/D VOLTAGES DISPLAYED OR IF NO OUTPUT AT ALL IS DESIRED.
C
C   TXT_FLG - 3
C    THIS PART IS USED TO TELL THE USER TO APPLY AGENT TO THE
C    SAMPLE AND HOW MUCH.
C
C   TXT_FLG - 4
C    THIS TELLS THE USER THAT A SPECIFIC LASER IS READY TO BE
C    CALCULATED.
C
C   PORT - THE SERIAL COMMUNICATIONS PORT SUPPLIED BY THE USER
C        SENT BACK TO THE CALLER
C
C   RESET - O          ! SENT FROM THE CALLER
C    DRAW THE GRAPHICS PANEL FOR TEXT
C
C   RESET - 1
C    DONT DO ANYTHING TO THE PANEL DRAW/DELETE
C
C   RESET - 2
C    DELETE THE PANEL AND TEXT
C
C   ANS  - CHARACTER NUMBER 1,2 OR 3.  FOR THE TYPE OF OUTPUT THE
C        USER WOULD LIKE TO HAVE FOR THIS EXPERIMENT.
C        SENT BACK TO THE CALLER
C
C   ANALYTE - THE CHEMICAL THAT IS TO BE APPLIED TO THE CURRENT
C         SAMPLE.  SENT IN FROM CALLER.
C
C   AMOUNT  - THE AMOUNT OF THE CHEMICAL THAT IS TO BE USED ON THE
C         CURRENT SAMPLE.  SENT IN FROM THE CALLER
C   IEXIT - FLAG THAT USER WANTS TO EXIT THE PROGRAM
C------------------------------------------

      CHARACTER E,SEG*3,A*5,A1*5,TEXT*80,PORT*10,ANS*1,FINISHED*1
      CHARACTER ANALYTE*20,AMOUNT*15

      INTEGER X,Y,TXT_FLG,RESET

```

```
      E - CHAR(27)


     WRITE(*,*)E//'LV0'
     WRITE(*,*)E//'LZ'


     ISEG - 900
     CALL INTRPT(ISEG,SEG)


     IF(RESET.EQ.2)THEN              ! CLEAR THE WHOLE VIEW

      WRITE(*,*)E//'SK'//SEG         ! DELETE SEGMENT 900
      ISEG - 901
      CALL INTRPT(ISEG,SEG)
      WRITE(*,*)E//'SK'//SEG         ! DELETE SEGMENT 900
      WRITE(*,*)E//'KN0'             ! RENEW THE VIEW
      GOTO 1010

     ELSEIF(RESET.EQ.0)THEN

      WRITE(*,*)E//'SE'//SEG         ! BEGIN SEGMENT 900
                                   ! USER WANTS A RED PANEL
      WRITE(*,*)E//'MP"'             ! PANEL COLOR RED
      WRITE(*,*)E//'MT7'             ! TEXT COLOR YELLOW
      WRITE(*,*)E//'ML1'             ! LINE COLOR WHITE
      X - 500
      Y - 2800
      CALL HTY(X,Y,A)
      WRITE(*,*)E//'LP'//A//'1'

      X - X + 3095
      CALL HTY(X,Y,A)
      WRITE(*,*)E//'LG'//A

      Y - Y - 2000
      CALL HTY(X,Y,A)
      WRITE(*,*)E//'LG'//A

      X - X - 3095
      CALL HTY(X,Y,A)
      WRITE(*,*)E//'LG'//A

      WRITE(*,*)E//'SC'

    ENDIF

C-----------------------------------------
C    THIS DRAWS THE TEXT IN THE PANEL
C-----------------------------------------

    ISEG - 901

```

```
  CALL INTRPT(ISEG,SEG)
WRITE(*,*)E//'SK'//SEG          ! DELETE SEGMENT 901
  WRITE(*,*)E//'KN0'           ! RENEW THE VIEW
WRITE(*,*)E//'SE'//SEG          ! BEGIN SEGMENT 901

IF(TXT_FLG.EQ.1)THEN               ! SET PORT PARAMETERS

X - 1000
Y - 2500
CALL HTY(X,Y,A)
WRITE(*,*)E//'LF'//A
TEXT - 'THE COMMUNICATIONS PORT MUST BE DEFINED WITH'
WRITE(*,*)E//'LTD0'//TEXT
Y - Y - 150
CALL HTY(X,Y,A)
WRITE(*,*)E//'LF'//A

TEXT - 'THE FOLLOWING PARAMETERS:'
WRITE(*,*)E//'LTD0'//TEXT

X - X + 200
Y - Y - 300
CALL HTY(X,Y,A)
WRITE(*,*)E//'LF'//A

TEXT - '9600  BAUD        NO  PARITY'
WRITE(*,*)E//'LTD0'//TEXT
Y - Y - 150
CALL HTY(X,Y,A)
WRITE(*,*)E//'LF'//A

TEXT - '8    BITS         1 STOP BIT'
WRITE(*,*)E//'LTD0'//TEXT

Y - Y - 150
CALL HTY(X,Y,A)
WRITE(*,*)E//'LF'//A
TEXT - 'NO ECHO           NO LOCAL ECHO'
WRITE(*,*)E//'LTD0'//TEXT

Y - Y - 150
CALL HTY(X,Y,A)
WRITE(*,*)E//'LF'//A
TEXT - 'PASSALL MODE '
WRITE(*,*)E//'LTD0'//TEXT

Y - Y - 300
CALL HTY(X,Y,A)
WRITE(*,*)E//'LF'//A
TEXT - 'PRESS RETURN FOR DEFAULT PORT <  TXA2: >'
WRITE(*,*)E//'LTD0'//TEXT

Y - Y - 350
CALL HTY(X,Y,A)
```

```
      WRITE(*,*)E//'LP'//A
      TEXT - 'ENTER THE SERIAL PORT NAME: '
      WRITE(*,*)E//'LTD0'//TEXT

      WRITE(*,*)E//'SC'

      WRITE(*,*)E//'LI722'        ! TEXT YELLOW  ON RED
      WRITE(*,*)E//'LL2'          ! DIALOG AREA 2 LINES
      WRITE(*,*)E//'LC:'          ! DIALOG AREA 10 CHARACTERS LONG

      X - 2650
      Y - Y - 90
      CALL HIY(X,Y,A)
      WRITE(*,*)E//'LX'//A

      WRITE(*,*)E//'LV1'
 50   WRITE(*,*)E//'LZ'
      READ(*,'(A5)',ERR - 50)PORT

C-----------------------------------------
C     CHECK FOR PORT DEFINITION ERRORS
C-----------------------------------------

      IF(PORT(1:1).EQ.CHAR(116))PORT(1:1) - 'T'
      IF(PORT(2:2).EQ.CHAR(120))PORT(2:2) - 'X'
      IF(PORT(3:3).EQ.CHAR(97))PORT(3:3)  - 'A'

      IF(PORT.EQ.' ')THEN           ! USER WANTS DEFAULT
       PORT - 'TXA2:'              ! SERIAL PORT
       GOTO 1000
      ENDIF

      IF(PORT(1:3).EQ.'TXA')THEN      ! THESE ARE THE SERIAL
                                   ! PORTS IM USING

      DO 60 I - 1,8                ! LOOP THRU 0 - 7
       IF(PORT(4:4).EQ.CHAR(47 + I))GOTO 70
 60   CONTINUE

      GOTO 50                      ! INPUT WAS BAD DO IT AGAIN
 70   PORT(5:5) - ':'              ! MAKE SURE YOU END WITH A COLON

      ELSE
       GOTO 50                     ! BAD INPUT
      ENDIF

C-----------------------------------------
C     HERE THE USER IS ASKED TO SELECT THE TYPE OF OUT PUT DESIRED
C-----------------------------------------

      ELSEIF(TXT_FLG.EQ.2)THEN

      X - 1000
      Y - 2500
```

```
       CALL HIY(X,Y,A)
      WRITE(*,*)E//'LP'//A
      TEXT = 'ENTER THE TYPE OF OUTPUT DESIRED:'
      WRITE(*,*)E//'LTD0'//TEXT

      Y = Y - 400
      CALL HIY(X,Y,A)
      WRITE(*,*)E//'LP'//A
      TEXT = '1.  REAL TIME TEKTRONIX GRAPHICS DISPLAY'
      WRITE(*,*)E//'LTD0'//TEXT

      Y = Y - 150
      CALL HIY(X,Y,A)
      WRITE(*,*)E//'LP'//A
      TEXT = '2.  REAL TIME A/D CHANNEL VOLTAGE OUTPUTS'
      WRITE(*,*)E//'LTD0'//TEXT


      Y = Y - 150
      CALL HIY(X,Y,A)
      WRITE(*,*)E//'LP'//A
      TEXT = '3.  NO DISPLAY OF DATA'
      WRITE(*,*)E//'LTD0'//TEXT

      WRITE(*,*)E//'SC'              ! END THE SEGMENT

      WRITE(*,*)E//'LI722'           ! TEXT YELLOW  ON RED
      WRITE(*,*)E//'LL2'             ! DIALOG AREA 2 LINES
      WRITE(*,*)E//'LC:'             ! DIALOG AREA 10 CHARACTERS LONG

      X = 2000
      Y = Y - 300
      CALL HIY(X,Y,A)
      WRITE(*,*)E//'LX'//A

      WRITE(*,*)E//'LV1'
 80    WRITE(*,*)E//'LZ'

      READ(*,'(A)',ERR=80)ANS

      IF(ANS.EQ.'1'.OR.ANS.EQ.'2'.OR.ANS.EQ.'3')GOTO 1000
      GOTO 80

C--------------------------------------------------
C     THIS TELLS THE USER TO PUT THE ANALYTE ON THE SAMPLE.  THE SAMPLE
C     HAS BEEN MOVED TO THE CORRECT POSITION TO ADD THE CHEMICAL
C--------------------------------------------------

      ELSEIF(TXT_FLG.EQ.3)THEN

      X = 1000
      Y = 2000
      CALL HIY(X,Y,A)
      WRITE(*,*)E//'LP'//A
```

```
      TEXT = 'THE SAMPLE IS NOW READY FOR THE '// ANALYTE

     WRITE(*,*)E//'LTD0'//TEXT

     Y = Y - 200
     CALL HIY(X,Y,A)
     WRITE(*,*)E//'LP'//A
     TEXT = 'PLEASE APPLY: '//AMOUNT//' TO THE SAMPLE'
     WRITE(*,*)E//'LTD0'//TEXT

     X = X + 650
     Y = Y - 300
     CALL HIY(X,Y,A)
     WRITE(*,*)E//'LP'//A
     TEXT = 'PRESS TO GO ON.'
     WRITE(*,*)E//'LTD0'//TEXT

     WRITE(*,*)E//'SC'              ! END THE SEGMENT

     READ(*,'(A)',ERR=1000)FINISHED
     READ(*,'(A)',ERR=1000)FINISHED

C-----------------------------------------

     ELSEIF(TXT_FLG.EQ.4)THEN

     X = 1200
     Y = 2300
     CALL HIY(X,Y,A)
     WRITE(*,*)E//'LP'//A

     TEXT = 'TUNE LASER No. '//ANS//' TO WAVELENGTH: '//ANALYTE
     WRITE(*,*)E//'LTD0'//TEXT

     X = 850
     Y = Y - 800
     CALL HIY(X,Y,A)
     WRITE(*,*)E//'LP'//A
     TEXT = ' PRESS RETURN WHEN THE CALIBRATION IS COMPLETE'
     WRITE(*,*)E//'LTD0'//TEXT

     WRITE(*,*)E//'SC'              ! END THE SEGMENT

     READ(*,'(A)',ERR=1000)FINISHED

     ELSE
      IEXIT = 0
     ENDIF

1000 WRITE(*,*)E//'LV0'          ! DISABLE DIALOG AREA
     WRITE(*,*)E//'LZ'          ! CLEAR DIALOG AREA
     WRITE(*,*)E//'LI144'       ! TEXT WHITE ON BLUE
     WRITE(*,*)E//'LL2'         ! DIALOG AREA 2 LINES
     X = 0
```

```
        Y - 0
      CALL HTY(X,Y,A)
      WRITE(*,*)E//'LX'//A          ! RESET DIALOG AREA POSITION
      WRITE(*,*)E//'LC80'           ! DIALOG AREA 80 CHARACTERS LONG
 1010  RESET - 0                    ! RESET THE PANEL DEL/DRAW FLG
      RETURN
      END
```

AIV.26 Analog APSD Software Modules: TERM_INFO Source Code.

```
      SUBROUTINE TERM_INFO(ITERM,NUM_PLANES,TOT_MEM,FREE_MEM,
     . IVERSION,OPT_NUM,OPT_INFO)

C-----------------------------------------------
C THIS ROUTINE WILL POLL THE TERMINAL THAT YOU ARE LOGGED IN ON TO SEE
C WHAT TEKTRONIX MODEL # IT IS (AND IF IT IS A TEK TERMINAL AT ALL), AND
C ALSO WILL TRY TO DETERMINE HOW MANY BIT PLANES THE TERMINAL HAS
C INSTALLED,
C THE TOTAL MEMORY INSTALLED, THE AVAILABLE MEMORY, AND THE FIRMWARE
C VERSION
C NUMBER FOR THE TERMINAL (AND ALSO THE FIRMWARE VERSION NUMBER FOR ONE
C ADDITIONAL OPTION, IF REQUESTED).
C-----------------------------------------------
C PARAMETERS:
C
C IUT         - THE LOGICAL UNIT # OF THE TERMINAL FOR I/O PURPOSES
C ITERM       - THE INTEGER CONTAINING THE TERM TYPE (I.E. '4125')
C NUM_PLANES  - THE ACTUAL # OF BIT PLANES IN THE TERMINAL. (RETURNS
C               NUM_PLANES=-1 IF CANNOT DETERMINE THE # PRESENT)
C TOT_MEM     - TOTAL AMOUNT OF MEMORY INSTALLED IN THE TERMINAL(K BYTES)
C FREE_MEM    - MEMORY PRESENTLY AVAILABLE FOR USE (K BYTES)
C IVERSION    - VERSION NUMBER OF THE STANDARD FIRMWARE IN THE TERMINAL
C OPT_NUM     - AN OPTIONAL FIRMWARE VERSION NUMBER REQUEST(0=NO REQUEST)
C OPT_INFO    - THE RETURNED VALUE FROM 'OPT_NUM' CALL ABOVE
C-----------------------------------------------
C
      CHARACTER E*1,VERSION*3,OPT_NUM*2,OPT_INFO*3
      CHARACTER A*20,A1*2,ATERM*3,B*1
      INTEGER TOT_MEM,FREE_MEM,OPT22,OPT23
C
C-----------------------------------------------
C
      E=CHAR(27)                  ! ESCAPE CHARACTER

C THE FOLLOWING WILL DETERMINE IF THE TERM IS A TEK GRAPHICS TERMINAL

      WRITE(*,*)E//'%!0'          ! PUT TERMINAL INTO TEK MODE
      WRITE(*,*)E//'LV0'          ! MAKE DIALOG AREA INVISIBLE
      WRITE(*,*)E//'IQ?T'         ! GET THE TERMINAL TYPE

 5    READ(*,1,ERR=5)A

      IF(A.EQ.' ')GOTO 100         ! USER HIT <CR>, SO NOT A TEK TERM.
      IF(A(1:2).NE.'?T')GOTO 100   ! INCORRECT TERMINAL RESPONSE

      IADD = ICHAR(A(3:3)) + 32
      IADD1 = ICHAR(A(4:4)) + 32

      ATERM=A(3:5)

      CALL DECODE(ATERM,ITERM)    ! CONVERT TEK CHAR CODE TO INTEGER
```

```
      IF(ITERM.EQ.4111)THEN          ! — FOR 4111 TERMINAL ONLY —

       WRITE(*,*)E//'IQ?P'           ! TOTAL MEMORY(# OF 16 BYTE BLOCKS)
10      READ(*,1,ERR=10)A             ! (4111 OPT 2C, PAGED MEMORY)


       ATERM=A(3:5)

       CALL DECODE(ATERM,TOT_MEM)     ! CONVERT TEK CHAR CODE TO INTEGER


       ATERM=A(6:8)
       CALL DECODE(ATERM,FREE_MEM)    ! CONVERT TEK CHAR CODE TO INTEGER

      ELSE                  ! — ALL OTHER TERMINALS —

       WRITE(*,*)E//'IQ?M'

20      READ(*,1,ERR=20)A

       ATERM=A(3:5)

       CALL DECODE(ATERM,TOT_MEM)     ! CONVERT TEK CHAR CODE TO INTEGER

       ATERM=A(6:8)
       CALL DECODE(ATERM,FREE_MEM)    ! CONVERT TEK CHAR CODE TO INTEGER

                            ! IN KB
      END IF

      WRITE(*,*)E//'IQ00'           ! GET THE STD FIRMWARE VERSION
      READ(*,1,ERR=30)A

30     VERSION=A(3:5)

      CALL DECODE(VERSION,IVERSION)  ! CONVERT TEK CHAR CODE TO INTEGER


      IF(ITERM.EQ.4115.OR.ITERM.EQ.4129.OR.ITERM.EQ.4128
     . .OR.ITERM.EQ.4225.OR.ITERM.EQ.4236)THEN

       WRITE(*,*)E//'IQ22'          ! GET THE OPT22 INFO
40      READ(*,1,ERR=40)A

       CALL DECODE(A(3:5),OPT22)    ! CONVERT TEK CHAR CODE TO INTEGER

       WRITE(*,*)E//'IQ23'          ! GET THE OPT23 INFO
50      READ(*,1,ERR=50)A

       CALL DECODE(A(3:5),OPT23)    ! CONVERT TEK CHAR CODE TO INTEGER

```

```
         IF(OPT22.EQ.0.AND.OPT23.EQ.1)THEN
           NUM_PLANES=8
         ELSE
           NUM_PLANES=4
         END IF

      ELSE
        NUM_PLANES=4
      END IF

      GOTO 1000                ! DONE, SO EXIT

100   CONTINUE                 ! NOT A TEK TERMINAL, SO EXIT

      ITERM = 0
      NUM_PLANES = -1

C------------------------------------------

1     FORMAT(A20)

1000  CONTINUE
      RETURN
      END
```

AIV.27 Analog APSD Software Modules: TIME4 Source Code.

```
      SUBROUTINE TIME4(DBUFF,TBUFF)

C-----------------------------------
C SUBROUTINE TO GET AN ASCII STRING WITH THE DATE AND TIME
C-----------------------------------

      CHARACTER*9 DBUFF
      CHARACTER*8 TBUFF

      CALL DATE(DBUFF)
      CALL TIME(TBUFF)

      RETURN
      END
```

AIV.28 Analog APSD Software Modules: TWAIT Source Code.

```
      SUBROUTINE TWAIT(IWTIM)
C--------------------------------------------
C SUBROUTINE TO WAIT FOR 'IWTIM' TENTHS OF A SECOND AND THEN RETURN
C--------------------------------------------
C ARNE A. JOHNSON, 21 APRIL 88
C--------------------------------------------
      REAL TIME1,FWTIM,TMIN
      MM-1
      IFLAG-0
C--------------------------------------------
C TIME DELAY LOOP (FOR 'IWTIM' TENTHS OF A SECOND)
C
      IF(IWTIM.EQ.0)GOTO 1000


100   CONTINUE
      IFLAG-IFLAG+1
      TIME1-SECNDS(0.0)
      FWTIM-FLOAT(IWTIM)
      TMIN-FWTIM/10.

50    CONTINUE

      DO 55 M-1,20          ! MEANINGLESS CALC. TO TAKE UP TIME
       MM-M*10+MM
55    CONTINUE

      DELTA-SECNDS(TIME1)
C     WRITE(*,*)DELTA
      IF (DELTA.LT.TMIN) GO TO 50
c     CLOSE(1)

1000  RETURN
      END


$
      SUBROUTINE TWAIT(IWTIM)
C--------------------------------------------
C SUBROUTINE TO WAIT FOR 'IWTIM' TENTHS OF A SECOND AND THEN RETURN
C--------------------------------------------
      REAL TIME1,FWTIM,TMIN
      MM-1
      IFLAG-0
C--------------------------------------------
C TIME DELAY LOOP (FOR 'IWTIM' TENTHS OF A SECOND)
C
      IF(IWTIM.EQ.0)GOTO 1000


100   CONTINUE
```

```
        IFLAG-IFLAG+1
      TIME1-SECNDS(0.0)
      FWTIM-FLOAT(IWTIM)
      TMIN-FWTIM/10.

 50   CONTINUE

      DO 55 M-1,20          ! MEANINGLESS CALC. TO TAKE UP TIME
       MM-M*10+MM
 55   CONTINUE

      DELTA-SECNDS(TIME1)
C      WRITE(*,*)DELTA
      IF (DELTA.LT.TMIN) GO TO 50
c     CLOSE(1)

1000  RETURN
      END
```

AIV.29 Analog APSD Software Modules: V45 Source Code.

```
      SUBROUTINE V45(ISHOTS,INUSE,ICNT,IS,SAMP,RINCR,RARM,REND,
     .ANGLE,STP)

C-----------------------------------------------
C     THIS MOD CAPTURES THE DATA FOR THE POLARIZERS IN POSITION
C     VERTICAL/45 deg. TWO ARRAYS ARE USED TO SAVE THE DATA.
C     BRAY - IS A SEQUENTIAL ARRAY STORINING ALL THE DATA RECEIVED FROM
C     THE A/D CONVERTER THRU THE ENTIRE ROTATION OF THE SAMPLE.
C
C     ERAY - IS A REPRESENTATION OF THE ENTIRE MATRIX. NINE ELEMENTS
C     WILL BE PASSED IN FROM THE A/D CONVERTER OF WHICH ONLY 3 WILL
C     BE NEW. THEY ARE : (9,10,12) FROM THE 1,2,4,9,10,12,13,14,16.
C     GAPS WILL BE LEFT IN THE ARRAY TO BE FILLED IN BY THE OTHER
C     TWO POSSIBLE POLARIZER SETTINGS.
C-----------------------------------------------


      CHARACTER SAMP*20,MSG*50,MESG*255,DIST*10,HEX*4
      CHARACTER CHAN*2,PORT*10

      REAL RDAT,ARAY,BRAY,CRAY,DRAY,ERAY,RINCR,REND,ANGLE,RARM

      INTEGER STP,ICNG,REPORT,TYPE,GETDAT

      DIMENSION GETDAT(16),RDAT(16),J1(3),HEX(16)

      COMMON /MATRIX/ARAY(3600),BRAY(3600),CRAY(3600),DRAY(3600),
     . ERAY(5800)

      DATA J1 /9,10,12/          ! NEW MATRIX ARRAY ELEMENTS

      ISWEEP = 2                 ! FLAG THAT THIS IS 2ND SWEEP
                                 ! OF THE POLARIZERS
C-----------------------------------------------
C     THIS MOVES THE POLARIZERS TO THE PROPER POSITION:
C     RECEIVER POLARIZER( AXIS 3 ) MOVES -225000 STEPS OR -45 DEGREES
C-----------------------------------------------

      WRITE(3,'(A4)')'1X0 '      ! RESET CUMM POSITION COUNTER
      CALL TWAIT(1)

      WRITE(3,'(A38)')'3PS 3ST1 3V10 3A10 3D-225000 3G 3CR C '

      READ(3,'(A50)')MSG(1:50)

      CALL TWAIT(1)

      WRITE(3,'(A5)')'3ST0 '
      CALL TWAIT(1)
      WRITE(3,'(A13)')'1PS 1ST1 1H '

```

```
        INUSE1 - 0
C------------------------------------------
C    HERE BEGINS THE LOOP WHERE SAMPLE DATA IS TAKEN AND STORED
C------------------------------------------

      DO 200 M - 1,ISHOTS          ! LOOP THRU THE SAMPLE ROTATION

C------------------------------------------
C    HERE IS WHERE THE A/D CONVERTER IS ASKED FOR THE DATA
C------------------------------------------


C    CALL TESTDAT(ISWEEP,ANGLE,RDAT)

      IC1 - 11
      TYPE - 3
      CALL DATEL(TYPE,REPORT,PORT,IC1,CHAN,RDAT,HEX)

C------------------------------------------
C    HERE EACH ARRAY IS SELECTED, WRITTEN TO AND INCREMENTED
C------------------------------------------

        DO 100 I - 10,1,-1
         BRAY(IS) - RDAT(I)        ! STORE THE 10 ARRAY ELEMENTS
         IS - IS - 1               ! ARAY SEQUENTIAL COUNTER
100      CONTINUE


C------------------------------------------
C    HERE ONLY THREE ELEMENTS OF THE ERAY MATRIX ARE NEW.  THEY ARE
C    ELEMENTS 9,10,12. THEY ARE STORED IN THE RDAT ARRAY IN POSITIONS
C    4,5,6 RESPECTIVELY.  I USE I1 TO SET THE CORRECT SEQUENCE.
C------------------------------------------

        I1 - 3                     ! START PT IN RDAT ARRAY
        DO 150 I - 1,3             ! LOOP THRU THE 3 NEW ELEMENTS
         K - J1(I) + ICNT          ! SELECT CORRECT ARRAY ELEMENT
         ERAY(K)  - RDAT(I + I1)   ! PLACE MATRIX DATA IN ERAY
150      CONTINUE

C------------------------------------------
C    THIS OUTPUTS THE DATA EITHER BY GRAPHICS OR A/D VOLTAGES OR NONE
C    AT ALL DEPENDING ON THE FLAG STP.
C------------------------------------------

        IF(STP.EQ.1)THEN           ! REAL TIME TEK GRAPHICS

         CALL DRAW_ELE(ISWEEP,ANGLE,RARM,RINCR,REND,RDAT,INUSE1)
         GOTO 160
        ENDIF

        IF(STP.EQ.3)GOTO 160       ! NO OUTPUT

        CALL VIEW(SAMP,ANGLE,IS,STP,IDIR)
```

```
160    IF(M.EQ.ISHOTS)GOTO 200
       ANGLE - ANGLE - RINCR     ! NEW SAMPLE ANGLE
       ICNT - ICNT - 16          ! DECREMENT THE ARRAY BY 16
C------------------------------------------------
       WRITE(3,'(A9)')'1G 1X1 C '
       READ(3,'(A50)',ERR - 200)MESG(1:50)
       CALL TWAIT(1)
       READ(MESG(14:22),'(BN,I9)',ERR - 200)MOTION
       ICNG - MOTION - MBUF
       MBUF - MOTION
170    FORMAT(5X,I4,10X,' ACCUMULATED MOTION: ',I9,
     .' RELATIVE MOTION: ',I9//)
D       WRITE(*,170)M,MOTION,ICNG
       WRITE(3,'(A4)')'1PS '
       CALL TWAIT(1)
200    CONTINUE                  ! LOOP THRU ROTATIONS
C------------------------------------------------
C      AT THIS POINT THE DATA COLLECTION FOR POLARIZERS POSITIONED AT
C      VERTICAL/45 deg IS COMPLETE.  WE NOW RETURN TO THE CALLER
C      WHERE THE NEXT POLARIZER SETTING WILL BE MADE AND THE SAMPLE WILL
C      BE RETURNED TO THE START POSITION.
C------------------------------------------------
       WRITE(3,'(A5)')'1ST0 '    ! DEENERGIZE SAMPLE MOTOR
400    RETURN
       END
```

AIV.30 Analog APSD Software Modules: VIEW Source Code.

```
      SUBROUTINE VIEW(SAMP,ANGLE,IS,STP,IDIR)

C------------------------------------------
C     THIS ROUTINE BRINGS ALL THE ARRAY DATA TO THE SCREEN FOR EACH
C     ANGLE THAT DATA IS TAKEN FOR.  THE REASON IS TO BE ABLE TO COMPARE
C     THE MATRIX DATA AS THRU THE FOUR POLARIZER CHANGES.
C------------------------------------------

      CHARACTER SAMP*20
      REAL A,B,C,D,E,F,POS
      INTEGER STP
      COMMON /MATRIX/A(1600),B(1600),C(1600),D(1600),E(3000)

10    FORMAT (13X,'ANGLE: ',F7.2,25X,'SAMPLE: ',A20,//)
20    FORMAT(14X,'V/V',7X,':',7X,'V/45',6X,':',7X,'45/V',6X,
     .':',6X,'45/45')
30    FORMAT(80('-'))
40    FORMAT(4X,I4,4X,F7.5,3(5X,':',5X,F7.5))
60    FORMAT(10X,'RETURN - CONT, < 99 > - DONT PAUSE,
     .< 100 > - NO VIEW')
      F - 0.00000

C------------------------------------------
C     THE DATA SENT TO THE SCREEN IS DEPENDENT ON THE DIRECTION OF THE
C     SCAN OF THE LASER.  IF THE SAMPLE IS GOING FROM LEFT TO RIGHT THEN
C     I SUBTRACT 9 FROM THE DATA ARRAY COUNTER.  OTHERWISE ITS OK THE
C     WAY IT IS.
C------------------------------------------

      IF(IDIR.EQ.1)IS - IS - 9

C------------------------------------------
C     HERE THE ENTIRE SCREEN IS WRITTEN TOO WITH FOUR FIELDS SERVING
C     AS THE FOUR ARRAYS.  IS - THE START OF THE MATRIX FOR THE
C     ANGLE SHOWING WHICH INCLUDES 16 ARRAY FIELDS.
C------------------------------------------

50    WRITE(*,10)POS,SAMP
      WRITE(*,20)
      WRITE(*,30)

      WRITE(*,40)1,A(1 + IS),B(1 + IS),C(1 + IS),D(1 + IS)
      WRITE(*,40)2,A(2 + IS),B(2 + IS),F,F
      WRITE(*,40)3,F,F,C(2 + IS),D(2 + IS)
      WRITE(*,40)4,A(3 + IS),B(3 + IS),C(3 + IS),D(3 + IS)
      WRITE(*,40)5,A(4 + IS),F,C(4 + IS),F
      WRITE(*,40)6,A(5 + IS),F,F,F
      WRITE(*,40)7,F,F,C(5 + IS),F
      WRITE(*,40)8,A(6 + IS),F,C(6 + IS),F
      WRITE(*,40)9,F,B(4 + IS),F,D(4 + IS)
      WRITE(*,40)10,F,B(5 + IS),F,F
```

```
        WRITE(*,40)11,F,F,F,D(5 + IS)
      WRITE(*,40)12,F,B(6 + IS),F,D(6 + IS)
      WRITE(*,40)13,A(7 + IS),B(7 + IS),C(7 + IS),D(7 + IS)
      WRITE(*,40)14,A(8 + IS),B(8 + IS),F,F
      WRITE(*,40)15,F,F,C(8 + IS),D(8 + IS)
      WRITE(*,40)16,A(9 + IS),B(9 + IS),C(9 + IS),D(9 + IS)

C-----------------------------------------------
C     THAT COMPLETES THE MATRIX VIEW.  HERE IM ADDING A READ STMT SO
C     THE USER CAN OPT TO SEE HOW THE DATA IS COMMING IN.  IF < 99 >
C     IS TYPED THIS READ WILL BE SKIPPED.
C     STP IS THE STOP FLAG.  AS LONG AS ITS 2 THEN THE DATA WILL BE
C     WRITTEN EVERY TIME.  IF < 99 > IS ISSUED THEN THE DATA WILL BE
C     WRITTEN BUT THE STP WONT STOP.  IF < 100 > IS ISSUED THEN STP
C     WILL NOTIFY THE CALLER NOT TO CALL THIS SUBROUTINE.  I'M LEAVING
C     AN OPTION FOR STP TO BE RESET TO ZERO BY THE CALLER(1st LINE BELOW)
C-----------------------------------------------

      IF(STP.EQ.2)INUSE = 0
      IF(INUSE.EQ.0)THEN
       WRITE(*,60)
       READ(*,'(A5)')STP
       IF(STP.EQ.99)INUSE = 1
      ENDIF


      IF(IDIR.EQ.1)IS = IS + 9
      RETURN
      END
```

AIV.31 Analog APSD Software Modules: VV Source Code.

```
      SUBROUTINE VV(ISHOTS,INUSE,ICNT,IS,SAMP,RINCR,RARM,REND,
     .ANGLE,STP,PORT)
C------------------------------------------
C     THIS MOD CAPTURES THE DATA FOR THE POLARIZERS IN POSITION
C     VERTICAL/VERTICAL. TWO ARRAYS ARE USED TO SAVE THE DATA.
C     ARAY - IS A SEQUENTIAL ARRAY STORINING ALL THE DATA RECEIVED FROM
C     THE A/D CONVERTER THRU THE ENTIRE ROTATION OF THE SAMPLE.
C
C     ERAY - IS A REPRESENTATION OF THE ENTIRE MATRIX. ONLY NINE
C     ELEMENTS WILL BE FILLED IN THIS ARRAY WITH THE OPTICS IN THIS
C     POSITION FOR EACH LASER SHOT. THEY ARE : 1,2,4,5,6,8,13,14,16.
C     GAPS WILL BE LEFT IN THE ARRAY TO BE FILLED IN BY THE OTHER
C     THREE POSSIBLE POLARIZER SETTINGS.
C     FEWER ELEMENTS WILL BE COLLECTED ON SUCCEDING SWEEPS OF THE
C     GONIOMETER ARM.
C------------------------------------------

      CHARACTER STR*9,GETDAT*4,CR,SAMP*20,A*80,DIST*10
      CHARACTER MESG*255,MSG*80,PORT*10,HEX*4
      CHARACTER CHAN*2,CHANNEL*2

      REAL RDAT,ARAY,BRAY,CRAY,DRAY,ERAY,RARM,ANGLE,RINCR,INC,REND

      INTEGER POS,STP,ISIZE,IOUT,REPORT,TYPE
      INTEGER*4 LENGTH,TIMEOUT,ICNG

      DIMENSION GETDAT(16),J(9),RDAT(16),HEX(16)

      COMMON /MATRIX/ARAY(3600),BRAY(3600),CRAY(3600),DRAY(3600),
     . ERAY(5800)

      DATA J /1,2,4,5,6,8,13,14,16/  ! MATRIX ARRAY ELEMENTS

      ISWEEP - 1              ! FLAG THAT THIS IS 1ST SWEEP
      ANGLE - RARM              ! ANGLE - START POSITION OF SAMP

C------------------------------------------
C     INUSE IS PASSED IN AS ZERO EVERY TIME A NEW LASER IS ENABLED
C     SO THAT ALL THE COUNTERS OF THE ARRAYS ARE CORRECT
C------------------------------------------

      IF(INUSE.EQ.0)THEN          ! INITIALIZE VARIABLES
       INUSE - 1             ! FLAG NOT TO COME BACK
       ICNT  - 0             ! 16 ELEMENT ARRAY COUNTER
       IS  - 0              ! SEQUENTIAL ARAY COUNTER
      ENDIF

C------------------------------------------
```

```
      C    THIS INITIALIZES THE SAMPLE STAGE MOVEMENT OF VELOCITY,
      C    ACCELERATION,  AND DISTANCE.
      C-----------------------------------------------

           INC - RINCR * 2000.          ! THIS IS THE ANGLE INCREMENT
           INCR - IFIX(INC)
           NUMBER - 10

           DO 50 I - 1,10
             IF(INCR.LT.NUMBER)GOTO 60
             NUMBER - NUMBER * 10
      50   CONTINUE

      C-----------------------------------------------
      C    IM USING THIS TO CLEAR THE READ BUFFER
      C-----------------------------------------------

      C    PORT - 'TXA2:'
      60   LENGTH - 255

           TIMEOUT - 1

           CALL READ_QIO(PORT,MESG,LENGTH,TIMEOUT,ISIZE,IOUT,CHANNEL,IUS)

      C-----------------------------------------------

           WRITE(3,'(A4)')'1X0 '        ! RESET CUMM POSITION CNT

           CALL MIX(INCR,DIST)

           IF(INCR.LT.10000)THEN
            WRITE(3,'(A27)')'1PS 1ST1 1V10 1A10 1D-'//DIST(1:I)//' '
           ELSE
            WRITE(3,'(A28)')'1PS 1ST1 1V10 1A10 1D-'//DIST(1:I)//' '
           ENDIF

      C    WRITE(*,'(A23)')'1ST1 1V10 1A10 1D-'//DIST(1:I)//' '

           CALL TWAIT(30)

           INUSE1 - 2                   ! INITIALIZE TEK DRAW FLAG

      C-----------------------------------------------
      C    HERE BEGINS THE LOOP WHERE SAMPLE DATA IS TAKEN AND STORED
      C-----------------------------------------------

           DO 200 M - 1,ISHOTS          ! LOOP THRU THE SAMPLE ROTATION

      C-----------------------------------------------
      C    HERE IS WHERE THE A/D CONVERTER IS ASKED FOR THE DATA
      C-----------------------------------------------


      C    CALL TESTDAT(ISWEEP,ANGLE,RDAT)
```

```
      IC1 = 11
      TYPE = 3                 ! STROBE THE A/D BOARD
      CALL DATEL(TYPE,REPORT,PORT,IC1,CHAN,RDAT,HEX)
                               ! GET HEX DATA FROM A/D BOARD

C-----------------------------------------------
C     HERE EACH ARRAY IS SELECTED, WRITTEN TO AND INCREMENTED
C-----------------------------------------------

      DO 100 I=1,10
        IS = IS + 1            ! ARAY SEQUENTIAL COUNTER
        ARAY(IS) = RDAT(I)     ! STORE THE 10 ARRAY ELEMENTS
        K = J(I) + ICNT        ! SELECT CORRECT ARRAY ELEMENT
        IF(I.EQ.10)GOTO 100    ! DO NOT STORE THE LAST DC ELE.
        ERAY(K) = RDAT(I)      ! PLACE MATRIX DATA IN ERAY
100   CONTINUE

C-----------------------------------------------
C     THIS PART DISPLAYS OUTPUT IN EITHER A/D VOLTAGES OR GRAPHICS
C     THE CALLER WILL DEFINE STP AS 1 FOR GRAPHICS, 2 FOR A/D VOLTAGES
C     OR 3 NOT YET DEFINED.
C-----------------------------------------------

      IF(STP.EQ.1)THEN         ! USER WANTS REAL GRAPHICS

       CALL DRAW_ELE(ISWEEP,ANGLE,RARM,RINCR,REND,RDAT,INUSE1)
       GOTO 150

      ENDIF

      IF(STP.EQ.3.)GOTO 150    ! SKIP VIEW FLAG

        CALL VIEW(SAMP,ANGLE,IS,STP,IDIR)

C-----------------------------------------------


150     IF(M.EQ.ISHOTS)GOTO 200   ! DO NOT MOVE SAMPLE ON LAST SHOT

        ANGLE = ANGLE + RINCR     ! INCREMENT ANGLE OF SAMPLE

        ICNT = ICNT + 16          ! INCREMENT THE ARRAY BY 16

C-----------------------------------------------

        WRITE(3,'(A9)')'1G 1X1 C '

        READ(3,'(A50)',ERR=200)MESG(1:50)

D        WRITE(*,*)M,' MESG(1:50) = ',MESG(1:50)

        CALL TWAIT(1)

```

```
           READ(MESG(14:22),'(BN,I9)',ERR = 200)MOTION

         ICNG = MOTION - MBUP
         MBUP = MOTION

  160   FORMAT(5X,I4,10X,' ACCUMULATED MOTION: ',I9,
     .'  RELATIVE MOTION:  ',I9/)

  D       WRITE(*,160)M,MOTION,ICNG
         WRITE(3,'(A4)')'1PS '
         CALL TWAIT(1)

  C-----------------------------------------

  200   CONTINUE                    ! LOOP THRU ROTATIONS

  C-----------------------------------------
  C     AT THIS POINT THE DATA COLLECTION FOR POLARIZERS POSITIONED AT
  C     VERTICAL/VERTICAL IS COMPLETE.  WE NOW RETURN TO THE CALLER
  C     WHERE THE NEXT POLARIZER SETTING WILL BE MADE AND THE SAMPLE WILL
  C     BE ROTATED IN REVERSE.
  C-----------------------------------------

        WRITE(*,'(A5)')'1ST0 '       ! DEENERGIZE SAMPLE STAGE

  400   RETURN
        END
```

AIV.32 Analog APSD Software Modules: YHI Source Code.

```
      SUBROUTINE YHI(A,X,Y,BUTT)

C----------------------------------------
C    THIS MOD CHANGES PACKED TEK 4957 CODES TO THEIR X,Y EQUIV.
C    WHEN A GIN DEVICE IS ACTIVATED IT SENDS TO THE HOST A 6 OR 12
C    CHARACTER LENGTH STRING.  THE FIRST CHARACTER IS A NUMBER OR
C    CHARACTER REPRESENTING THE KEY OR BUTTON PRESSED.  AS A MOUSE HAS
C    THREE BUTTONS THE FIRST CHARACTER WOULD BE A 1, 2 OR 3.
C    THE NEXT 5 CHARACTERS ARE THE X,Y SCREEN COORDINATES.
C    THIS ROUTINE PARSES THE FIRST SIX CHARACTERS THE FIRST IS
C    RETURNED IN BUTT AND THE X,Y ARE RETURNED AS INTEGERS.
C         IT IS INTERRESTING TO NOTE THAT THE FIVE CHARACTER STRING
C    THAT REPRESENTS X,Y CANNOT BE USED DIRECTLY IN OTHER COMMANDS.
C    THE LOCATIONS WOULD NOT BE THE SAME.  SO THE RETURNED X,Y THEN
C    HAS TO BE PASSED BACK THRU HIY.  THUS THE NAME OF THIS ROUTINE
C    IS THE MIRROR OF HIY...YHI.
C    THE MULTIPLICATIONS ARE FOR BIT SHIFTING
C
C----------------------------------------

      CHARACTER *6 A,BUTT*1
      INTEGER X,Y


      BUTT-A(1:1)    ! THIS IS USUALLY THE THE BUTTON NUMBER
      X-(ICHAR(A(5:5))-32) * 128 + (ICHAR(A(6:6))-32)*4
     $  + MOD(ICHAR(A(3:3)),4)
      Y-(ICHAR(A(2:2))-32) * 128 + (ICHAR(A(4:4))-32)*4
     $  + MOD((ICHAR(A(3:3))/4),4)
      RETURN
      END
```

Blank

APPENDIX V: RETRO/DISPLAY USER GUIDES AND SOURCE CODES.

RETRO/DISPLAY are numeric and graphic implementations of Full-Wave electromagnetic scattering theory. The theory was developed by Professor Ezekiel Bahar at the University of Nebraska-Lincoln. The reader is advised to read publications 30-34 listed in the **Literature Cited** section for details about this theory. Equations notated [A; B] imply that A is the **Literature Cited** reference number, and B the page number from that reference. The **Full Wave code** was written for a CRAY supercomputer operating under UNIX UNICOS.

AV.1 INPUT FILES.

Two input files are needed by RETRO. The first is "params 1," which is used in every run and contains filenames, surface information, and integration routine controls. The second "dielectric's input" file contains information to determine the material's relative dielectric constant, ϵ_r, as a function of wavelength, λ_0.

AV.1.1 "params."

The following information, in order, is stored in file "params": filename for commentary output, filename for data output, a 13 character description of the surface material, desired mean squared heights, mean squared slopes, radiation wavelengths, incident angles, code for getting ϵ_r data, and the error requirements for IMSL integration routines, QDAG and TWODQ. An example of a correctly edited file is shown at the end of this guide. The line by line format is as follows:

A - filename for commentary output (up to 13 characters)
B - filename for data output (up to 13 characters)
C - description of surface material (up to 13 characters)
D - no. of $<h^2>$'s, min($<h^2>$), step($<h^2>$) in μm^2 (integer,real,real)
E - lines E through F exist only if the no. of $<h^2>$'s < 0
F - here the min and step values are ignored in favor of reading
in a list of $<h^2>$ values separated by spaces
G - no. of σ_ξ^2's, min(σ_ξ^2), step(σ_ξ^2) (integer,real,real)
H - same as for $<h^2>$'s
I -
J - no. of λ_0's, min(λ_0), step(λ_0) in μm (integer,real,real)
K - same as for $<h^2>$'s
L -
M - no. of θ_0's, min(θ_0), step(θ_0) in degrees (intger,real,real)
N - same as for $<h^2>$'s
O -
P - A code for determining ϵ_r values (integer = 0 or 1)
Q - if the code on line P is 1 then the relative dielectric is put here (complex)
- otherwise the filename containing the ϵ_r's is here (up to 13 characters)
R - absolute and relative error criteria for QDAG (real,real)
S - absolute and relative error criteria for TWODQ (real,real)

The data on lines A,B,C,D,G,J,M,P,Q,R, and S must appear in "params." The data listed on lines E,F,H,I,K,L,N, and O may or may not be needed, depending on the sign of the first number on lines D,G,J, and M.

AV.1.2 Dielectrics Input File.

This file passes RETRO the information to calculate $\epsilon_r = \epsilon_r(\lambda_0)$. The filename is passed on line Q in the above format. The usual form of dielectric filenames is "material.nk," where "material" is a singular name for the substance making up the surface. The ".nk" implies that the real and imaginary part of the relative index of refraction, $n_r = n + ik$, where $n = n(\lambda_0)$ and $k = k(\lambda_0)$, is passed instead of $\epsilon_r = n_r^2$. The data is stored this way because n and k values are easier to find in tables.
The format for "material.nk" is as follows:

A - the number of wavelengths in list, m (integer)
B - $\lambda_0(1)$, $n(1)$,$k(1)$ (real,real,real)

.

.

Z - $\lambda_0(m)$, $n(m)$, $k(m)$ (real,real,real)

where $\lambda_0(l) < \lambda_0(l+1)$. Also, for the interpolation routine to work, m > 6. A sample file is shown at the end of this guide.

AV.2 OUTPUT FILES.

Two output files are created by RETRO, a commentary output and the data output. These filenames are defined on lines A and B of "params." The commentary file is primarily a debugging tool should something go wrong. The computed matrices and a header block are put in the data file.

The commentary output file is opened as unit 8 in RETRO. Unit 8 is then sent various values for tracing and debugging the program. The file becomes important if unknown run-time errors occur. Most of the write statements to unit 8 have been commented out, but they can be reinstated by editing RETRO and changing some "c"'s to blanks. During normal operation, the only significant information sent to this file is the list of ϵ_r's for each λ_0's after interpolation from the dielectric file. DISPLAY does not access this file for plotting.

AV.2.1 Data Output File.

This file contains two distinct sections: the heading block and the data block. The heading block is the key to finding the elements of Mueller matrix, $F(\langle h^2\rangle,\sigma_\xi^2,\lambda_0,\theta_0)$. The format of the heading block is as follows :

A - description of material ("params" line C)
B - no. of $\langle h^2\rangle$'s, σ_ξ^2's, λ_0's, and θ_0's ("params" lines D,G,J, and M)
C - $\langle h^2\rangle(1)$ . . . $\langle h^2\rangle$(no. of $\langle h^2\rangle$'s) ("params" line D or lines E and F)
D - $\sigma_\xi^2(1)$ . . . σ_ξ^2(no. of σ_ξ^2's) ("params" line G or lines H and I)
E - $\lambda_0(1)$ . . . λ_0(no. of λ_0's) ("params" line J or lines K and L)
F - $\theta_0(1)$ . . . θ_0(no. of θ_0's) ("params" line M or lines N and O)

The data block consists of two data lines per matrix. The first line contains three Q values corresponding to the auto-correlation functions Gaussian, N=8, and N=6. The second line contains six generally non-zero matrix elements divided by Q. In order, these correspond to f_{11}, f_{12}, f_{22}, f_{33}, f_{34}, and f_{44}. RETRO does not write $f_{21} = f_{12}$ or

$f_{43} = -f_{34}$. The other 8 matrix elements are zero.

RETRO calculates and writes the matrix data in a nested loop where $<h^2>$ varies more slowly than σ_ξ^2, which varies more slowly than λ_0, which varies more slowly than θ_0. The looping of each variable goes from the first value to the last value listed in the header (lines C to F). Given this information, DISPLAY can access any of the F's in the file.

AV.3 RUNNING INSTRUCTIONS.

RETRO is ready for execution after the input file "params" has been properly edited and the dielectric file for the surface material has been loaded. There are two ways of running the program, and the preferred method is based on how many matrices must be calculated. The number of Mueller matrices calculated is the number of $<h^2>$'s times the number of σ_ξ^2's times the number of λ_0's times the number of θ_0's. A short run is considered less than 100 matrix elements.

AV.3.1 Short Interactive Runs.

Runs that calculate less than 100 Mueller matrices can be done interactively by typing "retro.run" after the prompt. This command will compile, link, and execute the program using the data provided in file "params." This executable code is stored in file "retro.xqt." If you desire to run the program again at the same login after changing "params," you need only type "retro.xqt" after the prompt. This should be done instead of "retro.run" to prevent unnecessary compiling and linking. Finally, before exiting, you should remove "retro.xqt." The object file "retro.o" was removed by "retro.run" immediately following the linking procedure.

AV.3.2 Long Batch Runs.

For longer runs, a job needs to be submitted to a batch queue. For runs of less than 2000 matrices, type "qsub -eo -q small2 examp." If the number of matrices is greater than 2000, replace "small2" with "large2" in the previous string. Immediately after the submit command, the computer returns the job's assignment number, N. Write this down, and if you discover an error in "params" while the job is still running you can stop it by giving the command "qdel -k N," where N is the assignment number.

Files "exampl" and "examp2" are the run streams for the batched job. Only one run stream named "exampl" is allowed in the batch queue at a time. However, it is possible to have more than one batch job running at a time if copies of exampl are submitted in exactly the same way. The file "params" is read soon after the submit command, so it can be edited and the next job submitted while the first is still active. The same comments about removing "retro.xqt" apply here as in the previous section.
When a batch job is finished, an additional output file is created under the name exampl.oN, where N is defined as before. This is the job accounting report for the run, that includes the job's user time, system time, and billing units. For a complete description see the Cray manual's "jar" (job accounting report generation) command. A good idea is to append this accounting report to the commentary file created by the run. This is the file named on line B of "params."

AV.4 RELATIONSHIP OF PROGRAM VARIABLES TO THEORY.

The following is a break down of the various theoretical parameters followed by the relationship of program variables to these values. See reference 26 for further explanation.

AV.4.1 Computing Q:

$$Q = \frac{k_0^2}{\pi} Q'$$

$$Q' = 2\pi \int_0^{\infty} (\chi_2(r_d) - |\chi|^2) J_0(v_{xz} r_d) r_d \, dr_d$$

$$\chi_2 = exp(v_y^2 \langle h^2 \rangle (1 - r_{hh}'))$$

$$v_y = 2k_0 cos(\theta_0)$$

$$r_{hh}' = r_{hh}'(r_d)$$

$$|\chi|^2 = exp(-v_y^2 \langle h^2 \rangle)$$

$$v_{xz} = |2k_0 \sin(\theta_0)|$$

Q = Q
KOSQ = k_0^2
PI = π
RD = r_d
QPRIME = Q'
CHI2 = χ_2
VYSQH = $v_y^2 \langle h^2 \rangle$
VY = v_y
HMSQ = $\langle h^2 \rangle$
K0 = k_0
THETA = θ_0
CSTHT = $cos(\theta_0)$
SNTHT = $sin(\theta_0)$
RRSURF(RD) = $r_{hh}'(r_d)$
CHISQ = $|\chi|^2$
JSUB0 = $J_0(v_{xz} r_d)$
VXZ = v_{xz}

AV.4.2 Computing r_{hh}'.

For the Gaussian auto-correlation function:

$$r_{hh}(r_d) = exp\left(\frac{-r_d^2}{l_c^2}\right)$$

For the N=8 auto-correlation function:

$$r_{hh}(r_d) = \left[1 - \frac{3\zeta^2}{8} + \frac{\zeta^4}{32} + \frac{\zeta^6}{3072}\right]\zeta K_1(\zeta) + \left[\frac{1}{2} - \frac{\zeta^2}{4} - \frac{\zeta^4}{96}\right]\zeta^2 K_0(\zeta) \qquad ([33;45])$$

$$\zeta = r_d \kappa_8, \quad \kappa_8^2 = \frac{.4\sigma_\zeta^2}{<h^2>}$$

For the N=6 auto-correlation function:

$$r_{hh}(r_d) = \left[1 - \frac{3\zeta^2}{4} - \frac{\zeta^4}{96}\right]\zeta K_1(\zeta) + \left[\frac{1}{2} + \frac{3\zeta^2}{16}\right]\zeta^2 K_0(\zeta)$$

$$\zeta = r_d \kappa_6, \quad \kappa_6^2 = \frac{\sigma_\zeta^2}{<h^2>}$$

CLENSQ = l_c^2
R = ζ
RSQ = $\zeta 2$
R4TH = ζ^4
R6TH = ζ^6
KAPPA8 = κ_8
HMSQ = $<h^2>$
SIGS = σ_ζ^2
KAPPA6 = κ_6

AV.4.3 Computing σ_{ij}^{kl}/Q:

$$\frac{\sigma_{ij}^{kl}}{Q} = 2\int_{h_x=0}^{\infty}\int_{h_z=-\infty}^{\infty} \frac{D^{ij}\,D^{kr}}{(\bar{n}\cdot\bar{a}_y)^2} P_2\, P_S\, dh_x\, dh_z \qquad ([32;731\text{-}732])$$

$$P_2(h_x) = \frac{1}{1+C_0} U(cotan(\theta_0))$$

$$C_0 = \frac{1}{2}\frac{\sqrt{\sigma_\xi^2}}{\sqrt{\pi}} tan(\theta_0) exp(-\frac{cotan^2(\theta_0)}{\sigma_\xi^2}) - \frac{1}{2} erfc(\frac{cotan(\theta_0)}{\sqrt{\sigma_\xi^2}})$$

$$P_S(h_x,h_z) = \frac{1}{\pi\sigma_\xi^2} exp(-\frac{h_x^2+h_z^2}{\sigma_\xi^2})$$

IDD = σ_{ij}^{kl}
DPQMAG = $\dfrac{D^{ij}\,D^{kr}}{(\bar{n}\cdot\bar{a_y})^2}$
P2 = P_2
HX = h_x
HZ = h_z
COTT = $cotan(\theta_0)$
FACT = $\dfrac{1}{1+C_0}$
SIGSRT = $\sqrt{\sigma_\xi^2}$
PIC = $\dfrac{1}{2}\dfrac{\sqrt{\sigma_\xi^2}}{\sqrt{\pi}}$
R = $\dfrac{cotan(\theta_0)}{\sqrt{\sigma_\xi^2}}$, (in function ARGIDD only)
F1 = $exp(-\dfrac{cotan^2(\theta_0)}{\sigma_\xi^2})$
F2 = $erfc(\dfrac{cotan(\theta_0)}{\sqrt{\sigma_\xi^2}})$
C0 = C_0
FHXHZ = P_S
HXZSQ = $h_x^2 + h_z^2$

AV.4.4 Computing $\dfrac{D^{ij}\,D^{kr}}{(\bar{n}\cdot\bar{a_y})^2}$:

$$\bar{n}\cdot\bar{a_y} = cos(\gamma) \qquad ([31;446])$$

γ = the angle between $\bar{a_y}$ and the normal to the local scatter plane, $\bar{n}$.

$$D^{PQ} = \begin{cases} D_{11}, & PQ = VV \\ D_{12}, & PQ = VH \\ D_{21}, & PQ = HV \\ D_{22}, & PQ = HH \end{cases}$$

$$\mathbf{D} = \cos(\theta_0^n)\mathbf{T}\,\mathbf{F}\,\mathbf{T}$$

θ_0^n = the angle between the incident direction and $\tilde{n}$.

$$\mathbf{T} = \begin{bmatrix} \cos(\psi) & \sin(\psi) \\ -\sin(\psi) & \cos(\psi) \end{bmatrix} \qquad ([31;448])$$

ψ = the angle between the reference and local planes of incidence.

$$\mathbf{F} = \begin{bmatrix} F^{VV} & 0 \\ 0 & F^{HH} \end{bmatrix}$$

CSG = $\cos(\gamma)$
DPQ(i,j) = D_{ij}
CS0N = $\cos(\theta_0^n)$
SN0N = $\sin(\theta_0^n)$
CSSI = $\cos(\psi)$
SNSII = $\sin(\psi)$
CFVV = $\cos(\theta_0^n)F^{VV}$
CFHH = $\cos(\theta_0^n)F^{HH}$

AV.4.5 Computing F^{VV} and F^{HH}:

$$F^{VV} = \frac{(\mu_r \cos^2(\theta_1^n) - \sin^2(\theta_0^n))(1-\frac{1}{\epsilon_r}) + (1-\mu_r)}{(\cos(\theta_0^n) + \eta_r \cos(\theta_1^n))^2} \qquad ([31;446])$$

$$F^{HH} = \frac{(\epsilon_r \cos^2(\theta_1^n) - \sin^2(\theta_0^n))(1-\frac{1}{\mu_r}) + (1-\epsilon_r)}{(\cos(\theta_0^n) + \frac{1}{\eta_r}\cos(\theta_1^n))^2}$$

$$\mu_r = \frac{\mu_1}{\mu_0}$$

$$\epsilon_r = \frac{\epsilon_1}{\epsilon_0}$$

$$\eta_r = \frac{\sqrt{\mu_r}}{\sqrt{\epsilon_r}}$$

$$n_r = \sqrt{\epsilon_r \mu_r}$$

$$\sin(\theta_1^n) = \frac{\sin(\theta_0^n)}{n_r}$$

$$\cos^2(\theta_1^n) = 1-\sin^2(\theta_1^n)$$

$$\mathrm{ER} = \epsilon_r$$

$$\mathrm{UR} = \mu_r$$

$$\mathrm{RIR} = n_r$$

$$\mathrm{ETAR} = \eta_r$$

$$\mathrm{ERM} = 1-\epsilon_r$$

$$\mathrm{URM} = 1-\mu_r$$

$$\mathrm{ERMR} = 1-\frac{1}{\epsilon_r}$$

$$\mathrm{URMR} = 1-\frac{1}{\mu_r}$$

$$\mathrm{SN1N} = \sin(\theta_1^n)$$

$$CS1N = \cos(\theta_1^n)$$

$$DEN2 = \cos(\theta_0^n) + \eta_r \cos(\theta_1^n)$$

$$DEN3 = \cos(\theta_0^n) + \frac{1}{\eta_r}\cos(\theta_1^n)$$

$$S0IF = \sin^2(\theta_0^n)$$

$$C1IF = \cos^2(\theta_1^n)$$

AV.5 RETRO SOURCE CODE.

```
C
c This work was done for the CRDEC on the Aberdeen Proving Grounds
c Edgewood Area.
c This work was done by Craig M. Herzinger under contract 88MQ450.
c
c This program uses the Full-wave Theory for computing the scattering
c of a plane wave from a randomly rough surface.
c
c This program is for SINGLE scatter from an ISOTROPIC rough surface.
c This program is for BACKSCATTER only.
c This program is for DIFFUSE scattering only.
c
c Single scatter implies the reflected radiation struck the rough
c surface only one time and that multiple scattering is unaccounted for.
c An isotropic surface is considered to invariant to rotation and translation
c in terms of the average scattering.
c Backscatter implies the receiver and detector are at the same point,
c with the same orientation, at a point far from the surface.
c Only diffuse scattering is calculated because the coherent specular term
c which occurs at normal incidence for backscatter is dropped.
c
c This program calculates the 8 generally non-zero Mueller Mtx elements,
c F11,12,21,22,33,34,43, and 44, for use with the standard Stokes Vector
c notation.  Of these eight two pairs are degenerate, F12-F21 and F34--F43.
c The elements are calculated on a per solid angle basis so that the calculated
c Mueller Mtx is absolutely correct to within a scalar constant.  The scalar
c constant is based upon the size of the solid angle intercepted by the
c detector for a particular experimental setup.  For this work to be valid
c the detector must look at range of returned angles close enough to
c pure backscatter that the backscattered return is a good approximation
c of the entire reflected range.  Also, the solid angle intercepted by the
c detector must be invariant to incident angle and wavelength.
c
c The program first calculates the scattering mtx, S, for use with the
c modified Stokes Vector notation, and this is then transformed into the
c desired form.
c
c The elements of S can be written as a product of two values, Q, and
c a 2-d slope averaging integral.  This is allowed by the Full-wave
c Theory ONLY because slopes and heights are considered uncorrelated.
c Q is a function of incident angle, surface height auto-correlation
c function, and free space radiation wavelength.  The slope integral
c is function of incident angle, mean squared slope, and wavelength
c through the surfaces relative dielectric constant.
c
c For the assumed isotropic surface, Q, normally a 2-d integral can
c be rewritten in polar coordinates and transformed into a 1-d integral.
c Q, is computed by subroutine QCMP for 3 different spectral density/
c surface height auto-correlation functions.  The inputs for the auto-corr.
c functions are mean squared height and mean squared slope.  The 3
c functions are Gaussian, N-8, and N-6.
c
```

```
      c The slope averaging integrals account for all possible combinations of
c slopes in the x and z directions.  Considering various polarizations and
c phase relationships, 16 unique integrals are possible to complete S.
c But 8 integrals --> 0 because the integrand is odd.
c 3 others converge to 1 value, and another --> 0 because the integrand
c is proportional to the imaginary part of a real number.
c This leaves 5 unique 2-d integrals performed by subroutine IDDCMP.
c The 5 values could be used compute S but they are recombined in one
c step to form the Mueller mtx elements, Pij.
c
c The following version 10.0 IMSL routines are required:
c    QDAG, TWODQ, BSJ0, BSK0, BSK1, and ERFC.
c These are available from IMSL, customer relations, sixth floor,
c NBC Building 7500 Bellaire Boulevard, Houston, Texas  77036-5085, USA.
c Telephone (713)772-7927      Telex: 79-1923 IMSL INC HOU
c
c
c  Input unit 7 is assigned to a file params that must contain the
c  following input parameters:
c line01 - filename for program commentary from unit 8
c line02 - filename for output data from unit 9
c line03 - character*13 description of material being tested
c line04 - range or list of desired mean squared heights (in u^2)
c        - range or list of desired mean squared slopes (unitless)
c        - range or list of desired wavelengths (in u)
c        - range or list of desired incident angles (in deg.)
c             measured with respect to the normal of the reference plane
c        - prompt for dielectric of material
c        - the complex relative dielectric constant of the surface or
c          the name of the file containing list of constants
c        - integration requirements for QDAG
c        - integration requirements for TWODQ
c
c Unit 8 is sent various values for debugging the program and determining
c where something may have gone wrong.  It sends values for monitoring the
c integration routines.  Most writes to unit 8 have been commented out, but
c can be reinstated should the program have run time problems.
c
c Unit 9 is sent a heading block to access the Mueller Mtx data that follows.
c Elements of a Mueller Mtx for a given mean squared height, mean squared slope,
c radiation wavelength, and incident angle are printed on two lines.
c The first line contains the scalar Q values for the 3 different surface
c correlation functions.  The second lines contains six of mtx elements
c divided by Q.  The elements in order are P11,12,22,33,34 and 44.
c Any program making use of this output file must make the proper combinations
c and assignments to complete the total Mueller Mtx.
C
C
      REAL PI,HMSQB,HMSQS,SIGSB,SIGSS,WLENB,WLENS,ANGB,ANGS,
     1    HMSQ,CLEN,WLEN,CLENSQ,SIGS,K0,K0SQ,CSTHT,SNTHT,THETA,
     2    HMSQL(15),SIGSL(15),WLENL(100),ANGL(100),
     3    IDD(16),XS(16),P(16),Q(3),
     4    AERR1,RERR1,AERR2,RERR2
      REAL SETER
```

```
      REAL SVIDD(100,16)
C
      COMPLEX ERCMP,ER
C
      INTEGER NANG,NWLEN,NSIGS,NHMSQ,I1,I2,I3,I4,IL,
     1      DCODE,RCODE,IFLAG1
C
      CHARACTER*15 OUTNM,COMNM
      CHARACTER*13 MATNM
C
      COMMON/ONE/HMSQ,CLEN,CLENSQ,WLEN,THETA,SNTHT,CSTHT,SIGS,K0,PI
C
C
      PI=ABS(ATAN2(0.,-1.))
C
C  This block opens file params
C  It reads file names and opens files for commentary output and data output
C  It also reads in the identifying material name and puts in output
      OPEN(UNIT=7,FILE='params')
      READ(7,'(A)')COMNM
      OPEN(UNIT=8,FILE=COMNM)
      WRITE(8,*)'COMMENTARY FOR RETRO'
      READ(7,'(a)')OUTNM
      OPEN(UNIT=9,FILE=OUTNM)
      READ(7,'(a)')MATNM
      write(8,*)'Commentary for retro'
      write(8,*)'Output file is ',OUTNM
      write(8,*)'Material description is ',MATNM
C
C
C THESE FOUR BLOCKS READ IN THE SURFACE AND BACKSCATTER PARAMETERS
C IF THE FIRST VAL IS NEGATIVE, A LIST OF VALUES IS TO FOLLOW
C IF THE FIRST VAL IS POSITIVE, A RANGE WITH THAT MANY VALUES IS SPECIFIED
C THIS ROUTINE THEN EITHER READS THE LIST INTO AN ARRAY OR
C FILLS THE ARRAY USING THE BEGINNING AND STEP VALUES GIVEN
      READ(7,*)NHMSQ,HMSQB,HMSQS
      IF(NHMSQ.LT.0) THEN
       NHMSQ=-1*NHMSQ
       READ(7,*)(HMSQL(IL),IL=1,NHMSQ)
      ELSE
       DO 10, IL=1,NHMSQ
   10  HMSQL(IL)=HMSQB+(IL-1)*HMSQS
      ENDIF
      WRITE(8,*)NHMSQ,'  ','MEAN SQUARE HEIGHT VALUES READ IN'
C
      READ(7,*)NSIGS,SIGSB,SIGSS
      IF(NSIGS.LT.0)THEN
       NSIGS=-1*NSIGS
       READ(7,*)(SIGSL(IL),IL=1,NSIGS)
      ELSE
       DO 20, IL=1,NSIGS
   20  SIGSL(IL)=SIGSB+(IL-1)*SIGSS
      ENDIF
      WRITE(8,*)NSIGS,'  ','MEAN SQUARE SLOPE VALUES READ IN'
```

```
C
  READ(7,*)NWLEN,WLENB,WLENS
  IF(NWLEN.LT.0)THEN
   NWLEN--1*NWLEN
   READ(7,*)(WLENL(IL),IL-1,NWLEN)
  ELSE
   DO 30, IL-1,NWLEN
30  WLENL(IL)-WLENB+(IL-1)*WLENS
  ENDIF
  WRITE(8,*)NWLEN,'  ','WAVELENGTHS READ IN'
C
  READ(7,*)NANG,ANGB,ANGS
  IF(NANG.LT.0)THEN
   NANG--1*NANG
   READ(7,*)(ANGL(IL),IL-1,NANG)
  ELSE
   DO 40, IL-1,NANG
40  ANGL(IL)-ANGB+(IL-1)*ANGS
  ENDIF
  WRITE(8,*)NANG,'  ','INCIDENT ANGLES READ IN'
C
C
C THIS SECTION PLACES A HEADING BLOCK INTO THE OUTPUT FILE
C THE SURFACE AND BACKSCATTER PARAMETERS ARE WRITTEN INTO THE OUTPUT
C FILE SO THAT LATER ANALYSIS ROUTINES WILL HAVE AN INDEX TO THE DATA
  WRITE(9,*)MATNM
  WRITE(9,1000)NHMSQ,NSIGS,NWLEN,NANG
 1000 FORMAT(4I10.4)
  WRITE(9,1010)(HMSQL(IL),IL-1,NHMSQ)
  WRITE(9,1010)(SIGSL(IL),IL-1,NSIGS)
  WRITE(9,1010)(WLENL(IL),IL-1,NWLEN)
  WRITE(9,1010)(ANGL(IL),IL-1,NANG)
 1010 FORMAT(5E12.4)
C
C
C THIS BLOCK READS IN THE CODE FOR THE RELATIVE DIELECTRIC CONSTANTS TO BE
C USED.  THIS CODE AND THE DEGREES OF LAGRANGIAN INTERPOLATION TO BE USED
C IS PASSED TO ENTRY SETER OF FUNCTION ERCMP.  THE NEXT LINE OF INPUT IS
C READ INSIDE ENTRY SETER.
  READ(7,*)DCODE
  write(8,*)'Dielectric code read in ',dcode
  SETUP-SETER(DCODE,5)
C
C
C THIS LAST INPUT DATA IS THE RELATIVE AND ABSOLUTE ERROR REQUIREMENTS TO BE
C BY THE INTEGRATION ROUTINES QDAG AND TWODQ RESPECTIVELY.
  READ(7,*)AERR1,RERR1
  write(8,*)' Error criteria for QDAG read in ',aerr1,rerr1
  READ(7,*)AERR2,RERR2
  write(8,*)' error criteria for TWODQ read in ',aerr2,rerr2
C
C
C THESE FOUR NESTED LOOPS MAKE ALL COMBINATIONS OF SURFACES WITH
C WAVELENGTHS AND ANGLES OF INCIDENCE SPECIFIED.
```

```
C THE DESIRED PARAMETERS ARE USED TO COMPUTE THE MUELLER
C MATRIX ELEMENTS BY FIRST CALCULATING A SCALING 1-D INTEGRAL, Q,
C AND THEN CALCULATING 16 2-D INTEGRALS, IDD1..IDD16, THAT ARE COMBINED IN THE
C CORRECT MANNER TO GIVE THE MUELLER MATRIX ELEMENTS FOR THE
C STANDARD STOKES VECTOR NOTATION.
C IN REALITY FOR THIS WORK ONLY 5 2-D INTEGRALS NEED BE CALCULATED BUT
C THE PROGRAM IS SET UP GENERALLY.
      DO 50 I1=1,NHMSQ
       HMSQ=HMSQL(I1)
C
      DO 60 I2=1,NSIGS
       SIGS=SIGSL(I2)
C The correlation length can be calculated when hmsq and sigs are fixed
       CLENSQ=4.*HMSQ/SIGS
       CLEN=SQRT(CLENSQ)
C
      IOLD=0
      DO 70 I3=1,NWLEN
       WLEN=WLENL(I3)
C K0 and the relative dielectric constant can be computed when wlen is known
       ER=ERCMP(WLEN)
       K0=2*PI/WLEN
       K0SQ=K0*K0
       write(8,*)'Relative dielectric constant ',ER
C
      DO 80 I4=1,NANG
       ANG=ANGL(I4)
C Theta is the angle between the incident direction and the normal to the
C reference plane in radians
       THETA=PI*ANG/180.0
       CSTHT=COS(THETA)
       SNTHT=SIN(THETA)
C
C     write(8,*)'Angle(deg)  Wlen   Mean Sq Slope   Mean Sq Hgt'
      write(8,*)ANG,WLEN,SIGS,HMSQ
C
C
C THIS SECTION COMPUTES THE SCALAR Q VALUES FOR 3 AUTO-CORR FUNCTIONS
C RCODE=1 -> GAUSSIAN   RCODE=2 -> N=8   RCODE=3 -> N=6
C IF ALL Q VALUES ARE TREATED AS 0 THEN IFLAG1=0 AND THE IDD'S DO NOT
C NEED TO BE CALCULATED.
      IFLAG1=0
      DO 88 RCODE=1,3
      CALL QCMP(Q(RCODE),AERR1,RERR1,RCODE)
      IF(Q(RCODE).GT.0.)IFLAG1=1
   88 CONTINUE
C
C
C THIS SECTION COMPUTES THE 16 IDD VALUES NEEDED FOR THE MUELLER MTX.
C NOTE ONLY 5 DISTINCT INTEGRATIONS ARE DONE.  THE OTHERS ARE KNOWN FOR
C OTHER REASONS ASSIGNED TO THE FOLLOWING CONDITIONAL ASSIGNMENTS.
C IF IFLAG1=1 THEN THERE IS A REASON TO CALCULATE THESE VALUES
C IF Er IS HELD CONSTANT OVER A RANGE OF WAVELENGTHS THEN IDD'S ONLY
C NEED TO BE CALCULATED ONCE FOR EACH INCIDENT ANGLE.
```

```
          IF(IFLAG1.GT.0.) THEN
         IF(DCODE.NE.1.OR.IOLD.NE.1) THEN
          DO 90 IL=1,16
           IF(IL.EQ.1.OR.IL.EQ.2.OR.IL.EQ.4.OR.IL.EQ.5.OR.IL.EQ.6)THEN
            CALL IDDCMP(IDD(IL),AERR2,RERR2,IL)
           ELSE IF(IL.EQ.3.OR.IL.EQ.7) THEN
            IDD(IL)=IDD(2)
           ELSE IF(IL.EQ.8) THEN
            IDD(IL)=0.
           ELSE IF(IL.GE.8) THEN
            IDD(IL)=0.
           ENDIF
   90      CONTINUE
C If Er is constant then save IDD's for next wavelength
         IF(DCODE.EQ.1) THEN
          DO 93 IL=1,16
           SVIDD(I4,IL)=IDD(IL)
   93      CONTINUE
         ENDIF
        ELSE
C If Er is constant and the IDD's have been saved use them
         DO 92 IL=1,16
          IDD(IL)=SVIDD(I4,IL)
   92     CONTINUE
        ENDIF
       ENDIF
C
C
C THIS SECTION UTILIZED THE IDD VALUES TO COMPUTE THE MUELLER MTX
C ELEMENTS DIVIDED BY 1.
C F(1)=F11/Q, F(2)=F12/Q, .. F(5)=F21/Q, .. F(16)=F44/Q
C MANY OF THESE F VALUES ARE ZERO BUT THEY ARE CALCULATED HERE FOR
C COMPLETENESS. THEIR CALCULATION TIME IS MINUTE COMPARED TO THE ACTUAL
C INTEGRATIONS.
       F(1)=.5*(IDD(1)+IDD(2)+IDD(3)+IDD(4))
       F(2)=.5*(IDD(1)+IDD(2)-IDD(3)-IDD(4))
       F(3)=IDD(9)
       F(4)=-IDD(10)
       F(5)=.5*(IDD(1)-IDD(2)+IDD(3)-IDD(4))
       F(6)=.5*(IDD(1)-IDD(2)-IDD(3)+IDD(4))
       F(7)=IDD(13)
       F(8)=-IDD(14)
       F(9)=2.*IDD(11)
       F(10)=2.*IDD(15)
       F(11)=IDD(5)+IDD(7)
       F(12)=-IDD(6)+IDD(8)
       F(13)=2.*IDD(12)
       F(14)=2.*IDD(16)
       F(15)=IDD(6)*IDD(8)
       F(16)=IDD(5)-IDD(7)
C
C
C THE SCALAR Q VALUES ARE WRITTEN ON ONE LINE AND
C 6 MUELLER MATRIX ELEMENTS DIVIDED BY Q ARE WRITTEN ON THE NEXT.
```

```
C THE OTHER 10 ARE KNOWN TO BE ZERO FOR THIS WORK.
   WRITE(9,2000)Q(1),Q(2),Q(3)
   WRITE(9,2000)F(1),F(2),F(6),F(11),F(12),F(16)
2000 FORMAT(6E12.4)
   write(8,2001)Q(1),Q(2),Q(3)
   write(8,2001)IDD(1),IDD(2),IDD(4),IDD(5),IDD(6)
2001 FORMAT(6E12.4)
C
 80 CONTINUE
   IOLD-1
 70 CONTINUE
 60 CONTINUE
 50 CONTINUE
C
   REWIND(7)
   REWIND(8)
   REWIND(9)
   CLOSE(7)
   CLOSE(8)
   CLOSE(9)
   STOP
   END
C
C
C
C THIS SUBROUTINE DRIVES QDAG   TO COMPUTE Q
   SUBROUTINE QCMP(Q,AERR,RERR,RCODE)
   REAL VX,VY,VXZ,VYSQH,UPLIM,QPRIME,SETARG,ARG,ARG1,
  1    UPLIMD,UPLIM1,SETUPR
   REAL HMSQ,CLEN,CLENSQ,WLEN,THETA,SNTHT,CSTHT,SIGS,K0,PI
   REAL GCNT1
   INTEGER RCODE
   EXTERNAL ARG
  COMMON/ONE/HMSQ,CLEN,CLENSQ,WLEN,THETA,SNTHT,CSTHT,SIGS,K0,PI
C FOR EACH VALUE OF THETA COMPUTE COMPONENTS OF VECTOR V.
C THEN COMBINE WITH THE MEAN SQUARE HEIGHT AND SAVE IN ARG
C    WRITE(8,*)'SNTHT,CSTHT',SNTHT,CSTHT
   VX--2.*K0*SNTHT
C   VZ-0.
   VY-2*K0*CSTHT
   VXZ-ABS(VX)
   VYSQH-VY*VY*HMSQ
   SETARG-ARG1(VXZ,VYSQH)
   UPLIMD-SETUPR(RCODE)
   SETARG-GCNT1(0.)
C COMPUTE THE UPPERLIMIT ON THE INTEGRATION BY ASSUMING THE INTEGRAND
C DIES AWAY WITHIN UPLIMD CORRELATION LENGTHS.
   UPLIM-UPLIMD*CLEN
   UPLIM1-UPLIM/100.
 100 CONTINUE
C COMPUTE THE DOUBLE INTEGRAL WHERE ONE HALF -> 2PI
C THIS CAN BE DONE BY SWITCHING TO POLAR COORDINATES
C IF NO INTERGRAND IS FOUND THEN REDUCE INTEGRATION LIMITS TO FIND IT
   CALL QDAG(ARG,0.,UPLIM,AERR ,RERR,1,QPRIME,ERREST)
```

```
          QPRIME-QPRIME*2*PI
        IF(QPRIME.EQ.0.AND.UPLIM.GT.UPLIM1) THEN
         UPLIM-UPLIM*0.7
         GOTO 100
        ENDIF
C
C IF QPRIME > 0. THEN COMPUTE THE TOTAL Q
C IF QPRIME STILL EQUALS ZERO THEN INDICATE BY -998.
C IF QPRIME WAS LESS THAN ZERO INDICATE BY -999.
        IF (QPRIME.GT.0.)THEN
         Q-(K0*K0/PI)*QPRIME
        ELSE IF(QPRIME.EQ.0.)THEN
         Q--998.
        ELSE IF(QPRIME.LT.0.)THEN
         Q--999.
        ENDIF
C        WRITE(8,*)'Q-',Q,'   ACCESSED',GCNT1(0.)
        RETURN
        END
C
C
C
C THIS FUNCTION COMPUTES THE INTEGRAND OF Q FOR DCADRE.
C THIS FUNCTION HAS 3 ENTRY POINTS
C BSJ0 COMPUTES BESSEL FUNCTION J SUB 0
        FUNCTION ARG(RD)
        REAL BSJ0,RRSURF,CHI2,CHISQ,JSUB0,RD,VXZ,VYSQH
        REAL COUNT,DUMMY
        INTEGER RCODE,DRCODE
        SAVE
        COUNT-COUNT+1.
        JSUB0-BSJ0(VXZ*RD)
        CHI2-EXP((RRSURF(RD)-1.)*VYSQH)
        ARG-JSUB0*(CHI2-CHISQ)*RD
C        ARG-1.
        RETURN
C
C
C THIS SECOND ENTRY POINT SAVES SOME CONSTANTS  FOR A GIVEN INTEGRATION
C THESE VALUES ARE ONLY FUNCTIONS OF THE SURFACE PARAMETERS NOT OF RD
        ENTRY ARG1(DVXZ,DVYSQH)
        VXZ-DVXZ
        VYSQH-DVYSQH
        CHISQ-EXP(-VYSQH)
        ARG1-1.0
        RETURN
C
C
C THIS ENTRY RETURNS THE NUMBER OF ACCESSES SINCE LAST INQUIRY
        ENTRY GCNT1(DUMMY)
        GCNT1-COUNT
        COUNT-DUMMY
        RETURN
        END
```

```
      C
    C
    C THIS FUNCTION CALCULATES THE AUTOCORRELATION FUNCTION USED TO
    C MODEL THE RANDOMLY ROUGH SURFACE
    C BSK1 COMPUTES THE BESSEL FUNCTION K SUB 1
    C BSK0 COMPUTES THE BESSEL FUNCTION K SUB 0
        FUNCTION RRSURF(RD)
        REAL RD,R,RSQ,R4TH,R6TH,KAPPA6,KAPPA8,BSK1,BSK0,TERM1,TERM2
        REAL HMSQ,CLEN,CLENSQ,WLEN,THETA,SNTHT,CSTHT,SIGS,K0,PI
        INTEGER RCODE,DRCODE
      COMMON/ONE/HMSQ,CLEN,CLENSQ,WLEN,THETA,SNTHT,CSTHT,SIGS,K0,PI
        IF(RD.EQ.0.) THEN
         RRSURF-1.
         RETURN
        ENDIF
        IF(RCODE.EQ.1) THEN
         RRSURF-EXP(-RD*RD/CLENSQ)
         RETURN
        ELSE IF(RCODE.EQ.2) THEN
         R-RD*KAPPA8
         RSQ-R*R
         R4TH-RSQ*RSQ
         R6TH-RSQ*R4TH
         TERM1-(1.-3.*RSQ/8.+3.*R4TH/32.+R6TH/3072.)*R*BSK1(R)
         TERM2-(RSQ/2.-R4TH/4.-R6TH/96.)*BSK0(R)
      RRSURF-TERM1+TERM2
         RETURN
        ELSE IF(RCODE.EQ.3) THEN
         R-RD*KAPPA6
         RSQ-R*R
         R4TH-RSQ*RSQ
         TERM1-(1.-3.*RSQ/4.-R4TH/96.)*R*BSK1(R)
         TERM2-(RSQ/2.+3.*R4TH/16.)*BSK0(R)
         RRSURF-TERM1+TERM2
         RETURN
        ELSE
         WRITE(8,*)'ILLEGAL RCODE'
         STOP
        ENDIF
        RRSURF-0.
        RETURN
    C
    C
        ENTRY SETUPR(DRCODE)
        RCODE-DRCODE
        SETUPR-1.
        IF(RCODE.EQ.1) THEN
         SETUPR- 5.
        ELSE IF(RCODE.EQ.2) THEN
         KAPPA8-SQRT(1.6)/CLEN
         SETUPR-175./SQRT(1.6)
        ELSE IF(RCODE.EQ.3) THEN
         KAPPA6-1./CLEN
         SETUPR-175
```

```
          ENDIF
        RETURN
        END
C
C
C
C THIS SUBROUTINE DRIVES TWODQ TO COMPUTE IDD FOR DIJ*DKL IN INTEGRAND
C THE REAL PART IS COMPUTED FOR CODE- 1 OR 2, IMAGINARY CODE- 3
      SUBROUTINE IDDCMP(IDD,AERR,RERR,IL)
      REAL IDD,AERR,RERR,ERREST
      REAL HZMIN,HZMAX,HXMIN,HXMAX,HZMAX1
      REAL SDPQ,SP2,SPHXHZ,SZMIN,SZMAX
      REAL HMSQ,CLEN,CLENSQ,WLEN,THETA,SNTHT,CSTHT,SIGS,K0,PI
      REAL GCNT2
      INTEGER ICODE(16),ISUB(16),JSUB(16),KSUB(16),LSUB(16)
      INTEGER IL
      COMPLEX ERCMP,UR
      EXTERNAL ARGIDD,ZMIN,ZMAX
   COMMON/ONE/HMSQ,CLEN,CLENSQ,WLEN,THETA,SNTHT,CSTHT,SIGS,K0,PI
      DATA ICODE/1,1,1,1,2,3,2,3,2,3,2,3,2,3,2,3/
      DATA  ISUB/1,1,2,2,1,1,1,1,1,1,1,1,2,2,1,1/
      DATA  JSUB/1,2,1,2,1,1,2,2,1,1,1,1,1,1,2,2/
      DATA  KSUB/1,1,2,2,2,2,2,2,1,1,2,2,2,2,2,2/
      DATA  LSUB/1,2,1,2,2,2,1,1,2,2,1,1,2,2,2,2/
C
      UR-(1.,0.)
      SETUP-SDPQ(ICODE(IL),ISUB(IL),JSUB(IL),KSUB(IL),
     1      LSUB(IL),ERCMP(WLEN),UR,CSTHT,SNTHT)
      SETUP-SP2(SIGS,SNTHT,CSTHT,PI)
C     WRITE(8,*)'FACT- ',SETUP
      SETUP-SPHXHZ(PI,SIGS)
C     WRITE(8,*)'DENOM- ',SETUP
      SETUP-GCNT2(0.)
      HZMIN-0.
      HZMAX-5.*SQRT(SIGS)
      HZMAX1-HZMAX*.4
      HXMIN--HZMAX
      HXMAX-HZMAX
  199 SETUP-SZMIN(HZMIN)
      SETUP-SZMAX(HZMAX)
C
   CALL TWODQ(ARGIDD,HXMIN,HXMAX,ZMIN,ZMAX,AERR,RERR,1,IDD,ERREST)
      IDD-2.*IDD
C
C     WRITE(8,*)'IDD- ',IDD,'   ACCESSED',GCNT2(0.)
      IF (IDD.EQ.0.0.AND.HZMAX.GT.HZMAX1)THEN
        HZMIN-HZMIN*.7
        HZMAX-HZMAX*.7
        HXMIN-HXMIN*.7
        HXMAX-HXMAX*.7
        GOTO 199
      ENDIF
C
      RETURN
```

```
      END
C
C
C
C THESE FUNCTIONS CALCULATE THE HZ LIMITS FOR 2-D INTEGRATION BY TWODQ
      FUNCTION ZMIN(X1)
      REAL D1ZMIN,D2ZMIN
      SAVE
      ZMIN-D1ZMIN
      RETURN
      ENTRY SZMIN(D2ZMIN)
      D1ZMIN-D2ZMIN
      SZMIN-1.0
      RETURN
      END
C
      FUNCTION ZMAX(X1)
      REAL D1ZMAX,D2ZMAX
      SAVE
      ZMAX-D1ZMAX
      RETURN
      ENTRY SZMAX(D2ZMAX)
      D1ZMAX-D2ZMAX
      SZMAX-1.0
      RETURN
      END
C
C
C
C THIS FUNCTION CALCULATES THE ARGUMENT OF IDD FOR TWODQ
      FUNCTION ARGIDD(HX,HZ)
      REAL COUNT,DUMMY,HXZSQ
      REAL DENOM,DSIGS,SIGS,SNTHT,CSTHT,PI,PACT
      REAL SIGSRT,PIC,C0,TANT,COTT,R,P1,P2
      REAL ERPC
      SAVE
C
      COUNT-COUNT+1.
      HXZSQ-HX*HX+HZ*HZ
C
      IF (HX.LT.-COTT) THEN
        P2-0.
      ELSE
        P2-PACT
      ENDIF
C
      PHXHZ-EXP(-HXZSQ/SIGS)/DENOM
C
      ARGIDD-PHXHZ*P2*DPQMAG(HX,HZ,HXZSQ)
      RETURN
C
C
      ENTRY SPHXHZ(PI,DSIGS)
      SIGS-DSIGS
```

```
        DENOM-PP*SIGS
       SPHXHZ-DENOM
       RETURN
    C
    C
       ENTRY SP2(DSIGS,SNTHT,CSTHT,PI)
       SIGSRT-SQRT(DSIGS)
       PIC-.5*SIGSRT/SQRT(PI)
       IF(ABS(SNTHT).LT.1.E-10)THEN
        COTT-1.E30
        C0-0.
       ELSE
        TANT-SNTHT/CSTHT
        COTT-1/TANT
        R-COTT/SIGSRT
        F1-EXP(-R*R)
        F2-ERFC(R)
        C0-PIC*TANT*F1-.5*F2
       ENDIF
       FACT-1./(1.+C0)
       SP2-FACT
       RETURN
    C
    C
       ENTRY GCNT2(DUMMY)
       GCNT2-COUNT
       COUNT-DUMMY
       RETURN
       END
    C
    C
    C
    C THIS FUNCTION CALCULATE DIJ*DKL AS PART OF IDD'S INTEGRAND
       FUNCTION DPQMAG(HX,HZ,HXZSQ)
       COMPLEX ER,UR,RIR,ETAR,ERM,ERMR,ERMRR,URM,URMR,URMRR
       COMPLEX SN1N,CS1N,DEN2,DEN3,C1IF,CFVV,CFHH,B1,B2,B3,B4
       COMPLEX DPQ(2,2),DER,DUR
       REAL HX,HZ,HXZSQ,SNTHT,CSTHT,DSNTHT,DCSTHT
       REAL CSG,TNG,SNG,CS0N,SN0N,SNPDI,CSPDI,S0IF,C0IF
       REAL CSSI,SNSII,SNSIF
       INTEGER ICODE,IICODE,ISUB,IISUB,JSUB,JJSUB,KSUB,KKSUB,LSUB,LLSUB
       SAVE
    C
       CSG-1./SQRT(1.+HXZSQ)
       TNG-SQRT(HXZSQ)
       SNG-CSG*TNG
       IF(TNG.LT.1.E-5) THEN
        CS0N-CSTHT
        SN0N-SNTHT
       ELSE
        CSPDI--HX/TNG
        SNPDI- HZ/TNG
        CS0N-CSG*CSTHT-SNG*SNTHT*CSPDI
        SN0N-SQRT(1.-CS0N*CS0N)
```

```
      ENDIF
C
      IF(ABS(CS0N).LT.1.E-5) THEN
       DPQMAG-0.
       RETURN
      ENDIF
C
      SN1N-SN0N/RIR
      CS1N-CSQRT(1.-SN1N*SN1N)
      DEN2-CS0N+CS1N*ETAR
      DEN3-CS0N+CS1N/ETAR
      S0IF-SN0N*SN0N
      C0IF-CS0N
      C1IF-CS1N*CS1N
      CFVV-C0IF*((-UR*C1IF-S0IF)*ERMR-URM)/(DEN2*DEN2)
      CFHH-C0IF*((-ER*C1IF-S0IF)*URMR-ERM)/(DEN3*DEN3)
C
      IF(TNG.LT.1.E-5) THEN
       DPQ(1,1)-CFVV
       DPQ(1,2)-0.
       DPQ(2,1)-0.
       DPQ(2,2)-CFHH
      ELSE
       CSSI-(CSG*SNTHT+SNG*CSTHT*CSPDI)/SN0N
       SNSII--SNG*SNPDI/SN0N
       SNSIP--SNSII
       B1-CFVV*CSSI
       B2--CFHH*SNSII
       B3-CFVV*SNSII
       B4-CFHH*CSSI
       DPQ(1,1)-CSSP*B1-SNSIP*B2
       DPQ(1,2)-CSSP*B3-SNSIP*B4
       DPQ(2,1)-SNSIP*B1+CSSP*B2
       DPQ(2,2)-SNSIP*B3+CSSP*B4
      ENDIF
C
      IF(ICODE.EQ.1) THEN
       DPQMAG-(CABS(DPQ(ISUB,JSUB))/CSG)**2
      ELSE IF(ICODE.EQ.2) THEN
       DPQMAG-REAL(DPQ(ISUB,JSUB)*CONJG(DPQ(KSUB,LSUB)))/CSG/CSG
      ELSE
       DPQMAG-AIMAG(DPQ(ISUB,JSUB)*CONJG(DPQ(KSUB,LSUB)))/CSG/CSG
      ENDIF
      RETURN
C
C
      ENTRY SDPQ(IICODE,IISUB,JJSUB,KKSUB,LLSUB,DER,DUR,
     1       DCSTHT,DSNTHT)
      ICODE-IICODE
      ISUB-IISUB
      JSUB-JJSUB
      KSUB-KKSUB
      LSUB-LLSUB
      CSTHT-DCSTHT
```

```
          SNTHT-DSNTHT
        ER-DER
        UR-DUR
        RIR-CSQRT(ER*UR)
        ETAR-CSQRT(UR/ER)
        ERM-1.-ER
        ERMR-1.-1./ER
        ERMRR-ERMR*RIR
        URM-1.-UR
        URMR-1.-1./UR
        URMRR-URMR*RIR
        SDPQ-1.0
        RETURN
        END
C
C
C
C THIS FUNCTION COMPUTES THE VALUE ER IF INTERPOLATION IS DESIRED
C OTHERWISE IT JUST RETURNS THE CONSTANT VALUE GIVEN IN PARAMS.
        COMPLEX FUNCTION ERCMP(WLEN)
        INTEGER NPTS,DEG,I,J,MIN,MAX,DDEG,DNPTS
        REAL WN(200)
        REAL WLEN,WNP,FACTOR,LI,NR,KR
        COMPLEX D1ER(200),EREST
        CHARACTER*10 FNAME
        SAVE
C
        IF(NPTS.EQ.1)THEN
          ERCMP-D1ER(1)
          RETURN
        ELSE
C this converts wavelength in microns to wave number in 1/cm
          WNP-10000./WLEN
          I-(DEG+1) / 2
          IF(WN(I).GT.WNP) THEN
            MIN-1
            MAX-MIN+DEG
            GOTO 211
          ENDIF
  200     IF(WN(I).GT.WNP) GOTO 210
          I-I+1
          IF(I.EQ.NPTS-(DEG+1) / 2)THEN
            MAX-NPTS
            MIN-NPTS-DEG
            GOTO 211
          ENDIF
        GOTO 200
C
  210 MIN-I-DEG/2-1
        MAX-MIN+DEG
  211 FACTOR - 1.
C     WRITE(8,*)'MIN,MAX ',MIN,MAX
C
        DO 220 J- MIN,MAX
```

```
          IF(WNP.NE.WN(J))GOTO 220
         ERCMP-D1ER(J)
         RETURN
   220 FACTOR-FACTOR*(WNP-WN(J))
C
        EREST-(0.,0.)
        DO 230 I-MIN,MAX
         LI-FACTOR/(WNP-WN(I))
         DO 240 J- MIN,MAX
   240    IF(I.NE.J)LI-LI/(WN(I)-WN(J))
   230  EREST-EREST+D1ER(I)*LI
        ENDIF
C
        ERCMP-EREST
        RETURN
C
C
        ENTRY SETER(DNPTS,DDEG)
        NPTS-DNPTS
        DEG-DDEG
        IF(NPTS.EQ.1)THEN
         READ(7,*)D1ER(1)
        ELSE
         READ(7,'(A)')FNAME
         WRITE(8,*)'OPENING ',FNAME,' AS DIELECTRIC FILE'
         OPEN(UNIT-10,FILE-FNAME)
         READ(10,*)NPTS
         DO 277 I-1,NPTS
          READ(10,*)WN(I),NR,KR
          D1ER(I)-(1.,0.)*(NR*NR-KR*KR)-(0.,1.)*ABS(2.*NR*KR)
   277   CONTINUE
         CLOSE(10)
        ENDIF
        SETER-1.
        RETURN
        END
```

AV.6 SAMPLE CALCULATIONS: PARAMETERS INPUT, COMMENTARY AND DATA OUTPUT FILES.

"Params" is an input file for RETRO, containing topographical parameters about the scatterer and routine control information. UNIX output files "TestClc.d" and "TestClc.c" are created by RETRO for calculation inspections and trouble-shooting purposes. (1) "Test" implies the file was created by RETRO, and is not a combination of other output files, (2) upper case "C" implies that this is a composite clay material, (3) "0" and lower case "c" imply that mean-squared height and slope of the clay surface is 5 μm^2 and 0.5 respectively, and ".c" or ".d" at the end indicates commentary file or data file for DISPLAY, respectively.

AV.6.1 Sample data file "PARAMS." (See Appendix V.1.1)

line # (No leading spaces in first column)

```
1 TestClc.c
2 TestClc.d
3 composite clay
4 -1 0. 0.
5 1.5
6 -1 0. 0.
7 .5
8 71 9. 0.05
9 22 0. 4.
10 0
11 compos.nk
12 .000000001 .0000001
13 .00001 .005
```

AV.6.2 Sample data file TestClc.d. (See Appendix AV.1.4)

```
1.  composite clay
2.     0001       0001       0071       0022
3.  0.1500E+01
4.  0.5000E+00
5.  0.9000E+01 0.9050E+01 0.9100E+01 0.9150E+01 0.9200E+01
6.  0.9250E+01 0.9300E+01 0.9350E+01 0.9400E+01 0.9450E+01
7.  0.9500E+01 0.9550E+01 0.9600E+01 0.9650E+01 0.9700E+01
8.  0.9750E+01 0.9800E+01 0.9850E+01 0.9900E+01 0.9950E+01
9.  0.1000E+02 0.1005E+02 0.1010E+02 0.1015E+02 0.1020E+02
10. 0.1025E+02 0.1030E+02 0.1035E+02 0.1040E+02 0.1045E+02
11. 0.1050E+02 0.1055E+02 0.1060E+02 0.1065E+02 0.1070E+02
12. 0.1075E+02 0.1080E+02 0.1085E+02 0.1090E+02 0.1095E+02
13. 0.1100E+02 0.1105E+02 0.1110E+02 0.1115E+02 0.1120E+02
14. 0.1125E+02 0.1130E+02 0.1135E+02 0.1140E+02 0.1145E+02
15. 0.1150E+02 0.1155E+02 0.1160E+02 0.1165E+02 0.1170E+02
16. 0.1175E+02 0.1180E+02 0.1185E+02 0.1190E+02 0.1195E+02
17. 0.1200E+02 0.1205E+02 0.1210E+02 0.1215E+02 0.1220E+02
18. 0.1225E+02 0.1230E+02 0.1235E+02 0.1240E+02 0.1245E+02
19. 0.1250E+02
20. 0.0000E+00 0.4000E+01 0.8000E+01 0.1200E+02 0.1600E+02
21. 0.2000E+02 0.2400E+02 0.2800E+02 0.3200E+02 0.3600E+02
22. 0.4000E+02 0.4400E+02 0.4800E+02 0.5200E+02 0.5600E+02
```

```
23. 0.6000E+02 0.6400E+02 0.6800E+02 0.7200E+02 0.7600E+02
24. 0.8000E+02 0.8400E+02
25. 0.2446E+01 0.1649E+01 0.1901E+01
26. 0.2483E+00 -0.1437E-11 0.2336E+00 -0.2188E+00 -0.1539E-12 -0.2336E+00
27. 0.2412E+01 0.1605E+01 0.1896E+01
28. 0.2482E+00 0.3166E-03 0.2332E+00 -0.2181E+00 -0.5693E-03 -0.2332E+00
29. 0.2312E+01 0.1703E+01 0.2447E+01
30. 0.2478E+00 0.1243E-02 0.2320E+00 -0.2162E+00 -0.2228E-02 -0.2320E+00
.
.
.
.
.
.
.
3139. 0.3237E-02 0.4312E-02 0.8359E-02
3140. 0.6718E-01 0.1580E-01 0.6122E-01 -0.5657E-01 -0.4200E-02 -0.6252E-01
3141. 0.5769E-03 0.1179E-02 0.3512E-02
3142. 0.5872E-01 0.1476E-01 0.5334E-01 -0.4907E-01 -0.3925E-02 -0.5445E-01
3143. 0.6616E-04 0.3242E-03 0.1474E-02
3144. 0.4887E-01 0.1313E-01 0.4427E-01 -0.4053E-01 -0.3492E-02 -0.4513E-01
3145. 0.4011E-05 0.9443E-04 0.5812E-03
3146. 0.3741E-01 0.1075E-01 0.3380E-01 -0.3074E-01 -0.2857E-02 -0.3435E-01
3147. 0.7976E-07 0.2454E-04 0.1781E-03
3148. 0.2409E-01 0.7396E-02 0.2172E-01 -0.1961E-01 -0.1965E-02 -0.2198E-01
```

AV.6.3 Commentary file TestC1c.c. (See Appendix AV.1.3)

```
1 Commentary for retro
2 Output file is TestC1c.d
3 Material description is Composite Clay
4 1, Mean square height values read in
5 1, Mean square slope values read in
6 71, Wavelengths read in
7 22, Incident angles read in
8 Dielectric code read in 0
9 Opening compos.nk as dielectric file
10 The following relative dielectrics are used
11 wlen,Er 9.0, (0.230451874,-1.11635916)
12 wlen,Er 9.05, (0.127786514,-0.9956851)
13 wlen,Er 9.1, (-0.12072582,-0.933739048)
14 wlen,Er 9.15, (-0.349295173,-1.01144447)
15 wlen,Er 9.2, (-0.580000781,-1.14418849)
16 wlen,Er 9.25, (-0.813025525,-1.31932772)
17 wlen,Er 9.3, (-1.06799906,-1.5584366)
18 wlen,Er 9.35, (-1.33129798,-1.88047748)
19 wlen,Er 9.4, (-1.59029517,-2.30134942)
20 wlen,Er 9.45, (-1.82918509,-2.86734854)
21 wlen,Er 9.5, (-2.00403183,-3.61813835)
22 wlen,Er 9.55, (-2.00863271,-4.69301692)
23 wlen,Er 9.6, (-1.62000141,-5.93069545)
24 wlen,Er 9.65, (-7.286277419E-02,-6.88245115)
```

```
   wlen,Er 9.7, (1.9116751,-7.26745517)
wlen,Er 9.75, (2.69681864,-6.34696569)
.
.
.
.
.
.
.
wlen,Er 12.3, (2.19544571,-0.322365349)
wlen,Er 12.35, (2.14063373,-0.355664223)
wlen,Er 12.4, (2.1129991,-0.433485636)
wlen,Er 12.45, (2.10166741,-0.515307218)
wlen,Er 12.5, (2.106225,-0.586)
 Error criteria for QDAG read in   1.0E-09,  1.0E-07
  error criteria for TWODQ read in   1.0E-05,  5.0E-03

Cray Research's UNICOS Release 3.0.2

/amsaa is full, please clean up.
/secad is full, please clean up.

Input TERM type?  Tek 4115

 COMMAND REPORT

COMMAND  STARTED  USER-CPU  SYS-CPU  I/O-WAIT  ELAPSED   SBU'S
NAME      AT      [SECONDS] [SECONDS] [SECONDS] [SECONDS]

jad      10:04:49    0.01     0.03     0.00      0.09     0.00
segldr   10:04:50    2.94     1.29     0.04     10.95     0.00
sh       10:04:49    0.08     0.07     0.00   3556.54     0.00
jad      10:04:50    0.10     0.97     0.00   3556.13     0.00
a.out    10:05:01 1832.13     6.60     0.00   3545.12     0.00
a.out    15:08:32 1562.01     2.51     0.00   2457.13     0.00

PROCESS FLOW CHART
==================

parent  -> child   ...
=========================

jad
segldr
sh
jad
a.out
a.out

JOB ACCOUNTING REPORT
=====================
Operating System        : XXXX
User                    : XXXX
```

121	Group	: XXXX
122	Accounting Id	: XXXX
123	Job Id	: XXXX
124	Report starts	: 07/14/88 10:04:49
125	Report ends	: 07/14/88 15:49:29
126	CPU Time (User)	: 3397.2720 Seconds
127	CPU Time (System)	: 11.4611 Seconds
128	I/O Wait Time	: 0.0415 Seconds
129	Elapsed Time	: 20680 Seconds
130	CPU Time Memory Integral	: 0.8662 MWords * Seconds
131	I/O W-Time Memory Integral	: 0.1419 MWords * Seconds
132	Data Transferred	: 17.0252 MBytes
133	Logical I/O Requests	: 6454
134	Real I/O Requests	: 142
135	No. of Commands	: 6
136	Billing Units	: 0.0000

AV.7.1 DISPLAY User's Guide: STARTUP PROCEDURE.

To run DISPLAY, log on and type "display.run" after the prompt. This runstream compiles, links, and executes the program. DISPLAY's executable code is stored in the file display.xqt. If one desires to rerun DISPLAY during the same login session, type "retro.xqt." This will run the program again without unnecessary recompiling and relinking. Since retro.xqt is a very large file, it should be removed before logging off. The object file, display.o, is removed immediately following the linking process.

At the beginning of each run, the program will ask for the graphics terminal type. DISPLAY directly supports the Tektronics 4107 and 4115 device drivers. After the terminal type is defined, DISPLAY enters its main menu.

A note about Tektronics terminals. If the page border and words of a DISPLAY plot are drawn in dashed lines, the problem is that the CRAY has changed the terminal setup status. A solution to this problem is simply wait for the plot to finish then press the reset button on the terminal. This will clear the screen, self test, and reset the terminal's setup parameters. When the self test is finished, press <RETURN> to continue.

AV.7.2 MAIN MENU OPTIONS.

After the menu described below has been printed, the program prompts your input. As with all menu prompts in this program, type in the number of your choice and hit <RETURN>. If you type an illegal value, the menu will be reprinted and you should try again. After the main menu option is entered, additional menu prompts will guide you. If you make a mistake and do not want any of the options presented at a given menu, try typing in "0" to back up a menu.

The main menu has 13 options, given below.

DISPLAY MAIN MENU

11. Assign a surface for data access
12. Display info. on an assigned surface
13. Remove an assigned surface
21. Modify list of surfaces to use
22. Modify list of matrix elements to use
23. Modify list of wavelengths to use
24. Modify list of incident angles to use
31. Change analysis normalizations
32. Change plotting format features
41. Do 3-d plot
42. Do 2-d plot
43. Do Contour plot
99. Quit program

The first three options 11, 12 and 13 are grouped as the data file control section. Option 11 acquires the file name, auto-correlation function, mean squared height ($<h^2>$), and mean squared slope (σ_s^2) that defines a surface. It then assigns a number to that sample. Option 12 displays pertinent information on the assigned surfaces, thus you need not remember or write down all the assignment numbers. Option 13 allows one to remove a surface assignment.

The next four options, 21, 22, 23, and 24 maintain the directive lists used by the plotting routines. Option 21 builds a surface assignment list for use by three plotting options, determining which surface(s) to use for a particular plot. Option 22 maintains the list of matrix elements to use when plotting 2-d representations of the data. Option 23 maintains the list of fixed λ_0's for use when plotting 2-d angle relationships of the Mueller elements. Lastly, Option 24 maintains the list of fixed θ_0's used in plotting the matrix elements as a function of λ_0.

Option 31 controls which analysis operations are used when data is read.

In addition to f_{ij}, it is also possible to plot $\frac{f_{ij}}{f_{11}}$, $\frac{f_{ij}^{targ}}{f_{ij}^{base}}-1$, or $\frac{f_{ij}^{targ}}{f_{11}^{targ}}\frac{f_{11}^{base}}{f_{ij}^{base}}-1$.

Option 32 allows modification of plot features such as page size, line color, tickmark density, grid lines, curve thickness, and hidden lines on 3-d plots.

Options 41, 42, and 43 produce the actual plot after the data files have been assigned, the directive lists are built, and the plot format established. Option 41 plots, in a 3-d format, a single matrix element of a single surface as a function of λ_0 and θ_0. Option 42 graphs 2-d cross sections of 3-d plots with either θ_0 or λ_0 held constant. This can be done for multiple matrix elements, one per page. Option 43 draws a contoured representation of 3-d plots. Up to four surfaces can be compared on one page.

Option 99 terminates the program. When exiting "bob>," remember to delete the large file retro.xqt.

AV.7.3 SURFACE ASSIGNMENT OPTIONS.

These three routines connect DISPLAY to the desired data files. You must know the file names before you begin. Required information is obtained from prompted user inputs. The program works on a system of surface assignment numbers. A surface is defined by a file name, auto-correlation function (a special code is reserved for experimental data), $<h^2>$, and σ_s^2. Each surface has its own list of allowed θ_0's and λ_0's stored in the heading section of the data file. Up to ten surfaces can be assigned at one time.

AV.7.3.1 Option 11.

Option 11 accesses the file name, auto-correlation function, $<h^2>$, and σ_s^2 that define a surface, then assigns a number to that surface. This routine next reads in the allowed λ_0's and θ_0's. This option does not read the Mueller matrix elements, but retains the above information and associates it with the assignment number given to the surface. The matrix elements are read, as needed, for plotting.

The data file contains two distinct sections: the heading block and the data block. The heading block is the key to finding the elements of Mueller matrix, $F(<h^2>,\sigma_s^2,\lambda_0,\theta_0)$. The format of the heading block is as follows:

A - description of material
B - no. of <h^2>'s, σ_ξ^2's, λ_0's, and θ_0's
C - <h^2>(1) . . . <h^2>(no. of <h^2>'s)
D - $\sigma_\xi^2(1)$ . . . <h^2>(no. of σ_ξ^2's)
E - $\lambda_0(1)$ . . . λ_0(no. of λ_0's)
F - $\theta_0(1)$ . . . θ_0(no. of θ_0's)

The information listed on lines C through F, above, may actually occupy several lines in the data file.

For files created by RETRO the data block consists of two data lines per matrix. The first line contains three Q values corresponding to the auto-correlation functions Gaussian, N=8, and N=6. The second line contains six generally non-zero matrix elements divided by Q. In order, these correspond to f_{11}, f_{12}, f_{22}, f_{33}, f_{34}, and f_{44}. RETRO does not write $f_{21} = f_{12}$ or $f_{43} = -f_{34}$. The other 8 matrix elements are zero.

For files containing experimental data the data block consists of three data lines per matrix. The first line contains the 11, 12, 13, 14, 21, and 22 elements of **F**. The second line contains the 23, 24, 31, 32, 33, and 34 elements of **F**. The third lines contains the 41, 42, 43, and 44 elements of **F**.

The matrix data should be written in a nested loop where <h^2> varies more slowly than σ_ξ^2, which varies more slowly than λ_0, which varies more slowly than θ_0. The looping of each variable goes from the first value, to the last value listed in the header (lines C to F). Given this information, DISPLAY can access any of the elements of **F** on file.

AV.7.3.2 Option 12.

Option 12 displays pertinent information associated with the assignment numbers: the material name, auto-correlation function, <h^2>, σ_ξ^2, range on λ_0 and range on θ_0. This option is simply a reminder option of your choice of the assignment numbers.

AV.7.3.3 Option 13.

Option 13 is used to remove an assigned surface and free the assignment number. When the list of assigned surfaces is presented you are prompted to:

Enter the number of the surface you want to remove,
Enter 0 to return to the main menu,
Enter -99 to clear out all assignments.

For example, to remove the surface with assignment number 6, enter "6" <RETURN>. To exit without any further removals, enter "0" <RETURN>. Entering "-99" <RETURN> will eliminate all assignments. Option 13 can have an effect on options 21 and 31, should a material in the directive list or the base normalization material be deleted.

AV.7.4 PLOT DIRECTIVE LIST MODIFICATIONS.

These routines build lists of information used by the plotting options. Information on plotting options is stored so as to avoid entering repeated data for different plot options.

AV.7.4.1 Option 21.

Option 21 maintains the list of assignment numbers of the surfaces used in the plotting routines. The first element of this list is the 3-d plot feature. Up to four elements in the assignments list can produce a page of contour plots. The entire list, a maximum of seven, can be used when comparing 2-d contour plots.

The flow of Option 21 is similar to others in this group. On entry, the current list and a menu is displayed. If the list needs to be changed, just enter the number of assignment numbers for a new list. Prompts will guide you to enter a new list of assignment numbers separated by spaces on one line. These new values are checked and, if an illegal input is entered, returned to the start of this option (to reenter the correct list). A return is made to the main menu when the correct inputs are obtained.

AV.7.4.2 Option 22.

Option 22 controls the list of matrix elements used by the 2-d plotting option. Up to four matrix elements can be drawn on the same page in 2-d format (these element numbers are stored in this list). For example, to plot the matrix elements f_{11}, f_{22}, and f_{34} on one page, the list will contain "11" and "22" and "34." The method of entering a new list is the same as for Option 21.

AV.7.4.3 Option 23.

Option 23 maintains a list of wavelengths. These are the fixed wavelengths used when the f_{ij}'s are plotted versus θ_0 in the 2-d format. When a comparison is being made between surfaces, only the first wavelength is used, otherwise a comparison can be made between the wavelengths as a function of incident angle. The entry method is the same as above.

AV.7.4.4 Option 24

Option 24 is like 23, except here a list of fixed incident angles is maintained. This list is used when f_{ij} is plotted versus λ_0 in the 2-d format. The entry method is the same as above.

AV.7.5 OPTION 31, DATA ANALYSIS OPTIONS.

Two analysis techniques are available in DISPLAY. The first causes all f_{ij}'s to be divided by the dc element f_{11}. The second allows one to examine the percent difference between a target surface and a base surface, $\frac{f_{ij}^{targ}}{f_{ij}^{base}}-1$. When both target and base sample measurements are included the normalizations work in succession through division by f_{11}:

$$\frac{f_{ij}}{f_{11}},\ \frac{f_{ij}^{targ}}{f_{ij}^{base}}-1, \text{ and } \frac{f_{ij}^{targ}}{f_{11}^{targ}}\frac{f_{11}^{base}}{f_{ij}^{base}}-1.$$

To set up a desired analysis, follow the instructions given in the program.

AV.7.5.1 Division by f_{11}.

This feature is included to make the theoretical data more closely resemble normalized or non-normalized Mueller matrix representations (see Section 4.5.4). Division by f_{11} is performed as the Mueller matrices are being read in as a function of λ_0 and θ_0, before the comparison to a base material is done.

AV.7.5.2 Subtraction and Division of f_{ij}^{base}.

The percent difference simply determines when base and target surfaces, as a function of λ_0 and θ_0, are most dissimilar. This operation is performed after all the f_{ij}'s have been read in for the base and the target materials.

AV.7.6 OPTION 32, PLOT FORMAT OPTIONS.

This option allows change of various plot features including: page size, line color, tick mark density, grid density, curve thickness, and hidden line removal. The six sub-options are accessed through one menu.

The default page size is 12 by 10.5 inches. However, these dimensions have little to do with the absolute size of the plot coming off the printer. This option lets you change the relative size of the page.

For terminals that have color capabilities supported by Disspla, this program allows one to draw curves on the same set of axes in different colors. This makes comparisons easier, essential for contour plots. By default, multiple lines on the same set of axes are differentiated by line type (dots, dashes, etc.).

The number of tickmarks drawn per major division on the x and y axes can be controlled independently (for aesthetic purposes).

Dashed grid lines can be added to plots to make accurate readings easier. The number of grid lines per major division on both axes can be controlled. Too many grid lines will clutter the plot.

The thickness of curves on 2-d and contour plots can be adjusted depending on the output desired. For terminal display or paper plotting, a thickness of one is usually sufficient. For transparences, however, the thickness should be increased, especially if color is also used.

The final format option concerns hidden lines on 3-d plots. The default setting does not remove hidden lines (they can clutter the plot). This option must be executed to remove hidden line data in the 3-dimensional plot, or add them in latter plots.

AV.7.7 OPTION 41, 3-D PLOTTING.

When executed, this option draws a 3-d representation of $f_{ij}(\lambda_0,\theta_0)$. (The 3-d plots are always monochromatic.)

The assignment number of the surface used is obtained from the directive list built by Option 21. The program asks which matrix element is to be plotted (this value is not taken from the directive list), and the correct reply is row-column element numbers.

The program then requests a range of λ_0's and θ_0's for the base plane. When a legal base plane range is obtained data that fall within that range are mapped.

The last input required is the viewpoint of the plot. Disspla assumes the plot is in a cube with corners at (0,0,0) and (2,1,1) (arbitrary units). At (0,0,0), $\lambda_0 = \lambda_0(\text{min})$ and $\theta_0 = \theta_0(\text{min})$. At (2,1,1), $\lambda_0 = \lambda_0(\text{max})$ and $\theta_0 = \theta_0(\text{max})$. The bottom of the cube is $f_{ij} = f_{ij}(\text{min})$. The cube top is $f_{ij} = f_{ij}(\text{max})$. The program requires a viewpoint in terms of these arbitrary box units. Remember, the 1st coordinate is positioned on the λ_0 axis, the 2nd coordinate is positioned on the θ_0 axis, and the 3rd coordinate is positioned on the f_{ij} axis. The plot and various headings are then drawn. Hidden lines may or may not be removed, depending upon the setting in Option 32.

At this point you have three options: you can change the viewpoint for this matrix element, plot a different matrix element for the same surface and viewpoint, or return to the main menu.

AV.7.8 OPTION 42, 2-D PLOTTING.

This option is included because of the difficulty in reading values from 3-d plots, and because of the difficulty in making absolute comparison within and between plots. In general, 2-d plotting allows you to look at cross sections of 3-d plots where either λ_0 or θ_0 has been held constant. The matrix elements plotted are defined by the directive list of Option 22. Up to four matrix elements can be plotted on one page. The four possible plot types, as presented by DISPLAY, are:

1. Spectral with multiple fixed incident angles for 1 surface,
2. Spectral with fixed incident angle for 1+ surfaces,
3. Angular with multiple fixed wavelengths for 1 surface,
4. Angular with fixed wavelength for 1+ surfaces.

Plot type 1 will present $f_{ij}(\lambda_0,\theta_0^{fixed})$ for up to seven θ_0^{fixed} values on the same set of axes. The allowed θ_0^{fixed} values are defined in the directive list (Option 24). The range of λ_0's is entered after plot type 1 or 2 is selected. The first surface assignment number in the surfaces directive list is used.

Plot type 2 presents $f_{ij}(\lambda_0,\theta_0^{fixed})$ for up to seven different surfaces, defined in the directive list, on the same set of axes. θ_0^{fixed} is the first value from Option 24's directive list.

Plot type 3 presents $f_{ij}(\lambda_0^{fixed},\theta_0)$ for up to seven λ_0^{fixed} values on the axes. The allowed λ_0^{fixed} values are defined in the directive list in Option 23. The range of θ_0's is entered after plot type 3 or 4 is selected. The first surface assignment number in the surfaces directive list is used.

Plot type 4 draws $f_{ij}(\lambda_0^{fixed},\theta_0)$ for up to seven different surfaces, defined in the directive list, on the same set of axes. λ_0^{fixed} is the first value from Option 23's directive list.

Headings and legends are provided on the page to distinguish the curves when comparisons are made. Several plot features can be changed by using Option 31 before starting the plot. The process is the same as for 3-d plots. Press <RETURN> to continue.

Follow the program's instructions before leaving Option 42 to adjust the limits on the horizontal axis.

AV.7.9 OPTION 43, CONTOUR PLOTTING.

Contour plotting is included as a way to read more exact information and to compare materials, without fixing λ_0 or θ_0. This option draws nine level curves. Color is recommended for this option.

Like the 3-d option, the matrix element and range of the base grid is input at the start of this option. However, unlike the 3-d plots, up to four different surfaces defined in Option 21's directive list can be used. When multiple surfaces are used the level curves are the same for each plot. This means that not all plots will have 9 level curves.

Appropriate headings and legends are drawn to aid your interpretation of the graphs. Several plot features associated with 2-d plots are used, as defined by Option 32.

AV.8 DISPLAY SOURCE CODE.

```
C This program will plot 2-d, 3-d, and contoured representations of the
C Mueller Matrix elements as a function of wavelength and
C incident angle for theoretically calculated values and for
C experimental values when the proper input file form is used.
C Various surface materials and roughness parameters can be
C compared.
C Various normalizations will be available for visual analysis.
C
      Integer maxplt,maxcur,maxang,maxwln,maxsur,maxdat
      Parameter (maxplt=4,maxcur=7,maxang=25)
      Parameter (maxwln=71,maxsur=10,maxdat=20)
C
      Integer i,j,k,m,l1,l2
      Integer nhmeq,nsigs,nwlen(maxsur),nang(maxsur),nskip(maxsur)
      Integer numplt,numcur,ptype
      Integer fnum(maxplt),surf(maxcur),numpts(maxcur)
      Integer nangf, nwlenf
      Integer acode(maxsur)
      Integer itemp(maxcur)
      Integer elt,idx
      Integer index(maxplt),ixmn(maxplt),ixmx(maxplt)
      Integer iymn(maxplt),iymx(maxplt)
C
      Real hmeq(maxsur),hmsql(15),sigs(maxsur),sigsl(15)
      Real wlenl(maxsur,maxwln), angl(maxsur,maxang)
      Real angfx(maxcur), wlenfx(maxcur)
      Real xray(maxcur,maxwln),yray(maxcur,maxwln)
      Real ymin,ymax,xmin,xmax
      Real xray1(maxwln),yray1(maxwln)
      Real fst(maxdat,maxwln,maxang)
      Real fwrk(maxwln,maxang)
      Real rtemp(maxcur)
      Real lvl(11),delz
C
      Character*13 autonm(8),fname(maxsur),fn
      Character*13 matnm(maxsur)
C
C
      Common/one/fst
C
      Data fname/maxsur*'none'/
      Data autonm/'Gaussian','N-8','N-6','Gauss, Q only','N-8, Q only',
     *            'N-6, Q only','Pi/Q','Experimental'/
C
C
C Initialize main program variables
      nsurf=0
      nfnum=0
      nangf=0
      nwlenf=0
C
C Get the terminal being used
```

```
   30 write(6,1)
      write(6,*)' Enter the terminal you are using'
      write(6,*)'    1) Tek4115'
      write(6,*)'    2) Tek4107'
      write(6,*)'    3) HDS200'
      write(6,*)
      read(5,*)iterm
      if(iterm.lt.1.or.iterm.gt.3) goto 30
C
C Initialise main program variables and send them off to some subroutines
      call stplt2(iterm)
      call finit
      nticx=1
      nticy=1
      call stplt3(nticx,nticy)
      iflcol=0
      call setlin(iflcol)
      iflgrd=0
      ngrdx=0
      ngrdy=0
      call sgrd2d(iflgrd,ngrdx,ngrdy)
      call sgrdcn(iflgrd,ngrdx,ngrdy)
      xsize=12.
      ysize=10.5
      xscale=1.
      yscale=1.
      call sscl2d(xscale,yscale)
      call sscl3d(xscale,yscale)
      call stplt4(xscale,yscale)
      ndraw=1
      call stplt5(ndraw)
      n1=0
      n2=0
      call fnorm1(n1)
      call fnorm2(n2,' ',0,0)
      ihide=0
      call stplt6(ihide)
C
C
C********* MAIN ******************************************************
C
C Print the main menu and process the command
C
   50 write(6,1)
    1 format(/)
      write(6,*)'      PLOTTER MAIN MENU'
      write(6,*)
      write(6,*)' 11. Assign a surface for data access'
      write(6,*)' 12. Display info. on an assigned surfaces'
      write(6,*)' 13. Remove an assigned surface'
      write(6,*)' 21. Modify list of surfaces to use'
      write(6,*)' 22. Modify list of matrix elements to use'
      write(6,*)' 23. Modify list of wavelengths to use'
      write(6,*)' 24. Modify list of incident angles to use'
```

```
      write(6,*)' 31. Change analysis normalizations'
      write(6,*)' 32. Change plotting format features'
      write(6,*)' 41. Do 3-d plot'
      write(6,*)' 42. Do 2-d plot'
      write(6,*)' 43. Do Contour plot'
      write(6,*)' 99. Quit program'
      write(6,*)
      write(6,*)' Enter the number of your choice'
      read(5,*)i
      if(i.eq.11) goto 100
      if(i.eq.12) goto 200
      if(i.eq.13) goto 300
      if(i.eq.21) goto 400
      if(i.eq.22) goto 500
      if(i.eq.23) goto 600
      if(i.eq.24) goto 700
      if(i.eq.31) goto 800
      if(i.eq.32) goto 900
      if(i.eq.41) goto 1100
      if(i.eq.42) goto 1000
      if(i.eq.43) goto 1200
      if(i.eq.99) goto 1400
      goto 50
C
C
C********** Option 99 ******************
C
 1400 write(6,*)'Leaving program'
      stop
C
C********** Option 11 ******************
C
C This section finds an unused assignment number,
C reads in the filename and Q info needed to access the data later, and
C saves this information in arrays fname(),acode(),hmeq(),sigs(),nskip(),
C  wleni(), and angi().
C
  100 continue
c get assgn #
      m=1
  101 if(fname(m).eq.'none') goto 102
      m=m+1
      if(m.le.maxsur) goto 101
      write(6,1)
      write(6,*)'No free space to put surface information'
      write(6,*)'Remove a surface then try again'
      call rcont
      goto 50
C
c get filename and Q info, fn,acode(m)
  102 write(6,1)
      write(6,*)'Enter the material data file name'
      write(6,*)'Enter 0 to return to the main menu'
      read(5,'(a)')fn
```

```
      if(fn.eq.'0') goto 50
  110 write(6,1)
      write(6,*)'1. Gauss          2. N-8          3. N-6'
      write(6,*)'4. Gauss, Q only  5. N-8, Q only  6. N-6, Q only'
      write(6,*)'7. Pij/Q'
      write(6,*)'8. Experimental'
      write(6,*)
      write(6,*)'Enter the autocorrelation code for surface'
      read(5,*)i
      if(i.eq.0) goto 102
      if(i.lt.1.or.i.gt.8) goto 110
      acode(m)-i
c
c Read header info material name, allowed hmeq, sigs, ang and wlen
      write(6,*)'opening file ',fn
      open(unit-10,file-fn,status-'old',err-199)
      read(10,'(a)',end-198)matnum(m)
      read(10,*,end-198)nhmeq,nsigs,nwlen(m),nang(m)
      read(10,*,end-198)(hmeqi(i),i-1,nhmeq)
      read(10,*,end-198)(sigsi(i),i-1,nsigs)
      read(10,*,end-198)(wleni(m,i),i-1,nwlen(m))
      read(10,*,end-198)(angi(m,i),i-1,nang(m))
c
c Get users desired hmeq and sigs
c A general data file may have more than one of each
  120 write(6,1)
      write(6,*)'Allowed mean squared heights:'
      write(6,10010)(i,hmeqi(i),i-1,nhmeq)
10010 format(1x,2(i3,') ',e10.3,'  '))
      write(6, )
      write(6,*)'Enter the number of your choice'
      read(5,*)j
      if(j.lt.1.or.j.gt.nhmeq) goto 120
      hmeq(m)-hmeqi(j)
c
  130 write(6,1)
      write(6,*)'Allowed mean squared slopes:'
      write(6,10010)(i,sigsi(i),i-1,nsigs)
      write(6,*)
      write(6,*)'Enter the number of your choice'
      read(5,*)k
      if(k.lt.1.or.k.gt.nsigs) goto 130
      sigs(m)-sigsi(k)
c
c compute the number of lines to skip past header to get into the correct data
c for files generated by RETRO
      nskip(m)-2*((j-1)*nsigs+k-1)*nwlen(m)*nang(m)
c
c Print out the results of the assignment operation
      write(6,*)'Assigned surface #',m
      write(6,*)'Is from file',fn
      write(6,*)'Has mean squared height of',hmeq(m)
      write(6,*)'Has mean squared slope of',sigs(m)
      fname(m)-fn
```

```
          close(10)
        call rcont
        goto 50
C
c Error - most likely fn spelled wrong
  199 write(6,*)'An error occurred opening the file',fn
        call rcont
        goto 50
C
c Error - most likely RETRO didn't finish putting in data
  198 write(6,*)'Not enough data on unit 10 for file',fn
        call rcont
        goto 50
C
C********** Option 12 ********************
C
C This section prints out all the current assignments
C The specified autocorrelation,hmsq,sigs and stored wlen and ang ranges
C are displayed to refresh to users memory about assignments
  200 continue
        write(6,1)
        write(6,*)'     SURFACE ASSIGNMENTS'
        write(6,*)
        do 220 i=1,maxsur
          if(fname(i).eq.'none')then
            write(6,10210)i
10210 format(i3,' -- surface not assigned')
          else
            write(6,10220)i,matnm(i),autonm(acode(i))
            write(6,10230)hmsq(i),sigs(i)
            write(6,10240)wlenl(i,1),wlenl(i,nwlen(i)),
     *                    angl(i,1),angl(i,nang(i))
10220 format(i3,' -- material - ',a20,'     corr. fnc. - ',a)
10230 format(7x,'mean sq. height - ',f8.3,'  mean sq. slope - ',f8.3)
10240 format(7x,'wlen min,max ',2f8.3,'   inc ang min,max',2f7.2)
          endif
          write(6,*)
c put a page break every 5 assignment numbers
          if(i.eq.5.or.i.eq.10.or.i.eq.15) call rcont
  220 continue
        goto 50
C
C********** Option 13 ********************
C
C This routine frees assignment numbers for use with other surfaces
C This operation can affect options 21 and 31, and subroutine freads.
C The current assignments are displayed and the menu given.
C The user's option is read and executed
  300 continue
        write(6,1)
        write(6,*)'       ASSIGNED SURFACES'
        write(6,*)
        write(6,10319)'Assgn # ','Material     ','Corr. fnc.    ',
     *              ' mean sq hgt','mean sq slope'
```

```
 10319 format(1x,a8,2x,a13,2x,a13,2x,a13,2x,a13)
       do 303 i=1,maxsur
        if(fname(i).ne.'none') then
          write(6,10320)i,matnm(i),autonm(acode(i)),hmsq(i),sigs(i)
        endif
 10320 format(1x,i8,2x,a13,2x,a13,2x,f13.3,2x,f13.3)
  303 continue
       write(6,*)
  320 continue
       write(6,*)'Enter the number of the surface you want to remove'
       write(6,*)'Enter 0 to return to the main menu'
       write(6,*)'Enter -99 to clear out all assignments'
       read(5,*) i
       if(i.eq.-99) goto 390
       if(i.eq.0) goto 50
       if(i.lt.1.or.i.gt.maxsur) goto 300
C
c Free assignment number i only and update freadz
c Check to see if options 31 or 21 are affected
       fname(i)='none'
       call fupdt(i)
       if(i.eq.n2)then
        write(6,*)'Base surface for normalizations has been deleted'
        write(6,*)'Use option 31 to reassign base surface'
        write(6,*)
        n2=0
       endif
       do 380 j=1,nsurf
        if(surf(j).eq.i) then
          nsurf=0
          write(6,*)'The surfaces to use list has been cleared.'
          write(6,*)'Use main menu option 21 to rebuild'
          call rcont
        endif
  380 continue
       goto 300
C
c Free all assignment numbers and initialize freadz
c Option 21 is affected and 31 might be too
  390 do 392 i=1,maxsur
        fname(i)='none'
  392 continue
       call finit
       if(n2.ne.0) then
        write(6,*)'Base surface for normalizations has been deleted'
        n2=0
       endif
       nsurf=0
       write(6,*)'The surfaces to use list has been cleared.'
       write(6,*)'Use main menu option 21 to rebuild'
       call rcont
       goto 50
C
C********** Option 21 ******************
```

```
      C
      C The list of surfaces to be used in some 2-d and contour plots is built here
      C The 1st assignment # in the list is used by some 2-d and the 3-d routine
      C The current surfaces 'to use' list is printed
      C The menu is given, the user's choice read, and executed
       400 continue
         write(6,1)
         write(6,*)'        LIST OF SURFACES TO USE'
         write(6,*)
         write(6,10419)'List # ','Assgn #' ,'Material    ',
         *        'Corr. fnc.   ','  mean sq hgt','mean sq slope'
      10419 format(a7,2x,a8,2x,a13,2x,a13,2x,a13,2x,a13)
         do 403 i=1,nsurf
           m=surf(i)
           write(6,10420)i,m,matnm(m),autonm(acode(m)),hmsq(m),sigs(m)
      10420  format(i4,' -',i8,2x,a13,2x,a13,2x,f13.3,2x,f13.3)
       403 continue
         write(6,*)
       411 write(6,*)
         write(6,*)'Enter # of surfaces for list  or'
         write(6,*)'enter 0 to return to the main menu'
         read(5,*)j
         if(j.eq.0) goto 50
         if(j.lt.1.or.j.gt.maxsur) goto 411
      C
      c read the number of assignment numbers requested into a temp array
       420 write(6,*)
         write(6,*)'Enter the numbers of the assigned surfaces'
         write(6,*)'you want in the list.  Enter in desired order'
         write(6,*)'separated by spaces on one line.'
         write(6,*)'Enter all zeros to get out.'
         read(5,*)(itemp(i),i=1,j)
         do 430 i=1,j
           if(itemp(i).eq.0) goto 400
           if(itemp(i).lt.1.or.itemp(i).gt.maxsur) goto 420
           if(fname(itemp(i)).eq.'none') then
             write(6,*)'At least one surface is not assigned'
             goto 420
           endif
       430 continue
      C
      c If all values are legitimate copy them into storage array surf()
         nsurf=j
         do 440 i=1,nsurf
       440  surf(i)=itemp(i)
         goto 400
      C
      C********** Option 22 ********************
      C
      C This option builds the matrix elements 'to use' list that is necessary
      C to do 2-d plots with multiple mtx elements
      C The current list is presented in mtx notation F11, F12, F44 etc.,
      C however the data is stored linearly F11=f(1),F12=f(2),F44=f(16)
      C The menu is given, the choice read, and executed.
```

```
      500 continue
      write(6,1)
      write(6,*)'      LIST OF MATRIX ELEMENTS TO USE'
      write(6,*)
      do 510 i=1, nfnum
        l1=(fnum(i)-1)/4+1
        l2=fnum(i)-(l1-1)*4
        write(6,10510)i,l1,l2
10510 format(2x,i8,' = F',2i1)
  510 continue
  511 write(6,*)
      write(6,*)'Enter number of matrix elements for list or'
      write(6,*)'enter 0 to return to the main menu'
      read(5,*) j
      if(j.eq.0) goto 50
      if(j.lt.1.or.j.gt.maxplt) goto 511
C
c This reads the mtx notation for the elt's and converts to linear format
  515 write(6,*)
      write(6,*)'Enter the matrix element codes for the list'
      write(6,*)'Enter them in desired order separated by spaces'
      write(6,*)'e.g. if you want F11,F22, and F34 then enter'
      write(6,*)'11 22 34'
      write(6,*)
      read(5,*)(itemp(i),i=1,j)
      do 520 i=1,j
        if(itemp(i).eq.0) goto 500
        itemp(i)=itemp(i)-6*(itemp(i)/10)-4
        if(itemp(i).lt.1.or.itemp(i).gt.16) goto 515
  520 continue
C
c Copy the temp array into to the storage array
      nfnum=j
      do 540 i=1,nfnum
  540   fnum(i)=itemp(i)
      goto 500
C
C********** Option 23 ********************
C
C This section reads the list of fixed wavelengths to be used by
C the 2-d routine when cross sections of 3-d plots are made
C parallel to the inc. angle axis.
C The current list is given in micrometers, the menu is plotted,
C the choice read and executed
  600 continue
      write(6,1)
      write(6,*)'         WAVELENGTHS TO USE'
      write(6,*)
      do 610 i = 1,nwlenf
        write(6,10600)i,wlenfx(i)
10600 format(2x,i4,' = ',f7.3)
  610 continue
  611 write(6,*)
      write(6,*)'Enter the number of wavelengths for list or'
```

```
        write(6,*)'enter 0 to return to the main menu'
      read(5,*)j
      if(j.eq.0) goto 50
      if(j.lt.1.or.j.gt.maxcur) goto 611
c
c The desired number of wavelengths are read into a temp array
  615 write(6,*)
      write(6,*)'Enter the desired wavelengths in micro-meters'
      write(6,*)'separated by spaces on one line'
      read(5,*)(rtemp(i),i=1,j)
      do 620 i=1,j
        if(rtemp(i).eq.0.) goto 600
        if(rtemp(i).lt.0.) goto 615
  620 continue
c
c The temp array is copied if all values that are legitimate.
      nwlenf=j
      do 640 i=1,nwlenf
  640   wlenfx(i)=rtemp(i)
      goto 600
c
C********** Option 24 ******************
c
C This section reads in the list of inc angles to be used by the 2-d routine
C when cross sections parallel to the wavelength axis are being made.
C The current list is presented with the menu.
C The choice is read and executed.
  700 continue
      write(6,1)
      write(6,*)'           INCIDENT ANGLES TO USE'
      write(6,*)
      do 710 i= 1,nangf
        write(6,10700)i,angfx(i)
10700 format(2x,i4,' = ',f7.2)
  710 continue
  711 write(6,*)
      write(6,*)'Enter the number of incident angles for list or'
      write(6,*)'enter 0 to return to main menu'
      read(5,*)j
      if(j.eq.0) goto 50
      if(j.lt.1.or.j.gt.maxcur) goto 711
c
c Read values into a temp array and check them
  715 write(6,*)
      write(6,*)'Enter the desired incident angles in degrees'
      write(6,*)'separated by spaces on one line'
      read(5,*)(rtemp(i),i=1,j)
      do 720 i=1,j
        if(rtemp(i).lt.0.) goto 700
        if(rtemp(i).gt.90.) goto 715
  720 continue
c
c Copy temp array into storage if all vals OK
      nangf=j
```

```
        do 740 i=1,nangf
  740   angfx(i)=rtemp(i)
        goto 700
C
C********** Option 31 ******************
C
C Write out current set up for normalizations
C Display menu
C Get user command
  800 continue
      write(6,1)
      write(6,*)'     ANALYSIS NORMALIZATIONS'
      write(6,*)
      write(6,*)'  Current set up:'
      if(n1.eq.1) then
        write(6,*)'     Pij -> Pij/P11'
        write(6,*)'     All P11 elements scaled -> 1'
        write(6,*)
      endif
      if(n2.ne.0) then
        write(6,*)'     Pij(object) -> Pij(object)/Pij(base)-1'
        write(6,*)'     where the base surface is '
        write(6,10805)'Assgn#','Material','Auto corr','mean sq hgt',
     *              'mean sq slope'
10805 format(a8,a15,a15,a15,a15)
        write(6,10810)n2,matnm(n2),autonm(acode(n2)),hmsq(n2),sigs(n2)
10810 format(i8,a15,a15,f15.3,f15.3)
        write(6,*)
      else if(n1.eq.0) then
        write(6,*)'     No normalizations in place'
        write(6,*)
      endif
C
  810 write(6,*)'Enter 1 for Pij -> Pij/P11'
      write(6,*)'Enter 2 to cancel division by P11 element'
      write(6,*)'Enter 3 for Pij(object) -> Pij(object)/Pij(base)-1'
      write(6,*)'Enter 4 to cancel comparisons with a base surface'
      write(6,*)'Enter 0 to return to the main menu'
      write(6,*)
      read(5,*)i
      if(i.eq.0) goto 50
      if(i.lt.1.or.i.gt.4) goto 810
C
c n1=1 --> divide Pij by P11 at readin in readsf
c and pass this along through fnorm1
      if(i.eq.1) then
        n1=1
        call fnorm1(n1)
C
c n1=0 --> do NOT divide by P11
      else if(i.eq.2) then
        n1=0
        call fnorm1(n1)
C
```

```
      c n2=0 --> there is no base material
      c and reset this for fnorm2
          else if(i.eq.4) then
            n2=0
            call fnorm2(n2,' ',0,0)
      C
      c n2>0 --> there is a base material and its assignment number is n2
      c print menu and get base material's assignment number if desired.
      c pass information to fnorm2
          else if(i.eq.3) then
       820  write(6,*)
            write(6,*)'Enter the number of the assignment # of the surface'
            write(6,*)'you want to use as the base or'
            write(6,*)'Enter 0 to back up a menu'
            read(5,*)j
            if(j.eq.0) goto 810
            if(j.lt.0.or.j.gt.maxsur) goto 820
            if(fname(j).eq.'none') then
              write(6,*)'That surface is not assigned'
              goto 820
            endif
            n2=j
            call fnorm2(n2,fname(j),acode(j),nskip(j))
          endif
      c changing the normalization makes all old data obsolete (use finit)
          call finit
          goto 800
      C
      C********** Option 32 *******************
      C
      C This section allows modification of page size, color on/off,
      C tickmark density, grid lines, curve thickness, and hidden line removal.
      C The current setup is displayed in the menu.
       900 continue
          write(6,1)
          write(6,*)'    CURRENT PLOT PARAMETERS'
          write(6,*)
          write(6,10910)xsize,ysize
      10910 format(' The plot page is ',f4.1,'in. by ',f4.1,'in.')
          write(6,*)
      c
          if(iflcol.eq.1) then
            write(6,*)'Curves distinguished by color'
          else
            write(6,*)'Curves distinguished by line type'
          endif
          write(6,*)
      c
          write(6,10915)ntickx,nticky
      10915 format(' There are ',i1,' tick marks per major x-axis division',
         *  /,' There are ',i1,' tick marks per major y-axis division')
          write(6,*)
      c
          if(iflgrd.eq.1) then
```

```
        write(6,*)'Grid option for 2-d and contour plot is in effect'
       write(6,10920)ngrdx,ngrdy
 10920 format('   There are ',i1,' grid lines per major tick on the',
     *      'x-axis',/,'   There are ',i1,' grid lines per major tick',
     *      'on the y-axis')
      else
       write(6,*)'No grid lines'
      endif
      write(6,*)
c
      write(6,10930)ndraw
 10930 format(' Curves on 2-d and contour plots are drawn',i2,' times')
      write(6,*)
c
      if(ihide.eq.0) then
        write(6,*)' On 3-d plots all lines drawn'
      else
        write(6,*)' On 3-d plots hidden lines removed'
      endif
C
      write(6,*)
      write(6,*)
  920 write(6,*)'Enter 1 to change page size'
      write(6,*)'Enter 2 to change color/line-type option'
      write(6,*)'Enter 3 to change the tick mark density'
      write(6,*)'Enter 4 to modify the grid option'
      write(6,*)'Enter 5 to change curve thickness on 2-d and'//
     *      'Contour plots'
      write(6,*)'Enter 6 to modify 3-d hidden lines option'
      write(6,*)'Enter 0 to return to the main menu'
      write(6,*)
      read(5,*)i
      if(i.eq.0) goto 50
      if(i.lt.1.or.i.gt.6) goto 920
C
      write(6,*)
c This section reads in new page size xsize,ysize
c then resets the scale and passes then info via stplt4, secl2d, and secl3d
      if(i.eq.1) then
  930  write(6,*)'The normal page size is 12in by 10.5in'
        write(6,*)'Enter the x and y size of the page in inches'
        write(6,*)
        read(5,*)xsize,ysize
        if(xsize.lt.0.0.or.ysize.lt.0.0) goto 930
        xscale=xsize/12.
        yscale=ysize/10.5
        call stplt4(xscale,yscale)
        call secl2d(xscale,yscale)
        call secl3d(xscale,yscale)
C
c This option changes the way lines are distinguished
c ifcol=1 --> use color option, otherwise use line type option
c Write menu get a valid response for ifcol and pass along to setlin
      else if(i.eq.2) then
```

```
          if(iterm.eq.3) then
         write(6,*)'Color option not available on HDS200'
        else
  935    write(6,*)'Enter 1 to distinguish curves by color'
         write(6,*)'Enter 2 to distinguish curves by line-type'
         read(5,*)i
         if(i.lt.1.or.i.gt.2) goto 935
         if(i.eq.1) then
           iflcol=1
         else
           iflcol=0
         endif
         call setlin(iflcol)
        endif
c
c This allows changing the total number of tick marks between in a major divis
c Get legal values for x and y axes then pass along to stplt3
      else if(i.eq.3) then
  940   write(6,*)'Enter the number of tick marks per major division'
        write(6,*)'along the x and y axes in the allowed range 1..9'
        write(6,*)
        read(5,*)nticx,nticky
        if(nticx.lt.1.or.nticx.gt.9) goto 940
        if(nticky.lt.1.or.nticky.gt.9) goto 940
        call stplt3(nticx,nticky)
c
c This allows removing grid lines all together or controlling the number to draw
c get legal values the iflgrd, ngrdx, and ngrdy
c iflgrd=0 --> no grids otherwise use ngrdx and ngrdy as the number to use
c pass this info to sgrd2d and sgrdcn
      else if(i.eq.4) then
  945   write(6,*)'Enter 1 to turn the grid option off'
        write(6,*)'Enter 2 to turn on grid option and modify density'
        write(6,*)
        read(5,*)i
        if(i.lt.1.or.i.gt.2) goto 945
        if(i.eq.1) then
          iflgrd=0
        else
          iflgrd=1
  947     write(6,*)'Enter the number of grid lines per major division'
          write(6,*)'along the x and y axes in the allowed range 0..9'
          write(6,*)
          read(5,*)ngrdx,ngrdy
          if(ngrdx.lt.0.or.ngrdx.gt.9) goto 947
          if(ngrdy.lt.0.or.ngrdy.gt.9) goto 947
        endif
        call sgrd2d(iflgrd,ngrdx,ngrdy)
        call sgrdcn(iflgrd,ngrdx,ngrdy)
c
c This allows the user to change the number of times the curve is
c drawn over the same path.  This allows thicker curves especially for
c making transparences. Get info in ndraw and pass to stplt5.
      else if(i.eq.5) then
```

```
  950   write(6,*)'Enter the number of times the curves should be'
        write(6,*)'drawn to control thickness'
        write(6,*)'The allowed range is 1..5'
        read(5,*)ndraw
        if(ndraw.lt.1.or.ndraw.gt.5) goto 950
        call stplt5(ndraw)
c
c This allow the user to remove hidden lines from 3-d plots or to put them
c back in.  ihide=1 --> remove hidden lines.  Pass info to stplt6.
      else if(i.eq.6) then
  960   write(6,*)'Enter 1 to remove hidden lines from 3-d plots'
        write(6,*)'Enter 2 to draw all lines on 3-d plots'
        read(5,*)i
        if(i.lt.1.or.i.gt.2) goto 960
        if(i.eq.1) then
          ihide=1
        else
          ihide=0
        endif
        call stplt6(ihide)
      endif
      goto 900
c
c********** Option 41 ********************
c
c This option plots 2-d representations of data
c first make sure there is a material and a matrix element to plot
 1000 continue
      if(nsurf.lt.1) then
        write(6,*)'No surfaces in the "to use" list'
        write(6,*)'Use main menu option 21 and try again'
        call rcont
        goto 50
      endif
c
      if(nfnum.lt.1) then
        write(6,*)'No matrix elements in the to use list'
        write(6,*)'Use main menu option 22 and try again'
        call rcont
        goto 50
      endif
c
c print menu and get response
c Options 1 and 3 use only the 1st element in surfaces list but they use
c the entire angle list and wavelength list respectively to determine
c how many curves should be drawn per mtx element.
c Options 2 and 4 use the surfaces list to determine how many curves should be
c drawn per mtx element and the 1st element of the angle list and wavelength
c list respectively are used to fix that parameter to compare surfaces.
 1001 write(6,1)
      write(6,*)'Enter the type of plot you want'
      write(6,*)'  1. Spectral with multiple fixed inc. angles '
     *          //'for 1 surface'
      write(6,*)'  2. Spectral with a fixed inc. angle for 1+ surfaces'
```

```
        write(6,*)'   3. Angular with multiple fixed wavelengths '
     *      //'for 1 surface'
      write(6,*)'  4. Angular with a fixed wavelength for 1+ surfaces'
      write(6,*)
      read(5,*)ptype
      if(ptype.eq.0) goto 50
      if(ptype.lt.1.or.ptype.gt.4) goto 1001
c
c Options 1 and 2 make sure there is at least one fixed inc angle
      iflag3=0
      if(ptype.lt.3) then
       if(nangf.lt.1) then
        write(6,*)'No fixed incident angles in list'
        write(6,*)'Use main menu option 24 and try again'
        call rcont
        goto 50
       endif
c
c Options 3 and 4 make sure there is at least one fixed wavelength
      else
       if(nwlenf.lt.1) then
        write(6,*)'No fixed wavelengths in list'
        write(6,*)'Use main menu option 23 and try again'
        call rcont
        goto 50
       endif
      endif
c
c Get range for the x-axis of the plots
 1002 write(6,1)
      if(ptype.lt.3) then
       write(6,*)'Enter the range on wavelength in micro-meters'
       write(6,*)'Minimum and maximum separated by spaces on one line'
       read(5,*)xmin,xmax
       if(xmin.lt.0.0.or.xmin.ge.xmax) goto 1002
      else
       write(6,*)'Enter the range on incident angle in degrees.'
       write(6,*)'Minimum and maximum separated by spaces on one line'
       read(5,*)xmin,xmax
       if(xmin.lt.0.0.or.xmax.gt.90.0.or.xmin.ge.xmax) goto 1002
      endif
c iflag3=1 --> Plot already done once and only a change in x-range desired
      if(iflag3.eq.1) goto 1012
c
c
c Print out in words what will plotted to allow the user to back out
      write(6,1)
      write(6,*)'    PLOT SETUP'
      write(6,*)
      if(ptype.eq.1.or.ptype.eq.3) then
       m=surf(1)
       write(6,11062)'List #','Assgn #','Material','Corr. fnc.',
     *         'mean sq hgt','mean sq slope'
       write(6,11063)1,m,matnm(m),autonm(acode(m)),hmsq(m),sigs(m)
```

```
          write(6,*)
        endif
c
        write(6,11010)nfnum
        do 1003 i=1,nfnum
          l1=(fnum(i)-1)/4+1
          l2=fnum(i)-(l1-1)*4
 1003   write(6,11020)i,l1,l2
11010 format(' There will be ',i3,' mtx. elements displayed')
11020 format(i10,' -- P',2i1)
        write(6,*)
c
        if(ptype.gt.2) then
          write(6,11030)xmin,xmax
        else
          write(6,11040)xmin,xmax
        endif
11030 format(' Each mtx. element will be plotted vs. incident angle',
     *       /,' in the range from ',f5.1,' to ',f5.1,' degrees')
11040 format(' Each mtx. element will be plotted vs. wavelength',
     *       /,' in the range from ',f7.3,' to ',f7.3,' micro-meters')
c
        write(6,*)
        if(ptype.eq.1) then
          write(6,11050)nangf
          do 1004 i=1,nangf
 1004     write(6,11052)i,angfx(i)
11050 format(' For each mtx. element ',i3,' fixed incident angles',
     *       ' will be used')
11052 format(i10,' -- ',f5.1)
c
        else if(ptype.eq.3) then
          write(6,11054)nwlenf
          do 1006 i=1,nwlenf
 1006     write(6,11056)i,wlenfx(i)
11054 format(' For each mtx. element ',i3,' fixed wavelengths',
     *       ' will be used')
11056 format(i10,' -- ',f7.3)
c
        else
          write(6,11058)nsurf
          write(6,*)
          write(6,11062)'List #','Assgn #','Material','Corr. fnc.',
     *                  'mean sq hgt','mean sq slope'
          do 1008 i=1,nsurf
            m=surf(i)
            write(6,11063)i,m,matnm(m),autonm(acode(m)),hmsq(m),sigs(m)
 1008   continue
          write(6,*)
          if(ptype.eq.2) then
            write(6,11060)angfx(1)
          else
            write(6,11061)wlenfx(1)
          endif
```

```
          endif
 11058 format('  For each mtx. element ',i3,' surfaces will be used.')
 11060 format('  The fixed angle of incidence will be ',f5.1)
 11061 format('  The fixed wavelength will be ',f7.3)
 11062 format(a8,a8,2x,a13,2x,a13,2x,a13,2x,a13)
 11063 format(i4,' - ',i8,2x,a13,2x,a13,2x,f13.3,2x,f13.3)
c
c if the plot set up is correct continue otherwise back up
       write(6,*)
 1009  write(6,*)'Enter 1 if this is correct'
       write(6,*)'Enter 0 to return to the main menu'
       read(5,*)i
       if(i.eq.1) goto 1010
       if(i.eq.0) goto 50
       goto 1009
c
c
c numplt is the number of mtx elements to be use (up to 4)
c numcur is the number of curves per mtx elt dependent upon plot type (ptype)
 1010  continue
       numplt=nfnum
       if(ptype.eq.1) then
        numcur=nangf
       else if(ptype.eq.3) then
        numcur=nwlenf
       else
        numcur=nsurf
       endif
c
c iflag1=-1 —> only one surface material at one roughness scale
c iflag1=0  —> comparison of roughness parameters for one material
c iflag1=1  —> comparison of materials and roughness is assumed to be same
c iflag2=-1 —> only one fixed inc angle or fixed wavelength is used
c iflag2=0  —> multiple curves correspond to 1+ fixed wavelengths
c iflag2=1  —> multiple curves correspond to 1+ fixed inc angles
       if(numcur.eq.1) then
        iflag1=-1
        iflag2=-1
       else
        if(ptype.eq.2.or.ptype.eq.4) then
         if(matnm(surf(1)).eq.matnm(surf(2)).and.
     *    acode(surf(1)).eq.acode(surf(2))) then
          iflag1=1
         else
          iflag1=0
         endif
         iflag2=-1
        else
         if(ptype.eq.1) then
          iflag2=1
         else
          iflag2=0
         endif
         iflag1=-1
```

```
          endif
      endif
C
c This is the section where the actual plotting is called for
 1012 call stplot
      m=surf(1)
c this call draws heading display in the middle of the page
      call head2d(numplt,matnm(m),autonm(acode(m)),hmeq(m),
     *          sigs(m),wlenfx(1),angfx(1),iflag1,iflag2,ptype)
c this section decides which, if any, legends need to be drawn to
c distinguish the curves from one another
      if(iflag1.eq.0) then
        call lgmat(numcur,matnm,maxsur,surf,acode,autonm)
      else if(iflag1.eq.1) then
        call lgruf(numcur,hmeq,sigs,maxsur,surf)
      else if(iflag2.eq.0) then
        call lgwln(numcur,wlenfx)
      else if(iflag2.eq.1) then
        call lgang(numcur,angfx)
      endif
c if a base normalization is in effect then give its stats in a legend
      if(n2.ne.0) then
        call lgbase(matnm(n2),autonm(acode(n2)),hmeq(n2),sigs(n2))
      endif
C
c This section calls the data management routines to read in data files.
c one or more calls may be necessary to freeds
      if(ptype.eq.2.or.ptype.eq.4) then
        do 1013 i=1,nsurf
          m=surf(i)
          call freeds(m,nfnum,fnum,fname(m),acode(m),nskip(m))
 1013   continue
      else
        m=surf(1)
        call freeds(m,nfnum,fnum,fname(m),acode(m),nskip(m))
      endif
C
c this is the main loop for each mtx element plot draw all curves in succession
c curve extracts the curve data from storage, interpolates as necessary, and
c puts the values to be plotted in xray and yray.
c The first index in xray(..) and yray(..) is the curve number
      do 1020 i=1,numplt
        if(ptype.eq.2.or.ptype.eq.4) then
          do 1025 j=1,numcur
            m=surf(j)
            call curve(j,j,m,fnum(j),angl,wlenl,nang(m),nwlen(m),
     *              angfx,wlenfx,ptype,numpts,xray,yray)
 1025     continue
        else
          m=surf(1)
          call curve(1,numcur,m,fnum(i),angl,wlenl,nang(m),nwlen(m),
     *            angfx,wlenfx,ptype,numpts,xray,yray)
        endif
C
```

Appendix V

```
c This section finds min and max y value and sets up limits for plots
c Then the physical origin and scale of plot is set
c the axes are drawn and labeled
      call rng2d(numcur,xray,xmin,xmax,yray,numpts,maxcur,maxwin,
     *          ymin,ymax)
      call prtn2d(numplt,i)
      call axes2d(ptype,fnum(i),xmin,xmax,ymin,ymax,n1,n2)
      do 1030 j=1,numcur
c for each curve change the line's appearance is changed and the curve drawn
        call liness(j)
        call dolt2d(j,numpts(j),xray,yray,xray1,yray1,maxcur,maxwin)
 1030   continue
      call engr2d
 1020 continue
      call enplot
C
 1040 write(6,*)'Enter 1 to change x-axis range for same plot format'
      write(6,*)'Enter 0 to return to the main menu'
      read(5,*)i
      if(i.eq.1) goto 1002
      if(i.eq.0) goto 50
      goto 1040
C
C********** Option 42 ********************
C
C This section does 3-d plotting
c make sure there is a surface to use
 1100 continue
      if(nsurf.lt.1) then
        write(6,*)'No surfaces in the to use list'
        write(6,*)'Use main menu option 21 and try again'
        call rcont
        goto 50
      endif
      iflag1=0
C
c get the desired mtx element
 1101 write(6,1)
      write(6,*)'Enter the mtx element position you want using'
      write(6,*)'the matrix notation'
      write(6,*)'e.g. if you want P11, P22, or P34 then enter'//
     *        ' 11 or 22 or 34'
      write(6,*)'Enter 0 to return to the main menu'
      read(5,*)i
      if(i.eq.0) goto 50
      elt=i-6*(i/10)-4
      if(elt.lt.1.or.elt.gt.16) goto 1101
c iflag1=1 --> view pt and base range stays the same but mtx element changes
c skip unnecessary menus
      if(iflag1.eq.1) goto 1128
C
 1102 iflag2=0
      write(6,1)
      write(6,11110)i
```

```
 11110 format(' This plot will be for mtx. element F',i2)
      write(6,*)
c
      write(6,11062)'List #','Assgn #','Material','Corr. fnc.',
     *          'mean sq hgt','mean sq slope'
      m=surf(1)
      write(6,11063)1,m,matnm(m),autonm(acode(m)),hmsq(m),sigs(m)
      write(6,*)
c
c get range for x-axis which has wavelength info
 1115 write(6,1)
      write(6,*)'Enter the range on wavelength (min max)'
      write(6,*)'Enter 1 1 for auto ranging'
      read(5,*)xmin,xmax
      if(xmin.eq.xmax) goto 1120
      if(xmin.lt.0.0.or.xmin.ge.xmax) goto 1115
c
c get range for y-axis which has the inc angle info
 1120 write(6,*)'Enter the range on incident angle (min max)'
      write(6,*)'Enter 1 1 for auto ranging'
      read(5,*)ymin,ymax
      if(ymin.eq.ymax) goto 1125
      if(ymin.lt.0.0.or.ymax.gt.90.0.or.ymin.ge.ymax) goto 1120
c
c get view point in terms of the plot box size and location
 1125 write(6,*)'Enter the vu point (xpos,ypos,zpos) in box units'
      write(6,*)'The display box has corners at (0,0,0) and (2,1,1)'
      write(6,*)'Box point (0,0,0) has values (x,y,z)=(xmin,ymin,zmin)'
      write(6,*)'Box point (2,1,1) has values (x,y,z)=(xmax,ymax,zmax)'
      write(6,*)'x = wavelength,  y = inc. ang,  z = Fij'
      write(6,*)'Enter 0 0 0 to back up'
      write(6,*)'Enter -3 3 3 for normal viewing'
      read(5,*)vx,vy,vz
      if(vx.eq.0.0.and.vy.eq.0.0.and.vz.eq.0.) goto 1102
      if((0.le.vx.and.vx.le.2.).and.(0.le.vy.and.vy.le.1.).and.
     *  (0.le.vz.and.vz.le.1.)) then
        write(6,*)'Do not put the view point inside the display box'
        write(6,*)
        goto 1125
      endif
c iflag2=1 --> base range and mtx elt stayed the same but the view pt changed
c skip unnecessary operations
      if(iflag2.eq.1) goto 1150
c
 1109 write(6,*)'Enter 1 if this is correct'
      write(6,*)'Enter 0 to return to the main menu'
      read(5,*)i
      if(i.eq.1) goto 1128
      if(i.eq.0) goto 50
      goto 1109
c
c
c this makes sure the proper data is available
 1128 m=surf(1)
```

```
        j-fnum(1)
      fnum(1)-elt
      call freadz(m,1,fnum,fname(m),acode(m),nskip(m))
      fnum(1)-j
      call ffind(m,elt,idx)
      write(6,*)'m,elt,idx',m,elt,idx
c
c this determines which data pts are needed to cover the user's range,
c otherwise autoranging takes all data points.
      if(xmin.eq.xmax) then
        ixmin-1
        ixmax-nwlen(m)
c
c otherwise find smallest x range that covers the user's range
      else
        ixmin-1
 1160   if(xmin.gt.wlenk(m,ixmin).and.ixmin.lt.nwlen(m)) then
          ixmin-ixmin+1
          goto 1160
        endif
        ixmax-nwlen(m)
 1162   if(xmax.lt.wlenk(m,ixmax).and.ixmax.gt.1) then
          ixmax-ixmax-1
          goto 1162
        endif
      endif
      if(ixmin.gt.1) ixmin-ixmin-1
      if(ixmax.lt.nwlen(m)) ixmax-ixmax+1
c reset range to correspond to plot
c DISSPLA should support an unevenly space base grid through the SURTRN
c command but I couldn't get it to work so the xmin and xmax range need
c to be reset so even base grid spacing can be used.
      xmin-wlenk(m,ixmin)
      xmax-wlenk(m,ixmax)
c
c makes sure there are enough pts to make a decent looking plot
      write(6,*)'ixmax,ixmin',ixmax,ixmin
      write(6,*)'xmin,xmax',xmin,xmax
      if((ixmax-ixmin).lt.6) then
        write(6,*)'Not enough wavelengths for a good plot'
        write(6,*)
        if(ixmin.eq.1.and.ixmax.eq.nwlen(m)) goto 50
        goto 1115
      endif
c
c do the same thing as the x-axis
      if(ymin.eq.ymax) then
        iymin-1
        iymax-nang(m)
c
      else
        iymin-1
 1164   if(ymin.gt.angl(m,iymin).and.iymin.lt.nang(m)) then
          iymin-iymin+1
```

```
          goto 1164
        endif
        iymax=nang(m)
 1166   if(ymax.lt.angl(m,iymax).and.iymax.gt.1) then
          iymax=iymax-1
          goto 1166
        endif
      endif
      if(iymin.gt.1) iymin=iymin-1
      if(iymax.lt.nang(m)) iymax=iymax+1
      ymin=angl(m,iymin)
      ymax=angl(m,iymax)
c
c makes sure there are enough pts to make a decent looking plot
      if((iymax-iymin).lt.6) then
        write(6,*)'Not enough incident angles for a good plot'
        write(6,*)
      if(iymin.eq.1.and.iymax.eq.nang(m)) goto 50
        goto 1120
      endif
c
c find the min and max of Fij then set up limits
      call rng3d(idx,ixmin,ixmax,iymin,iymax,zmin,zmax)
      write(6,*)'zmin,zmax',zmin,zmax
      call setlim(xmin,xmax,x1,x2,10.)
      call setlim(ymin,ymax,y1,y2,10.)
      call setlim(zmin,zmax,z1,z2,10.)
      write(6,*)'x1,x2,y1,y2,z1,z2'
      write(6,*)x1,x2,y1,y2,z1,z2
c
 1150 call stplot
c draw center caption
      call head3d(1,matnm(m),autonm(acode(m)),hmsq(m),sigs(m)
     *           ,elt,n1,n2)
      if(n2.ne.0) then
        call lgbase(matnm(n2),autonm(acode(n2)),hmsq(n2),sigs(n2))
      endif
c set up plot area, draw the axes, and then do the plot
      call prtn3d(vx,vy,vz)
      call axes33(x1,x2,y1,y2,z1,z2)
      call doit3d(ixmin,ixmax-ixmin+1,iymin,iymax-iymin+1,
     *           angl,winl,m,idx,fwrk)
      call enplot
c
c Allow various changes without going to then main menu
 1180 write(6,1)
      write(6,*)'Enter 1 for another view point'
      write(6,*)'Enter 2 for a different mtx element'
      write(6,*)'Enter 0 to return to the main menu'
      read(5,*)i
      if(i.eq.1) then
        iflag2=1
        goto 1125
      else if(i.eq.2) then
```

```
          iflag1=1
        goto 1101
      else if(i.eq.0) then
        goto 50
      endif
      goto 1180
C
C*********** Option 43 ********************
C
C This section does contour plotting
c make sure there is a surface to use
 1200 continue
      if(nsurf.lt.1) then
        write(6,*)'No surfaces in the to use list'
        write(6,*)'Use main menu option 21 and try again'
        call rcont
        goto 50
      endif
C
c get mtx element data
 1201 write(6,1)
      write(6,*)'Enter the mtx element position you want using'
      write(6,*)'the matrix notation'
      write(6,*)'e.g. if you want F11, F22, or F34 then enter'//
     *          ' 11 or 22 or 34'
      write(6,*)'Enter 0 to return to the main menu'
      read(5,*)i
      if(i.eq.0) goto 50
      elt=i-6*(i/10)-4
      if(elt.lt.1.or.elt.gt.16) goto 1201
C
c print what the plot will be of
 1202 continue
      write(6,1)
      write(6,12110)i
12110 format(' These plots will be for mtx. element F',i2)
      write(6,*)
C
c Use no more than the 1st four elements of surfaces list
      if(nsurf.gt.4) then
        numplt=4
      else
        numplt=nsurf
      endif
C
      write(6,12119)'Assgn # ',' Material ',' Corr. fnc. ',
     *          ' mean sq hgt ','mean sq slope'
12119 format(a8,2x,a13,2x,a13,2x,a13,2x,a13)
      do 1203 l=1,numplt
        m=surf(l)
        write(6,12120)m,matnm(m),autonm(acode(m)),hmsq(m),sigs(m)
12120   format(i8,2x,a13,2x,a13,2x,f13.3,2x,f13.3)
 1203 continue
      write(6,*)
```

```
c
c get wavelength and inc angle range
 1215 write(6,1)
      write(6,*)'Enter the range on wavelength (min max)'
      read(5,*)xmin,xmax
      if(xmin.lt.0.0.or.xmin.ge.xmax) goto 1215
c
 1220 write(6,*)'Enter the range on incident angle (min max)'
      read(5,*)ymin,ymax
      if(ymin.lt.0.0.or.ymax.gt.90.0.or.ymin.ge.ymax) goto 1220
c
c let the user out if he sees a mistake
 1228 write(6,*)'Enter 1 if this is correct'
      write(6,*)'Enter 0 to return to the main menu'
      read(5,*)i
      if(i.eq.1) goto 1229
      if(i.eq.0) goto 50
      goto 1228
c
c read in the data for all the plots
c find the base grid bounded by the user's range
c Find zmax and zmin over all the plots
 1229 zmax--1.e36
      zmin-1.e36
      do 1269 l-1,numplt
       m-surf(l)
       j-fnum(1)
       fnum(1)-elt
       call freadz(m,1,fnum,fname(m),acode(m),nskip(m))
       fnum(1)-j
       call ffind(m,elt,idx)
       write(6,*)'m,elt,idx',m,elt,idx
       txmin-1
 1260  if(xmin.gt.wlenk(m,txmin).and.txmin.lt.nwlen(m)) then
        txmin-txmin+1
        goto 1260
       endif
       txmax-nwlen(m)
 1262  if(xmax.lt.wlenk(m,txmax).and.txmax.gt.1) then
        txmax-txmax-1
        goto 1262
       endif
c
c contour plots look bad without a lot of base points
      write(6,*)'txmax,txmin',txmax,txmin
      write(6,*)'xmin,xmax',xmin,xmax
      if((txmax-txmin).lt.10) then
       write(6,*)'Not enough wavelengths for a good plot'
       write(6,*)
       if(txmin.eq.1.and.txmax.eq.nwlen(m)) goto 50
       goto 1215
      endif
c
       tymin-1
```

```
 1264  if(ymin.gt.angl(m,iymin).and.iymin.lt.nang(m)) then
         iymin=iymin+1
         goto 1264
       endif
       iymax=nang(m)
 1266  if(ymax.lt.angl(m,iymax).and.iymax.gt.1) then
         iymax=iymax-1
         goto 1266
       endif
c
      if((iymax-iymin).lt.10) then
       write(6,*)'Not enough incident angles for a good plot'
       write(6,*)
      if(iymin.eq.1.and.iymax.eq.nang(m)) goto 50
       goto 1220
      endif
c
c this call gets min and max for one plot only
      call rng3d(idx,ixmin,ixmax,iymin,iymax,zmin2,zmax2)
      write(6,*)'zmin2,zmax2',zmin2,zmax2
      if(zmin2.lt.zmin) zmin=zmin2
      if(zmax2.gt.zmax) zmax=zmax2
c save base grid ranges for each plot separately
      index(i)=idx
      ixmn(i)=ixmin
      ixmx(i)=ixmax
      iymn(i)=iymin
      iymx(i)=iymax
 1269 continue
c
c set up limits and decide what 9 common contours to use on all the plots
      call setlim(xmin,xmax,x1,x2,10.)
      call setlim(ymin,ymax,y1,y2,10.)
      call setlim(zmin,zmax,z1,z2,10.)
      write(6,*)'x1,x2,y1,y2,z1,z2'
      write(6,*)x1,x2,y1,y2,z1,z2
      delz=(z2-z1)/10.
      do 1270 i=1,9
       lvl(i)=delz*(i)+z1
 1270 continue
c
      call stplot
c same heading as 3d plots
      call head3d(nsurf,matnm(m),autonm(acode(m)),hmeq(m),sigs(m)
     *            ,elt,n1,n2)
c put a legend for the contour lines in
      call lgcn(lvl)
      if(n2.ne.0) then
       call lgbase(matnm(n2),autonm(acode(n2)),hmeq(n2),sigs(n2))
      endif
c do each plot in sequence but use a common set of contour levels
      do 1280 i=1,numplt
       ns=surf(i)
       call prtn2d(numplt,i)
```

```
        call axescn(x1,x2,y1,y2,nsurf,matnm(m),autonm(acode(m))
     *           ,hmsq(m),sigs(m))
      call dolkcn(ixmn(l),ixmx(l)-ixmn(l)+1,iymn(l),iymx(l)-iymn(l)+1,
     *           angl,wlenl,m,index(l),lvl,z1,z2)
       call engr2d
 1280 continue
      call enplot
      goto 50
C
C In the main segment of this program no calls have been made directly
C to DISSPLA.  This means another plotting library can be used if the
C subroutines are rewritten to perform the same tasks as described
C after this.  Those subroutines with direct calls to DISSPLA are noted
C
      end
C********** End of MAIN ********************
C
C
C********** Subroutine rcont **************************************************
C
      subroutine rcont
c let the user look at something and prompt him to continue
      write(6,*)
      write(6,*)'enter <RETURN> to continue'
      read(5,*)
      write(6,*)
      return
      end
C
C
C********** Subroutines freadz, fnorm2, fnorm1, finit, fupdt ********************
C
C
C This routine has NO Disspla calls
C
C These subroutines along with ffind make up the data management section
C Since the data files can be very long this section stores only the
C requested information.  Instead of storing all Fij for each surface
C only selected mtx elts are stored.
C Storage is in array fst(index,iang,iwin)
C index is the number assigned by freadz and can be determined for a
C given surface assignment number and a given mtx elt by ffind
C Up to maxdat indices are available.  If fst fills up then the oldest
C index is removed to make room for the next one.
C For each index the surface assignment number, the mtx elt num, and age are saved
C in arrays surfl, eltl, and agel.
C
      subroutine freadz(m,nelt,fnum,fname,acode,nskip)
      integer maxplt,maxcur,maxang,maxwin,maxsur,maxdat
      integer dacode,acode2,dnskip,nskip2
      parameter(maxplt=4,maxcur=7,maxang=25)
      parameter(maxwin=71,maxsur=10,maxdat=20)
      integer m,index(maxplt),nelt
      integer fnum(maxplt),elt(maxplt),acode,nskip
```

```
      Integer surfi(maxdat),elti(maxdat),agei(maxdat)
      Integer i,j,date,oldest,idx
      Integer n1,n2,dn1,dn2,mtemp
      Character*13 fname,dfname,fname2
      Common/four/surfi,elti
      save
C
C
c if a base material is desired its values are stored with assignment # 99
c so the info can be used for subtractions elsewhere
      if(n2.ne.0) then
       iflag1-1
       mtemp-m
       m-99
      endif
 2112 continue
C
      k-0
c every access increase date so that age of an index can be determined
      date-date+1
c see if requested read already exists
      do 2110 i-1,nelt
       idx-0
       do 2115 j-1,maxdat
        if(surfi(j).eq.m.and.elti(j).eq.fnum(i)) idx-j
 2115  continue
C
c only read if you have to and put data into the oldest index
c also k mtx elements can be read in one shot so
c more than one index may be available
      if(idx.eq.0) then
       k-k+1
       elt(k)-fnum(i)
       oldest-10000
       do 2130 j-1,maxdat
        if(agei(j).lt.oldest) then
         oldest-agei(j)
         index(k)-j
        endif
 2130  continue
c store indexing information and date it
       surfi(index(k))-m
       elti(index(k))-fnum(i)
       agei(index(k))-date
      endif
 2110 continue
C
c k is then number of mtx elts to read
      if(k.gt.0) then
       if(iflag1.eq.1) then
c if reading base data don't do the base normalization because you'd get 0
         call readelf(index,elt,k,fname2,acode2,nskip2,n1,0)
       else
c otherwise do the reading with the appropriate normalizations
```

```
          call readef(index,elt,k,fname,acode,nskip,n1,n2)
        endif
      endif
c
c if base data read in then go back and do the data for plotting
      if(iflag1.eq.1) then
        iflag1=0
        m=mtemp
        goto 2112
      endif
      return
c
c
c this entry accepts new base normalization info and saves it for freadz
      entry fnorm2(dn2,dfname,dacode,dnskip)
      n2=dn2
      fname2=dfname
      acode2=dacode
      nskip2=dnskip
      return
c
c
c this entry accepts new flag for F11 normalization and saves it
      entry fnorm1(dn1)
      n1=dn1
      return
c
c
c this entry is used to throw away all old information when normalizations
c have been changed or at startup to clear the memory
      entry finit
      date=0
      do 2140 i=1,maxdat
        surfi(i)=-1
        elti(i)=-1
        agei(i)=-1
 2140 continue
      return
c
c
c this entry is used to throw away information pertaining to a given
c assignment number that has been deleted.
      entry fupdt(m)
      do 2150 i=1,maxdat
        if(surfi(i).eq.m) then
          surfi(i)=-1
          elti(i)=-1
          agei(i)=-1
        endif
 2150 continue
      return
      end
c
c
```

```
C********** Subroutine ffind ************************************************
C
C This routine has NO Disspla calls
C
C This routine interrogates indexing info to find the index for a given
C assignment # and mtx elt combination
      subroutine ffind(m,elt,index)
      Integer maxplt,maxcur,maxang,maxwin,maxsur,maxdat
      parameter(maxplt-4,maxcur-7,maxang-25)
      parameter(maxwin-71,maxsur-10,maxdat-20)
      Integer m,elt,index,k
      Integer surfi(maxdat),elti(maxdat)
      Common/four/surfi,elti
C
c find match to requested data
      do 2120 k-1,maxdat
       if(surfi(k).eq.m.and.elti(k).eq.elt) then
         index-k
         return
       endif
 2120 continue
      index-0
      return
      end
C
C
C********** Subroutine readef ***********************************************
C
C This routine has NO Disspla calls
C
C This subroutine reads in data into array fst(...) using the given
C index list, elt list, filename, correlation code, and normalizations provided
C Two reading loops are provided to read in experimental and theoretical data
C
      subroutine readef(index,elt,nelt,fname,acode,nskip,n1,n2)
      integer maxplt,maxcur,maxang,maxwin,maxsur,maxdat
      parameter(maxplt-4,maxcur-7,maxang-25)
      parameter(maxwin-71,maxsur-10,maxdat-20)
      Integer index(maxplt),elt(maxplt),nelt
      Integer i,lang,'wlen,nskip
      Integer nh,ns,nwlen,nang
      Integer acode,n1,n2,idx2
      Real fst(maxdat,maxwin,maxang)
      Real a,b,c,d,e,g,dum(16),q(3)
      Character*13 fname
C
      Common/one/fst
C
c read in the header information
      open(11,file-fname,status-'old')
      read(11,*,end-911)
      read(11,*,end-911)nh,ns,nwlen,nang
      read(11,*,end-911)(x,i-1,nh)
      read(11,*,end-911)(x,i-1,ns)
```

```
         read(11,*,end=911)(x,i=1,nwlen)
      read(11,*,end=911)(x,i=1,nang)
c
c skip over the proper number of data lines for theoretical data
      if (nskip.gt.0) then
      do 2010 i=1,nskip
        read(11,*,end=911)
 2010 continue
      endif
c
c read in exp data
c read extra lines for experimental data
c read in 16 values on 3 lines for each iwlen and iang
c save only the desired ones into fst(...)
      if(acode.eq.8) then
        do 2011 i=1,nskip/2
          read(11,*,end=911)
 2011   continue
        do 2020 iwlen=1,nwlen
          do 2020 iang=1,nang
            read(11,*,end=911)(dum(i),i=1,16)
            do 2020 i=1,nelt
              fst(index(i),iwlen,iang)=dum(elt(i))
 2020       continue
c
c read in theoretical data.
c for RETRO produced files these 8 elts are assumed 0.
c read in 3 Q values and 6 unique elts for each iwlen and iang.
c for P11 normalization the Q values cancel out.
c a Q<0 is assumed --> 0.
c do P11 normalization.
c combine Q's back into the elts.
c save only desired mtx elts in fst(...).
      else
        dum(3)=0.
        dum(4)=0.
        dum(7)=0.
        dum(8)=0.
        dum(9)=0.
        dum(10)=0.
        dum(13)=0.
        dum(14)=0.
c
        do 2040 iwlen=1,nwlen
          do 2040 iang=1,nang
            read(11,*,end=911)q(1),q(2),q(3)
c
          if(n1.eq.1) then
            q(1)=1.
            q(2)=1.
            q(3)=1.
          else
            if(q(1).lt.0.) q(1)=0.
            if(q(2).lt.0.) q(2)=0.
```

```
            if(q(3).lt.0.) q(3)=0.
         endif
          read(11,*,end=911)a,b,c,d,e,g
c
          if(n1.eq.1) then
           b=b/a
           c=c/a
           d=d/a
           e=e/a
           g=g/a
           a=1.
          endif
c
          dum(1)=a
          dum(2)=b
          dum(5)=b
          dum(6)=c
          dum(11)=d
          dum(12)=e
          dum(15)=-e
          dum(16)=g
c
          do 2040 i=1,nelt
           if(acode.lt.4) then
            fat(index(i),iwlen,iang)=q(acode)*dum(elt(i))
           else if(acode.lt.7) then
            fat(index(i),iwlen,iang)=q(acode-3)
           else
            fat(index(i),iwlen,iang)=dum(elt(i))
           endif
 2040   continue
        endif
c
c do base normalizations
c find where the base data is stored
c go through and do subtractions
        if(n2.ne.0) then
         do 2050 i=1,nelt
          call ffind(99,elt(i),idx2)
          do 2050 j=1,nwlen
           do 2050 k=1,nang
      fat(index(i),j,k)=(fat(index(i),j,k)-fat(idx2,j,k))/fat(index2,j,k)
 2050   continue
        endif
c
c close and rewind file
        close(11)
        return
c
c print out an error in reading data but don't kill program
 911  write(6,*)'not enough data on unit 11'
        write(6,*)'returning to program anyway'
        write(6,*)'proceed at your own risk'
        call rcont
```

```
        return
      end
C
C
C********** Subroutines stplot, stplt2, stplt3, stplt4, stplt5, stplt6 **********
C
C This routine has DISSPLA routines
C
C stplot nominates the terminal, sets page, and many plot features
C stplt2, stplt3, stplt4, stplt5, and stplt6 pass info to stplot
      subroutine stplot
      Integer iterm,diterm,ntckx,dntckx,ntcky,dntcky,dndraw,ndraw
      Integer ihide,dihide
      Real xsc,ysc,xscale,yscale
      save
C
c nominate 1 of three terminal types
      if(iterm.eq.1) then
        call tk41(4115)
      else if(iterm.eq.2) then
        call tk41(4107)
      else if(iterm.eq.3) then
        call vt240
      endif
C
c st a new page, send disspla output to fort.7, set up page, scale for headings
c and legends, setup delimiters for embedded commands for messag ,
c set up macros to do mean square height and mean square slope symbols
c set up tick marks, curve thickness, and set hidden line removal status
c
      call bgnpl(0)
      call setdev(7,7)
      call page(12.*xsc,10.5*ysc)
      call bscale(xsc,ysc)
      call nochek
      call compbx
      call basalf('STANDARD')
      call mixalf('INSTRUCTION')
      call s1use('M11)'//char(37)//'(M1)H(EH.5)2(EXHXM11)'//
     *          char(38)//'(MX',29)
      call s2use('M7)S(MXH.5L.25)S(B1LXHXEH.5)2(EXHXMX',33)
      call xticks(ntckx)
      call yticks(ntcky)
      call thicrv(ndraw)
      if(ihide.eq.0) call nohide
      return
C
C
c pass in terminal code
      entry stplt2(diterm)
      iterm=diterm
      return
C
C
```

```
c pass in tick mark density
      entry stplt3(dntckx,dntcky)
      ntckx=dntckx
      ntcky=dntcky
      return
C
C
c pass in plot scaling
      entry stplt4(xscale,yscale)
      xsc=xscale
      ysc=yscale
      return
C
C
c pass in curve thickness parameter
      entry stplt5(dndraw)
      ndraw=dndraw
      return
C
C
c pass in hidden line removal status
      entry stplt6(dihide)
      ihide=dihide
      return
      end
C
C
C********** Subroutine enplot ***********************************************
C
C This routine has DISSPLA calls
C
C this routine end the current page layout
      subroutine enplot
      call endpl(0)
      close(7)
      call rcont
      return
      end
C
C
C********** Subroutine head2d ***********************************************
C
C This routine has DISSPLA calls
C
C This routine puts the center heading on the page
C if1 and if2 correspond to iflag1 and iflag2 in option 41
C
      subroutine head2d(numplt,matnm,autonm,hmsq,sigs,
     *                wlen,ang,if1,if2,ptype)
      Integer numplt,if1,if2,ptype
      Real hmsq,sigs,wlen,ang
      Character*13 matnm
      Character*13 autonm
      Character*1 achar
```

```
C
C
      if(numplt.eq.1) then
        schar=' '
      else
        schar='S'
      endif
C
c set up plot area and location to be ready for messag commands
      call complx
      call physor(0.,0.)
      call area2d(10.5,9.0)
C
c put in 1st and 2nd lines using the self centering headin routines
c 1st line is main title
      call headin('BACKSCATTER MUELLER MATRIX ELEMENT'//schar,35,1.6,4)
c 2nd line is the material used line
c indicating a comparison
      if(if1.eq.0) then
        call headin('Comparison of Materials*',100,1.3,4)
c or the actual one used
      else
        call headin(matnm//'    '//autonm//'*',100,1.3,4)
      endif
      call headin(' *',100,1.3,4)
      call headin(' *',100,1.3,4)
C
c put in a line describing rough parameters for this page
c either a statement of comparison or
      if(if1.eq.1) then
        call messag('(H1.3)Comparison of Roughness Parameters*',
     *              100,3.0,9.5)
c the actual values used
      else
        call messag('(H1.3MXZ1) - *',100,3.7,9.5)
        call realno(hmeq,104,'ABUT','ABUT')
        call messag('(HXM7)M(M1)M(EH.5)2(EXHXMX)*',100,'ABUT','ABUT')
        call messag('(H1.3)      (Z2) - (HXMX)*',100,'ABUT','ABUT')
        call realno(sigs,104,'ABUT','ABUT')
      endif
C
c the 4th line provides info about the fixed inc ang or fixed wavelength
      if(if2.eq.1) then
        call messag('(H1.3)Comparison of Incident Angles*',100,3.5,9.2)
c
      else if(if2.eq.0) then
        call messag('(H1.3)Comparison of Wavelengths*',100,3.7,9.2)
c
c put in the inc ang if only one used
      else
        if(ptype.lt.3) then
          call messag('(H1.3M7)Q(M1H.5L)0(LXHXMX) - *',100,4.8,9.2)
          call realno(ang,104,'ABUT','ABUT')
          call messag('(M1EH.5)O(MXEXHX)*',100,'ABUT','ABUT')
```

```
C
c put in the wavelength if only one used
      else
       call messag('(H1.3M7)L(M1H.5L)0(LXHXMX) - *',100,4.8,9.2)
       call realno(wlen,104,'ABUT','ABUT')
       call messag('(H.8M7)M(M1)M(HXMX)*',100,'ABUT','ABUT')
      endif
     endif
c reset so that next routine can have its own sub plot area
     call endgr(0)
     return
     end
C
C
C********** Subroutine lgmat **********************************************
C
C This routine has DISSPLA calls
C
C This routine prints the list of materials/corr. fnc used in a comparison
C
      subroutine lgmat(numcur,matnm,maxsur,surf,acode,autonm)
      integer numcur,i,surf(numcur),maxsur,acode(maxsur)
      Character*13 matnm(maxsur)
      Character*13 autonm(5)
C
c define a subplot area and display the info, changing color or line type
      call physor(0.,0.)
      call area2d(10.5,9.0)
      call messag('(H1.3)Materials*',100,9.9,9.85)
      do 3010 i=1,numcur
       call lineaz(i)
       call strtpt(8.6,9.9-.2*i)
       call connpt(9.2,9.9-.2*i)
       call reset('DASH')
       call messag(matnm(surf(i))//'  '//autonm(acode(surf(i)))//
     *            '*',100,9.3,9.8-.2*i)
 3010 continue
      call reset('SETCLR')
c end this subplot
      call endgr(0)
      return
      end
C
C
C*********** Subroutine lgruf *********************************************
C
C This routine has DISSPLA calls
C
C This routine prints the roughness scales if a comparison is being made
C
      subroutine lgruf(numcur,hmeq,sigs,maxsur,surf)
      Integer numcur,maxsur,surf(numcur),i
      Real hmeq(maxsur),sigs(maxsur)
C
```

```
c set up subplot area and display info while changing color or line type
   call physor(0.,0.)
   call area2d(10.5,9.0)
   call messag('(H1.3Z1). (Z2)*',100,10.2,10.15)
   do 3020 i=1,numcur
    call linesz(i)
    call strtpt(9.3,10.2-.2*i)
    call connpt(9.9,10.2-.2*i)
    call reset('DASH')
    call realno(hmeq(surf(i)),103,10.,10.1-.2*i)
    call messag('(H.8M7)M(M1)M(MXHXEH.5)2(EXHX). *',
 *          100,'ABUT','ABUT')
    call realno(sigs(surf(i)),104,'ABUT','ABUT')
 3020 continue
   call reset('SETCLR')
   call endgr(0)
   return
   end
C
C
C********** Subroutine lgwin ***********************************************
C
C
C This routine has DISSPLA calls
C
C This routine prints the multiple fixed wavelengths if needed
C Very similar format to other legend makers
C
   subroutine lgwin(numcur,wlenfx)
   Integer numcur,i
   Real wlenfx(numcur)
C
   call physor(0.,0.)
   call area2d(10.5,9.0)
   call messag('(H1.3M7)L(M1H.5L)0(LXHX)*',100,10.2,10.15)
   do 3030 i=1,numcur
    call linesz(i)
    call strtpt(9.3,10.2-.2*i)
    call connpt(9.9,10.2-.2*i)
    call reset('DASH')
    call realno(wlenfx(i),104,10.,10.1-.2*i)
    call messag('(H.8M7)M(M1)M(HXMX)*',100,'ABUT','ABUT')
 3030 continue
   call reset('SETCLR')
   call endgr(0)
   return
   end
C
C
C********** Subroutine lgang ***********************************************
C
C This routine has DISSPLA calls
C
C This routine prints out the multiple fixed inc. angles if comparing
```

```
C very similar to other legend makers
C
      subroutine lgang(numcur,angfx)
      Integer numcur,i
      Real angfx(numcur)
C
      call physor(0.,0.)
      call area2d(10.5,9.0)
      call messag('(H1.3M7)Q(M1H.5L)0(LXHX)*',100,10.2,10.15)
      do 3040 i=1,numcur
       call linesz(i)
       call strtpt(9.3,10.2-.2*i)
       call connpt(9.9,10.2-.2*i)
       call reset('DASH')
       call realno(angfx(i),104,10.,10.1-.2*i)
       call messag('(M1EH.5)O(MXEXHX)*',100,'ABUT','ABUT')
 3040 continue
      call reset('SETCLR')
      call endgr(0)
      return
      end
C
C
C********** Subroutine linesz, setlin ****************************************
C
      subroutine linesz(i)
      Integer i,j,iflcol,dflcol
      Real r1(4),r2(6),r3(3),r4(8)
      Data r1/11.,3.,3.,3./
      Data r2/8.,3.,2.,2.,2.,3./
      Data r3/5.,10.,5./
      Data r4/4.,3.,2.,3.,2.,3.,4.,3./
C
      if(iflcol.eq.1) then
       j=i-(i/7)*7
       if(j.eq.1) call setclr('BLUE')
       if(j.eq.2) call setclr('GREEN')
       if(j.eq.4) call setclr('CYAN')
       if(j.eq.3) call setclr('RED')
       if(j.eq.5) call setclr('MAGENTA')
       if(j.eq.6) call reset('SETCLR')
       if(j.eq.0) call setclr('YELLOW')
       return
     else
C
       j=i-(i/6)*6
       if(j.eq.0) call dash
       if(j.eq.1) call reset('DASH')
       if(j.eq.2) call mrscod(.4,4,r1)
       if(j.eq.3) call mrscod(.4,6,r2)
       if(j.eq.4) call mrscod(.4,3,r3)
       if(j.eq.5) call mrscod(.48,8,r4)
       return
     endif
```

```
C
C
      entry setlin(dficol)
      ificol-dficol
      return
      end
C
C
C********** Subroutine setlim ***************************************************
C
C This routine has NO disspla calls
C
C This routine returns ymin and ymax which should be better limits for
C setting up an axis than the absolute values y1 and y2 passed in.
C round controls how many sig figs ymin and ymax will contain.
C round-10 --> 2 sig figs,  round-1000 --> 4 sig figs
C
      subroutine setlim(y1,y2,ymin,ymax,round)
      Integer l1,l2,itemp
      Real y1,y2,ymin,ymax,round
      Real y1p,y2p,y1pp,y2pp,ytemp
C
c find true min and max if values switched around
      if(y1.gt.y2) then
        ytemp-y1
        y1-y2
        y2-ytemp
      endif
C
c special cases exist if y1-y2
      if(y1.eq.y2) then
        if(y1.eq.0.) then
          ymin--1.
          ymax-1.
        else if(y1.lt.0.) then
          ymin-1.5*y1
          ymax-.5*y1
        else
          ymin-.5*y1
          ymax-1.5*y1
        endif
        return
      else
C
c otherwise break the numbers down
      if(y1.eq.0.) y1-1.e-36
      if(y2.eq.0.) y2-1.e-36
      l1-ifix(alog10(abs(y1))+40)-40
      l2-ifix(alog10(abs(y2))+40)-40
      if(l1.gt.l2) then
        itemp-l1
        l1-l2
        l2-itemp
      endif
```

Appendix V

```
C
c y1p and y2p have a maximum of 1 place left of the dec pt except
c in the special circumstance when that place is not a true sig fig
 3500  y1p=y1/(10.**float(l2))
       y2p=y2/(10.**float(l2))
       if((ifix(y1p).eq.0.and.ifix(y2p).eq.1).or.
     *   (ifix(y1p).eq.-1.and.ifix(y2p).eq.0)) then
        l2=l2-1
        goto 3500
       endif
C
c round the numbers off on a common basis
c let plot limits fall inside absolute values just slightly if need be
       y1pp=ifix(y1p*round)/round
       if(y1.lt.0.0.and.y1pp.gt.y1p+.01/round) y1pp=y1pp-(1./round)
       y2pp=ifix(y2p*round)/round
       if(y2.gt.0.0.and.y2pp.lt.y2p-.01/round) y2pp=y2pp+(1./round)
C
c remove the normalization factor
       ymin=y1pp*(10.**l2)
       ymax=y2pp*(10.**l2)
      endif
      return
      end
C
C
C********** Subroutine carve ***********************************************
C
C This routine has NO Disspla calls
C
C this routine takes info out of data storage and puts it into arrays
C for plotting.  The data is stored in 2-d format, carve pulls the data out
C as a fnc of wlen at a fixed inc ang or it pulls data out as a fnc of inc ang
C at a fixed wavelength.  The fixed inc ang or wavelength need not lie directly
C on a grid because this routine linearly interpolates the between grids
C
C This routine fills curve numbers k1 to k2 of arrays xray and yray
C either k1=1 and k2=numcur or k1=k2
C
      subroutine carve(k1,k2,m,fnum1,angl,wlenl,nang1,nwlen1,
     *            angfx,wlenfx,ptype,numpts,xray,yray)
      Integer maxplt,maxcur,maxang,maxwln,maxsur,maxdat
      Parameter (maxplt=4,maxcur=7,maxang=25)
      Parameter (maxwln=71,maxsur=10,maxdat=20)
      Integer k,k1,k2,m,fnum1,nang1,nwlen1,numpts(maxcur),ptype
      Real facti
      Real angl(maxsur,maxang),wlenl(maxsur,maxwln)
      Real angfx(maxcur),wlenfx(maxcur)
      Real xray(maxcur,maxwln),yray(maxcur,maxwln)
      Real fst(maxdat,maxwln,maxang)
C
      Common/one/fst
C
c get index of data storage
```

```
      call ffind(m,fnum1,idx)
c get data as a function of wavelength for
    if(ptype.lt.3) then
     do 4000 k=k1,k2
c determine fixed inc angle
      if(ptype.eq.1) then
       afx=angfx(k)
      else
       afx=angfx(1)
      endif
      numpts(k)=nwlen1
c find what grid corresponds to the inc angle or find grids for interpolation
      j=1
      if(angl(m,j).gt.afx) goto 4040
 4020  j=j+1
      if(angl(m,j).gt.afx) goto 4030
      if(angl(m,j).eq.afx) goto 4040
      if(j.eq.nang1) goto 4040
      goto 4020
c
c do the interpolation to get data as a function of wavelength
 4030  j=j-1
      facti=(afx-angl(m,j))/(angl(m,j+1)-angl(m,j))
      do 4050 i=1,nwlen1
       xray(k,i)=wlenl(m,i)
       yray(k,i)=(fst(idx,i,j+1)-fst(idx,i,j))*facti+fst(idx,i,j)
 4050  continue
      goto 4000
c
c inc angle hit grid line exactly or requested inc ang fell outside grid range
 4040  do 4060 i=1,nwlen1
       xray(k,i)=wlenl(m,i)
       yray(k,i)=fst(idx,i,j)
 4060  continue
 4000 continue
c
c get data as a function of inc angle
    else
     do 4100 k=k1,k2
c determine the fixed wavelength to use
      if(ptype.eq.3) then
       wfx=wlenfx(k)
      else
       wfx=wlenfx(1)
      endif
      numpts(k)=nang1
c find the correct grid or grids to interpolate between
      j=1
      if(wlenl(m,j).gt.wfx) goto 4140
 4120  j=j+1
      if(wlenl(m,j).gt.wfx) goto 4130
      if(wlenl(m,j).eq.wfx) goto 4140
      if(j.eq.nwlen1) goto 4140
      goto 4120
```

```
  C
c do the interpolation
 4130    j=j-1
        facti=(wfx-wlenk(m,j))/(wlenk(m,j+1)-wlenk(m,j))
        do 4150 i=1,nang1
          xray(k,i)=angl(m,i)
          yray(k,i)=(fat(idx,j+1,i)-fat(idx,j,i))*facti+fat(idx,j,i)
 4150    continue
        goto 4100
C
c interpolation not needed
 4140    do 4160 i=1,nang1
          xray(k,i)=angl(m,i)
          yray(k,i)=fat(idx,j,i)
 4160    continue
 4100  continue
      endif
      return
      end
C
C
C********** Subroutine rng2d ************************************************
C
C This routine has NO disspla calls
C
C this routine determines the absolute max and min of the y value
C on a given 2-d plot for a single mtx element in the appropriate x-axis range
C
      subroutine rng2d(numcur,xray,xmin,xmax,yray,numpts,
     *              maxcur,maxwin,ymin,ymax)
      Integer numcur,maxcur,maxwin,numpts(maxcur)
      Integer i,j
      Real xray(maxcur,maxwin),yray(maxcur,maxwin),ymin,ymax,xmin,xmax
C
c normal min max search strategy
      ymin=1.e37
      ymax=-1.e37
      do 4200 i=1,numcur
       do 4200 j=1,numpts(i)
        if(xray(i,j).ge.xmin.and.xray(i,j).le.xmax) then
          if(yray(i,j).gt.ymax) ymax=yray(i,j)
          if(yray(i,j).lt.ymin) ymin=yray(i,j)
        endif
 4200 continue
      return
      end
C
C
C********** Subroutines prtn2d, sect2d ****************************************
C
C This routine has DISSPLA calls
C
C This routine sets up the subplot area for 2-d and contour plots by dividing
C the page into 1 or 4 section.  For each plt number a different section
```

```
 C is set up for use
C
    subroutine prtn2d(numplt,plt)
    Integer numplt,plt
    Real xsc,ysc,xscale,yscale
C
c scaling must be done on physical origin
    if(numplt.eq.1) then
     call physor(1.5*xsc,1.5*ysc)
     call height(.21)
    else
     if(plt.eq.1) call physor(1.*xsc,5.25*ysc)
     if(plt.eq.2) call physor(6.9*xsc,5.25*ysc)
     if(plt.eq.3) call physor(1.*xsc,1.*ysc)
     if(plt.eq.4) call physor(6.9*xsc,1.*ysc)
c the scaling must be reset again for multiple subplots
     call bscale(xsc*3./6.5,ysc*4.5/9.)
     call height(.35)
    endif
    call area2d(9.5,6.5)
    return
C
C
c this entry passes the proper scale in from the main program
    entry secl2d(xscale,yscale)
    xsc=xscale
    ysc=yscale
    return
    end
C
C
C********** Subroutines axes2d, agrd2d ****************************************
C
C This routine has DISSPLA calls
C
C This routine draws and labels the axes for 2-d plots
C
    subroutine axes2d(ptype,fnum1,xmin,xmax,ymin,ymax,n1,n2)
    Integer ptype,fnum1,irow,icol,n1,n2
    Integer iflgrd,dflgrd,ngrdx,ngrdy,dngrdx,dngrdy
    Real xmin,xmax,x1,x2,ymin,ymax,y1,y2
C
c choose proper name for x-axis
    if(ptype.lt.3) then
     call xname("(F4)Wavelength in (M7)M(M1)M",100)
    else
     call xname("(F4)Angle of Incidence in degrees",100)
    endif
C
    call yaxang(0.)
    call yname(" ",100)
c put y-axis label on top of graph
    call messag(" ",100,-.3,6.7)
    call mxlbel(fnum1,n1,n2)
```

```
C
c choose limits and set up draw axes
      call setlim(xmin,xmax,x1,x2,10.)
      call setlim(ymin,ymax,y1,y2,2.)
      call simpbx
      call graf(x1,(x2-x1)/10.,x2,y1,(y2-y1)/5.,y2)
      call compbx
c put in grid lines if indicated
      if(ifgrd.eq.1) then
        call dash
        call grid(ngrdx,ngrdy)
        call reset('DASH')
      endif
      return
C
C
c pass in the correct number of grid lines for the x and y axes
      entry sgrd2d(dfigrd,dngrdx,dngrdy)
      ifgrd=dfigrd
      ngrdx=dngrdx
      ngrdy=dngrdy
      return
      end
C
C
C********** Subroutine makef ************************************************
C
C This routine has DISSPLA calls
C
C This routine prints Fij, Fij/F11, Fij/Fij(base)-1, or [Fij/F11]/[Fij/F11(base)]-1
C dependent upon what normalizations are if effect
C
      subroutine makef(elt,n1,n2)
      Integer elt,n1,n2
      Character*1 cint(4)
      Data cint/'1','2','3','4'/
      irow=(elt-1)/4+1
      icol=elt-(irow-1)*4
c print Fij
      call messag('(F4)F(H.5)'//cint(irow)//cint(icol)//'$'
     *          ,100,'ABUT','ABUT')
c print F11
      if(n1.eq.1) then
        call messag('(F4)F(H.5)11$',100,'ABUT','ABUT')
      endif
c print base Fij
      if(n2.ne.0) then
        call messag('-(F4)F(H.5)'//cint(irow)//cint(icol)//
     *           '(B2HXE.8H.6)base(B2)$',100,'ABUT','ABUT')
c print base F11
      if(n1.eq.1) then
        call messag('(F4)F(H.5)11(B2HXE.8H.6)base$'
     *           ,100,'ABUT','ABUT')
      endif
```

```
        endif
      return
      end
C
C
C********** Subroutine lgbase *********************************************
C
C This routine has DISSPLA calls
C
C This routine prints out the info about the base in upper left corner
C It prints name, corr. fnc, height, and slop info
C
      subroutine lgbase(matnm,autonm,hmaq,sigs)
      Real hmaq,sigs
      Character*13 matnm,autonm
C
c set up subplot area
      call physor(0.,0.)
      call area2d(11.5,10.5)
      call height(.14)
      call messag('Base material -$',100,.2,9.7)
      call messag(matnm//'$',100,.2,9.5)
      call messag(autonm//'$',100,.2,9.3)
      call messag('(Z1) - $',100,.2,9.1)
      call realno(hmaq,104,'ABUT','ABUT')
      call messag('(M7)M(M1)M(EH.5)2$',100,'ABUT','ABUT')
      call messag('(Z2) - $',100,.2,8.9)
      call realno(sigs,104,'ABUT','ABUT')
      call endgr(0)
      return
      end
C
C
C********** Subroutine doit2d *********************************************
C
C This routine has DISSPLA calls
C
C This routine actually calls the curve drawing routines.
C Variable dimensioning of xray1 and yray1 are used because curve()
C only wants the correctly dimensioned outputs.
C
      subroutine doit2d(icur,numpts,xray,yray,xray1,yray1,maxcur,maxwin)
      Integer icur,numpts,maxcur,maxwin
      Real xray(maxcur,maxwin),yray(maxcur,maxwin)
      Real xray1(numpts),yray1(numpts)
C
c copy into variably dimensioned arrays
      do 4400 i=1,numpts
       xray1(i)=xray(icur,i)
       yray1(i)=yray(icur,i)
 4400 continue
c subplot set up by prtn2d
      call grace(.1)
      call curve(xray1,yray1,numpts,0)
```

```
        call reset('SETCLR')
      call reset('DASH')
      return
C
C
c this call is used to finish off a mtx plot when all curves drawn
      entry engr2d
      call endgr(0)
      return
      end
C
C
C********** Subroutine doit3d ************************************************
C
C this routine has DISSPLA calls
C
C this routines puts the data for the 3-d plot in a variably dimensioned
C 2-d array to match the requirements of surmat()
C
      subroutine doit3d(ixmin,ixpts,iymin,iypts,angl,wlenl,m,idx,fwrk)
      Integer maxplt,maxcur,maxang,maxwin,maxsur,maxdat
      Parameter (maxplt=4,maxcur=7,maxang=25)
      Parameter (maxwin=71,maxsur=10,maxdat=20)
      Integer ixmin,ixmax,iymin,iymax,ixpts,iypts,m,idx
      Integer iwork(2*(maxwin*maxang)+4)
      Real angl(maxsur,maxang),wlenl(maxsur,maxwin)
      Real fat(maxdat,maxwin,maxang),fwrk(ixpts,iypts)
      Real x33mat,y33mat,setx3,sety3,setup
      External x33mat,y33mat
      Common/one/fat
C
      ixmax=ixmin+ixpts-1
      iymax=iymin+iypts-1
      iypts=iymax-iymin+1
c     setup=setx3(m,wlenl,ixmin)
c     setup=sety3(m,angl,iymin)
C
c copy data from storage into smaller array
      do 4600 j=ixmin,ixmax
        do 4600 k=iymin,iymax
          fwrk(j-ixmin+1,k-iymin+1)=fat(idx,j,k)
 4600 continue
C
c do plot
      call surmat(fwrk,1,ixpts,1,iypts,iwork)
      return
      end
C
C
C********** Functions x33mat, setx ******************************************
C
C This routine has NO disspla calls
C
C This routine returns the wavelength for a given x index to contour plotting
```

```
C routine.  This function was to be x3dmat and be used with surtrn() but
C that subroutine does not seem to work in disspla on the cray2.
C
      function x33mat()
      Integer maxsur,maxwin,m,j
      parameter(maxsur=10,maxwin=71)
      Integer ixmin,dixmin
      Real wienl1(maxwin),dwienl(maxsur,maxwin)
      save
C
c simple isn't it
      x33mat=wienl1(j+ixmin-1)
      return
C
C
c this entry copies in the list of wavelengths for the given surface assgn #
      entry setx3(m,dwienl,dixmin)
      ixmin=dixmin
        do 4700 i=1,maxwin
          wienl1(i)=dwienl(m,i)
 4700 continue
      setx3=1.
      return
      end
C
C
C********** Function y33mat, sety *******************************************
C
C This function has NO disspla calls
C
C this routine is just like x33mat except that it deals with the inc angles
C
      function y33mat()
      Integer maxsur,maxang,m,j
      parameter(maxsur=10,maxang=25)
      Integer diymin,iymin
      Real angl1(maxang),dangl(maxsur,maxang)
      save
C
      y33mat=angl1(j+iymin-1)
      return
C
C
      entry sety3(m,dangl,diymin)
      iymin=diymin
        do 4800 i=1,maxang
          angl1(i)=dangl(m,i)
 4800 continue
      sety3=1.
      return
      end
C
C
C********** Subroutine mg3d *************************************************
```

```
C
C This routine has NO disspla calls
C
C this is a normal min max search process in the desired range for the plot
C
      subroutine rng3d(idx,ixmin,ixmax,iymin,iymax,zmin,zmax)
      Integer maxplt,maxcur,maxang,maxwin,maxsur,maxdat
      Parameter (maxplt=4,maxcur=7,maxang=25)
      Parameter (maxwin=71,maxsur=10,maxdat=20)
      Integer idx,ixmin,ixmax,iymin,iymax,i,j
      Real zmin,zmax,fst(maxdat,maxwin,maxang)
C
      Common/one/fst
C
      zmin=1.e36
      zmax=-1.e36
      do 5000 i=ixmin,ixmax
       do 5000 j=iymin,iymax
        if(fst(idx,i,j).gt.zmax) zmax=fst(idx,i,j)
        if(fst(idx,i,j).lt.zmin) zmin=fst(idx,i,j)
 5000 continue
      return
      end
C
C
C********** Subroutine head3d ************************************************
C
C This routine has DISSPLA calls
C
C this routine prints the center heading for 3d plots and contour plots
C
      subroutine head3d(nsurf,matnm,autonm,hmsq,sigs,elt,n1,n2)
      Integer elt,nsurf
      Real hmsq,sigs
      Character*13 matnm,autonm
C
c setup subplot area
      call physor(0.,0.)
      call area2d(12.,10.5)
      call height(.26)
      call messag('Backscatter Mueller Matrix Element $',100,0.9,10.)
      call makef(elt,n1,n2)
      call height(.21)
c if there is only one material on a contour or for all 3-d plots, print it
      if(nsurf.eq.1) then
       call messag(matnm//'   '//autonm//'$',100,3.2,9.75)
       call height(.17)
       call messag('(Z1) - $',100,3.1,9.45)
       call realno(hmsq,104,'ABUT','ABUT')
       call messag('(H.7M7)M(M1)M(EH.5)2(EXHXMX)$',100,'ABUT','ABUT')
       call messag('     (Z2) - $',100,'ABUT','ABUT')
       call realno(sigs,104,'ABUT','ABUT')
      else
c otherwise print comparison
```

```
        call messag('Comparison of Materials$',100,3.4,9.75)
      endif
      call reset('HEIGHT')
      call endgr(0)
      return
      end
C
C
C********** Subroutines prtn3d, sacl3d ****************************************
C
C This routine has DISSPLA calls
C
C This routine sets up the page area for displaying the 3-d plot
C
      subroutine prtn3d(vx,vy,vz)
      Real vx,vy,vz,xsc,ysc,xscale,yscale
C
c scaling of physical origin is important
c setting view pt is also done here
      call physor(1.*xsc,.25*ysc)
      call titl3d(' $',100,11.,9.5)
      call volm3d(2.,1.,1.)
      call vuabs(vx,vy,vz)
      return
C
C
c this entry passes in the scaling factor to use
      entry sacl3d(xscale,yscale)
      xsc=xscale
      ysc=yscale
      return
      end
C
C
C********** Subroutine axes33 *************************************************
C
C This routine has DISSPLA calls
C
C This routine setups for and plots the 3 axes on 3-d plots
C
      subroutine axes33(x1,x2,y1,y2,z1,z2)
      Real x1,x2,y1,y2,z1,z2
C
      call zaxang(-90.)
      call yaxang(0.)
      call height(.21)
c name them
      call x3name('(F4)Wavelength in (M7)M(M1)M$',100)
      call y3name('(F4)Incident Angle in deg.$',100)
      call z3name(' $',100)
      call simplx
c draw them
      call graf3d(x1,(x2-x1)/5.,x2,y1,(y2-y1)/5.,y2,z1,(z2-z1)/5.,z2)
      call complx
```

```
      return
      end
C
C
C********** Subroutine lgcn ***********************************************
C
C This routine has DISSPLA calls
C
C this routine prints the legend of the contour levels in the upper right corner
C
      subroutine lgcn(lvl)
      Real lvl(11)
C
      call physor(0.,0.)
      call area2d(11.5,10.5)
      do 5900 i=1,5
       call linesz(i)
       call strtpt(9.7,10.2-.2*i)
       call connpt(10.2,10.2-.2*i)
       call reset('DASH')
       call realno(lvl(i*2-1),-3,10.3,10.1-.2*i)
 5900 continue
      call reset('SETCLR')
      call endgr(0)
      return
      end
C
C
C********** Subroutines axescn, sgrdcn ****************************************
C
C This routine has DISSPLA calls
C
C this routine names and plots the axes for contour plots
C
      subroutine axescn(x1,x2,y1,y2,nsurf,matnm,autonm,hmsq,sigs)
      Integer nsurf,iflgrd,dflgrd,ngrdx,ngrdy,dngrdx,dngrdy
      Real x1,x2,y1,y2,hmsq,sigs
      Character*13 matnm,autonm
C
c name them
      call yaxang(0.)
      call yname('(F4)Incident Angle in deg.$',100)
      call xname('(F4)Wavelength in (M7)M(M1)M$',100)
      call simplx
c plot them
      call graf(x1,(x2-x1)/10.,x2,y1,(y2-y1)/5.,y2)
      call complx
      call height(.21)
C
c if materials are being compared write material data over each plot
      if(nsurf.gt.1) then
       call messag(matnm//' '//autonm//' $',100,0.,6.6)
       call messag('(Z1) - $',100,'ABUT','ABUT')
       call realno(hmsq,104,'ABUT','ABUT')
```

```
        call messag('(M7)M(M1)M(EH.5)2*',100,'ABUT','ABUT')
      call messag('      (Z2) - *',100,'ABUT','ABUT')
      call realno(sigs,104,'ABUT','ABUT')
      call reset('HEIGHT')
     endif
c draw grid lines if desired
     if (iflgrd.eq.1) then
      call dash
      call grid(ngrdx,ngrdy)
      call reset('DASH')
     endif
     return
c
c
c this entry passes grid line info in
     entry sgrdcn(dflgrd,dngrdx,dngrdy)
     iflgrd-dflgrd
     ngrdx-dngrdx
     ngrdy-dngrdy
     return
     end
c
c
C********** Subroutine doitcn ************************************************
c
C This routine has NO disspla calls
c
C This routine determines the point on each of 9 contour levels of the
C corresponding 3-d plot
c
     subroutine doitcn(ixmin,dixpts,iymin,diypts,angl,wlenl,msurf,
    *              idx,lvl,z1,z2)
     Integer maxplt,maxcur,maxang,maxwln,maxsur,maxdat
     parameter(maxplt-4,maxcur-7,maxang-25)
     parameter(maxwln-71,maxsur-10,maxdat-20)
     Integer ixmin,ixpts,dixpts,ixmax,iymin,diypts,iypts,iymax
     Integer msurf,idx,ilvl
     Integer idk(4),idl(4)
     Integer hk(1000),hl(1000),hd(1000),n
     Real angl(maxsur,maxang),wlenl(maxsur,maxwln)
     Real fwrk(maxwln,maxang)
     Real fst(maxdat,maxwln,maxang)
     Real lvl(11),z1,z2,level
     Real xray(1000),yray(1000)
c
     Common/one/fst
     Common/five/level,ihits,hk,hl,hd,fwrk,ixpts,iypts
     Data idk,idl/0,1,0,-1,1,0,-1,0/
c
c move proper range of data from storage into the working 2-d array
     ixpts-dixpts
     iypts-diypts
     ixmax-ixmin+ixpts-1
     iymax-iymin+iypts-1
```

```
        setup=setx3(msurf,wienl,ixmin)
        setup=sety3(msurf,angl,iymin)
        do 6100 j=ixmin,ixmax
          do 6100 k=iymin,iymax
            fwrk(j-ixmin+1,k-iymin+1)=fsl(idx,j,k)
 6100 continue
c
c loop through each of 9 levels
c assigning colors or dashed lines to each
        do 6110 ilvl= 9,1,-1
          ihits=0
          level=lvl(ilvl)
          if(ilvl.eq.2*(ilvl/2)) then
            icolor=10
          else
            icolor=ilvl/2+1
          endif
c
c find all places where the contour line pass between adjacent wavelengths
        do 6112 k=1,ixpts-1
          do 6112 l=1,iypts
            call hitme(k,l,1)
            call hitme(k+1,l,3)
 6112 continue
c
c find all place where the contour line pass between adjacent inc angles
        do 6114 l=1,iypts-1
          do 6114 k=1,ixpts
            call hitme(k,l,0)
            call hitme(k,l+1,2)
 6114 continue
c
c
c iflag1=0 --> working on edge to edge contours
c iflag1=1 --> working on closed contours
        iflag1=0
        kav=0
 6130 if(ihits.eq.0) goto 6110
        kav=-1
        ipts=0
        if(iflag1.eq.1) goto 6133
c find an edge entry point
        n=1
 6131 if(hd(n).eq.0.and.hk(n).eq.1) goto 6135
        if(hd(n).eq.1.and.hl(n).eq.iypts) goto 6135
        if(hd(n).eq.2.and.hk(n).eq.ixpts) goto 6135
        if(hd(n).eq.3.and.hl(n).eq.1) goto 6135
        if(n.eq.ihits) goto 6133
        n=n+1
        goto 6131
c
c pick a hit as starting for close contour, save it and copy it
 6133 iflag1=1
        kav=hk(1)
```

```
      lsv-hl(1)
      idsv-hd(1)
      ihits-ihits+1
      hk(ihits)-ksv
      hl(ihits)-lsv
      hd(ihits)-idsv
      n-1
c
c save interpolated x and y values in the plotting array
 6135 id-hd(n)
      k-hk(n)
      l-hl(n)
      kp-k+idk(id+1)
      lp-l+idl(id+1)
      ipts-ipts+1
      x1-x33mat(k)
      y1-y33mat(l)
      delf-(level-fwrk(k,l))/(fwrk(k+idk(id+1),l+idl(id+1))-fwrk(k,l))
      xray(ipts)-(x33mat(k+idk(id+1))-x1)*delf+x1
      yray(ipts)-(y33mat(l+idl(id+1))-y1)*delf+y1
c remove hit from list
      hd(n)-hd(ihits)
      hk(n)-hk(ihits)
      hl(n)-hl(ihits)
      ihits-ihits-1
c see if you are at the end of edge to edge
      if(iflag1.eq.0) then
      if((id.eq.0.and.k.eq.ixpts).or.(id.eq.2.and.k.eq.1).or.
     *  (id.eq.1.and.l.eq.1).or.(id.eq.3.and.l.eq.iypts)) then
       call curvcn(xray,yray,icolor,ipts)
       goto 6130
      endif
c see if you are at the end of a closed contour
      else
      if(k.eq.ksv.and.l.eq.lsv.and.id.eq.idsv) then
       if(ipts.gt.1) then
       call curvcn(xray,yray,icolor,ipts)
       goto 6130
       endif
      endif
      endif
c
c determine what the next hit should be
c this is the meat of the code but it is not really possible to document here
c the best way to understand it is to do a pencil and paper run
      idp-incd(id)
      kpp-kp+idk(idp+1)
      lpp-lp+idl(idp+1)
      if(fwrk(kpp,lpp).gt.level) then
        knew-kpp
        lnew-lpp
        idnew-incd(incd(idp))
      else
        kppp-k+idk(idp+1)
```

```
          lppp-l+idl(idp+1)
        if(fwrk(kppp,lppp).gt.level) then
          knew-kppp
          lnew-lppp
          idnew-id
        else
          knew-k
          lnew-l
          idnew-idp
        endif
      endif
C
c find the new hit in hit list
      n-1
 6138 if(hk(n).eq.knew.and.hl(n).eq.lnew.and.hd(n).eq.idnew) goto 6135
      if(n.eq.ihits) then
        goto 6110
      endif
      n-n+1
      goto 6138
 6110 continue
      return
      end
C
C
C********** Subroutine hitme ************************************************
C
C This routine has NO disspla calls
c
c this routine determines if a base pt and a direction constitutes a hit for
c the given contour level
c
      subroutine hitme(k,l,m)
      Integer maxwin,maxang
      Parameter(maxwin-71,maxang-25)
      Integer k,l,m,hk(1000),hl(1000),hd(1000),ixpts,iypts
      Integer idk(4),idl(4)
      Real level,fwrk(maxwin,maxang)
C
      Common/five/level,ihits,hk,hl,hd,fwrk,ixpts,iypts
      Data idk,idl/0,1,0,-1,1,0,-1,0/
C
      if(fwrk(k,l).ge.level) then
        kp-k+idk(m+1)
        lp-l+idl(m+1)
        if(fwrk(kp,lp).lt.level) then
          ihits-ihits+1
          hk(ihits)-k
          hl(ihits)-l
          hd(ihits)-m
        endif
      endif
      return
      end
```

```
C
C
C********** Function incd ****************************************************
C
C This routine is a mod 4 function
C
      function incd(idd)
      integer idd
      incd=idd+1-4*((idd+1)/4)
      return
      end
C
C
C********** Subroutine curvcn ***********************************************
C
C This routine has DISSPLA calls
C
C This routines plots the contours as any other 2-d curve
C a special color handling section makes every other contour a dashed line
C
      subroutine curvcn(xray,yray,icolor,ipts)
      integer icolor,ipts
      Real xray(ipts),yray(ipts)
      if(icolor.eq.10) then
        call dash
      else
        call lincsa(icolor)
      endif
      call curve(xray,yray,ipts,0)
      call reset('SETCLR')
      call reset('DASH')
      return
      end
```

Blank

APPENDIX VI: DETECT/DECIDE2 USER GUIDES AND SOURCE CODES.

AVI.1.1 Introduction to DETECT.

Program DETECT, written by Mark Haugland, analyzes backscatter Mueller matrices calculated by RETRO. Both DETECT and RETRO are written in FORTRAN 77 and set up to run on a Cray 2 under the UNICOS operating system. The program serves two purposes: (1) it locates optimum angles of incidence and wavelengths for use in discriminating between a contaminated and a dry surface; (2) it then identifies those Mueller matrix elements that can be used to discriminate between contaminated and dry surfaces at optimum beams angles of incidence and beam wavelengths.

AVI.1.1.1 Outline of DETECT's functions.

DETECT functions in the following manner:

1. Read in two output files from RETRO.
   a) Either normalize all matrix elements to f_{11} or use the selected auto-correlation function.
2. Write a heading block to the output file (unit 9).
3. Compute Equations (37) and (39). The program does not enforce the condition that these values must be larger than $|x|_{max}$ and $|y|_{max}$. The user must decide what these values are and make judgements accordingly.
   a) Sort the terms in Equations (37) and (39) so that they are in decreasing order of magnitude.
4. Write to unit 9.
   a) The subscripts of non-zero terms in Equation (39) in the order of largest to smallest terms. The subscript of the largest term is followed by the subscript of the next largest term and so on,
   b) the subscripts of all non-zero terms in (37) in the order of largest to smallest term. The subscript of the largest term is followed by the subscript of the next largest term and so on,
   c) Equation (37), largest term, subscript of largest term,
   d) Equation (39), largest term, subscript of largest term,
   e) the angle and wavelength at which (37) took on its maximum value over the range of wavelengths and angles in the RETRO output files and the maximum value,
   f) the angle and wavelength at which (39) took on its maximum value over the range of wavelengths and angles in the RETRO output files and the maximum value,
   g) the largest value assumed by any term in (37), the subscript of that term, the angle and wavelength at which that value occurred,
   h) the largest value assumed by any term in (39), the subscript of that term, the angle and wavelength at which that value occurred.
5. Find all local maxima and minima in k (imaginary refractive index) for both materials,
6. write all local maxima and minima in k for each material to unit 9.

AVI.1.1.2 Input Files.

DETECT needs five input files. The first file is 'FILENAMES,' which is used in every run and contains names of the four other files required by DETECT. In addition to names of the other input files, 'FILENAMES' contains code numbers that determine which auto-correlation function the program accesses for the background material, which auto-correlation function the program uses for the target material, or whether to normalize all Mueller matrix elements to f_{11}

AVI.1.1.2.1 FILENAMES.

An example of a correctly edited file is given at the end of this section. The line-by-line format is as follows:

A- filename containing RETRO output for the background (base) material (up to 15 characters);

B- filename containing RETRO output for the target material (up to 15 characters);

C- filename for data output, unit 9 (15 characters max);

D- filename for commentary output, unit 8 (15 characters max);

E- filename containing the background material's index of refraction as a function of wavelength (15 characters or less);

F- filename containing the target material's index of refraction as a function of wavelength (15 characters or less);

G- The code number (IBCOR) determines whether the auto-correlation function for the background material is Gaussian (IBCOR=1), N=8 (IBCOR=2), or N=6 (IBCOR=3) (integer);

H- The code number ITCOR determines whether the auto-correlation function for the target material is Gaussian (ITCOR=1), N=8 (ITCOR=2), or N=6 (ITCOR=3) (integer);

I- The code number NORM equals zero for no normalization and 1 for normalizing all Mueller matrix elements to f_{11}. That is, a value of 1 on this line causes the program to analyze $\frac{f_{ij}}{f_{11}}$ rather than f_{ij} (integer).

AVI.1.1.2.2 Accessing RETRO output files.

Program DETECT analyzes RETRO's output data as a function of beam wavelength and incident angle. The wavelengths and incident angles in RETRO's output file for background and target materials must be identical. If not the same, DETECT will quit and write an error message to unit 8. For more information regarding files generated by RETRO, see Appendix V.

AVI.1.1.2.3 Index of refraction files.

These files are the same files RETRO accesses to compute the dielectric constant for each material.

AVI.1.1.3 Output files.

DETECT creates two output files. The first file is for commentary output should something go wrong, the second is the DETECT data file. The names for these files are the inputs to program DETECT found on lines C and D of 'FILENAMES.'

AVI.1.1.3.1 Commentary output file.

This file is opened as unit 8. Unit 8 is sent various values for debugging the program if something should go wrong during the computer run. With normal operations the only written input to unit 8 are error messages indicating if wavelengths and angles of incidence in each RETRO output file are not equal or if the number on line I of 'FILENAMES' is not 0 or 1. In DETECT, several write statements to unit 8 have been commented out. They may be reinstated by removing "C" in the leftmost column on the line where the write statement appears.

AVI.1.1.3.2 Data output file.

This file (written to unit 9) contains four distinct sections. The header block appears at the beginning of the file. The second section contains data from the analysis of RETRO's output. The third section contains maximum quantities contained in the second section. The final section contains local maxima and minima of k-values for both materials. The format of the heading block is as follows.

A- A description of the base material,

B- and the target material.

C- Present if line I in 'FILENAMES' is 1. This line is a comment to remind the user that RETRO output was normalized to f_{11}.

D- Present if line I in 'FILENAMES' is 0. This line has the same value as that of line G of 'FILENAMES,' to remind the user which auto-correlation function for the background material DETECT used in its analysis.

E- This line is the same as that of line D, for the target material.

F- The number of $\langle h^2\rangle$'s, σ_s^2's, λ's, and θ_0's used by RETRO. If the number of $\langle h^2\rangle$'s or σ_s^2's is greater than 1, the second and third sections of this file are repeated in a nested loop where σ_s^2 varies more rapidly than $\langle h^2\rangle$.

G- A comment that the next line is the $\langle h^2\rangle$'s for the background material.

H- $\langle h^2\rangle$'s for the background material.

I- A comment that the next line is the σ_s^2's for the background material.

J- σ_s^2's for the background material.

K-

L-

M-

N- Lines K, L, M, and N are the same as lines G, H, I, and J, for the target material.

O- A comment that the next line(s) list the wavelengths,

P- and those wavelengths used.

Q- A comment that the next line(s) list the incident angles,

R- and those incident angles used.

The second section of the data output file is program DETECT's output of the Mueller matrices. These data are generated in a nested loop where θ_0 varies more rapidly than λ. In this section, there are three lines of data for each pair of Mueller matrices read in. In addition, every time the wavelength is incremented, a line 'WAVELENGTH string' is written with wavelength value as the string in μm. The format of this section is as follows:

A2- Wavelength is written here each time it changes.

B2- The subscripts of all non-zero components of $\bar{y}$ Equation (39) are listed in decreasing order of the magnitude of the component. As an example, suppose $y_5 > y_1 > y_4$ and $y_2 = y_3 = y_6 = 0$. For the $\bar{y}$ described above, this line would read 5 1 4.

C2- This line is the same as B2, except it is associated with $\bar{x}$ of Equation (37).

D2- Values: $|\bar{x}|$, $|\bar{y}|$, j, $(x_j)_{max}$, i, and $(y_i)_{max}$

where;

$|\bar{x}|$ is calculated from Equation (37),

$|\bar{y}|$ is calculated from Equation (39),

$(x_j)_{max}$ is the largest component of $\bar{x}$,

and $(y_i)_{max}$ is the largest component of $\bar{y}$.

The third section in the data output file is repeated once for each combination of $<h^2>$ and σ_s^2 read in. This section gives the user a general idea where to find the most interesting information in the second section. There are four lines of data in this section.

A3- Values θ_0, λ, $(x_k)_{max}$, k.

$(x_k)_{max}$ is the largest value any component of $\bar{x}$ assumed over all combinations of the incident angles and wavelengths examined.

θ_0 and λ are the incident angle and wavelength at which $(x_k)_{max}$ occurred.

B3- Values θ_0, λ, $(y_k)_{max}$, k.

Same as A3 except for $\bar{y}$.

C3- Values θ_0, λ, $|\bar{x}|_{max}$.

θ_0 and λ are the angle of incidence and wavelength values for which $|\bar{x}|$ was maximum over all combinations of the incident angles and wavelengths examined.

D3- Values θ_0, λ, $|\bar{y}|_{max}$.

Same as C3 except for $\bar{y}$.

The fourth and final section of this file lists information regarding the indices of refraction for the background and target materials. The purpose of this section is to identify the correlation between resonant wavelengths and interesting behavior in each material's Mueller matrices. Four subsections are ordered as follows:

A4- Wavelengths and values for local maxima of k for the base (background) material.

B4- Wavelengths and values for local minima of k for the base material.

C4- Wavelengths and values for local maxima of k for the target material.

D4- Wavelengths and values for local minima of k for the target material.

AVI.1.2 Instructions for running the program.

Appendix VI

The steps required to run DETECT are now presented. DETECT can be executed after input file 'FILENAMES' has been properly edited and all of the other input files are loaded. Execution time is short, thus there is no need to run DETECT in the batch mode. Typing "detect.run" after the UNICOS prompt "bob>" will compile, link, and execute the program accessing data stored in 'FILENAMES.' The executable code is stored in "detect.xqt." To run the program again during the same login after changing 'FILENAMES,' simply type "detect.xqt." Do this rather than typing "detect.run" to avoid recompilation and linking. Before logging off, remove 'detect.xqt,' for it is a very large file. The object file 'detect.o' was removed by 'detect.run' immediately following linkage.

AVI.1.3 DETECT SOURCE CODE.

```
C This work was done for the CRDEC on the Aberdeen Proving Grounds
C Edgewood Area.
C This work was done by S. Mark Haugland under contract DAAD0589P0427.
C
C The purpose of this program is to aid the user in locating combinations
C of backscatter angle and wavelength of optimal use in discriminating
C between a background (base) surface and the surface with
C an optically thick layer of contaminant on top of it (target material).
C The term optically thick restricts the
C Mueller matrix elements which are of any use to those that are
C larger in magnitude for the contaminant than for the background.
C
C DETECT functions in the following manner:
C 1. Open the file FILENAMES and read the following input variables in:
C   IN1NM- the name of RETRO output file for the background material
C   IN2NM- the name of RETRO output file for the target material
C   OUTNM- the name of the file this program writes results to.
C   COMNM- the name of the file this program writes commentary
C     output to
C   BASENK- the name of the file containing the background material's
C     index of refraction as a function of wavelength
C   TARGNK- the name of the file containing the target material's
C     index of refraction as a function of wavelength
C   IBCOR- code number that selects the auto-correlation function
C     to use in constructing the Mueller matrices for the
C     background material
C   ITCOR- code number that selects the auto-correlation function
C     to use in constructing the Mueller matrices for the
C     target material
C   NORM- code number to determine whether DETECT will process
C     Mueller matrices or Mueller matrices normalized to
C     P11.
C      2. Open and read 2 output files from RETRO. These files contain
C   information needed to construct a set of Mueller matrices
C   for the target and the background materials.
C      a) either normalize to P11 or construct unnormalized Mueller
C   matrices using the auto-correlation functions requested
C   by IBCOR and ITCOR.
C 3. Write a heading block to the data output file (unit 9)
C 4. Compute data discussed in the DETECT user's manual.
C 5. Write data to unit 9.
C 6. Open the files containing the index of refraction as a
C   function of wavelength for the base material and find
C   all local maxima and minima of k in that file. Write
C   these values and the corresponding wavelengths to unit 9.
C 7. Open the files containing the index of refraction as a
C   function of wavelength for the target material and find
C   all local maxima and minima of k in that file. Write
C   these values and the corresponding wavelengths to unit 9.
C
      PROGRAM DETECT
```

```
      REAL HMSQLT(15),HMSQLB(15),WLENB(100),WLENT(100),ANGLB(100)
      REAL ANGLT(100),SIGSLB(15),SIGSLT(15),FB(6,100,100)
      REAL FT(6,100,100),FB1(6),FT1(6),QB(3),QT(3),DIFF(6),RATLOG(6)
      REAL ANKMIN(40),ANKMAX(40),WLNMIN(40),WLNMX(80)
      REAL WLNMN(80),ANGLMN(80),ANGLMX(80),FMAX(80),FMIN(80),ARR(6)
      INTEGER IBCOR,ITCOR,NORM,NHMSQB,NSIGSB,NWLENB,NANGB,NHMSQT
      INTEGER NSIGST,NWLENT,NANGT,NMIN,NMAX,KKA(6)
      CHARACTER*15 IN1NM,IN2NM,OUTNM,COMNM,BASENK,TARGNK,FNAME
      CHARACTER*13 BASEMAT,TARGMAT
C
C This block reads in inputs from FILENAMES and opens all of them except
C the files containing the indices of refraction for the target and
C background materials which are opened in the subroutine SORTNK.
C
      OPEN(UNIT=7,FILE='FILENAMES')
      READ(7,'(A)')IN1NM
      OPEN(UNIT=10,FILE=IN1NM)
      READ(7,'(A)')IN2NM
      OPEN(UNIT=11,FILE=IN2NM)
      READ(7,'(A)')OUTNM
      OPEN(UNIT=9,FILE=OUTNM)
      READ(7,'(A)')COMNM
      OPEN(UNIT=8,FILE=COMNM)
      READ(7,'(A)')BASENK
      READ(7,'(A)')TARGNK
C
C This block reads in control variables from unit 7 which determine
C which auto-correlation function to use for the base material and which
C auto-correlation function to use for the target material. IBCOR determines
C which auto-corr function to use for the base. IBCOR=1,2,3 for Gaussian,N=8
C ,and N=6 respectively. ITCOR performs the same function as IBCOR except for
C the target material. There are two options for normalizing the Mueller matrix
C elements. NORM=0 no normalization is used, NORM=1 all Mueller matrix elements
C are normalized to F11.
C
      READ(7,*)IBCOR
      READ(7,*)ITCOR
      READ(7,*)NORM
C
C This block reads the heading block from units 10 and 11.  Units 10 and 11 are
C the output from two separate runs of RETRO. Unit 10 contains the data on
C the base material and unit 11 contains the data on the target material.
C
      READ(10,'(A)')BASEMAT
      READ(11,'(A)')TARGMAT
      READ(10,1000)NHMSQB,NSIGSB,NWLENB,NANGB
      READ(11,1000)NHMSQT,NSIGST,NWLENT,NANGT
 1000 FORMAT(4I10.4)
      READ(10,1010)(HMSQLB(IL),IL=1,NHMSQB)
      READ(10,1010)(SIGSLB(IL),IL=1,NSIGSB)
      READ(10,1010)(WLENB(IL),IL=1,NWLENB)
      READ(10,1010)(ANGLB(IL),IL=1,NANGB)
      READ(11,1010)(HMSQLT(IL),IL=1,NHMSQT)
      READ(11,1010)(SIGSLT(IL),IL=1,NSIGST)
```

```
      READ(11,1010)(WLENT(IL),IL-1,NWLENT)
      READ(11,1010)(ANGLT(IL),IL-1,NANGT)
 1010 FORMAT(5E12.4)
C
C This block checks if the wavelengths and the angles of incidence for the
C base and the target materials are the same. If the wavelengths and the angles
C are not the same for both sets of data, the program stops and writes an error
C message to unit 8.
C
      IF(NANGB.NE.NANGT)GO TO 990
      IF(NWLENT.NE.NWLENB)GO TO 990
      DO 77 IL-1,NWLENB
      D-WLENB(IL)-WLENT(IL)
      IF(ABS(D).GE..001)GO TO 990
   77 CONTINUE
      DO 78 IL-1,NANGB
      D-ANGLB(IL)-ANGLT(IL)
      IF(ABS(D).GE..001)GO TO 990
   78 CONTINUE
C
C This section puts a heading in the output file so that the user can index
C the data.
C
      WRITE(9,9000)BASEMAT
 9000 FORMAT('BASE MATERIAL',A13)
      WRITE(9,9001)TARGMAT
 9001 FORMAT('TARGET MATERIAL',A13)
      IF(NORM.EQ.1)THEN
      WRITE(9,9010)
 9010 FORMAT('ALL MATRIX ELEMENTS ANALYZED ARE NORMALIZED TO F11')
      ELSE IF(NORM.EQ.0)THEN
      WRITE(9,9011)IBCOR
 9011 FORMAT('CODE FOR AUTO-CORR. FUNCTION FOR THE BACKGROUND
     * MATERIAL',I5)
      WRITE(9,9012)ITCOR
 9012 FORMAT('CODE FOR AUTO-CORR. FUNCTION FOR THE TARGET
     * MATERIAL',I5)
      ELSE
      WRITE(8,8222)
 8222 FORMAT('ILLEGAL NORM')
      GO TO 992
      ENDIF
      WRITE(9,1000)NHMSQB,NSIGSB,NWLENB,NANGB
      WRITE(9,*)'<h**2> for the base material'
      WRITE(9,1010)(HMSQLB(IL),IL-1,NHMSQB)
      WRITE(9,*)'mean square slopes for the base material'
      WRITE(9,1010)(SIGSLB(IL),IL-1,NSIGSB)
      WRITE(9,*)'<h**2> for the target material'
      WRITE(9,1010)(HMSQLT(IL),IL-1,NHMSQB)
      WRITE(9,*)'mean square slopes for the target material'
      WRITE(9,1010)(SIGSLT(IL),IL-1,NSIGSB)
      WRITE(9,*)'wavelengths used'
      WRITE(9,1010)(WLENB(IL),IL-1,NWLENB)
      WRITE(9,*)'incident angles'
```

```
 WRITE(9,1010)(ANGLB(IL),IL-1,NANGB)
C
C This block reads in the actual data needed to construct the Mueller matrices
C for both the base and target materials.  Each matrix element is tested for
C angle and wavelength responses. The mean square height and slope are held
C constant for all tests. The meaning of the subscripts for the arrays
C FB(K,I3,I4) and FT(K,I3,I4) are as follows: K is the matrix element
C (K-1—>F11,K-2—>F12, K-3—>F22, K-4—>F33, K-5—>F34, K-6—>F44.
C I3 relates to the wavelength and I4 relates to the incident angle.
C The data is read in for each mean square height and slope as a function of
C angle and wavelength. The tests are performed on the data for each wavelength
C and angle.
C
  DO 90 I1-1,NHMSQB
  DO 91 I2-1,NSIGSB
  T2STMX-0.
  DIFFMX-0.
  RADMX-0.
  DO 92 I3-1,NWLENB
  WRITE(9,9100)WLENB(I3)
 9100 FORMAT('WAVELENGTH',E12.4)
  DO 93 I4-1,NANGB
 2000 FORMAT(6E12.4)
  READ(10,2000)(QB(IL),IL-1,3)
  READ(10,2000)(FB1(IL),IL-1,6)
  READ(11,2000)(QT(IL),IL-1,3)
  READ(11,2000)(FT1(IL),IL-1,6)
C
C This section computes and normalizes the Mueller matrix elements
C and does some analyses.
C
  DO 94 K-1,6
  IF(NORM.EQ.0)THEN
  FB(K,I3,I4)-QB(IBCOR)*FB1(K)
  FT(K,I3,I4)-QT(ITCOR)*FT1(K)
  ELSE
  FB(K,I3,I4)-FB1(K)/FB1(1)
  FT(K,I3,I4)-FT1(K)/FT1(1)
  ENDIF
C
C Careful with putting a zero in the denominator. Should one of the
C matrix elements be zero, assign a value of zero to the term that it
C belongs in and proceed as usual. Also, set all terms which have
C FB>FT equal to zero because the target is assumed optically thick.
C
  IF(ABS(FT(K,I3,I4)).GT.ABS(FB(K,I3,I4)))THEN
  DIFF(K)-FT(K,I3,I4)-FB(K,I3,I4)
   IF(FB(K,I3,I4).NE.0..AND.FT(K,I3,I4).NE.0.)THEN
   RAT-SQRT(ABS(FT(K,I3,I4)*FB(K,I3,I4)))
   RATLOG(K)-DIFF(K)/RAT
   ELSE
   RATLOG(K)-0.
   ENDIF
  ELSE
```

```
      DIFF(K)=0.
      RATLOG(K)=0.
      ENDIF
C     WRITE(8,8001)ABS(FB(K,I3,I4)),ABS(FT(K,I3,I4)),DIFF(K)
C     IF(I3.EQ.NWLENB.AND.I4.EQ.NANGB)WRITE(8,8001)DIFF(K),RATLOG(K)
 8001 FORMAT(3E12.4)
   94 CONTINUE
C
C This block executes all operations that are done on the mtx. elements
C holding the angle and wavelength constant.  This type of test should
C give one an idea which wavelength and angle of incidence to use, and
C which Mueller matrix elements are of most use.
C
      CALL AMTXELE(RATLOG,ARR,KKA,NOF)
      WRITE(9,9019)(KKA(MM),MM=1,NOF)
C     WRITE(8,9019)KKA
C     WRITE(8,8777)RATLOG
      CALL AMTXELE(DIFF,ARR,KKA,NOF)
      WRITE(9,9019)(KKA(MM),MM=1,NOF)
 8777 FORMAT(6E12.4)
C     WRITE(8,9019)KKA
C     WRITE(8,8777)DIFF
 9019 FORMAT(6I3)
      CALL T2STAT(DIFF,RATLOG,LMAX,DIFFMAX,RADMAX,LRMAX,RADIUS,RLOG)
      WRITE(9,9022)RADIUS,RLOG,LMAX,DIFFMAX,LRMAX,RADMAX
 9022 FORMAT(2E12.4,I10.4,E12.4,I10.4,E12.4)
C This if block finds the maximum difference between any two matrix elements
C and saves the wavelength and incident angle at which the maximum difference
C occurred.  The matrix element subscript is also saved.
      IF(ABS(DIFFMAX).GT.ABS(DIFFMX))THEN
      DIFFMX=DIFFMAX
      LMX=LMAX
      I3MX=I3
      I4MX=I4
      ENDIF
C This if block finds the largest term in the array RETAOG that occurred.
C the wavelength, incident angle, and subscript are also saved.
      IF(ABS(RADMAX).GT.ABS(RADMX))THEN
      RADMX=RADMAX
      LRMX=LRMAX
      I3RMX=I3
      I4RMX=I4
      ENDIF
C This if block finds the incident angle and wavelength at which the Euclidean
C distance is at a maximum between the set of data for the target material and
C the base material.
      IF(RADIUS.GT.T2STMX)THEN
      T2STMX=RADIUS
      I3RDMX=I3
      I4RDMX=I4
      ENDIF
C This block finds the incident angle and wavelength at which
C RLOG took on its maximum value.
      IF(RLOG.GT.RLOGMX)THEN
```

```
      RLOGMX-RLOG
      I3RLMX-I3
      I4RLMX-I4
      ENDIF
   93 CONTINUE
   92 CONTINUE
      WRITE(9,9101)ANGLB(I4MX),WLENB(I3MX),DIFFMX,LMX
      WRITE(9,9101)ANGLB(I4RMX),WLENB(I3RMX),RADMX,LRMX
      WRITE(9,9101)ANGLB(I4RDMX),WLENB(I3RDMX),T2STMX
      WRITE(9,9101)ANGLB(I4RLMX),WLENB(I3RLMX),RLOGMX
 9101 FORMAT(3E12.4,I10.4)
   91 CONTINUE
   90 CONTINUE
C
C This section finds all local maxima and minima in the imaginary part of
C index of refraction of both materials being compared. The purpose for this
C is to identify resonant wavelengths and to investigate the relationship
C between the index of refraction and the dependence of the Mueller matrix
C on wavelength.
C
      FNAME-BASENK
      CALL SORTNK(FNAME,NMIN,NMAX,WLNMX,WLNMIN,ANKMIN,ANKMAX)
      WRITE(9,9110)
 9110 FORMAT('Wavelengths and values of maxima in k for the base
     *material')
      DO 170 II-1,NMAX
      WRITE(9,9111)WLNMX(II),ANKMAX(II)
  170 CONTINUE
 9111 FORMAT(2E12.4)
      WRITE(9,9112)
 9112 FORMAT('Wavelengths and values of minima in k for the base
     *material')
      DO 171 II-1,NMIN
      WRITE(9,9111)WLNMIN(II),ANKMIN(II)
  171 CONTINUE
      FNAME-TARGNK
      CALL SORTNK(FNAME,NMIN,NMAX,WLNMX,WLNMIN,ANKMIN,ANKMAX)
      WRITE(9,9113)
 9113 FORMAT('Wavelengths and values of maxima in k for the target
     *material')
      DO 172 II-1,NMAX
  172 WRITE(9,9111)WLNMX(II),ANKMAX(II)
      WRITE(9,9114)
 9114 FORMAT('Wavelengths and values of minima in k for the target
     *material')
      DO 173 II-1,NMIN
  173 WRITE(9,9111)WLNMIN(II),ANKMIN(II)
      GO TO 991
  990 WRITE(8,8000)
 8000 FORMAT('INCORRECT INPUT DATA CHECK UNITS 10 AND 11 AND MAKE'/
     1'SURE THAT THE BASE AND TARGET ARE BEING COMPARED FOR THE'/
     2'SAME WAVELENGTHS AND INCIDENT ANGLES')
  992 CLOSE(7)
      CLOSE(8)
```

```
   CLOSE(9)
  CLOSE(10)
  CLOSE(11)
  991 STOP
  END
C
C*******SUBROUTINE MTXELE*********
C
C This routine sorts the arrays containing the terms in the discrimination
C tests of T2STAT in order of largest to smallest. The purpose of this is
C to  find out which matrix elements are most useful at a particular
C angle of incidence and wavelength.
C
  SUBROUTINE AMTXELE(X,Y,K,N)
  INTEGER K(6),KK(6)
  REAL X(6),Y(6)
  DATA KK/1,2,3,4,5,6/
  Y-X
  K-KK
C  WRITE(8,8888)Y
C  WRITE(8,8888)X
 8888 FORMAT(6E12.4)
  DO 100 M-1,5
  DO 100 L-1,6-M
  IF(ABS(Y(L)).LE.ABS(Y(L+1)))THEN
  T-Y(L)
  Y(L)-Y(L+1)
  Y(L+1)-T
  JT-K(L)
  K(L)-K(L+1)
  K(L+1)-JT
  ENDIF
  100 CONTINUE
  N-0
  DO 101 J-1,6
  IF(Y(J).NE.0)N-N+1
  101 CONTINUE
  RETURN
  END
C
C *****SUBROUTINE T2STAT*********
C This routine the magnitude of the vectors (arrays) DIFF and RATLOG.
C It also finds the largest entry and the subscript of that entry for
C each array.
C
  SUBROUTINE T2STAT(DIFF,RATLOG,LMAX,DIFFMAX,RADMAX,
  *LRMAX,RADIUS,RLOG)
  DIMENSION DIFF(6),RATLOG(6)
  RAD-0.
  RADMAX-0.
  RADRAT-0.
  DIFFMAX-0.
  DO 330 L-1,6
  RAD-RAD+DIFF(L)*DIFF(L)
```

```
      RADRAT=RADRAT+RATLOG(L)*RATLOG(L)
      IF(ABS(RATLOG(L)).GT.ABS(RADMAX))THEN
      RADMAX=RATLOG(L)
      LRMAX=L
      ENDIF
      IF(ABS(DIFF(L)).GT.ABS(DIFFMAX))THEN
      DIFFMAX=DIFF(L)
      LMAX=L
      ENDIF
  330 CONTINUE
      RADIUS=SQRT(RAD)
      RLOG=SQRT(RADRAT)
C     WRITE(8,8800)RADMAX
 8800 FORMAT('RADMAX',E12.4)
      RETURN
      END
C
C*******SUBROUTINE SORTNK*******
C
C This routine opens the file containing the values of the index of refraction
C as a function of wavelength. Then the imaginary part of the index and the
C wavelength that corresponds to it are read in and stored in arrays. The real
C part of the index is not saved.  The rest of the routine sorts through the
C array containing the imaginary part of the index and finds all maxima and
C minima. If there is an interval over which there are local extrema and
C several points share the extreme value, then the program will write the
C extreme value at the two endpoints of the interval, and it is implicitly
C understood that k is constant over that interval.  Also the first and last
C values in the file containing the index of refraction as a function of
C wavelength is counted as a maximum or minimum depending on whether
C k is increasing or decreasing at those points.
C
      SUBROUTINE SORTNK(FNAME,NMIN,NMAX,WLNMX,WLNMIN,ANKMIN,ANKMAX)
      CHARACTER*15 FNAME
      REAL WLNMIN(40),WLNMX(40),ANKMIN(40),ANKMAX(40),WLN(500),NK(500)
      INTEGER NMIN,NMAX
      OPEN(UNIT=12,FILE=FNAME)
      READ(12,*)NPTS
      DO 300 M=1,NPTS
      READ(12,*)WLN(M),ANR,NK(M)
C     WRITE(8,8888)WLN(M),ANR,NK(M)
 8888 FORMAT(3E12.4)
  300 CONTINUE
      NMIN=0
      NMAX=0
      IF(NK(1).GT.NK(2))THEN
      NMAX=NMAX+1
      WLNMX(NMAX)=WLN(1)
      ANKMAX(NMAX)=NK(1)
      ELSE
      NMIN=NMIN+1
      WLNMIN(NMIN)=WLN(1)
      ANKMIN(NMIN)=NK(1)
      ENDIF
```

```
 M-2
99 DO 301 J-M,NPTS-1
 IF(NK(J).GT.NK(J-1).AND.NK(J).GT.NK(J+1))THEN
 NMAX-NMAX+1
 WLNMX(NMAX)-WLN(J)
 ANKMAX(NMAX)-NK(J)
 ELSE IF(NK(J).LT.NK(J-1).AND.NK(J).LT.NK(J+1))THEN
 NMIN-NMIN+1
 WLNMIN(NMIN)-WLN(J)
 ANKMIN(NMIN)-NK(J)
C
C Careful about drawing a false conclusion if two neighboring entries in
C the data file are equal.
C
 ELSE IF(NK(J).GT.NK(J-1).AND.NK(J).EQ.NK(J+1))THEN
 NMAX-NMAX+1
 WLNMX(NMAX)-WLN(J)
 ANKMAX(NMAX)-NK(J)
  JJ-J
  MM-1
  DO 302 MMM-J+1,NPTS-1
  MM-MM+1
  IF(NK(MMM).GT.NK(MMM+1))THEN
  NMAX-NMAX+1
  WLNMX(NMAX)-WLN(MMM)
  ANKMAX(NMAX)-NK(MMM)
  GO TO 303
  ENDIF
  IF(NK(MMM).LT.NK(MMM+1))NMAX-NMAX-1
  IF(NK(MMM).LT.NK(MMM+1))GO TO 303
302  CONTINUE
ELSE IF(NK(J).LT.NK(J-1).AND.NK(J).EQ.NK(J+1))THEN
NMIN-NMIN+1
WLNMIN(NMIN)-WLN(J)
ANKMIN(NMIN)-NK(J)
 JJ-J
 MM-1
 DO 305 MMM-J+1,NPTS-1
 MM-MM+1
 IF(NK(MMM).LT.NK(MMM+1))THEN
 NMIN-NMIN+1
 WLNMIN(NMIN)-WLN(MMM)
 ANKMIN(NMIN)-NK(MMM)
 GO TO 303
 ENDIF
 IF(NK(MMM).GT.NK(MMM+1))NMIN-NMIN-1
 IF(NK(MMM).GT.NK(MMM+1))GO TO 303
305  CONTINUE
ENDIF
301 CONTINUE
IF(NK(NPTS).GT.NK(NPTS-1))THEN
NMAX-NMAX+1
WLNMX(NMAX)-WLN(NPTS)
ANKMAX(NMAX)-NK(NPTS)
```

```
  ELSE
 NMIN=NMIN+1
 WLNMIN(NMIN)=WLN(NPTS)
 ANKMIN(NMIN)=NK(NPTS)
 ENDIF
 GO TO 304
 303 M=JJ+MM
 GO TO 99
 304 CLOSE(12)
 RETURN
 END
```

AVI.1.4 SAMPLE CALCULATIONS: PARAMS AND FILENAMES INPUTS, DATA OUTPUT FILES.

AVI.1.4.1 This is the "params" file for the composite (soil) surface.

```
TestC1c.c
TestC1c.d
composite
-1 0. 0.
.3
-1 0. 0.
.5
71 9.0 .05
22 0. 4.
0
compos.nk
.000001 .00001
.0001 .005i
```

AVI.1.4.2 This is the "params" file for the SF96-contaminated surface.

```
TestS1c.c
TestS1c.d
sf96
-1 0. 0.
.3
-1 0. 0.
.5
71 9.0 .05
22 0. 4.
0
sf96.nk
.000001 .00001
.0001 .005
```

AVI.1.4.3 This is an example of a correctly edited version of FILENAMES.

```
TestC1c.d
TestS1a.d
testdetect.out
commdetect.out
compos.nk
sf96.nk
1
1
0
```

AVI.2 DECIDE2

AVI.2.1 Running DECIDE2.

This section contains a general outline of program DECIDE2. Input and output files are presented on a line-by-line basis. Instructions for running the program and its access to sample input and output files are given at the end of this section.

AVI.2.1.1 Functional outline of DECIDE2.

Program DECIDE2 functions in the following manner.

1. Read the input variables.
   a) Quit and write a message to unit 8 if there is a conflict.
2. Write a heading block to the output file (unit 9).
3. Open and read files containing the indices of refraction for the target and background materials.
   a) Compute the relative permittivity $\epsilon_r = (n-ik)^2$ for each material as a function of wavelength.
   b) Locate and save all resonant wavelengths of the target material.
   c) Locate and save all resonant wavelengths of the background material.
   d) Locate and save all wavelengths corresponding to local maxima on the function $k^t - k^b$, where k^t and k^b are the imaginary parts of the indices of refraction for the target and background materials, respectively.
4. Locate initial wavelengths and examine.
   a) The first two are the primary resonant wavelengths for each material.
   b) The third wavelength corresponds to $(k^t - k^b)_{max}$. If this wavelength is equal to either of the first two, the program uses that wavelength that corresponds to the second largest value for a local maximum on $k^t - k^b$.
5. For both materials, compute the Mueller matrices as functions of incident angle for given wavelength selections in (4). The following procedure is executed each time a pair of Mueller matrices is computed.
   a) Interpolate to find ϵ_r for each material (do this only once per wavelength).
   b) Calculate Equations (37) and (39).
   c) Arrange the terms in Equations (37) and (39) in decreasing order of magnitude.
   d) Write to unit 9:
      1) the subscripts of all non-zero terms in Equation (39);
      2) the subscripts of all non-zero terms in Equation (37);
      3) Equation (39), largest term in Equation (39), and the subscript of that term;
      4) and Equation (37), largest term in Equation (37), and the subscript of that term.
   e) A check that the user specified limit on the number of Mueller matrices the program is allowed to compute has not been exceeded.
   f) A check whether the values of Equations (37) or (39) have increased over their previous maximum value.

6. Examine the Mueller elements at adjacent wavelengths and angles computed in (5) for the largest value in Equation (37) or Equation (39). The user must decide which Equation the program bases its decisions on. Adjacent wavelengths and angles are all possible combinations of $\lambda_0 + j\lambda_{step}$ and $\theta_0 + k\theta_{step}$, j, k = -1,0,1.
   a) Check to make sure that calculations are not being repeated. That is, do not compute data for the same angle/wavelength pair twice. If data has already been computed for the wavelength/angle pair in question, go on to the next pair.
   b) Compute ϵ_r for each material (do this once per wavelength).
   c) Compute the Mueller matrices.
      1) If the maximum number of Mueller matrices has been exceeded, quit program.
   d) Calculate Equations (37) and (39).
   e) Sort terms in Equations (37) and (39).
   f) Write to unit (9):
      1) the subscripts of all non-zero terms in Equation (39);
      2) the subscripts of all non-zero terms in Equation (37);
      3) Equation (39), largest term in Equation (39), and the subscript of that term;
      4) and Equation (37), largest term in Equation (37), and the subscript of that term.
   g) Check if the values of Equations (37) and (39) are improved.
7. If after completing step 6 an improvement in the results of Equation's (37) or (39) are detected, go back to step 6. If not, proceed to step 8.
8. Write:
   a) to unit 8 'THE PROGRAM STOPPED BECAUSE NO IMPROVEMENT IN THE RESULTS OF ROUTINE T2STAT HAS BEEN DETECTED';
   b) and to unit 9 'THE MOST PROMISING ANGLE AND WAVELENGTH PAIR IS' (the wavelength and angle are written next to the statement).
9. The Mueller matrices, at this point, for each material have been stored in arrays. To organize this output, its data is sorted then written in a nested loop where $\lambda_1 < \lambda_2 < \ldots < \lambda_n$ and $\theta_1 < \theta_2 < \ldots < \theta_m$. The sorted Mueller elements are written to unit 9 in a nested loop where θ varies more rapidly than λ.
10. Once an optimum angle/wavelength pair λ_0, θ_0 is found, the program computes and analyzes 121 additional Mueller matrices for the target and background materials. These computations are made for every combination of 11 wavelengths and 11 incident angles. The wavelengths are given by $\lambda_0 + j \times 0.05$, j = -5,-4,...,5. The incident angles are given by $\theta_0 + j \times 4$, j = -5,-4,...,5. That is, the wavelength is incremented 11 times in .05 μm steps centered about λ_0, and the incident angle is incremented 11 times in 4 degree steps centered about θ_0. The results of this part of the program are written to units 10, 11, and 12 in a format that is readable by the plotting package DISSPLAY.
    a) Write the Mueller matrices for the target material to unit 10.
    b) Write the Mueller matrices for the background material to unit 11.
    c) Write the results of the Mueller matrix elements analyses to unit 12.
11. Quit the program.

AVI.2.2 Input Files.

Program DECIDE2 needs three input files. The first file, 'DATAIN2,' is used in every run and contains all information needed to control the program. The remaining input files contain indices of refraction of target and background materials as functions of wavelength.

AVI.2.2.1 DATAIN2.

An example of a correctly edited file is given at the end of this section. The line-by-line format is as follows (for multiple entries per line, separate each entry by spaces):

A- filename containing the background material's index of refraction as a function of wavelength (up to 15 characters);

B- filename containing the target material's index of refraction as a function of wavelength (up to 15 characters);

C- filename for data output, unit 9 (15 characters max);

D- filename for commentary output, unit 8 (15 characters max);

E- a descriptive name of the background material (13 characters and no more);

F- and a descriptive name of the target material (13 characters and no more).

G- Values of $<h^2>$ and σ_s^2. These are the mean square height in (μm^2) and mean square slope, respectively, for the background material. (real, real)

H- Values of $<h^2>$ and σ_s^2. These are the mean square height in (μm^2) and the mean square slope, respectively, for the target material. (real, real)

I- The number of θ's, with θ_{min}, then θ_{step}. These values must be in degrees. The program uses this information to fill an array with all incident angles used in the computations. These angles are given by $\theta_k = \theta_{min} + k\,\theta_{step}$, k=0,1,..., *#of* θ's -1. (integer, real, real)

J- The parameters λ_{min}, λ_{max}, λ_{step} These values must be in μm. The program does not consider wavelengths which are $< \lambda_{min}$ or $> \lambda_{max}$ in any of its operations. (real, real, real)

K- The code number RCODEB. It determines which auto-correlation function the program will use for the background material, or to normalize the Mueller matrices to f_{11}. RCODEB=0 (normalize to f_{11}), RCODEB= 1 (Gaussian), RCODEB=2 (N=8), RCODEB=3 (N=6). (integer)

L- The code number, RCODET. It determines which auto-correlation function the program will use for the target material, or to normalize the Mueller matrices to f_{11}. RCODET=0 (normalize to f_{11}), RCODET= 1 (Gaussian), RCODET=2 (N=8), RCODET=3 (N=6). (integer)

M- Values AERR1 and RERR1. These values are the error requests for the IMSL routine QDAG. AERR1 is the absolute error, and RERR1 is the relative error. The IMSL routine QDAG computes the full wave quantity Q. (real, real)

N- Values AERR2 and RERR2. These values are the error requests for the IMSL routine TWODQ. AERR2 is the absolute error, and RERR2 is the relative error. The IMSL routine TWODQ computes the full wave quantity σ_{ij}^{H} [1]. (real, real)

O- Value NQUIT. The program will compute no more than NQUIT Mueller matrices. (integer)

P- Value NCRIT. If NCRIT=1, the program bases its decisions on Equation (37). If NCRIT=0, the program bases its decisions on Equation (39). (integer)

Q- A file given on this line is open to unit 10. 121 Mueller matrices for the target material are written to this file in a format that is readable by the plotting package DISPLAY. (character 15)

R- A file given on this line is open to unit 11. 121 Mueller matrices for the background material are written to this file in a format that is readable by the plotting package DISPLAY. (character 15)

S- A file given on this line is open to unit 12. The results this analyses of 121 Mueller matrices, units 10 and 11, are written to this file in a format that is readable by DISPLAY. (character 15).

AVI.2.2.2 Index of refraction files.

For each run, DECIDE2 reads two files of refractive index data. The names of these files are passed to lines A and B of the file DATAIN2. They are required for two reasons: (1) relative permittivity $\epsilon_r = (n-ik)^2$ is computed by interpolation at every wavelength the program computes Mueller matrices (both λ and ϵ_r are passed to the subroutine RETRO); and (2) the index of refraction specifies where resonant wavelengths occur.

The index of refraction files have the following format:

A- number of wavelengths in the list (integer);

B- λ_1, n_1, k_1 (real, real, real);

......

......

Z- and λ_m, n_m, k_m (real, real, real),

where $\lambda_j < \lambda_{j+1}$. Also, for the interpolation routine to work, m>6. A sample file is shown at the end of this guide.

AVI.2.3 Output Files.

Program DECIDE2 creates five output files. The first containing Mueller elements and the second contains commentary remarks. Data written to the third, fourth, and fifth files is in a format readable by the plotting program DISPLAY. The names for these files are the inputs to DECIDE2, found on lines C and D of 'DATAIN2.'

AVI.2.3.1 Commentary output file.

This file resides on unit 8. Its name is passed on line D of 'DATAIN2.' Unit 8 receives various values for debugging the program in the event of its failure. Several write statements to unit 8 exist in this program, most have been commented out. If necessary, they may be reinstated by removing the C in the leftmost column of the line the write statement appears. During normal operation DECIDE2 writes descriptive messages to unit 8 for the following reasons.

1. The input variable NCRIT appearing on line P of 'DATAIN2' is <0 or >2, the program writes 'ILLEGAL NCRIT' and then stops.

2. The program will write 'THE PROGRAM STOPPED BECAUSE NO IMPROVEMENT IN THE RESULTS OF THE ROUTINE T2STAT HAS BEEN DETECTED' once it had determined the optimum angle/wavelength pair. When this message is written to unit 8 the run has been successful and the program should quit.
3. In the event that one of the input variables RCODEB or RCODET (lines K and L of 'DATAIN2') is 0 and the other is non-zero the program writes 'ILLEGAL COMBINATION FOR RCODEB AND RCODET' then quits.
4. If either RCODEB or RCODET are negative or greater than 3 the subroutine RETRO will write 'ILLEGAL RCODE' and kill the program.
5. Should the number of Mueller matrices calculated exceed NQUIT (line O of DATAIN2) the program will write 'IF YOU WANT TO COMPUTE MORE MUELLER MATRICES, YOU WILL HAVE TO INCREASE THE INPUT VARIABLE NQUIT' and quits.

AVI.2.3.2 Data output file.

This file (unit 9) contains 3 distinct sections; the header block, the analysis of the Mueller matrices, and the Mueller matrix elements as computed by subroutine RETRO. The format of the header block is as follows:

A- a description of the background and target materials;

B- $<h^2>$, σ_s^2 of the background material;

C- $<h^2>$, σ_s^2 of the target material;

D- λ_{min}, λ_{max}, and λ_{step}. The program does not initially compute all wavelengths between λ_{min} and λ_{max} in increments of λ_{step}. It first selects three wavelengths between λ_{min} and λ_{max}. These wavelengths are those for which k^t, k^b, and $k^t - k^b$ are maximum over the interval [$\lambda_{min}, \lambda_{max}$] (the k's are the imaginary parts of the index of refraction for the t-target and b-background materials).

E- A list of all incident angles used by the program. This program computes and analyzes Mueller matrices for both materials at each of these incident angles for the first three wavelengths it selects. The program finds the wavelength λ_0 and angle θ_0 that resulted in the largest value for the discrimination criterion (see line P of 'DATAIN2'). From there, the program computes data at angles and wavelengths near λ_0 and θ_0 until it finds the optimum angle/wavelength pair (Section AVI.1).

F- Values RCODEB and RCODET. These values are input on lines K and L of 'DATAIN2.'

G- Value NCRIT. This value is input on line P of 'DATAIN2.'

The second section in this file contains two similar subsections. Both subsections contain data resulting from the analysis of Mueller matrices. The difference between each subsection is the way the program chooses the wavelengths and incident angles used in subsequent analyses.

In the first subsection of this file data is written in a nested loop where θ varies more rapidly than λ. The inner loop is repeated once for each incident angle written on line E of the header block. The outer loop is repeated three times. First, the wavelength of primary resonance for the background material is determined. Second, the wavelength of primary resonance for the target material is obtained. Finally, the third and final time through the outer loop selects the optimum wavelengths as discussed in the previous sections. (See also line D of the header block).

In the second subsection of this file, the program looks for angles and wavelengths near the angle and wavelength in the previous subsection that produced the largest value for Equation (37) (NCRIT=1) or Equation (39) (NCRIT=0 see line P of 'DATAIN2'). If the program finds an angle/wavelength pair that results in a larger value for Equation (37) or (39) it further searches angles and wavelengths in the neighbc.hood of that pair. This search goes on until the program notices no improvement in the alue of Equation (37) or (39).

The format for both subsections is given below.

A2- Both incident angle and the wavelength are written. This line reads: "WAVELENGTH="λ "INCIDENT ANGLE="θ.

B2- The subscripts of all non-zero components of $\bar{y}$ Equation (39) are listed in decreasing order of the magnitude of the component they represent. As an example, suppose $y_5 > y_1 > y_4$ and $y_2 = y_3 = y_6 = 0$. For the $\bar{y}$ described above, this line would read 5 1 4.

C2- This line is the same as B2, except it is associated with $\bar{x}$ Equation (37).

D2- $|\bar{x}|$, $|\bar{y}|$, j, $(x_j)_{max}$, i, $(y_i)_{max}$

where;

$|\bar{x}|$ is calculated from Equation (37),

$|\bar{y}|$ is calculated from Equation (39),

$(x_j)_{max}$ is the largest component of $\bar{x}$,

and $(y_i)_{max}$ is the largest component of $\bar{y}$,

E2- This program tells the user which incident angle and wavelength resulted in the best discrimination between the background and target materials. The line reads 'THE MOST PROMISING ANGLE AND WAVELENGTH PAIR IS' λ, θ.

DECIDE2 writes all of the Mueller elements computed to the third data block. They are written in a nested loop where θ varies more rapidly than λ. Both θ and λ increase monotonically from their minimum to their maximum value. The format is as follows:

A3- values λ and θ;

B3- values $v_1^t, v_2^t, v_3^t, v_4^t, v_5^t$, and v_6^t;

C3- values $v_1^b, v_2^b, v_3^b, v_4^b, v_5^b$, and v_6^b.

AVI.2.3.3 Mueller matrix output files.

The third and fourth output files are opened as units 10 and 11, respectively. Units 10 and 11 are also used by the program as the index of refraction files for both materials. The program closes units 10 and 11 after indices of refraction data files are read in, and reopens them according to the names on lines Q and R of DATAIN2. The purpose of these files is to make selections of Mueller elements read by the plot program DISPLAY. The third output file contains Mueller elements for the target material, while the fourth output file contains elements for the background material. Both files have 121 Mueller matrices that are formatted in a way identical to RETRO's output. The wavelengths and incident angles used to compute these data are chosen as described in Section AVI.2.1.1.

When using DISPLAY to plot data in either of these files be aware that only one autocorrelation function was computed, selected by input variables on lines K and L of DATAIN2. Also, be aware here that inputs on lines K and L of DATAIN2 determine if DECIDE2 computes and analyzes $\frac{f_{ij}}{f_{11}}$ or f_{ij}.

AVI.2.3.4 Analysis output file.

This output file contains results of analyses of data in the third and fourth output files. These data are in a format readable by DISPLAY. When DISPLAY asks which matrix element analysis to plot enter 1 1 for a plot of Equation (37) relating to element f_{11} and enter 2 1 for a plot of Equation (39), element f_{21}, and so on.

AVI.2.4 Instructions for running DECIDE2.

The steps required to run program DECIDE2 are presented in this section. After the input file 'DATAIN2' has been properly edited and all other input files are loaded, DECIDE2 is ready for execution. DECIDE2 may be run in either batch mode or interactively. The number of θ (line I of 'DATAIN2') and λ_{step} (line J of 'DATAIN2') parameters determine run time. As a rule of thumb run the program in the batch mode if the number of θ's > 30 or if λ_{step} < .02 over the normal 9-12 μm band of wavelengths.

AVI.2.4.1 Interactive runs.

Typing "decide2.run" after the prompt will compile, link, and execute the program using data stored in 'DATAIN2.' Executable code becomes stored in "decide2.xqt"; simple type this command to iterate a program run during the same login after changing 'DATAIN2.' Before logging off, remove 'decide2.xqt' for it is a rather large file. The object file 'decide2.o' was removed by 'decide2.run' immediately following linking.

AVI.2.4.2 Batch runs.

Typing "examp2dec" after the prompt runs this program in the batch mode using the data stored in 'DATAIN2.' The file "examp2dec" submits the file "decide2.bat" to the batch que. For this reason, the user must make sure that the file "decide2.bat" is in storage on before typing "examp2dec."

AVI.2.5 DECIDE2 SOURCE CODE.

```
C This work was done for the CRDEC on the Aberdeen Proving Grounds
C Edgewood Area.
C This work was done by S. Mark Haugland under contract DAAD05-89-P-0427.
C
C The purpose of this program is to locate optimum combinations of
C wavelength and incident angle for use in distinguishing between
C a contaminated and a dry surface. This program assumes the
C contaminant forms an optically thick layer on the background surface.
C
C DECIDE2 computes and analyzes Mueller matrices every time it calls
C the subroutine RETRO. These data are used to distinguish between
C the background (base) material and the target (contaminant) material.
C Each material's Mueller matrix is a function of incident angle
C , wavelength, mean square height, and mean square slope.
C DECIDE2 locates the primary resonant wavelength for each material
C and the wavelength at which the imaginary part of the index of refraction
C for each material differ by the greatest amount. At each of these
C wavelengths, DECIDE2 computes Mueller matrices for each material as
C a function of incident angle. Immediately following computation,
C each pair of Mueller matrices is analyzed (see the user manual for
C details). The program identifies the combination of these wavelengths
C and incident angles that resulted in the best discrimination between the
C two surfaces. Next, angle/wavelength pairs near the one that resulted in the
C best discrimination are examined. If there is an increase in the
C discrimination at some of these angles, the program remembers the one
C that resulted in the greatest increase. Angles near that one are examined
C next. This process is repeated until there is no further increase in
C discrimination.
C
C This program uses some IMSL routines. For more information about
C the IMSL routines, see the subroutine RETRO.
C
  PROGRAM DECIDE2
  REAL ANG(100),ANB(500),ANT(500),AKB(500),AKT(500),KDIFF(500)
  REAL WLNB(500),WLNT(500),HMSQB,SIGSB,HMSQT,SIGST,WLN(100,100),
  *ANG1,ANGINC,WLMAX,WLMIN,WLSTEP,WLNMAXT(40),WLNMINT(40)
  REAL FFT(6),FFB(6),DIFF(6),RATLOG(6),WLNWLN(100),FT(6,100,100)
  REAL AKMAXB(40),WLNMAXD(40),WLNMIND(40),AKMIND(40),AKMAXD(40)
  REAL AKMINT(40),AKMAXT(40),WLNMAXB(40),WLNMINB(40),AKMINB(40)
  REAL FB(6,100,100),ARR(6)
  INTEGER RCODEB,RCODET,NPTSB,NPTST,NANGMX,KKA(6)
  COMPLEX ERB(500),ERT(500),ERESTT,ERESTB,CNT(500),CNB(500)
  COMPLEX NKTEST,NKBEST
 CHARACTER*15 OUTNM, COMNM, BACKNK, TARGNK, TARGMTX, BACKMTX, ANALMTX
  CHARACTER*13 BACKMAT,TARGMAT
  COMMON/TWO/WLMIN,WLMAX
C
C Read names of input and output files and open them
C
  OPEN(UNIT-7,FILE-'DATAIN2')
  READ(7,'(A)')BACKNK
```

```
      OPEN(UNIT=10,FILE=BACKNK)
    READ(7,'(A)')TARGNK
    OPEN(UNIT=11,FILE=TARGNK)
    READ(7,'(A)')OUTNM
    OPEN(UNIT=9,FILE=OUTNM)
    READ(7,'(A)')COMNM
    OPEN(UNIT=8,FILE=COMNM)
  C
    PI=ABS(ATAN2(0.,-1.))
  C
  C Read names of materials to be examined, and information needed to
  C characterize them.
  C
    READ(7,'(A)')BACKMAT
    READ(7,'(A)')TARGMAT
    READ(7,*)HMSQB,SIGSB
    READ(7,*)HMSQT,SIGST
    READ(7,*)NANGMX,ANG1,ANGINC
    IF(NANGMX.LT.0)THEN
    NANGMX=-NANGMX
    READ(7,*)(ANG(IL),IL=1,NANGMX)
    ELSE
    DO 40 IL=1,NANGMX
    40 ANG(IL)=ANG1+(IL-1)*ANGINC
    ENDIF
    READ(7,*)WLMIN,WLMAX,WLSTEP
    WRITE(9,9002)BACKMAT,TARGMAT
  9002 FORMAT(2A15)
    WRITE(9,*)HMSQB,SIGSB
    WRITE(9,*)HMSQT,SIGST
    WRITE(9,9000)WLMIN,WLMAX,WLSTEP
  9000 FORMAT(5E12.4)
    WRITE(9,9000)(ANG(IL),IL=1,NANGMX)
  C
  C RCODEB and RCODET are the code variables that set the auto-correlation function
C for the background and target materials respectively. It is allowed for the
C target and background materials to have different auto-corr functions.RCODEB,
C T=1-->Gaussian, =2-->N=8, =3-->N=6. There is an option to normalize
  C all mtx. elements to P11 this will be the case if both RCODEB and RCODET=0
  C Thus both RCODEB and RCODET equaling 0 causes the program to analyze Pij/P11
  C rather than Pij.
  C
    READ(7,*)RCODEB
    READ(7,*)RCODET
    IF(RCODEB.EQ.0.AND.RCODET.NE.0)GO TO 991
    IF(RCODET.EQ.0.AND.RCODEB.NE.0)GO TO 991
    WRITE(9,9003)RCODEB,RCODET
  9003 FORMAT('CODE FOR AUTO-CORR FUNCTIONS AND NORMALIZATION', 2I5)
  C
  C Read the error controls for the integration routines. AERR1 and RERR1
  C are for QDAG and AERR2 and RERR2 are for TWODQ.
  C
    READ(7,*)AERR1,RERR1
    READ(7,*)AERR2,RERR2
```

```
C
C NCRIT=1 --> decisions are made based on the
C Euclidean distance between the Mueller matrix for the target and the Muller
C matrix for the background material. NCRIT=0--> decisions are made based on
C a test which is independent of the size variations in Muller matrix elements
C in the same Mueller matrix. NQUIT is the maximum number of Muller matrices
C you will this program to compute in a single run.
C
  READ(7,*)NQUIT
  READ(7,*)NCRIT
  WRITE(9,9001)NCRIT
 9001 FORMAT('NCRIT',I5)
  IF(NCRIT.LT.0.OR.NCRIT.GT.2)THEN
  WRITE(8,8767)
 8767 FORMAT('ILLEGAL NCRIT')
  STOP
  ENDIF
C
C The files TARGMTX and BACKMTX are written Mueller matrices for the
C target and background materials. These matrices are written in a
C way DISSPLAY can plot them. Once the optimum angle and wavelength
C has been identified, the program computes a rectangle of Mueller matrices
C centered at that point. The rectangle contains a range of 40 degrees in 4
C degree intervals for theta, and .5 microns in .05 micron intervals for
C lambda. These matrices are analyzed by the routine T2STAT and these
C results are written ANALMTX in a form that they can be plotted by
C DISSPLAY.
C
C Open TARGMTX, and BACKMTX as units 10 and 11 respectively. Do this after
C 10 and 11 have been closed by the part of the program that reads
C the indices of refraction in. Open ANALMTX as unit 12.
  READ(7,'(A)')TARGMTX
  READ(7,'(A)')BACKMTX
  READ(7,'(A)')ANALMTX
C
C Read in the index of refraction for the background material and compute the
C relative permittivity.
C
  READ(10,*)NPTSB
  DO 41 IL=1,NPTSB
  READ(10,*)WLNB(IL),ANB(IL),AKB(IL)
  X=ANB(IL)
  Y=AKB(IL)
  CNB(IL)=(1.,0.)*X+(0.,-1.)*Y
  ERB(IL)=(1.,0.)*(X*X-Y*Y)-(0.,1.)*ABS(2.*X*Y)
C  WRITE(8,8001)WLNB(IL),CNB(IL)
 41 CONTINUE
  CLOSE(10)
C
C Read in the index of refraction for the target material and compute the
C relative permittivity.
C
  READ(11,*)NPTST
  DO 42 IL=1,NPTST
```

```
        READ(11,*)WLNT(IL),ANT(IL),AKT(IL)
      X=ANT(IL)
      Y=AKT(IL)
      ERT(IL)=(1.,0.)*(X*X-Y*Y)-(0.,1.)*ABS(2.*X*Y)
 2121 FORMAT(3E12.4)
      CNT(IL)=(1.,0.)*X+(0.,-1.)*Y
C     WRITE(8,8001)WLNT(IL),CNT(IL)
   42 CONTINUE
      CLOSE(11)
C
C Find all local maxima and minima in the absolute value of the imaginary
C part of the index of refraction for each material.
C
CALL SORTNK(NMINT, NMAXT, WLNMAXT, WLNMINT, AKMINT, AKMAXT, NPTST,
     *AKT,WLNT)
      CALL SORTNK(NMINB,NMAXB,WLNMAXB,WLNMINB,AKMINB,AKMAXB,NPTSB,
     *AKB,WLNB)
C
C Find all local maxima and minima in the difference of k for the target
C material and k for the background material. There is a technical
C problem if the optical constants of each material are not known at the
C same wavelengths. This is handled by interpolating the function that
C is known at fewer wavelengths at all wavelengths for which the other set
C of optical constants is known.
C
C     WRITE(8,*)NPTSB
      IF(NPTSB.GT.NPTST)THEN
      DO 43 I=1,NPTSB
      WLEN=WLNB(I)
      IF(WLEN.GT.WLNT(NPTST))GO TO 43
      CALL ERCMP(WLNT,CNT,WLEN,NKTEST)
      KDIFF(I)=ABS(AIMAG(NKTEST))-AKB(I)
C     IF(ABS(NKTEST).GT.ABS(CNB(I)))KDIFF(I)=ABS(NKTEST-CNB(I))
C     WRITE(8,8812)I,WLEN,KDIFF(I),NKTEST
C     WRITE(8,8812)I,WLEN,NKTEST
 8812 FORMAT(I5,5E12.4)
   43 CONTINUE
CALL SORTNK(NMIND, NMAXD, WLNMAXD, WLNMIND, AKMIND, AKMAXD, NPTSB,
     *KDIFF,WLNB)
      ELSE
      DO 44 I=1,NPTST
      WLEN=WLNT(I)
      IF(WLEN.GT.WLNB(NPTSB))GO TO 44
      CALL ERCMP(WLNB,CNB,WLEN,NKBEST)
      KDIFF(I)=AKT(I)-ABS(AIMAG(NKBEST))
C     IF(ABS(NKBEST).LT.ABS(CNT(I)))KDIFF(I)=ABS(NKBEST-CNT(I))
C     WRITE(8,8812)I,WLEN,KDIFF(I),NKBEST
   44 CONTINUE
CALL SORTNK(NMIND, NMAXD, WLNMAXD, WLNMIND, AKMIND, AKMAXD, NPTST,
     *KDIFF,WLNT)
      ENDIF
C     DO 444 NN=1,NMAXD
C     WRITE(8,8009)NN,WLNMIND(NN),AKMIND(NN)
C 444 CONTINUE
```

```
  C DO 445 NN-1,NMINT
C  WRITE(8,8009)WLNMINT(NN),AKMINT(NN)
C 445 CONTINUE
C
C Start computing Mueller matrices. Start with the wavelengths that correspond
C to the primary resonance for each material. Also look at the wavelength
C for which the difference between the imaginary part of the index of refraction
C for the target material and that for the base material is at a maximum.
C
   WLNWLN(1)-WLNMAXB(1)
   IF(WLNWLN(1).NE.WLNMAXT(1))THEN
   WLNWLN(2)-WLNMAXT(1)
   ELSE
   WLNWLN(2)-WLNMAXT(2)
   ENDIF
   JK-2
   DO 45 JJJ-1,NMAXD
   J-JJJ
   IF(WLNMAXD(J).LT.1.E-05)GO TO 45
   IF(AKMAXD(J).LT.0.)GO TO 45
   IF(WLNWLN(2).NE.WLNMAXD(J).AND.WLNWLN(1).
   *NE.WLNMAXD(J))THEN
   JK-3
   WLNWLN(3)-WLNMAXD(J)
   GO TO 46
   ENDIF
  45 CONTINUE
  46 MWLN-0
   MATCNT-0
   T2STMX-0.
   RADMX-0.
   DO 89 J-1,JK
   MWLN-MWLN+1
   WLN(J,1)-WLNWLN(J)
   CALL ERCMP(WLNB,ERB,WLN(J,1),ERESTB)
C  PRINT 1111,WLN(J,1),ERESTB
 1111 FORMAT(3E12.4)
   CALL ERCMP(WLNT,ERT,WLN(J,1),ERESTT)
C  WRITE(8,8001)WLN(J,1),ERESTT,ERESTB
 8001 FORMAT('wavelength index ',5E12.4)
   DO 89 I-1,NANGMX
   WLN(J,I)-WLN(J,1)
C  WRITE(8,*)WLN(J,I),ANG(I)
   WRITE(9,9569)WLN(J,I),ANG(I)
 9569 FORMAT('WAVELENGTH- ',E14.6,' INCIDENT ANGLE- ',E14.6)
   CALL RETRO(ERESTT,HMSQT,SIGST,WLN(J,I),ANG(I),AERR1,RERR1,AERR2
   *,RERR2,RCODET,PFT,PI)
   CALL RETRO(ERESTB,HMSQB,SIGSB,WLN(J,I),ANG(I),AERR1,RERR1,AERR2
   *,RERR2,RCODEB,PFB,PI)
 9200 FORMAT('PFT OVER PFB',6E10.4)
   MATCNT-MATCNT+2
C
C Save and analyze results from retro.
C
```

```
      DO 78 K=1,6
      FT(K,J,I)=FFT(K)
      FB(K,J,I)=FFB(K)
   78 CONTINUE
      CALL T2STAT(DIFF,RATLOG,LMAX,DIFFMAX,RADMAX,
     *LRMAX,RADIUS,RLOG,FFT,FFB)
      CALL AMTXELE(RATLOG,ARR,KKA,NOF)
      WRITE(9,9099)(KKA(NNB),NNB=1,NOF)
      CALL AMTXELE(DIFF,ARR,KKA,NOF)
      WRITE(9,9099)(KKA(NNB),NNB=1,NOF)
      WRITE(9,9098)RADIUS,RLOG,LMAX,DIFFMAX,LRMAX,RADMAX
 9098 FORMAT(2E12.4,I10.4,E12.4,I10.4,E12.4)
 9099 FORMAT(6I3)
 8019 FORMAT(2E12.4)
 8003 FORMAT('FF',E12.4)
      CALL SAVE(NCRIT,T2STMX,RADMX,J,I,I3MX,I4MX,I3RDMX,I4RDMX,
     *JWLSET,RADIUS,RLOG)
      IF(MATCNT.GE.NQUIT)GO TO 999
   89 CONTINUE
  777 IF(NCRIT.EQ.0)I3SET=I3RDMX
      IF(NCRIT.EQ.1)I3SET=I3MX
      IF(NCRIT.EQ.0)I4SET=I4RDMX
      IF(NCRIT.EQ.1)I4SET=I4MX
C     WRITE(8,8121)I3SET,I4SET,WLN(I3SET,I4SET),ANG(I4SET)
 8121 FORMAT('I3SET,I4SET',2I6,2E12.4)
C     WRITE(8,8122)T2STMX,RADMX
 8122 FORMAT('RADIUS RLOG',2E12.4)
C
C Look at wavelengths and angles of incidence close to those for which
C things look most promising.
C
      KILL=1
      DO 79 J=-1,1
      W=WLNWLN(I3SET)+J*WLSTEP
      IF(W.LT.WLMIN)W=WLMIN
      IF(W.GT.WLMAX)W=WLMAX
      CALL ERCMP(WLNB,ERB,W,ERESTB)
      CALL ERCMP(WLNT,ERT,W,ERESTT)
      DO 80 K=-1,1
      IF(J.EQ.0.AND.K.EQ.0)GO TO 80
      KKK=K+I4SET
      IF(KKK.LE.0.OR.KKK.GT.NANGMX)GO TO 80
      CALL WLANGSET(W,MWLN,WLNWLN,WLN,J,K,I3SET,I4SET,NCALC,I3,I4)
C     WRITE(8,*)NCALC
      IF(NCALC.EQ.2)GO TO 80
C     WRITE(8,*)'Wlen=',WLN(I3,I4),ANG(I4)
      WRITE(9,9569)WLN(I3,I4),ANG(I4)
      CALL RETRO(ERESTT,HMSQT,SIGST,WLN(I3,I4),ANG(I4),AERR1,RERR1,
     *AERR2,RERR2,RCODET,FFT,FI)
      CALL RETRO(ERESTB,HMSQB,SIGSB,WLN(I3,I4),ANG(I4),AERR1,RERR1,
     *AERR2,RERR2,RCODEB,FFB,FI)
      MATCNT=MATCNT+2
C
C Save and analyze results from retro.
```

```
C
DO 791 KG-1,6
FT(KG,I3,I4)-FFT(KG)
FB(KG,I3,I4)-FFB(KG)
791 CONTINUE
CALL T2STAT(DIFF,RATLOG,LMAX,DIFFMAX,RADMAX,
*LRMAX,RADIUS,RLOG,FFT,FFB)
CALL AMTXELE(RATLOG,ARR,KKA,NOF)
WRITE(9,9099)(KKA(NNB),NNB-1,NOF)
CALL AMTXELE(DIFF,ARR,KKA,NOF)
WRITE(9,9099)(KKA(NNB),NNB-1,NOF)
WRITE(9,9098)RADIUS,RLOG,LMAX,DIFFMAX,LRMAX,RADMAX
CALL SAVE(NCRIT,T2STMX,RADMX,I3,I4,I3MX,I4MX,I3RDMX,I4RDMX,
*JWLSET,RADIUS,RLOG)
IF(JWLSET.EQ.2)KILL-2
C WRITE(9,*)'kill,matcnt ',KILL,MATCNT
80 CONTINUE
79 CONTINUE
IF(KILL.EQ.1)GO TO 996
IF(MATCNT.GE.NQUIT)GO TO 999
GO TO 777
996 WRITE(8,*)'PROGRAM STOPPED BECAUSE NO IMPROVEMENT IN THE'
WRITE(8,*)'RESULTS OF ROUTINE T2STAT HAS BEEN DETECTED'
GO TO 998
991 WRITE(8,8000)
8000 FORMAT('ILLEGAL COMBINATION FOR RCODEB AND RCODET')
GO TO 1090
999 IF(MATCNT.GE.NQUIT)WRITE(8,8080)
8080 FORMAT('IF YOU WANT TO COMPUTE MORE MATRICES YOU WILL HAVE TO'
*,/,' INCREASE THE INPUT VARIABLE NQUIT')
C
C Arrange the Mueller matrices so that they can be written to unit 9
C in a nested loop in which ANG(I4) varies more rapidly than WLNWLN(I3).
C
998 WRITE(9,9013)WLN(I3SET,I4SET),ANG(I4SET)
9013 FORMAT('THE MOST PROMISING WAVELENGTH AND ANGLE PAIR IS',2E12.4)
DO 98 J-1,MWLN-1
DO 98 I3-1,MWLN-J
IF(WLNWLN(I3).GT.WLNWLN(I3+1))THEN
T1-WLNWLN(I3)
WLNWLN(I3)-WLNWLN(I3+1)
WLNWLN(I3+1)-T1
DO 111 I4-1,NANGMX
T1-WLN(I3,I4)
WLN(I3,I4)-WLN(I3+1,I4)
WLN(I3+1,I4)-T1
DO 111 K-1,6
T1-FT(K,I3,I4)
FT(K,I3,I4)-FT(K,I3+1,I4)
FT(K,I3+1,I4)-T1
T1-FB(K,I3,I4)
FB(K,I3,I4)-FB(K,I3+1,I4)
FB(K,I3+1,I4)-T1
111 CONTINUE
```

```
        ENDIF
   98 CONTINUE
C Write Mueller Matrices for each material to unit (9)
      DO 112 I3-1,MWLN
      DO 112 I4-1,NANGMX
      IF(WLN(I3,I4).GT.1.E-05)THEN
      WRITE(9,9999)WLNWLN(I3),ANG(I4)
 9999 FORMAT(2E14.6)
      WRITE(9,9998)(PT(K,I3,I4),K-1,6)
      WRITE(9,9998)(PB(K,I3,I4),K-1,6)
 9998 FORMAT(6E12.4)
      ENDIF
  112 CONTINUE
C Determine the 11 wavelengths and 11 angles that the program computes
C and analyzes Mueller matrices for. These matrices and the results of
C the analysis are written to units 10, 11, and 12.
      CALL SETHETWL(ANG,WLNWLN,I3SET,I4SET)
      OPEN(UNIT-10,FILE-TARGMTX)
      OPEN(UNIT-11,FILE-BACKMTX)
      OPEN(UNIT-12,FILE-ANALMTX)
C Write heading blocks to units 10, 11, and 12 so DISSPLAY can read the
C data contained in those files.
      WRITE(10,'(A)')TARGMAT
      WRITE(11,'(A)')BACKMAT
      WRITE(12,'(A)')BACKMAT
      NH-1
      NS-1
      NANG-11
      NWLEN-11
      WRITE(10,1000)NH,NS,NWLEN,NANG
      WRITE(11,1000)NH,NS,NWLEN,NANG
      WRITE(12,1000)NH,NS,NWLEN,NANG
 1000 FORMAT(I10.4)
      WRITE(10,1001)HMSQT
      WRITE(10,1001)SIGST
      WRITE(10,1001)(WLNWLN(I3),I3-1,11)
      WRITE(10,1001)(ANG(I4),I4-1,11)
      WRITE(11,1001)HMSQB
      WRITE(11,1001)SIGSB
      WRITE(11,1001)(WLNWLN(I3),I3-1,11)
      WRITE(11,1001)(ANG(I4),I4-1,11)
      WRITE(12,1001)HMSQB
      WRITE(12,1001)SIGSB
      WRITE(12,1001)(WLNWLN(I3),I3-1,11)
      WRITE(12,1001)(ANG(I4),I4-1,11)
 1001 FORMAT(5E12.4)
C Compute and analyze the matrices.
      DO 113 I3-1,11
      W-WLNWLN(I3)
      CALL ERCMP(WLNB,ERB,W,ERESTB)
      CALL ERCMP(WLNT,ERT,W,ERESTT)
      DO 113 I4-1,11
      CALL RETRO(ERESTT,HMSQT,SIGST,W,ANG(I4),AERR1,RERR1,
     *AERR2,RERR2,RCODET,PPT,PI)
```

```
      CALL RETRO(ERESTB,HMSQB,SIGSB,W,ANG(I4),AERR1,RERR1,
     *AERR2,RERR2,RCODEB,FFB,FI)
     CALL T2STAT(DIFF,RATLOG,LMAX,DIFFMAX,RADMAX,
     *LRMAX,RADIUS,RLOG,FFT,FFB)
     Q1=1.
     Q2=1.
     Q3=1.
     WRITE(10,1002)Q1,Q2,Q3
     WRITE(11,1002)Q1,Q2,Q3
     WRITE(12,1002)Q1,Q2,Q3
     WRITE(10,1002)(FFT(I),I=1,6)
     WRITE(11,1002)(FFB(I),I=1,6)
     WRITE(12,1002)RADIUS,RLOG,DIFFMAX,RADMAX,DIFF(1),RATLOG(1)
 1002 FORMAT(6E12.4)
 113 CONTINUE
     CLOSE(7)
     CLOSE(8)
     CLOSE(9)
     CLOSE(10)
     CLOSE(11)
     CLOSE(12)
 1090 STOP
     END
C
C**********SUBROUTINE ERCMP************
C
C THIS SUBROUTINE COMPUTES THE VALUE ER FOR USE IN RETRO.
C Since this routine interpolates complex functions, it is also used
C to interpolate the index of refraction. This routine is a function in the
C original version of Retro and was taken of that program and modified for use
C here. The argument list is as follows:
C        WLN- An array containing the wavelengths at which the index of
C          refraction is known. This information was read in from unit
C   10 or 11.
C        ER- This is a complex array containing the relative permittivities
C   as a function of wavelength. All wavelengths are in microns.
C  WLEN-The wavelength in microns for which the relative permittivity is
C   desired.
C  EREST-The relative permittivity returned to the main program.
C
     SUBROUTINE ERCMP(WN,ER,WLEN,EREST)
     INTEGER NPTS,DEG,I,J,MIN,MAX,DDEG,DNPTS
     REAL WN(500)
     REAL WLEN,WNP,FACTOR,LI
     COMPLEX ER(500),EREST
C
C this converts wavelength in microns to wave number in 1/cm. COMMENT
C OUT AND WORK WITH THE WAVELENGTH RATHER THAN THE WAVENUMBER!!!!!
C  WNP=10000./WLEN
     WNP=WLEN
     DEG=3
     I=(DEG+1)/2
C  WRITE(8,8002)WLEN
     IF(WN(I).GT.WNP) THEN
```

```
      MIN-1
      MAX-MIN+DEG
      GOTO 211
     ENDIF
 200 IF(WN(I).GT.WNP) GOTO 210
     I-I+1
     IF(I.EQ.NPTS-(DEG+1) / 2)THEN
      MAX-NPTS
      MIN-NPTS-DEG
      GOTO 211
     ENDIF
    GOTO 200
C
 210 MIN-I-DEG/2-1
    MAX-MIN+DEG
 211 FACTOR - 1.
C  WRITE(8,*)'MIN,MAX ',MIN,MAX
C
 8002 FORMAT(E12.4)
    DO 220 J- MIN,MAX
     IF(WNP.NE.WN(J))GOTO 220
    EREST-ER(J)
     RETURN
 220 FACTOR-FACTOR*(WNP-WN(J))
C
    EREST-(0.,0.)
    DO 230 I-MIN,MAX
     LI-FACTOR/(WNP-WN(I))
     DO 240 J- MIN,MAX
 240  IF(I.NE.J)LI-LI/(WN(I)-WN(J))
 230  EREST-EREST+ER(I)*LI
    RETURN
    END
C*******************************************************************
C**********SUBROUTINE RETRO***************
C
C THIS WORK WAS ALTERED BY S.M. HAUGLAND ON 6-23-89.
C THE REVISIONS TO RETRO ALLOW IT TO BE CALLED BY ANOTHER PROGRAM
C AS A SUBROUTINE.
C
c This work was done for the CRDEC on the Aberdeen Proving Grounds
c Edgewood Area.
c This work was done by Craig M. Herzinger under contract 88MQ450.
c
c This program uses the Full-wave Theory for computing the scattering
c of a plane wave from a randomly rough surface.
c
c This program is for SINGLE scatter from an ISOTROPIC rough surface.
c This program is for BACKSCATTER only.
c This program is for DIFFUSE scattering only.
c
c Single scatter implies the reflected radiation struck the rough
c surface only one time and that multiple scattering is unaccounted for.
c An isotropic surface is considered to invariant to rotation and translation
```

```
c in terms of the average scattering.
c Backscatter implies the receiver and detector are at the same point,
c with the same orientation, at a point far from the surface.
c Only diffuse scattering is calculated because the coherent specular term
c which occurs at normal incidence for backscatter is dropped.
c
c This program calculates the 8 generally non-zero Mueller Mtx elements,
c F11,12,21,22,33,34,43, and 44, for use with the standard Stokes Vector
c notation. Of these eight two pairs are degenerate, F12-F21 and F34--F43.
c The elements are calculated on a per solid angle basis so that the calculated
c Mueller Mtx is absolutely correct to within a scalar constant. The scalar
c constant is based upon the size of the solid angle intercepted by the
c detector for a particular experimental setup. For this work to be valid
c the detector must look at range of returned angles close enough to
c pure backscatter that the backscattered return is a good approximation
c of the entire reflected range. Also, the solid angle intercepted by the
c detector must be invariant to incident angle and wavelength.
c
c The program first calculates the scattering mtx, S, for use with the
c modified Stokes Vector notation, and this is then transformed into the
c desired form.
c
c The elements of S can be written as a product of two values, Q, and
c a 2-d slope averaging integral. This is allowed by the Full-wave
c Theory ONLY because slopes and heights are considered uncorrelated.
c Q is a function of incident angle, surface height auto-correlation
c function, and free space radiation wavelength. The slope integral
c is function of incident angle, mean squared slope, and wavelength
c through the surfaces relative dielectric constant.
c
c For the assumed isotropic surface, Q, normally a 2-d integral can
c be rewritten in polar coordinates and transformed into a 1-d integral.
c Q, is computed by subroutine QCMP for 3 different spectral density/
c surface height auto-correlation functions. The inputs for the auto-corr.
c functions are mean squared height and mean squared slope. The 3
c functions are Gaussian, N-8, and N-6.
c
c The slope averaging integrals account for all possible combinations of
c slopes in the x and z directions. Considering various polarizations and
c phase relationships, 16 unique integrals are possible to complete S.
c But 8 integrals -> 0 because the integrand is odd.
c 3 others converge to 1 value, and another -> 0 because the integrand
c is proportional to the imaginary part of a real number.
c This leaves 5 unique 2-d integrals performed by subroutine IDDCMP.
c The 5 values could be used compute S but they are recombined in one
c step to form the Mueller mtx elements, Fij.
c
c The following version 10.0 IMSL routines are required:
c   QDAG, TWODQ, BSJ0, BSK0, BSK1, and ERFC.
c These are available from IMSL, customer relations, sixth floor,
c NBC Building 7500 Bellaire Boulevard, Houston, Texas 77036-5085, USA.
c Telephone (713)772-7927   Telex: 79-1923 IMSL INC HOU
c
c
```

```
c
c Unit 8 is sent various values for debugging the program and determining
c where something may have gone wrong. It sends values for monitoring the
c integration routines. Most writes to unit 8 have been commented out, but
c can be reinstated should the program have run time problems.
c
C
      SUBROUTINE RETRO(EREST, HHMSQ, SSIGS, WWLEN, ANG, AERR1, RERR1, AERR2
     *, RERR2, RCODE, FF, PPI)
      REAL PI
     1  HMSQ,CLEN,WLEN,CLENSQ,SIGS,K0,K0SQ,CSTHT,SNTHT,THETA,
     3  IDD(16),F(16),FF(6),Q(3),
     4  AERR1,RERR1,AERR2,RERR2
C Don't use SVIDD because as a called routine RETRO does not remember
C what the last set of parameters input to it were. The main program
C does this.
C  REAL SVIDD(100,16)
C
      COMPLEX ER,EREST
C
      INTEGER IL,DCODE,RCODE,IFLAG1
      COMMON/ONE/HMSQ,CLEN,CLENSQ,WLEN,THETA,SNTHT,CSTHT,SIGS,K0,PI
C
C
C THE DESIRED PARAMETERS ARE USED TO COMPUTE THE MUELLER
C MATRIX ELEMENTS BY FIRST CALCULATING A SCALING 1-D INTEGRAL, Q,
C AND THEN CALCULATING 16 2-D INTEGRALS, IDD1..IDD16, THAT ARE COMBINED IN THE
C CORRECT MANNER TO GIVE THE MUELLER MATRIX ELEMENTS FOR THE
C STANDARD STOKES VECTOR NOTATION.
C IN REALITY FOR THIS WORK ONLY 5 2-D INTEGRALS NEED BE CALCULATED BUT
C THE PROGRAM IS SET UP GENERALLY.
C The correlation length can be calculated when hmsq and sigs are fixed
      HMSQ-HHMSQ
      WLEN-WWLEN
      PI-PPI
      SIGS-SSIGS
      CLENSQ-4.*HMSQ/SIGS
      CLEN-SQRT(CLENSQ)
C   PRINT *,HMSQ,SIGS,WLEN,ANG,RCODE,PI
C
C  IOLD-0
      K0-2*PI/WLEN
      K0SQ-K0*K0
C   write(8,*)'Relative dielectric constant ',ER
C
C Theta is the angle between the incident direction and the normal to the
C reference plane in radians
      THETA-PI*ANG/180.0
      CSTHT-COS(THETA)
      SNTHT-SIN(THETA)
C
C  write(8,*)'Angle(deg)  Wlen  Mean Sq Slope  Mean Sq Hgt'
C  write(8,*)ANG,WLEN,SIGS,HMSQ
C
```

```
 C
C THIS SECTION COMPUTES THE SCALAR Q VALUES FOR 3 AUTO-CORR FUNCTIONS
C RCODE-1 -> GAUSSIAN  RCODE-2 -> N-8  RCODE-3 -> N-6 RCODE-0->
C YOU ARE NORMALIZING TO F11 AND DO NOT NEED A Q VALUE BECAUSE Q IS THE
C SAME FOR ALL MATRIX ELEMENTS.
C IF ALL Q VALUES ARE TREATED AS 0 THEN IFLAG1-0 AND THE IDD'S DO NOT
C NEED TO BE CALCULATED.
  IFLAG1-0
  IF(RCODE.NE.0)CALL QCMP(Q(RCODE),AERR1,RERR1,RCODE)
  IF(RCODE.EQ.0)Q(RCODE)-1.
  IF(Q(RCODE).GT.0.)IFLAG1-1
C
C
C THIS SECTION COMPUTES THE 16 IDD VALUES NEEDED FOR THE MUELLER MTX.
C NOTE ONLY 5 DISTINCT INTEGRATIONS ARE DONE. THE OTHERS ARE KNOWN FOR
C OTHER REASONS ASSIGNED TO THE FOLLOWING CONDITIONAL ASSIGNMENTS.
C IF IFLAG1-1 THEN THERE IS A REASON TO CALCULATE THESE VALUES
C IF Er IS HELD CONSTANT OVER A RANGE OF WAVELENGTHS THEN IDD'S ONLY
C NEED TO BE CALCULATED ONCE FOR EACH INCIDENT ANGLE.
C
C Set DCODE equal to zero so as not to use SVIDD
  DCODE-0
  IOLD-0
C
  IF(IFLAG1.GT.0.) THEN
C   IF(DCODE.NE.1.OR.IOLD.NE.1) THEN
   DO 90 IL-1,16
   IF(IL.EQ.1.OR.IL.EQ.2.OR.IL.EQ.4.OR.IL.EQ.5.OR.IL.EQ.6)THEN
    CALL IDDCMP(IDD(IL),AERR2,RERR2,IL,EREST)
   ELSE IF(IL.EQ.3.OR.IL.EQ.7) THEN
    IDD(IL)-IDD(2)
   ELSE IF(IL.EQ.8) THEN
    IDD(IL)-0.
   ELSE IF(IL.GE.8) THEN
    IDD(IL)-0.
   ENDIF
 90 CONTINUE
C If Er is constant then save IDD's for next wavelength
C   IF(DCODE.EQ.1) THEN
C    DO 93 IL-1,16
C    SVIDD(I4,IL)-IDD(IL)
C 93  CONTINUE
C   ENDIF
C  ELSE
C If Er is constant and the IDD's have been saved use them
C   DO 92 IL-1,16
C    IDD(IL)-SVIDD(I4,IL)
C 92  CONTINUE
C  ENDIF
  ENDIF
C
C
C THIS SECTION UTILIZED THE IDD VALUES TO COMPUTE THE MUELLER MTX
C ELEMENTS DIVIDED BY Q
```

```
C F(1)=F11/Q, F(2)=F12/Q .. F(5)=F21/Q .. F(16)=F44/Q
C MANY OF THESE F VALUES ARE ZERO BUT THEY ARE CALCULATED HERE FOR
C COMPLETENESS. THEIR CALCULATION TIME IS MINUTE COMPARED TO THE ACTUAL
C INTEGRATIONS.
  F(1)=.5*(IDD(1)+IDD(2)+IDD(3)+IDD(4))
  F(2)=.5*(IDD(1)+IDD(2)-IDD(3)-IDD(4))
  F(3)=IDD(9)
  F(4)=-IDD(10)
  F(5)=.5*(IDD(1)-IDD(2)+IDD(3)-IDD(4))
  F(6)=.5*(IDD(1)-IDD(2)-IDD(3)+IDD(4))
  F(7)=IDD(13)
  F(8)=-IDD(14)
  F(9)=2.*IDD(11)
  F(10)=2.*IDD(15)
  F(11)=IDD(5)+IDD(7)
  F(12)=-IDD(6)+IDD(8)
  F(13)=2.*IDD(12)
  F(14)=2.*IDD(16)
  F(15)=IDD(6)*IDD(8)
  F(16)=IDD(5)-IDD(7)
C
C The Muller matrix elements are computed in this section. Only the 6
C independent matrix elements are passed to the main program.a

C   write(8,2001)Q(RCODE)
C   write(8,2001)F(1),F(2),F(6),F(11),F(12),F(16)
  IF(RCODE.EQ.0)THEN
  FF(1)=1.
  FF(2)=F(2)/F(1)
  FF(3)=F(6)/F(1)
  FF(4)=F(11)/F(1)
  FF(5)=F(12)/F(1)
  FF(6)=F(16)/F(1)
  ELSE
  FF(1)=F(1)*Q(RCODE)
  FF(2)=F(2)*Q(RCODE)
  FF(3)=F(6)*Q(RCODE)
  FF(4)=F(11)*Q(RCODE)
  FF(5)=F(12)*Q(RCODE)
  FF(6)=F(16)*Q(RCODE)
  ENDIF
 2001 FORMAT(6E12.4)
C
  RETURN
  END
C
C
C
C THIS SUBROUTINE DRIVES QDAG  TO COMPUTE Q
  SUBROUTINE QCMP(Q,AERR,RERR,RCODE)
  REAL VX,VY,VXZ,VYSQH,UPLIM,QPRIME,SETARG,ARG,ARG1,
  1  UPLIMD,UPLIM1,SETUPR
  REAL HMSQ,CLEN,CLENSQ,WLEN,THETA,SNTHT,CSTHT,SIGS,K0,PI
  REAL GCNT1
```

```
       INTEGER RCODE
      EXTERNAL ARG
      COMMON/ONE/HMSQ,CLEN,CLENSQ,WLEN,THETA,SNTHT,CSTHT,SIGS,K0,PI
C FOR EACH VALUE OF THETA COMPUTE COMPONENTS OF VECTOR V.
C THEN COMBINE WITH THE MEAN SQUARE HEIGHT AND SAVE IN ARG
C  WRITE(8,*)'SNTHT,CSTHT',SNTHT,CSTHT
      VX--2.*K0*SNTHT
C  VZ-0.
      VY-2*K0*CSTHT
      VXZ-ABS(VX)
      VYSQH-VY*VY*HMSQ
      SETARG-ARG1(VXZ,VYSQH)
      UPLIMD-SETUPR(RCODE)
      SETARG-GCNT1(0.)
C COMPUTE THE UPPERLIMIT ON THE INTEGRATION BY ASSUMING THE INTEGRAND
C DIES AWAY WITHIN UPLIMD CORRELATION LENGTHS.
      UPLIM-UPLIMD*CLEN
      UPLIM1-UPLIM/100.
  100 CONTINUE
C COMPUTE THE DOUBLE INTEGRAL WHERE ONE HALF --> 2PI
C THIS CAN BE DONE BY SWITCHING TO POLAR COORDINATES
C IF NO INTERGRAND IS FOUND THEN REDUCE INTEGRATION LIMITS TO FIND IT
      CALL QDAG(ARG,0.,UPLIM,AERR ,RERR,1,QPRIME,ERREST)
      QPRIME-QPRIME*2*PI
      IF(QPRIME.lt.1.e-08.AND.UPLIM.GT.UPLIM1) THEN
       UPLIM-UPLIM*0.7
       GOTO 100
      ENDIF
C
C IF QPRIME > 0. THEN COMPUTE THE TOTAL Q
C IF QPRIME STILL EQUALS ZERO THEN INDICATE BY -998.
C IF QPRIME WAS LESS THAN ZERO INDICATE BY -999.
      IF (QPRIME.GT.0.)THEN
       Q-(K0*K0/PI)*QPRIME
      ELSE IF(QPRIME.EQ.0.)THEN
       Q--998.
      ELSE IF(QPRIME.LT.0.)THEN
       Q--999.
      ENDIF
C  WRITE(8,*)'Q-',Q,'  ACCESSED',GCNT1(0.)
      RETURN
      END
C
C
C
C THIS FUNCTION COMPUTES THE INTEGRAND OF Q FOR DCADRE.
C THIS FUNCTION HAS 3 ENTRY POINTS
C BSJ0 COMPUTES BESSEL FUNCTION J SUB 0
      FUNCTION ARG(RD)
      REAL BSJ0,RRSURF,CHI2,CHISQ,JSUB0,RD,VXZ,VYSQH
      REAL COUNT,DUMMY
      INTEGER RCODE,DRCODE
      SAVE
      COUNT-COUNT+1.
```

```
      JSUB0-BSJ0(VXZ*RD)
      CHI2-EXP((RRSURF(RD)-1.)*VYSQH)
      ARG-JSUB0*(CHI2-CHISQ)*RD
C     ARG-1.
      RETURN
C
C
C THIS SECOND ENTRY POINT SAVES SOME CONSTANTS FOR A GIVEN INTEGRATION
C THESE VALUES ARE ONLY FUNCTIONS OF THE SURFACE PARAMETERS NOT OF RD
      ENTRY ARG1(DVXZ,DVYSQH)
      VXZ-DVXZ
      VYSQH-DVYSQH
      CHISQ-EXP(-VYSQH)
      ARG1-1.0
      RETURN
C
C
C THIS ENTRY RETURNS THE NUMBER OF ACCESSES SINCE LAST INQUIRY
      ENTRY GCNT1(DUMMY)
      GCNT1-COUNT
      COUNT-DUMMY
      RETURN
      END
C
C
C THIS FUNCTION CALCULATES THE AUTOCORRELATION FUNCTION USED TO
C MODEL THE RANDOMLY ROUGH SURFACE
C BSK1 COMPUTES THE BESSEL FUNCTION K SUB 1
C BSK0 COMPUTES THE BESSEL FUNCTION K SUB 0
      FUNCTION RRSURF(RD)
      REAL RD,R,RSQ,R4TH,R6TH,KAPPA6,KAPPA8,BSK1,BSK0,TERM1,TERM2
      REAL HMSQ,CLEN,CLENSQ,WLEN,THETA,SNTHT,CSTHT,SIGS,K0,PI
      INTEGER RCODE,DRCODE
      COMMON/ONE/HMSQ,CLEN,CLENSQ,WLEN,THETA,SNTHT,CSTHT,SIGS,K0,PI
      IF(RD.EQ.0.) THEN
       RRSURF-1.
       RETURN
      ENDIF
      IF(RCODE.EQ.1) THEN
       RRSURF-EXP(-RD*RD/CLENSQ)
       RETURN
      ELSE IF(RCODE.EQ.2) THEN
       R-RD*KAPPA8
       RSQ-R*R
       R4TH-RSQ*RSQ
       R6TH-RSQ*R4TH
       TERM1-(1.-3.*RSQ/8.+3.*R4TH/32.+R6TH/3072.)*R*BSK1(R)
       TERM2-(RSQ/2.-R4TH/4.-R6TH/96.)*BSK0(R)
       RRSURF-TERM1+TERM2
       RETURN
      ELSE IF(RCODE.EQ.3) THEN
       R-RD*KAPPA6
       RSQ-R*R
       R4TH-RSQ*RSQ
```

```
        TERM1-(1.-3.*RSQ/4.-R4TH/96.)*R*BSK1(R)
       TERM2-(RSQ/2.+3.*R4TH/16.)*BSK0(R)
       RRSURF-TERM1+TERM2
       RETURN
      ELSE
       WRITE(8,*)'ILLEGAL RCODE'
       STOP
      ENDIF
      RRSURF-0.
      RETURN
C
C
      ENTRY SETUPR(DRCODE)
      RCODE-DRCODE
      SETUPR-1.
      IF(RCODE.EQ.1) THEN
       SETUPR- 5.
      ELSE IF(RCODE.EQ.2) THEN
       KAPPA8-SQRT(1.6)/CLEN
       SETUPR-175./SQRT(1.6)
      ELSE IF(RCODE.EQ.3) THEN
       KAPPA6-1./CLEN
       SETUPR-175.
      ENDIF
      RETURN
      END
C
C
C
C THIS SUBROUTINE DRIVES TWODQ TO COMPUTE IDD FOR DIJ*DKL IN INTEGRAND
C THE REAL PART IS COMPUTED FOR CODE- 1 OR 2, IMAGINARY CODE- 3
      SUBROUTINE IDDCMP(IDD,AERR,RERR,IL,EREST)
      REAL IDD,AERR,RERR,ERREST
      REAL HZMIN,HZMAX,HXMIN,HXMAX,HZMAX1
      REAL SDPQ,SP2,SPHXHZ,SZMIN,SZMAX
      REAL HMSQ,CLEN,CLENSQ,WLEN,THETA,SNTHT,CSTHT,SIGS,K0,PI
      REAL GCNT2
      INTEGER ICODE(16),ISUB(16),JSUB(16),KSUB(16),LSUB(16)
      INTEGER IL
      COMPLEX EREST,UR
      EXTERNAL ARGIDD,ZMIN,ZMAX
      COMMON/ONE/HMSQ,CLEN,CLENSQ,WLEN,THETA,SNTHT,CSTHT,SIGS,K0,PI
      DATA ICODE/1,1,1,1,2,3,2,3,2,3,2,3,2,3,2,3/
      DATA ISUB/1,1,2,2,1,1,1,1,1,1,1,1,2,2,1,1/
      DATA JSUB/1,2,1,2,1,1,2,2,1,1,[illegible],1,1,1,2,2/
      DATA KSUB/1,1,2,2,2,2,2,2,1,1,2,2,2,2,2,2/
      DATA LSUB/1,2,1,2,2,2,1,1,2,2,1,1,2,2,2,2/
C
      UR-(1.,0.)
      SETUP-SDPQ(ICODE(IL),ISUB(IL),JSUB(IL),KSUB(IL),
     1    LSUB(IL),EREST,UR,CSTHT,SNTHT)
      SETUP-SP2(SIGS,SNTHT,CSTHT,PI)
C     WRITE(8,*)'FACT- ',SETUP
      SETUP-SPHXHZ(PI,SIGS)
```

```
C  WRITE(8,*)'DENOM- ',SETUP
   SETUP-GCNT2(0.)
   HZMIN-0.
   HZMAX-5.*SQRT(SIGS)
   HZMAX1-HZMAX*.4
   HXMIN--HZMAX
   HXMAX-HZMAX
 199 SETUP-SZMIN(HZMIN)
   SETUP-SZMAX(HZMAX)
C
   CALL TWODQ(ARGIDD,HXMIN,HXMAX,ZMIN,ZMAX,AERR,RERR,1,IDD,ERREST)
   IDD-2.*IDD
C
C  WRITE(8,*)'IDD- ',IDD,' ACCESSED',GCNT2(0.)
   IF (IDD.EQ.0.0.AND.HZMAX.GT.HZMAX1)THEN
    HZMIN-HZMIN*.7
    HZMAX-HZMAX*.7
    HXMIN-HXMIN*.7
    HXMAX-HXMAX*.7
    GOTO 199
   ENDIF
C
   RETURN
   END
C
C
C
C THESE FUNCTIONS CALCULATE THE HZ LIMITS FOR 2-D INTEGRATION BY TWODQ
   FUNCTION ZMIN(X1)
   REAL D1ZMIN,D2ZMIN
   SAVE
   ZMIN-D1ZMIN
   RETURN
   ENTRY SZMIN(D2ZMIN)
   D1ZMIN-D2ZMIN
   SZMIN-1.0
   RETURN
   END
C
   FUNCTION ZMAX(X1)
   REAL D1ZMAX,D2ZMAX
   SAVE
   ZMAX-D1ZMAX
   RETURN
   ENTRY SZMAX(D2ZMAX)
   D1ZMAX-D2ZMAX
   SZMAX-1.0
   RETURN
   END
C
C
C
C THIS FUNCTION CALCULATES THE ARGUMENT OF IDD FOR TWODQ
   FUNCTION ARGIDD(HX,HZ)
```

```
      REAL COUNT,DUMMY,HXZSQ
      REAL DENOM,DSIGS,SIGS,SNTHT,CSTHT,PI,FACT
      REAL SIGSRT,PIC,C0,TANT,COTT,R,F1,F2
      REAL ERFC
      SAVE
C
      COUNT-COUNT+1.
      HXZSQ-HX*HX+HZ*HZ
C
      IF (HX.LT.-COTT) THEN
       F2-0.
      ELSE
       F2-FACT
      ENDIF
C
      PHXHZ-EXP(-HXZSQ/SIGS)/DENOM
C
      ARGIDD-PHXHZ*F2*DPQMAG(HX,HZ,HXZSQ)
      RETURN
C
C
      ENTRY SPHXHZ(PI,DSIGS)
      SIGS-DSIGS
      DENOM-PI*SIGS
      SPHXHZ-DENOM
      RETURN
C
C
      ENTRY SF2(DSIGS,SNTHT,CSTHT,PI)
      SIGSRT-SQRT(DSIGS)
      PIC-.5*SIGSRT/SQRT(PI)
      IF(ABS(SNTHT).LT.1.E-10)THEN
       COTT-1.E30
       C0-0.
      ELSE
       TANT-SNTHT/CSTHT
       COTT-1/TANT
       R-COTT/SIGSRT
       F1-EXP(-R*R)
       F2-ERFC(R)
       C0-PIC*TANT*F1-.5*F2
      ENDIF
      FACT-1./(1.+C0)
      SF2-FACT
      RETURN
C
C
      ENTRY GCNT2(DUMMY)
      GCNT2-COUNT
      COUNT-DUMMY
      RETURN
      END
C
C
```

```
C
C THIS FUNCTION CALCULATE DIJ*DKL AS PART OF IDD'S INTEGRAND
  FUNCTION DPQMAG(HX,HZ,HXZSQ)
  COMPLEX ER,UR,RIR,ETAR,ERM,ERMR,ERMRR,URM,URMR,URMRR
  COMPLEX SN1N,CS1N,DEN2,DEN3,C1IF,CFVV,CFHH,B1,B2,B3,B4
  COMPLEX DPQ(2,2),DER,DUR
  REAL HX,HZ,HXZSQ,SNTHT,CSTHT,DSNTHT,DCSTHT
  REAL CSG,TNG,SNG,CS0N,SN0N,SNPDI,CSPDI,S0IF,C0IF
  REAL CSSI,SNSII,SNSIF
  INTEGER ICODE,IICODE,ISUB,IISUB,JSUB,JJSUB,KSUB,KKSUB,LSUB,LLSUB
  SAVE
C
  CSG=1./SQRT(1.+HXZSQ)
  TNG=SQRT(HXZSQ)
  SNG=CSG*TNG
  IF(TNG.LT.1.E-5) THEN
   CS0N=CSTHT
   SN0N=SNTHT
  ELSE
   CSPDI=-HX/TNG
   SNPDI= HZ/TNG
   CS0N=CSG*CSTHT-SNG*SNTHT*CSPDI
   SN0N=SQRT(1.-CS0N*CS0N)
  ENDIF
C
  IF(ABS(CS0N).LT.1.E-5) THEN
   DPQMAG=0.
   RETURN
  ENDIF
C
  SN1N=SN0N/RIR
  CS1N=CSQRT(1.-SN1N*SN1N)
  DEN2=CS0N+CS1N*ETAR
  DEN3=CS0N+CS1N/ETAR
  S0IF=SN0N*SN0N
  C0IF=CS0N
  C1IF=CS1N*CS1N
  CFVV=C0IF*((-UR*C1IF-S0IF)*ERMR-URM)/(DEN2*DEN2)
  CFHH=C0IF*((-ER*C1IF-S0IF)*URMR-ERM)/(DEN3*DEN3)
C
  IF(TNG.LT.1.E-5) THEN
   DPQ(1,1)=CFVV
   DPQ(1,2)=0.
   DPQ(2,1)=0.
   DPQ(2,2)=CFHH
  ELSE
   CSSI=(CSC*SNTHT+SNG*CSTHT*CSPDI)/SN0N
   SNSII=-SNG*SNPDI/SN0N
   SNSIF=-SNSII
   B1=CFVV*CSSI
   B2=-CFHH*SNSII
   B3=CFVV*SNSII
   B4=CFHH*CSSI
   DPQ(1,1)=CSSI*B1-SNSIF*B2
```

```
      DPQ(1,2)-CSSP*B3-SNSIP*B4
      DPQ(2,1)-SNSIP*B1+CSSP*B2
      DPQ(2,2)-SNSIP*B3+CSSP*B4
     ENDIF
C
     IF(ICODE.EQ.1) THEN
      DPQMAG-(CABS(DPQ(ISUB,JSUB))/CSG)**2
     ELSE IF(ICODE.EQ.2) THEN
      DPQMAG-REAL(DPQ(ISUB,JSUB)*CONJG(DPQ(KSUB,LSUB)))/CSG/CSG
     ELSE
      DPQMAG-AIMAG(DPQ(ISUB,JSUB)*CONJG(DPQ(KSUB,LSUB)))/CSG/CSG
     ENDIF
     RETURN
C
C
     ENTRY SDPQ(IICODE,IISUB,JJSUB,KKSUB,LLSUB,DER,DUR,
     1    DCSTHT,DSNTHT)
     ICODE-IICODE
     ISUB-IISUB
     JSUB-JJSUB
     KSUB-KKSUB
     LSUB-LLSUB
     CSTHT-DCSTHT
     SNTHT-DSNTHT
     ER-DER
     UR-DUR
     RIR-CSQRT(ER*UR)
     ETAR-CSQRT(UR/ER)
     ERM-1.-ER
     ERMR-1.-1./ER
     ERMRR-ERMR*RIR
     URM-1.-UR
     URMR-1.-1./UR
     URMRR-URMR*RIR
     SDPQ-1.0
     RETURN
     END
C******END OF RETRO*******
C
C
C *****SUBROUTINE T2STAT*****
C This subroutine computes the basic statistics used in the analyses.
C For more information see the user manual Equations (1)-(5)
C
     SUBROUTINE T2STAT(DIFF,RATLOG,LMAX,DIFFMAX,RADMAX,
     *LRMAX,RADIUS,RLOG,PFT,PFB)
     DIMENSION DIFF(6),RATLOG(6),PFT(6),PFB(6)
     RAD-0.
     RADMAX-0.
     RADRAT-0.
     DIFFMAX-0.
     DO 330 L-1,6
     IF(ABS(PFT(L)).GT.ABS(PFB(L)))THEN
     DIFF(L)-PFT(L)-PFB(L)
```

```
          IF(FFT(L).NE.0..AND.FFB(L).NE.0.)THEN
   RATLOG(L)-DIFF(L)/SQRT(ABS(FFB(L)*FFT(L)))
   ELSE
   RATLOG(L)-0.
   ENDIF
  ELSE
  DIFF(L)-0.
  RATLOG(L)-0.
  ENDIF
  RAD-RAD+DIFF(L)*DIFF(L)
  RADRAT-RADRAT+RATLOG(L)*RATLOG(L)
  IF(ABS(RATLOG(L)).GT.ABS(RADMAX))THEN
  RADMAX-RATLOG(L)
  LRMAX-L
  ENDIF
  IF(ABS(DIFF(L)).GT.ABS(DIFFMAX))THEN
  DIFFMAX-DIFF(L)
  LMAX-L
  ENDIF
 330 CONTINUE
  RADIUS-SQRT(RAD)
  RLOG-SQRT(RADRAT)
C  WRITE(8,8800)RADMAX
 8800 FORMAT('RADMAX',E12.4)
  RETURN
  END
C
C******SUBROUTINE SORTNK******
C
C The routine sorts through the
C array containing the imaginary part of the index and finds all maxima and
C minima. If there is an interval over which there are local extrema and
C several points share the extreme value, then the program will store the
C extreme value at the two endpoints of the interval, and it is implicitly
C understood that k is constant over that interval. The array containing
C the resonant values (maxima), ANKMAX is rearranged in the last section
C of this routine so that its entries are in order of decreasing k rather
C than in order of increasing wavelength.
C
  SUBROUTINE SORTNK(NMIN,NMAX,WLNMX,WLNMIN,ANKMIN,ANKMAX,NPTS,
  *NK,WLN)
  COMMON/TWO/WLMIN,WLMAX
  REAL WLNMIN(40),WLNMX(40),ANKMIN(40),ANKMAX(40),WLN(500),NK(500)
  INTEGER NMIN,NMAX
C To avoid looking at wavelengths outside of the range defined by WLMAX
C and WLMIN, find out which wavelengths in WLN are outside of the range
C and exclude their corresponding entries in NK.
C  PRINT *,WLMIN,WLMAX
  WLNMX-0.
  ANKMAX-0.
  NLESS-1
  NMORE-1
  DO 777 JL-1,NPTS
  IF(WLN(JL).LT.WLMIN)NLESS-NLESS+1
```

```
        IF(WLN(JL).GT.WLMAX)NMORE=NMORE+1
  777 CONTINUE
C     WRITE(8,8222)NLESS,NMORE
 8222 FORMAT('NLESS, NMORE',2I5)
      NMIN=0
      NMAX=0
      IF(NK(NLESS).GT.NK(NLESS+1))THEN
      NMAX=NMAX+1
      WLNMX(NMAX)=WLN(NLESS)
      ANKMAX(NMAX)=NK(NLESS)
      ELSE
      NMIN=NMIN+1
      WLNMIN(NMIN)=WLN(NLESS)
      ANKMIN(NMIN)=NK(NLESS)
      ENDIF
      M=NLESS+1
C     WRITE(8,*)NMORE,NLESS
   99 DO 301 J=M,NPTS-NMORE
      IF(NK(J).GT.NK(J-1).AND.NK(J).GT.NK(J+1))THEN
      NMAX=NMAX+1
      WLNMX(NMAX)=WLN(J)
      ANKMAX(NMAX)=NK(J)
      ELSE IF(NK(J).LT.NK(J-1).AND.NK(J).LT.NK(J+1))THEN
      NMIN=NMIN+1
      WLNMIN(NMIN)=WLN(J)
      ANKMIN(NMIN)=NK(J)
C
C Careful about drawing a false conclusion if two neighboring entries in
C the data file are equal.
C
      ELSE IF(NK(J).GT.NK(J-1).AND.NK(J).EQ.NK(J+1))THEN
      NMAX=NMAX+1
      WLNMX(NMAX)=WLN(J)
      ANKMAX(NMAX)=NK(J)
       JJ=J
       MM=1
       DO 302 MMM=J+1,NPTS-1
       MM=MM+1
       IF(NK(MMM).GT.NK(MMM+1))THEN
       NMAX=NMAX+1
       WLNMX(NMAX)=WLN(MMM)
       ANKMAX(NMAX)=NK(MMM)
       GO TO 303
       ENDIF
       IF(NK(MMM).LT.NK(MMM+1))NMAX=NMAX-1
       IF(NK(MMM).LT.NK(MMM+1))GO TO 303
  302  CONTINUE
      ELSE IF(NK(J).LT.NK(J-1).AND.NK(J).EQ.NK(J+1))THEN
      NMIN=NMIN+1
      WLNMIN(NMIN)=WLN(J)
      ANKMIN(NMIN)=NK(J)
       JJ=J
       MM=1
       DO 305 MMM=J+1,NPTS-1
```

```
      MM-MM+1
     IF(NK(MMM).LT.NK(MMM+1))THEN
     NMIN-NMIN+1
     WLNMIN(NMIN)-WLN(MMM)
     ANKMIN(NMIN)-NK(MMM)
     GO TO 303
     ENDIF
     IF(NK(MMM).GT.NK(MMM+1))NMIN-NMIN-1
     IF(NK(MMM).GT.NK(MMM+1))GO TO 303
 305  CONTINUE
   ENDIF
 301 CONTINUE
   IF(NK(NPTS-NMORE+1).GT.NK(NPTS-NMORE))THEN
   NMAX-NMAX+1
   WLNMX(NMAX)-WLN(NPTS-NMORE+1)
   ANKMAX(NMAX)-NK(NPTS-NMORE+1)
   ELSE
   NMIN-NMIN+1
   WLNMIN(NMIN)-WLN(NPTS-NMORE+1)
   ANKMIN(NMIN)-NK(NPTS-NMORE+1)
   ENDIF
   GO TO 304
 303 M-JJ+MM
   GO TO 99
C
C Rearrange the elements in WLNMAX so that they are in decreasing order
C rather than in order of increasing wavelength. If a case occurs where
C two elements of WLNMAX are equal, handle it by setting the one with the
C largest subscript equal to zero.
C
 304 DO 321 I-1,NMAX-1
   LAST-NMAX-I
   DO 321 K-1,LAST
   IF(ANKMAX(K).EQ.ANKMAX(K+1))ANKMAX(K+1)-0.
   IF(ANKMAX(K).LT.ANKMAX(K+1))THEN
   TEMP-ANKMAX(K)
   TEMPWL-WLNMX(K)
   ANKMAX(K)-ANKMAX(K+1)
   WLNMX(K)-WLNMX(K+1)
   ANKMAX(K+1)-TEMP
   WLNMX(K+1)-TEMPWL
   ENDIF
 321 CONTINUE
   RETURN
   END
C
C****SUBROUTINE SAVE****
C This routine determines whether or not the data computed by RETRO is more
C useful from the standpoint of the test specified by NCRIT than the currently
C most useful set of data. Since discrepancies arise between the 2 tests,
C performed by T2STAT, the variable NCRIT determines which test is used in
C decision making.
C
   SUBROUTINE SAVE(NCRIT,T2STMX,RADMX,I3,I4,I3MX,I4MX,I3RDMX,I4RDMX
```

```
     *,JWLSET,RADIUS,RLOG)
C    WRITE(8,888)I3,I4,RADIUS,RLOG
888 FORMAT('FROM SAVE',2I5,2E12.4)
     JWLSET-1
     IF(RADIUS.GE.T2STMX)THEN
     T2STMX-RADIUS
     I3MX-I3
     I4MX-I4
     IF(NCRIT.EQ.1)JWLSET-2
     ENDIF
     IF(RLOG.GE.RADMX)THEN
     RADMX-RLOG
     I3RDMX-I3
     I4RDMX-I4
     IF(NCRIT.EQ.0)JWLSET-2
     ENDIF
     RETURN
     END
C
C*****SUBROUTINE WLANGSET*****
C
C This routine checks if Muller matrices have been computed for the point
C at wavelength W and incident angle ANG(K+I4SET). If Muller matrices have
C already been calculated for this point, then the program will not recompute
C them (NCALC-2). If the program has yet to compute Muller matrices for this
C wavelength and incident angle, it will (NCALC-1).
C
     SUBROUTINE WLANGSET(W,MWLN,WLNWLN,WLN,J,K,I3SET,I4SET,NCALC,
     *I3,I4)
     REAL WLNWLN(100),WLN(100,100)
     NCALC-2
     KKK-I4SET+K
C If J-0 then some incident angles have already been computed for
C this wavelength. It is hence necessary to determine if ANG(K+I4SET)
C has been used at this wavelength. The wavelengths and incident angles
C at which Muller matrices for those wavelengths have been calculated
C are stored in WLN so if WLN(.,.) is zero then that Muller matrix has
C not yet been computed.
     IF(J.EQ.0)THEN
          IF(WLN(I3SET,KKK).LT.1.E-05)THEN
      WLN(I3SET,KKK)-W
      I3-I3SET
      I4-KKK
      NCALC-1
      RETURN
      ELSE
      RETURN
      ENDIF
     ELSE
C If J.NE.0 then there is a chance that this wavelength has not been used
C at all in which case there is no chance that the same Muller matrix will
C be computed twice and any incident angle is o.k.
     DO 99 L-1,MWLN
     X-W-WLNWLN(L)
```

```
        IF(ABS(X).LT.1.E-05)THEN
        IF(WLN(L,KKK).LT.1.E-05)THEN
        WLN(L,KKK)-W
        I3-L
        I4-KKK
        NCALC-1
        RETURN
        ELSE
        RETURN
        ENDIF
       ENDIF
   99 CONTINUE
      ENDIF
C If this line has been reached then the wavelength W has yet to be
C used.
      MWLN-MWLN+1
      WLNWLN(MWLN)-W
      WLN(MWLN,KKK)-W
      I3-MWLN
      I4-KKK
      NCALC-1
      RETURN
      END
C
C******SUBROUTINE MTXELE*******
C
C This routine sorts the arrays containing the terms in the discrimination
C tests of T2STAT in order of largest to smallest. The purpose of this is
C to tell you which matrix elements are most useful at a particular
C angle of incidence and wavelength.
C
      SUBROUTINE AMTXELE(X,Y,K,N)
      INTEGER K(6),KK(6)
      REAL X(6),Y(6)
      DATA KK/1,2,3,4,5,6/
      Y-X
      K-KK
C     WRITE(8,8888)Y
C     WRITE(8,8888)X
 8888 FORMAT(6E12.4)
      DO 100 M-1,5
      DO 100 L-1,6-M
      IF(ABS(Y(L)).LE.ABS(Y(L+1)))THEN
      T-Y(L)
      Y(L)-Y(L+1)
      Y(L+1)-T
      JT-K(L)
      K(L)-K(L+1)
      K(L+1)-JT
      ENDIF
  100 CONTINUE
      N-0
      DO 101 J-1,6
      IF(Y(J).NE.0)N-N+1
```

```
      101 CONTINUE
      RETURN
      END
C
C******SUBROUTINE SETHETWL*******
C
C This routine determines which angles of incidence and wavelengths
C the program uses to compute and analyze Mueller matrices in an
C orderly fashion so they can be written in a form that
C DISSPLAY can read and plot them.
C
      SUBROUTINE SETHETWL(ANG,WLNWLN,I3SET,I4SET)
      DIMENSION ANG(100),WLNWLN(100)
      COMMON/TWO/WLMIN,WLMAX
      A=ANG(I4SET)
      IF(A.LT.20.)THEN
      DO 666 I=1,11
      ANG(I)=4.*(I-1)
  666 CONTINUE
      ELSE IF(A.GT.68.)THEN
      DO 667 I=1,11
      ANG(I)=48.+4.*(I-1)
  667 CONTINUE
      ELSE
      DO 671 I=-5,5
      ANG(I)=A+4.*I
  671 CONTINUE
      ENDIF
      W=WLNWLN(I3SET)
      IF(W-.25.LE.WLMIN)THEN
      X=W-WLMIN
      M=IFIX(X/.05)
      DO 668 I=-M,-M+10
      WLNWLN(I+M+1)=W+I*.05
  668 CONTINUE
      ELSE IF((W+.25).GT.WLMAX)THEN
      X=WLMAX-W
      M=IFIX(X/.05)
      DO 669 I=-M,-M+10
      WLNWLN(12-I-M-1)=W-.05*I
  669 CONTINUE
      ELSE
      DO 670 I=-5,5
      WLNWLN(I+6)=W+.05*I
  670 CONTINUE
      ENDIF
      RETURN
      END
```

AVI.2.6 SAMPLE CALCULATIONS: INPUT AND OUTPUT DATA OUTPUT FILES.

AVI.2.6.1 Input File "DATAIN2."

```
compos.nk
sf96.nk
decide2.out
decide2.comm2
compos
sf96
20. .5
20. .5
11 0. 8.
9.0 12.5 .05
1
1
.00000001 .0000001
.0001 .005
700
0
dic.d
cdi.d
acdi.d
```

AVI.2.6.2 Output File "dic.d."

```
sf96
     0001
     0001
     0011
     0011
  0.2000E+02
  0.5000E+00
  0.1195E+02  0.1200E+02  0.1205E+02  0.1210E+02  0.1215E+02
  0.1220E+02  0.1225E+02  0.1230E+02  0.1235E+02  0.1240E+02
  0.1245E+02
  0.0000E+00  0.4000E+01  0.8000E+01  0.1200E+02  0.1600E+02
  0.2000E+02  0.2400E+02  0.2800E+02  0.3200E+02  0.3600E+02
  0.4000E+02
  0.1000E+01  0.1000E+01  0.1000E+01
  0.5562E-01 -0.2157E-13  0.5521E-01 -0.5480E-01  0.1661E-11 -0.5521E-01
  0.1000E+01  0.1000E+01  0.1000E+01
  0.5552E-01  0.8135E-04  0.5510E-01 -0.5467E-01 -0.6637E-04 -0.5510E-01
  0.1000E+01  0.1000E+01  0.1000E+01
  0.5519E-01  0.3183E-03  0.5473E-01 -0.5427E-01 -0.2597E-03 -0.5473E-01
  0.1000E+01  0.1000E+01  0.1000E+01
  0.5451E-01  0.6884E-03  0.5399E-01 -0.5348E-01 -0.5617E-03 -0.5399E-01
  0.1000E+01  0.1000E+01  0.1000E+01
.
.
.
.
```

```
.
.
.
.

249  0.5398E+00 0.6587E-01 0.4800E+00 -0.4222E+00 -0.3128E-01 -0.4820E+00
250  0.1000E+01 0.1000E+01 0.1000E+01
251  0.4630E+00 0.6710E-01 0.4059E+00 -0.3522E+00 -0.3167E-01 -0.4092E+00
252  0.1000E+01 0.1000E+01 0.1000E+01
253  0.3753E+00 0.6222E-01 0.3245E+00 -0.2781E+00 -0.2926E-01 -0.3289E+00
254  0.1000E+01 0.1000E+01 0.1000E+01
255  0.2836E+00 0.5238E-01 0.2418E+00 -0.2051E+00 -0.2460E-01 -0.2468E+00
```

AVI.2.6.3 Output File "cdi.d."

```
 1  compos
 2     0001
 3     0001
 4     0011
 5     0011
 6  0.2000E+02
 7  0.5000E+00
 8  0.1195E+02 0.1200E+02 0.1205E+02 0.1210E+02 0.1215E+02
 9  0.1220E+02 0.1225E+02 0.1230E+02 0.1235E+02 0.1240E+02
10  0.1245E+02
11  0.0000E+00 0.4000E+01 0.8000E+01 0.1200E+02 0.1600E+02
12  0.2000E+02 0.2400E+02 0.2800E+02 0.3200E+02 0.3600E+02
13  0.4000E+02
14  0.1000E+01 0.1000E+01 0.1000E+01
15  0.2454E+00 -0.1181E-11 0.2394E+00 -0.2335E+00 -0.2653E-10 -0.2394E+00
16  0.1000E+01 0.1000E+01 0.1000E+01
17  0.2447E+00 0.7180E-03 0.2386E+00 -0.2325E+00 -0.7195E-04 -0.2386E+00
18  0.1000E+01 0.1000E+01 0.1000E+01
19  0.2426E+00 0.2798E-02 0.2360E+00 -0.2295E+00 -0.2800E-03 -0.2360E+00
20  0.1000E+01 0.1000E+01 0.1000E+01
21  0.2385E+00 0.6014E-02 0.2313E+00 -0.2241E+00 -0.6000E-03 -0.2313E+00
22  0.1000E+01 0.1000E+01 0.1000E+01
.
.
.
.
.

249  0.1381E+00 0.1315E-01 0.1323E+00 -0.1265E+00 -0.3216E-02 -0.1323E+00
250  0.1000E+01 0.1000E+01 0.1000E+01
251  0.1190E+00 0.1346E-01 0.1133E+00 -0.1078E+00 -0.3279E-02 -0.1135E+00
252  0.1000E+01 0.1000E+01 0.1000E+01
253  0.9686E-01 0.1250E-01 0.9171E-01 -0.8690E-01 -0.3039E-02 -0.9204E-01
254  0.1000E+01 0.1000E+01 0.1000E+01
255  0.7345E-01 0.1052E-01 0.6916E-01 -0.6530E-01 -0.2552E-02 -0.6958E-01
```

AVI.2.6.4 Output File "acdi.d."

```
1   compos
2      0001
3      0001
4      0011
5      0011
6   0.2000E+02
7   0.5000E+00
8   0.1195E+02 0.1200E+02 0.1205E+02 0.1210E+02 0.1215E+02
9   0.1220E+02 0.1225E+02 0.1230E+02 0.1235E+02 0.1240E+02
10  0.1245E+02
11  0.0000E+00 0.4000E+01 0.8000E+01 0.1200E+02 0.1600E+02
12  0.2000E+02 0.2400E+02 0.2800E+02 0.3200E+02 0.3600E+02
13  0.4000E+02
14  0.1000E+01 0.1000E+01 0.1000E+01
15  0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00
16  0.1000E+01 0.1000E+01 0.1000E+01
17  0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00
18  0.1000E+01 0.1000E+01 0.1000E+01
19  0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00
20  0.1000E+01 0.1000E+01 0.1000E+01
21  0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00 0.0000E+00
22  0.1000E+01 0.1000E+01 0.1000E+01
.
.
.
.
.
249 0.7040E+00 0.4320E+01 0.4018E+00 -0.2798E+01 0.4018E+00 0.1472E+01
250 0.1000E+01 0.1000E+01 0.1000E+01
251 0.5957E+00 0.4291E+01 0.3440E+00 -0.2786E+01 0.3440E+00 0.1466E+01
252 0.1000E+01 0.1000E+01 0.1000E+01
253 0.4771E+00 0.4270E+01 0.2785E+00 -0.2781E+01 0.2785E+00 0.1461E+01
254 0.1000E+01 0.1000E+01 0.1000E+01
255 0.3566E+00 0.4256E+01 0.2101E+00 -0.2782E+01 0.2101E+00 0.1456E+01
```

AVI.2.6.5 Output File "decide2.comm2."

```
PROGRAM STOPPED BECAUSE NO IMPROVEMENT IN THE
RESULTS OF ROUTINE T2STAT HAS BEEN DETECTED
```

AVI.2.6.6 Output File "decide2.out."

```
1   compos      sf96
2  20.0, 0.5
3  20.0, 0.5
4   0.9000E+01 0.1250E+02 0.5000E-01
5   0.0000E+00 0.8000E+01 0.1600E+02 0.2400E+02 0.3200E+02
6   0.4000E+02 0.4800E+02 0.5600E+02 0.6400E+02 0.7200E+02
7   0.8000E+02
8  CODE FOR AUTO-CORR FUNCTIONS AND NORMALIZATION   1   1
```

```
    NCRIT  0
WAVELENGTH=  0.961539E+01 INCIDENT ANGLE=  0.000000E+00
 5
 5
 0.5633E-09 0.1773E+02    0005 -0.5633E-09    0005 -0.1773E+02
.
.
.
.
.
WAVELENGTH=  0.124533E+02 INCIDENT ANGLE=  0.640000E+02
 5 2 1 6 3 4
 1 6 3 4 2 5
 0.4224E-02 0.4238E+01    0001  0.2582E-02    0005 -0.2845E+01
WAVELENGTH=  0.124533E+02 INCIDENT ANGLE=  0.720000E+02
 5 2 1 6 3 4
 1 6 3 4 2 5
 0.1444E-03 0.4220E+01    0001  0.8904E-04    0005 -0.2852E+01
WAVELENGTH=  0.124533E+02 INCIDENT ANGLE=  0.800000E+02
 5 2 1 6 3 4
 1 6 3 4 2 5
 0.6171E-06 0.4182E+01    0001  0.3839E-06    0005 -0.2841E+01
WAVELENGTH=  0.956539E+01 INCIDENT ANGLE=  0.000000E+00
 5
 5
 0.1975E-09 0.1138E+02    0005 -0.1975E-09    0005 -0.1138E+02
WAVELENGTH=  0.956539E+01 INCIDENT ANGLE=  0.800000E+01
 0.0000E+00 0.0000E+00    0005  0.0000E+00    0005  0.0000E+00
WAVELENGTH=  0.966539E+01 INCIDENT ANGLE=  0.000000E+00
 5
 5
 0.6568E-09 0.1835E+02    0005 -0.6568E-09    0005 -0.1835E+02
WAVELENGTH=  0.966539E+01 INCIDENT ANGLE=  0.800000E+01
 0.0000E+00 0.0000E+00    0005  0.0000E+00    0005  0.0000E+00
WAVELENGTH=  0.971539E+01 INCIDENT ANGLE=  0.000000E+00
 5
 5
 0.8643E-09 0.1759E+02    0005 -0.8643E-09    0005 -0.1759E+02
WAVELENGTH=  0.971539E+01 INCIDENT ANGLE=  0.800000E+01
 0.0000E+00 0.0000E+00    0005  0.0000E+00    0005  0.0000E+00
THE MOST PROMISING WAVELENGTH AND ANGLE PAIR IS 0.9665E+01 0.0000E+00
 0.956539E+01 0.000000E+00
 0.2823E+00 -0.2309E-11 0.2741E+00 -0.2658E+00 -0.1959E-09 -0.2741E+00
 0.1944E+01 0.1177E-08 0.1743E+01 -0.1543E+01 0.1537E-11 -0.1743E+01
.
.
.
```

.

.

270	0.1266E+00	0.2852E-01	0.1050E+00	-0.8727E-01	-0.1246E-01	-0.1089E+00
271	0.3218E-01	0.5420E-02	0.2998E-01	-0.2813E-01	-0.1361E-02	-0.3034E-01
272	0.124688E+02	0.560000E+02				
273	0.3198E-01	0.8327E-02	0.2593E-01	-0.2120E-01	-0.3660E-02	-0.2725E-01
274	0.8155E-02	0.1569E-02	0.7530E-02	-0.7033E-02	-0.3944E-03	-0.7659E-02
275	0.124688E+02	0.640000E+02				
276	0.3662E-02	0.1095E-02	0.2911E-02	-0.2333E-02	-0.4840E-03	-0.3083E-02
277	0.9366E-03	0.2051E-03	0.8583E-03	-0.7973E-03	-0.5162E-04	-0.8757E-03
278	0.124688E+02	0.720000E+02				
279	0.1272E-03	0.4365E-04	0.9951E-04	-0.7751E-04	-0.1933E-04	-0.1052E-03
280	0.3265E-04	0.8156E-05	0.2973E-04	-0.2740E-04	-0.2055E-05	-0.3033E-04
281	0.124688E+02	0.800000E+02				
282	0.5544E-06	0.2175E-06	0.4282E-06	-0.3204E-06	-0.9612E-07	-0.4466E-06
283	0.1429E-06	0.4078E-07	0.1294E-06	-0.1180E-06	-0.1027E-07	-0.1315E-06