Computational and Instrumental Developments in Quantitative Auger Electron Analysis

By: ***Andrew Robert Jackson***

Submitted for the Degree of
Doctor of Philosophy

Department of Electronics
The University of York
Heslington
YORK
YO1 5DD

Dedicated to my grandfather, Mr. E. F. Thurston BSc CChem FRSC whose enthusiasm for life, and love of science has given me great inspiration and encouragement since my earliest days.

ABSTRACT

The technique of quantitative Auger electron spectroscopy (AES) is central to modern surface analysis. Development, both in terms of new instrumental apparatus and the theoretical basis of Auger analysis, has been the subject of intense research. The work presented here investigates two of the current issues surrounding Auger electron spectroscopy and microscopy.

The modelling of electron-solid interaction is reviewed, and investigations are carried out into the two well established computational techniques, transport theory and Monte Carlo simulation. The transport mean free path, λ_{tr} and the inelastic mean free path, λ_i describe electron transport in solids to the first order. Variations in these parameters with energy and atomic number are explored with a view to identifying trends and establishing the extent to which generalisations are valid.

Although transport theory calculations have been shown to give an accurate representation of true electron behaviour, their application is largely limited to homogeneous materials. Monte Carlo modeling provides us with a more rigorous treatment of complex experimental conditions. A new Monte Carlo model is presented which allows extension of existing simulations to incorporate heterogeneous multi-layered samples.

The design of integrated circuits is an extremely fast moving technology, with routine manufacture of nanometric feature sizes now becoming a reality. The second part of this work is devoted to the design of an angle resolved electron spectrometer with a very high resolution field emission electron probe. It is intended that high resolution analysis, coupled with the ability to resolve the azimuthal component of electron trajectories, will offer new insight into the surface features of ultra large scale integrated circuits.

TABLE OF CONTENTS

CHAPTER THREE

SYSTEMATIC TRENDS IN THE TRANSPORT MEAN FREE PATH WITH ENERGY AND ATOMIC NUMBER 38

CHAPTER FOUR

MONTE CARLO SIMULATION OF THE DEPTH DISTRIBUTION FUNCTION IN MULTILAYERED STRUCTURES 60

CHAPTER FIVE

EXPERIMENTAL APPARATUS 96

CHAPTER SIX

THE ANGLE-RESOLVED DETECTOR HEAD 126

CHAPTER EIGHT

CHAPTER NINE

LIST OF FIGURES

ACKNOWLEDGEMENTS

Firstly, I would like to thank my supervisors, Prof. M. M. El-Gomati and Prof. J. A. D. Matthew, who have been instrumental to my work, both in terms of their scientific input and continual encouragement. Prof. El-Gomati first introduced me to surface analysis, and his broad knowledge of the subject has been a source of reference and direction to me throughout my research at York.

This work would not have been possible without the financial help of the W. W. Smith Bequest and the University of York, and for this I am very grateful.

The Electronics and Physics departments at York are fortunate to be supported by excellent engineering facilities. I would particularly like to thank I. Will and J. Cramer of the electronics workshop for their expert engineering and maintenance skills. Many thanks are also due to B. Wilkinson and R. Easton of the precision engineering workshop for their work on the fabrication of the electron gun and multi-channel detector head described in this thesis. Also, I would like to thank P. Durkin of the graduate workshop, and M. Law for his assistance in the clean room.

The working atmosphere in the electron devices research group is the best I have encountered. Thanks go to S. Johnson, T. Wells, Dr. P. Tenney, Dr. T. El-Bakush and all those who took part in the work there, for making my time at York so enjoyable. I would especially like to acknowledge the work of A. Gelsthorpe, who has taken on the task of continuing my research and provided me with useful information and data since I left the group.

The close working relationship between the Electronics and Physics departments at York has been a great benefit to me. I would like to thank Prof. M. Prutton, Dr. R Roberts, Dr. I. Berkshire, Dr. M. Wenham and all my colleagues in the department of Physics for many useful discussions.

I would like to thank my family for their immeasurable emotional and material support over the years, and in particular my wife, Alison, for her immense support in recent months as I have been working to complete my thesis whilst working full time.

DECLARATION AND PUBLICATIONS

The work presented in this thesis is that of the author, except where the contributions of others have been acknowledged explicitly in the main text, or by means of references.

A list of publications arising from the material presented here is given below.

A. R. Jackson, M. M. El-Gomati, J. A. D. Matthew, and P. J. Cumpson (1997), *Surf. Interface Anal.*,**25**, 341

J. A. D. Matthew, A. R. Jackson and M. M. El-Gomati (1997), *J. Electron Spectrosc. Relat. Phenom.*, **85**, 205

CHAPTER ONE

INTRODUCTION

The proliferation of the desktop computer into almost every working environment over the past ten years illustrates the increasing importance of high density electronic devices in everyday life. Since 1990, the minimum feature size commonly achieved in mass-production devices has reduced from the order of one micron to hundreds of nanometres, and this trend is set to continue towards one nanometre feature sizes within the next 30 years. Five years ago, high performance processors typically contained in the region of three million transistors; the latest Pentium II processors contain more than five million transistors with a minimum feature size of 250 nanometres, less than half that of their predecessors. Naturally, these improvements in fabrication techniques have a direct impact on the performance of available technology, evident in the dramatic increase in the computing power available at a given price.

The study of surfaces has been instrumental in the accelerated growth of the computing industry, through an improved understanding of influential surface-specific phenomena leading to more efficient and technologically advanced device production techniques. In general, surface science is the study of the surface structure, elemental composition and distribution in the top few atomic layers of a material.

Many techniques exist for the examination of surface properties, all of which are based on the observation of particles or radiation emitted from a surface when irradiated with an energy source of electrons or energetic ions. Precisely which of the available techniques is chosen, depends on the information required by the analyst. For example, low energy electron diffraction (LEED) gives structural information about atomic arrangements at surfaces (Higatsberger, 1981), whilst secondary ion mass spectrometry (SIMS) has been shown to be a very sensitive technique for determining chemical composition (William, 1979). Surface sensitivity is an important consideration when choosing an analytical technique. Of the currently available techniques, X-ray photoelectron spectroscopy (XPS) and Auger electron

spectroscopy (AES) are the most surface sensitive, with information depths restricted to the top few monolayers.

The Auger effect takes its name from Pierre Auger who first noticed the phenomenon in 1923 whilst performing cloud chamber experiments (Auger 1925). Essentially, the energy analysis of Auger electrons allows identification of the chemical composition of a material. After this discovery, it was more than forty years before the Auger effect was to become widely exploited as an experimental technique. The availability of ultra-high-vacuum (UHV) apparatus in the 1960s saw the start of widespread experimental research into the Auger effect, and concurrent extensive theoretical work has also been carried out to gain further understanding of the technique.

Development of both theoretical and experimental aspects of AES is the subject of this thesis. The work is split into two distinct sections. The first describes computational developments into quantitative analysis, based on transport theory calculations and Monte Carlo simulation. The second section details the design, development and testing of a novel angle-resolved electron energy analyser. The analytical instrument is supported by a comprehensive software package, which allows integration of the Auger spectrometer with a computer controlled secondary electron microscope (SEM) based on a compact secondary electron detector designed by the author.

Chapter two gives a general introduction to the science of Auger electron spectroscopy. After an outline of the physical processes involved in Auger emission, the experimental apparatus required for AES is examined, together with a review of the most commonly used electron energy analysers. Attention is then turned to the theoretical background to AES. In recent years, better understanding of the processes leading to a measured Auger signal has made accurate quantitative Auger electron analysis possible. Monte Carlo simulation and transport theory have become accepted techniques for modelling electron-solid interaction, and have been shown to be valuable tools in quantitative analysis. These techniques agree well in the case of heterogeneous sample geometries where the so-called "no memory" criterion is met.

For more complex multi-layered heterogeneous structures, transport theory approximations become invalid and more complex Monte Carlo approaches must be adopted.

The inelastic mean free path, λ_i and the transport mean free path, λ_{tr} are important parameters which form the basis of quantitative analysis. The physics of electron transport is complex and strongly influenced by factors such as atomic number and electron energy; as a result, the mean free paths show corresponding variations according to experimental conditions. Chapter three describes an investigation of systematic trends in the transport mean free path which emerge, over a range of atomic numbers and energies of particular interest for AES and X-ray photoelectron spectroscopy (XPS) studies. An attempt is made to explain the origins of observed trends with a view to establishing the extent to which valid generalisations and assumptions can be made.

Quantitative analysis by Auger electron spectroscopy or X-ray photoelectron spectroscopy depends on the accurate determination of the depth scale for electron emission. The construction of a detailed analytical model describing electron-solid interaction is almost impossible due to the enormous number of possible direction changes and energy losses an electron can encounter between its emission and detection. Monte-Carlo simulations allow us to more accurately model such effects by using an iterative process, randomly selecting between possible scattering events from a range of equally likely possibilities (Joy, 1988). Monte Carlo simulations have been the subject of a substantial amount of research since their introduction by Von Neumann during the Manhattan project in the mid 1940s. Chapter four describes the development of a rapid Monte Carlo simulation aimed at determining the depth distribution function (DDF), based on the trajectory reversal approach (Werner, 1991; Cumpson, 1993). The algorithms presented here represent an extension to previous work in the literature by allowing the simulation of more complex heterogeneous samples. A comparison is made between transport theory results based on the work of the Chapter three, and results of the Monte Carlo simulations.

Chapter five describes the experimental apparatus forming the basis of the angle resolved electron energy analyser. The instrument consists of a commercially available spectrometer, supported by a diffusion-pump-based ultra high vacuum system. The design of a compact secondary electron detector is discussed, together with the integration of a new high resolution ion gun.

Attention is then turned to the modification of a cylindrical mirror electron energy analyser (CMA), with the intention of offering resolution not only of the electron energy, but also of the azimuthal angle of emission. This development is made possible by the design of a sectioned detector head which directly replaces the channeltron as the collector in the spectrometer. Chapter six covers this work as well as describing the design and integration of a multi-channel charge sensitive head amplifier which supplies signals to the computer interfacing system.

Chapter seven is devoted to the design of a versatile, user-friendly computer control system for the spectrometer. This also includes the interfacing hardware, low level control software and man-machine interface. The software is based around an easy to use graphical environment from which the user can gather secondary electron images as well as spectra, images and linescans from the angle-resolved CMA instrument. Because the position of the electron beam is controlled in software, scanning electron images can be used to identify specific regions of interest for further investigation by Auger analysis.

The instrument described in chapters five, six and seven has been shown to give useful experimental results. Chapter eight is a further investigation of the achievable results from the present system, showing preliminary spectra sets and images. Shortcomings of the present design are identified, and the author's proposals for further research into overcoming these are discussed.

CHAPTER TWO

BACKGROUND TO AUGER ELECTRON SPECTROSCOPY

2.0 Introduction

In a broad sense, the process of electron spectroscopy involves the bombardment of a sample by an energetic excitation source, followed by the energy analysis of the emitted electrons. Examination of this energy distribution reveals characteristic features, the location and magnitude of which lead to information about the elemental composition of the sample.

Auger electron spectroscopy (AES) is now a well established technique for quantitative chemical characterisation of the first few atomic layers of a surface. It allows identification of the atoms present in a sample by measurement of the characteristic energies of their Auger electrons. All elements other than hydrogen and helium can be identified with AES.

This chapter provides an overview of Auger analysis, then introduces the reader to some of the currently available techniques used for collection of Auger spectra and interpretation of experimental results.

2.1 The Auger Effect

Figure 2.1 outlines the process leading to the emission of an Auger electron. Firstly, a core atomic level of binding energy E_A is ionised by an incident energetic beam of either electrons or photons. The resulting core level vacancy is then filled by an electron falling from a less strongly bound energy level (E_B). The excess energy (E_A-E_B) may then be released from the atom as a photon, or transferred to an electron in a higher energy level (E_C) which is subsequently ejected into the vacuum.

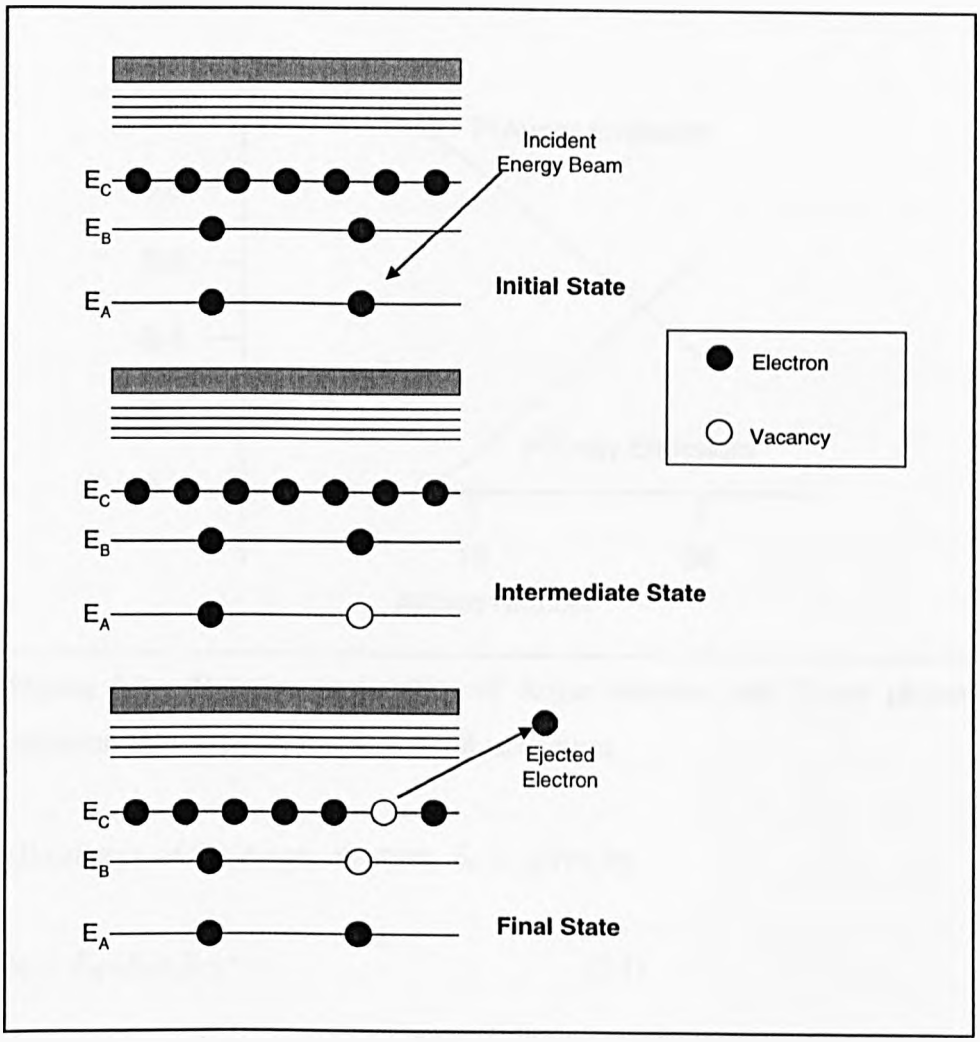

Figure 2.1 : Schematic diagrams showing the process leading to the emission of an Auger electron

Electrons ejected by this process are termed Auger electrons, and their energy is important because it is specific to the binding energies associated with the energy levels of the atom from which they originated. Assuming the initial core-level ionisation has occurred, Auger and X-ray photon emission are competing processes. Precisely which of the two processes dominates varies with atomic number for a particular initial-state hole level. The approximate probability of X-ray and Auger emission for an initial K shell ionisation is shown in Figure 2.2. In this case, the Auger process dominates for $Z<30$ and is the exclusive process for $Z<15$. For initial holes in other shells the Auger process dominates for higher atomic numbers (Chang, 1974).

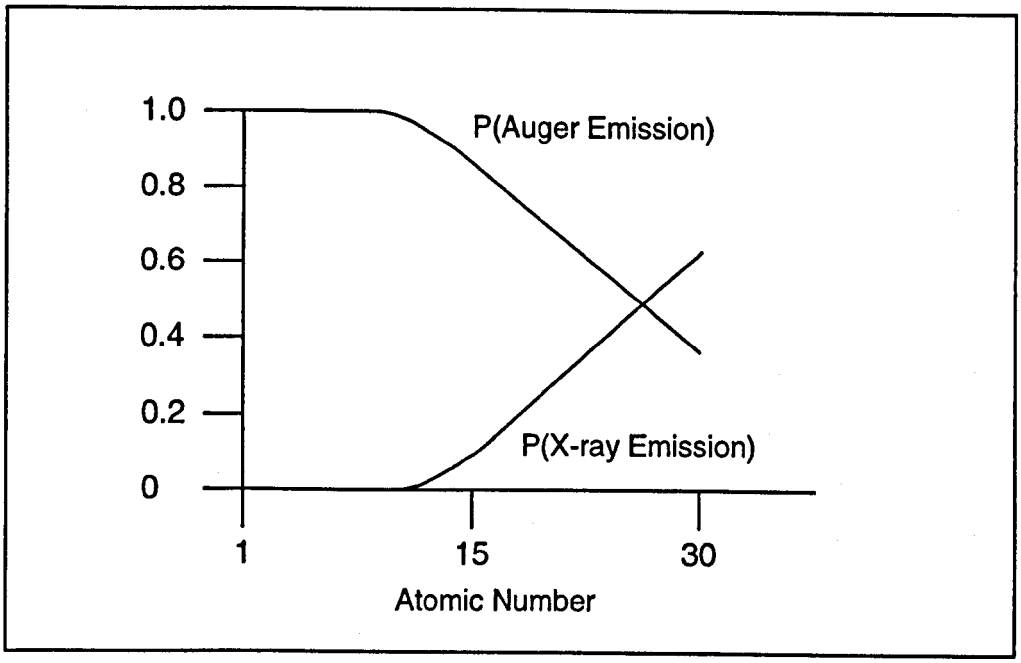

Figure 2.2 : Relative probability of Auger electron and X-ray photon emission following an initial K-shell ionisation

The kinetic energy of the Auger electron, E_k, is given by

$$E_k = E_A-(E_B+E_C)^* \qquad (2.1)$$

where E_I are the binding energies of the i^{th} atomic energy level. The term $(E_B+E_C)^*$ is slightly larger than (E_B+E_C) because it relates to the corresponding binding energy in the presence of a hole in level E_A (Briggs and Rivière, 1990). Examination of this formula explains the elemental specificity of the Auger technique. As equation 2.1 shows, the energy of the Auger signal is dependent only on the atomic energy levels of the atom from which it originates. Since there are no two elements with the same set of atomic binding energies, this gives a unique set of Auger energies for each element in the periodic table (with the exception of Hydrogen and Helium which only have electrons in their K shell and therefore cannot undergo Auger emission). The physically correct expression for Auger emission is

$$E_k = E_A-(E_B+E_C)- \tau +R^a+R^{ea} \qquad (2.2)$$

where E_A, E_B and E_C are the binding energies of the energy shells corresponding to the particular Auger process, τ represents the Coulomb interaction energy between the electrons, R^a is the intra-atomic relaxation energy resulting from electron removal and R^{ea} is the extra-atomic relaxation term associated with the response of the solid to the removal of electrons.

The Auger energies of relevant transitions for all elements are now well documented, rendering the use of predictive formulae unnecessary. An early example of this is the work of Davis *et al.* (1976) who published a comprehensive table of the principle Auger peaks for all elements in the range of 3-92 and electron energies less than 2400eV. There are several more recent examples of published standard spectra (McGuire, 1979; Shiokawa *et al.*, 1979; Sekine *et al.*, 1982).

For meaningful comparisons to be drawn from the published data, calibration of the spectrometer is of prime importance. A method has been described (Seah *et al.*, 1990) by which energy scale calibration can be achieved by measurement of the energy of the low-energy Cu doublet, and this method results in a value which is independent of instrument resolution. Calibration of the intensity scale can be carried out by comparison of measured spectra with standard spectra of the same element (Seah *et al., 1989).* Auger spectra have recently been obtained on a calibrated spectrometer and will be made available shortly in both handbook and electronic form (Barthés-Labrousse, 1998).

2.2 Surface Specificity and UHV

The widespread use of AES as an analytical technique has gone hand in hand with the development and increased availability of Ultra High Vacuum (UHV) equipment. The requirement for this is explained by consideration of Figure 2.3, showing the variation in attenuation length with electron energy (Seah and Dench 1979).

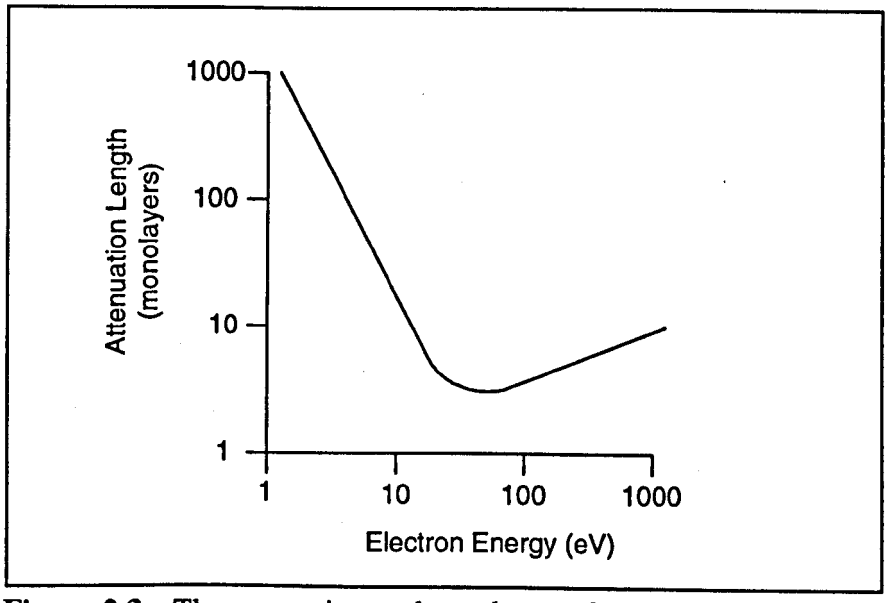

Figure 2.3 : The approximate dependence of attenuation length on the electron energy

The attenuation length refers to the distance an electron can travel in the matrix before it loses sufficient energy to no longer contribute to the Auger peak in the spectrum through inelastic scattering processes. The attenuation length therefore defines the maximum depth from which useful Auger analysis can be carried out (e.g. a source at a depth three times the attenuation length will give an Auger signal approximately $1/20^{th}$ of the corresponding surface atoms for electron emission normal to the surface). Examination of Figure 2.3 reveals that in the energy region of interest for Auger analysis, typical attenuation lengths are of the order of 2 – 10 monolayers, making the technique inherently surface specific.

As a result of this sensitivity, for successful analysis to be carried out we must ensure that the experiment can be carried out in less than the time taken for a monolayer of contaminating molecules to form on the sample surface. This requirement is discussed in detail in Chapter five, but essentially we can consider that currently available

vacuum technology allows pressures in the region of 10^{-10} mbar to be routinely achieved. Under these conditions, a monolayer will form over the course of several hours which is sufficient time for most experiments to be carried out.

The requirement for UHV also imposes constraints on the materials used in the construction of analytical tools intended for use inside the clean environment. The aim is to limit the amount of outgassing within the vacuum system. This is of particular importance when the material is placed in an area of restricted pumping, and it is important that the physical design of surface analysis equipment takes into consideration the pumping speeds required to maintain satisfactory vacuum conditions.

Some of the most commonly favoured materials are molybdenum, tantalum, gold (mainly for conductors and seals), machinable glass ceramic and PEEK (an aromatic machinable polymer). Construction methods are also limited; suitable techniques include spot welding, argon arc welding and vacuum brazing.

2.3 The Instrumentation of AES

The heart of any spectrometer system is the electron energy analyser. The type of analyser used can determine many characteristics of the overall achievable performance including energy resolution, angular acceptance angle and sensitivity. In addition, a source of excitation is required to stimulate the emission of Auger electrons from the sample under examination, and the characteristics of this have a significant bearing on the overall performance of the system. Notably, the spatial resolution of the spectrometer is mainly controlled by the spot size of the incident excitation source.

Under atmospheric conditions, surface contaminants commonly build up on a sample in a matter of seconds. It is essential to be able to clean and prepare a sample in the same UHV environment as analysis, in order to preserve the cleanliness of the surface for the duration of the experiment. Finally, provision must be made to allow the manipulation of a sample in the vacuum environment.

2.3.1 Sources for Auger Excitation

Several methods exist for stimulating the emission of Auger electrons, the aim being to irradiate the sample with sufficient energy to result in initial core level ionisation. It is possible to achieve this either by bombardment of a surface with energetic electrons, X-rays or more rarely, ions.

The two most commonly used X-ray sources are the Al$K\alpha$ and the Mg$K\alpha$ lines of 1487eV and 1254eV with line widths of 0.9eV and 0.8eV respectively (Thompson *et al.*, 1995). X-rays sources are less commonly used than electrons partly because electrons can be focused more effectively to give higher spatial resolution, but also due to the extra cost involved.

There are two types of electron source currently in use: thermionic emitters and field emitters. Thermionic emitters consist of a heated filament (typically tungsten), the high temperature of which allows electrons to overcome the work function of the material an escape to the vacuum.

Field emission sources use a very fine (sub micron) tip with a high electrostatic potential to lower the work function to the point at which electrons can tunnel into the vacuum (El-Gomati, 1983; El-Gomati *et al.*, 1985 and references therein). Because the electrons are emitted from a single point, field emission sources produce a higher current density than thermionic emitters and therefore the achievable spatial resolution is generally superior.

The main disadvantage of a field emission electron source is that a pressure in the region of 10^{-9} mbar is needed for stable operation, as opposed to approximately 10^{-3} mbar for a thermionic source. Higher pressures can cause static discharge between the tip of the field emitter and the high voltage extractor, destroying the emitter. For this reason, automated safety measures are usually designed, which disable the high voltage supplies in the event of a pressure rise.

2.3.2 Sample Cleaning and Preparation

In general, a sample is transferred to the vacuum system under atmospheric pressure where it will invariably become contaminated with a thin layer of adsorbates and often an oxide layer. For successful surface analysis to be carried out, the acceptable level of contamination is determined by the sensitivity of the technique. For AES, the limit of detection is approximately 0.1% (Rivière, 1990).

Several methods exist for achieving the required degree of surface cleanliness. Fracturing a sample along a specific crystal orientation can expose a clean surface for examination, or other mechanical means can be used to erode the surface such as scraping with a clean razor blade.

Heating the sample stimulates the desorption of surface contaminants, but may result in segregation of impurities to the surface. For this reason, the sample is often subjected to subsequent bombardment by energetic noble gas ions, typically argon ions of energies between 500eV and 5keV (Rivière, 1990). This bombardment, also referred to as sputtering, essentially results in the removal of the top layer of material with the intention of exposing the clean sample underneath. Often, the sample is treated with repeated iterations of heating and ion bombardment so that contaminants

segregated to the surface during the heating cycle are removed in a subsequent bombardment cycle.

In addition to cleaning the surface of a sample, ion bombardment can be used to produce a specific surface geometry. By deflection of an ion beam under computer control, it is possible to cut away successive layers of surface material to produce geometric structures such as bevels and craters. This process requires careful characterisation of the ion source to determine sputtering yield and spot size so that controlled experiments can be carried out.

2.3.3 Electron Energy Analysers

The process of Auger analysis relies on the acquisition of accurate and precise electron energy spectra. The task of the electron energy analyser is to measure the kinetic energy distribution of the electrons arriving within its solid angle of detection. Three such analysers have found widespread popularity over the years: the retarding field analyser, the cylindrical mirror analyser and the concentric hemispherical analyser. These are discussed in detail elsewhere (Roy and Carette, 1977; Rivière, 1990), but will be described briefly in this section.

2.3.3.1 The Retarding Field Analyser

The retarding field analyser (RFA) was the first experimental arrangement to be used for observation of Auger spectra, and is depicted schematically in Figure 2.4. Essentially, the RFA is a high pass filter and operates by presenting a potential barrier through which only electrons over a certain energy can pass and ultimately be detected. The RFA was originally adapted from the optical apparatus used for LEED experiments (Chang, 1974), and consists of two grids, one grounded and the other held at a variable potential, followed by a hemispherical screen acting as a collector.

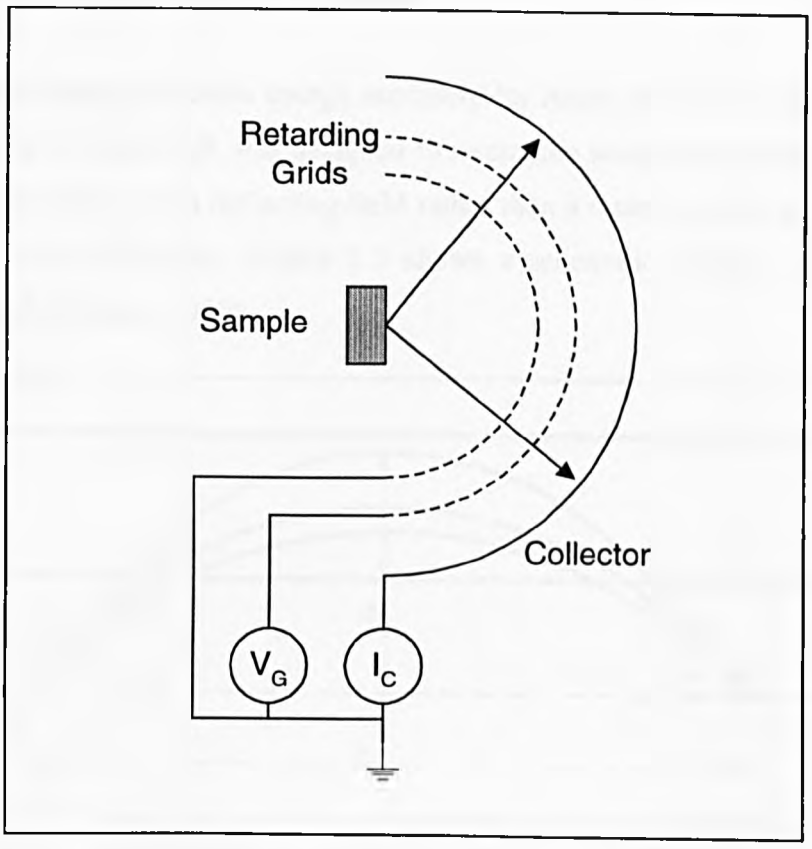

Figure 2.4 : Schematic diagram of the retarding field analyser

The current, *I(E)* at the detector for a retarding potential ε and primary beam energy ε_p, is given by

$$I(\varepsilon) = \int_{\varepsilon}^{\varepsilon_p} N(E)dE \qquad (2.3)$$

The RFA therefore acts as a high pass filter and as the retarding potential is decreased, the background signal increases in a cumulative manner. This results in a poor signal to noise ratio, particularly at low energies where Auger analysis is often performed. For this reason the RFA is not widely used other than to complement LEED detection with Auger analysis, where the same detector arrangement can be used for both studies.

2.3.3.2 The Cylindrical Mirror Analyser

One of the more popular electron energy analysers for Auger study is the cylindrical mirror analyser (CMA), which was designed to overcome some of the shortcomings of the RFA. The CMA uses a deflecting field rather than a retarding field of the RFA to selectively collect electrons. Figure 2.5 shows a schematic diagram of a basic single-pass CMA (Rivière, 1990).

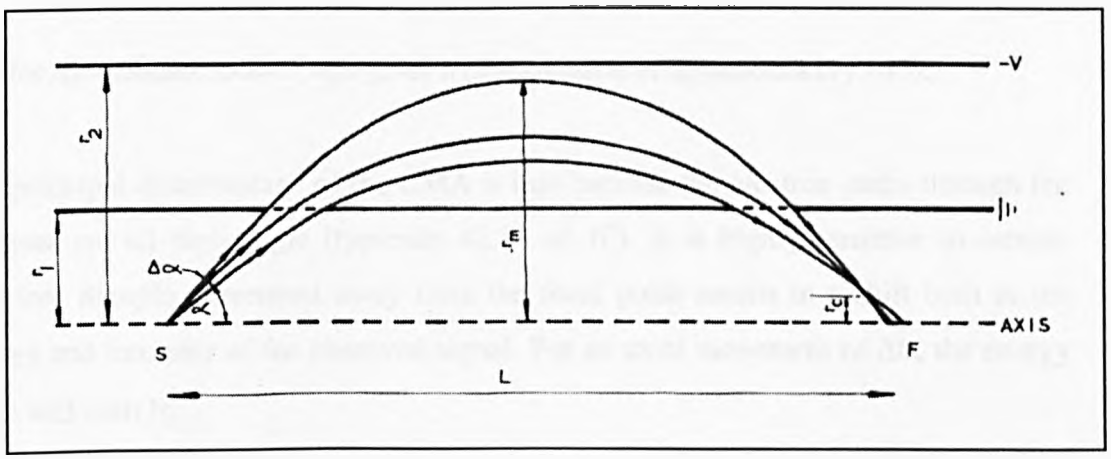

Figure 2.5 : (from Rivière, 1990) Schematic diagram of the cylindrical mirror analyser

In essence the design consists of two co-axially mounted cylinders of radii r_1 and r_2. The innermost cylinder is grounded whilst the outer cylinder is held at a negative voltage, -V according to the desired pass energy. The inner cylinder has slots at either end; electrons from S pass into the space between the cylinders and are deflected by the negative voltage on the outer cylinder, following parabolic trajectories according to their energy. The electrons of energy corresponding to the 'pass energy' of the analyser (for a given outer cylinder voltage) leave the space between the cylinders and pass through the exit aperture to be brought to a focus at F.

Sweeping the voltage on the outer cylinder under computer control whilst monitoring the number of counts (or current depending on the specific design) from the analyser allows an energy spectrum to be plotted. The focal relationship is given by

$$E_0 = \frac{Ke}{\ln(r_2 / r_1)} V \qquad (2.4)$$

where K is a constant, E_o the pass energy and V the voltage on the outer cylinder.

The CMA has a much improved signal-to-noise ratio compared with the RFA, but is also desirable because of its high transmission efficiency resulting from a large solid angle of collection. The transmission of the CMA is given by (Rivière, 1990)

$$T = 2\sin\alpha\delta\alpha = 1.346\delta\alpha \qquad (2.5)$$

and for the standard $\delta\alpha = 6^o$ this gives a transmission of approximately 14%.

The principal disadvantage of the CMA is that because the electron paths through the analyser are all high angle (typically 42.3° +/- 6°) it is highly sensitive to sample position. Sample movement away from the focal point results in a shift both in the energy and intensity of the observed signal. For an axial movement of ΔL, the energy scale will shift by

$$\Delta E = E_0 \frac{\Delta L}{5.6 r_1} \qquad (2.6)$$

which, for the CMA used in this study where r_1=27mm, results in a shift of approximately 6.6eV for every 1mm error in axial position at 1000eV.

For small values of the angular spread, δα, the energy resolution of the CMA is given by (Rivière, 1990)

$$\frac{\Delta E}{E_0} = 2.75(\delta\alpha)^3 \qquad (2.7)$$

The characteristics of the CMA analyser will re-examined in Chapter five.

2.3.3.3 The Concentric Hemispherical Analyser

Figure 2.6 shows a schematic diagram of the concentric hemispherical analyser (CHA). Voltages V_1 and V_2 are applied to the inner and outer hemispheres respectively such that electrons of a specific pass energy are deflected through the space between them and brought to a focus on the opposite side.

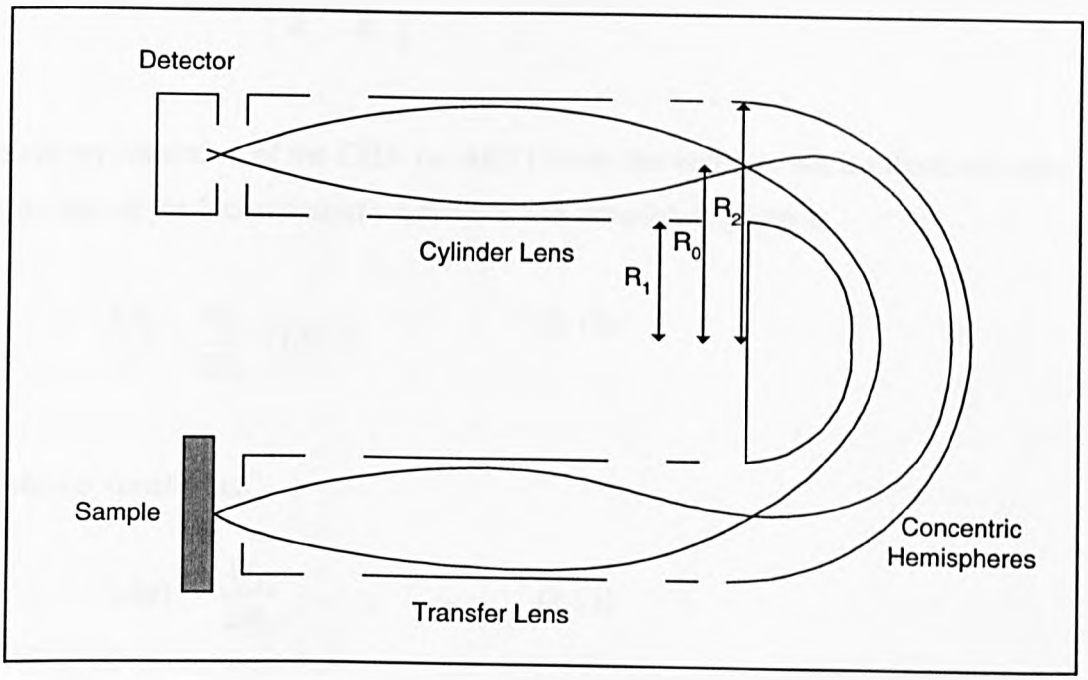

Figure 2.6 : Schematic diagram of the concentric hemispherical analyser

The median equipotential surface occurs at a radius R_0 with a potential V_0 where (Rivière, 1990; Purcell, 1938)

$$V_0 = \frac{V_1 R_1 + V_2 R_2}{2R_0} \qquad (2.8)$$

If electrons of energy $E=eV_0$ are injected to the median surface through a slit located midway between the hemispherical electrodes, then they will follow circular orbits according to the following expressions

$$V_1 = V_0\left[3 - 2\left(\frac{R_0}{R_1}\right)\right] \qquad (2.9)$$

and

$$V_2 = V_0\left[3 - 2\left(\frac{R_0}{R_2}\right)\right] \qquad (2.10)$$

from which

$$V_2 - V_1 = V_0\left(\frac{R_2}{R_1} - \frac{R_1}{R_2}\right) \qquad (2.11)$$

The energy resolution of the CHA for AES (where the entrance slit is effectively zero, i.e. the size of the focused electron beam at the sample), is given as

$$\frac{\Delta E_b}{E} = \frac{W_2}{2R_0} + (\delta\alpha^2) \qquad (2.12)$$

which is optimal when

$$(\delta\alpha)^2 = \frac{W_2}{2R_0} \qquad (2.13)$$

where W_2 is the width of the exit slit and $\delta\alpha$ is the angular spread of electrons. Note that the energy resolution of the CHA depends on the square of this angular spread, as opposed to the CMA for which it varies as the cube of $\delta\alpha$.

To allow operation of the CHA with a practical working space around the sample, the transfer lens focuses electrons from the sample onto the entrance slit of the two concentric hemispheres. Another cylindrical lens arrangement carries the electrons from the hemispheres to the detector, this second lens allows the removal of any secondary electrons generated by electron interactions with the hemisphere walls (Roy and Carette 1977, 1990).

The CHA can be operated either at constant pass energy, or constant retardation mode. In constant pass energy mode, the transfer lens retardation is varied to scan

through the desired energy range and produce N(E)/E. In constant retardation mode, the potential on the hemispheres is varied in order to produce an N(E) spectrum.

2.3.4 Comparison of Electron Energy Analysers

Whilst the RFA is favoured for its simple design and abstraction from existing LEED apparatus, its low signal-to-noise ratio makes it unsuitable for many Auger studies. For this reason, the CMA and CHA are now by far the most widely used analysers in both Auger electron and X-ray photoelectron spectroscopy.

The sensitivity to sample position of the CMA represents its main drawback, and this characteristic is not exhibited by the CHA. Increasing the radius of the inner cylinder of the CMA reduces the apparent energy shift for a given sample displacement. This problem can be overcome by the use of a double pass CMA, where the first electrostatic lens defines the source for the second (Palmberg, 1974). The double pass arrangement also serves to eliminate spurious signals caused by internal scattering.

The principal advantage of the CMA over the CHA is its high transmission (for a given energy resolution). In the case of the CHA this can be improved upon with the incorporation of a transfer lens with a high solid angle of collection and high magnification.

2.4 The Angle Resolved CMA

As we have already seen, the CMA and CHA offer by far the most appealing performance characteristics of the available electron energy analysers used in AES studies. For this reason, they have become the exclusive tools of analysts in this field. The CMA is generally favoured for its simple design, high transmission and efficiency, whilst the CHA, although a more complex instrument, is much less sensitive to sample position and is more versatile in certain experimental scenarios.

The aim of part of the work described in this thesis is to implement an angle resolving electron energy analyser of high spatial resolution. The CMA was chosen as a basis for this design, since the azimuthal component of the electrons trajectory is preserved as it passes through the analyser.

2.4.1 The Exploitation of Angle Resolved Analysis

Angle resolved spectroscopy has found acceptance in the world of surface science as a useful tool in a variety of situations including the study of crystal structures (Chambers *et al.*, 1985; Idzera *et al.*, 1988). In the case of Auger electron spectroscopy, angle resolved analysis gives us the ability to gather both structural and chemical information simultaneously. The angle resolved instrument proposed in this thesis is intended to facilitate the elimination of spectral artefacts in the examination of samples with complex topographical features.

The facility to select an isolated azimuthal range of detection when using a CMA for scanning Auger microscopy is likely to be particularly advantageous when the sample geometry contains deep trenches or raised particles. Figure 2.7 illustrates this condition for the case of a deep trench in a simple single-overlayer system. The electron beam strikes the bottom of the trench (Z_B), but because the CMA detector and electron gun are co-axially mounted, much of the useful signal is shadowed by the wall of the trench. In this instance, the useful signal is likely to be lost in the background noise from the shadowed region.

Using a CHA this problem can be overcome, since the electron gun and the analyser can be positioned independently (although there will obviously always be a physical limitation imposed by the size of these devices). However, in the case of an angle resolved CMA the signal collected from the shadowed region can simply be ignored, vastly improving the signal-to-noise ratio in this experiment. The sectioned detector proposed in Chapter six will provide this selectivity without the need for post-processing of data.

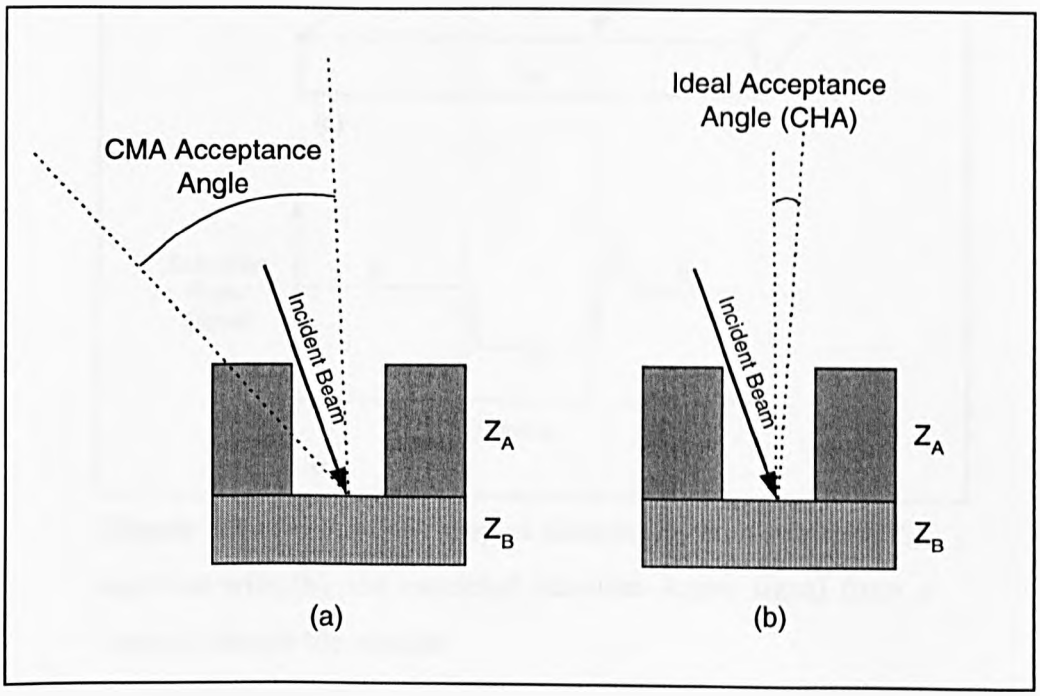

Figure 2.7 : Examination of trenches in a single overlayer system using (a) the CMA and (b) the CHA electron energy analysers

It is well understood that surface roughness can lead to artifacts in Auger images and line scans (El-Gomati *et al.,* 1988), but to some extent it may be possible to resolve these if the signals can be split into azimuthal sections. Figure 2.8a shows an example of such a structure, consisting of a raised strip of element Z_A on a substrate Z_B. Figure 2.8b illustrates the expected Auger signal from the substrate as an electron beam is scanned across the raised strip. Conversely, Figure 2.9 illustrates the same experiment, but this time separating out the signals gathered in semiconical sections in shadowed and unshadowed regions as the electron beam scans the surface of the sample.

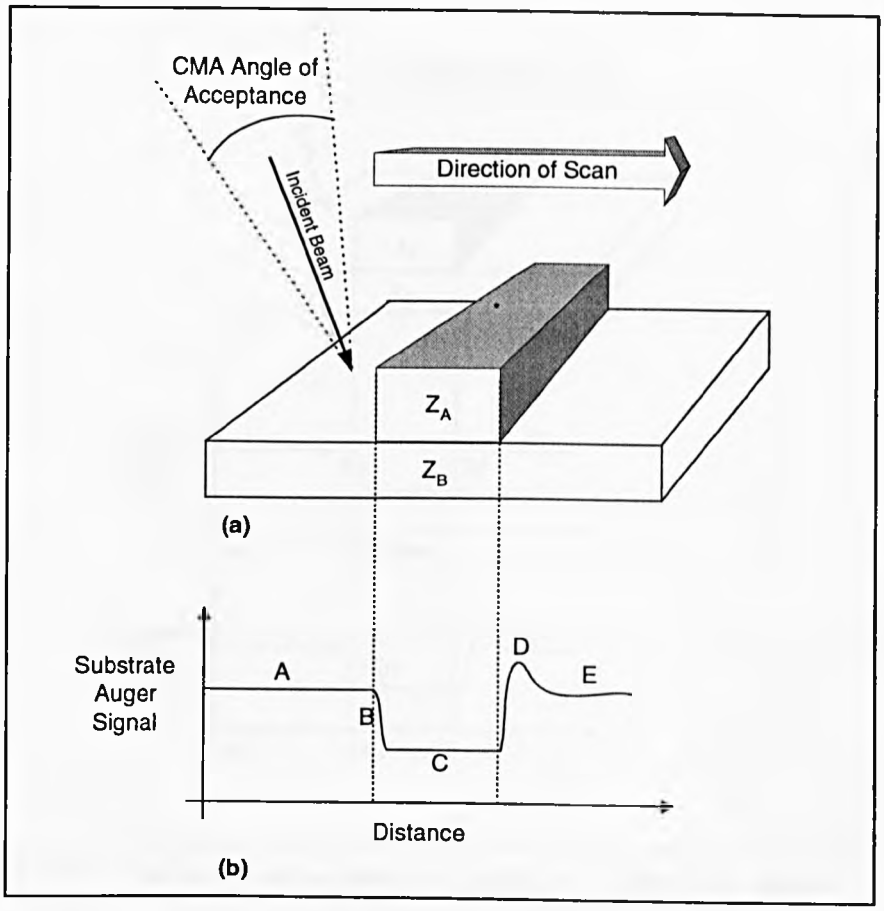

Figure 2.8 : (a) A raised strip of element Z_A on a substrate Z_B, together with (b), the expected substrate Auger signal from a linescan across the sample

The expected substrate Auger signal is as follows (El-Gomati *et al.*, 1988):

(A) The signal is initially exclusively from the substrate.

(B) As the electron beam climbs the side of the strip, the signal gradually drops.

(C) Whilst the beam is striking the top of the strip, the substrate Auger signal is at its lowest level.

(D) At this point, due to the off-normal angle of incidence of electrons and the pear shaped interaction volume, electrons emerging from the side of the raised strip strike the substrate and actually produce an enhanced Auger signal due to their grazing incidence and reduced energy leading to an enhanced ionisation cross section.

(E) The substrate signal returns to its steady state when the primary beam is clear of the raised strip.

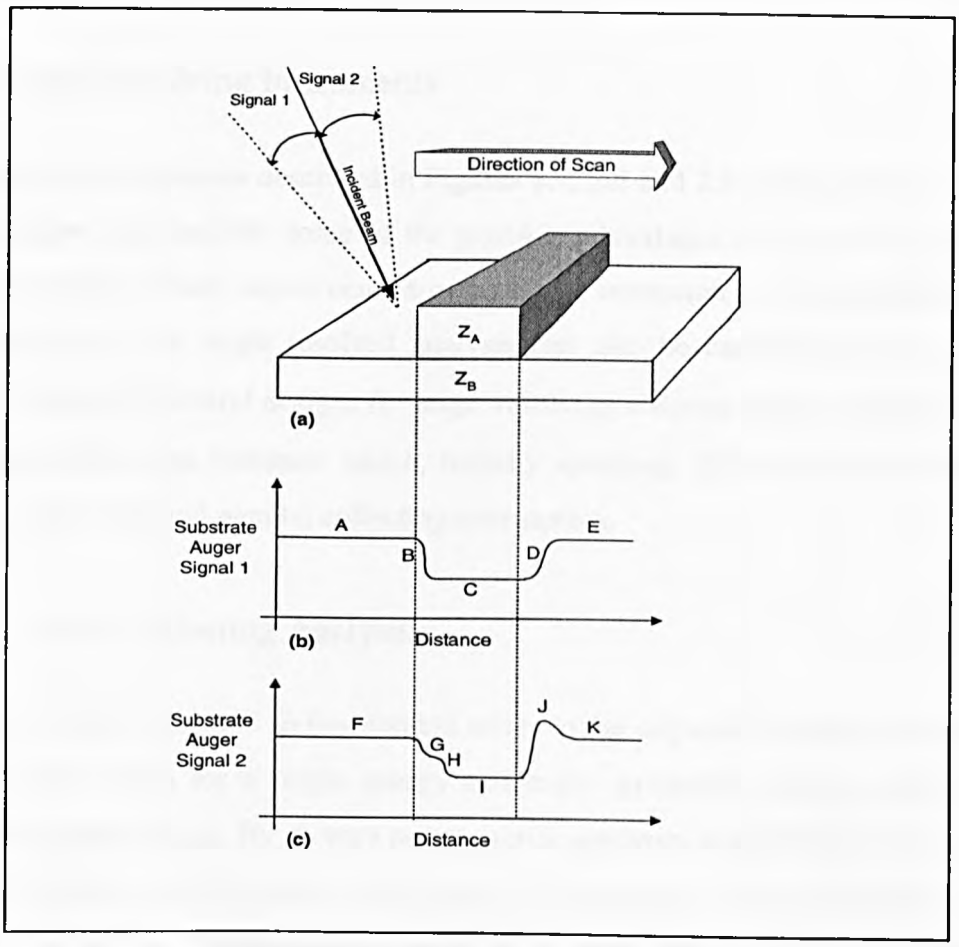

Figure 2.9 : The same experiment as shown in Figure 2.8, except displaying the Auger signal from (b) behind and (c) in front of the scanning beam in the direction of the scan

The features are as follows :

(A), (E), (F) and (K) The beam is striking the substrate directly.

(C) and **(I)** are produced when the beam is striking the top of the strip.

(G) The primary electron beam is falling just short of the raised strip, but some of the returning Auger electrons are lost in the strip.

(H) The primary beam is moving up the wall of the strip, the interaction volume with the substrate is diminishing.

(B) The same effect as (G) and (H), except because this is the signal from behind the beam, the effect described in (G) is not so prominent and the two effects blur together.

(J) The yield enhancement effect occurs as seen in Figure 2.8.

(D) Because this signal is collected behind the primary beam, the yield enhancement effect is shadowed by the raised strip.

2.4.2 Angle Resolving Instruments

Although the experiments described in Figures 2.7, 2.8 and 2.9 are hypothetical, their consideration does indicate some of the possible advantages to be gained by angle resolved analysis. These experiments are specific to resolution of the azimuthal angle of emission, ϕ, but angle resolved analysis can also be carried out in the polar emission angle, θ. Several designs for angle resolving electron energy analysers have been proposed in the literature which, broadly speaking, fall into the two distinct categories of serial and parallel collecting instruments.

2.4.2.1 Serial Collecting Analysers

The term 'serial collection' in this context refers to the stepwise process of collecting spectral information for a single energy and angle, gradually sweeping across the desired parameter range. By its very nature, serial spectrum acquisition can be a time intensive process, which in turn may impact on the quality of results obtained. The build up of surface contamination may occur at a sufficiently fast rate to become significant over the course of an experiment. Furthermore, mechanically induced artefacts will be more pronounced the longer an experiment takes to complete.

One of the most simple experimental arrangements for the collection of angle resolved Auger spectra is based on the retarding field analyser. Mróz *et al.* (1988) used a modified LEED apparatus, with both the sample and electron gun held on an adjustable platform to allow rotation around two perpendicular axes. The resulting Auger signal is collected through a small hole in the LEED optics, giving a very limited angle of collection. The angle resolved spectra can be obtained by repeatedly gathering the desired Auger signals whilst stepping through the desired energy and angle range. This arrangement is shown in Figure 2.10.

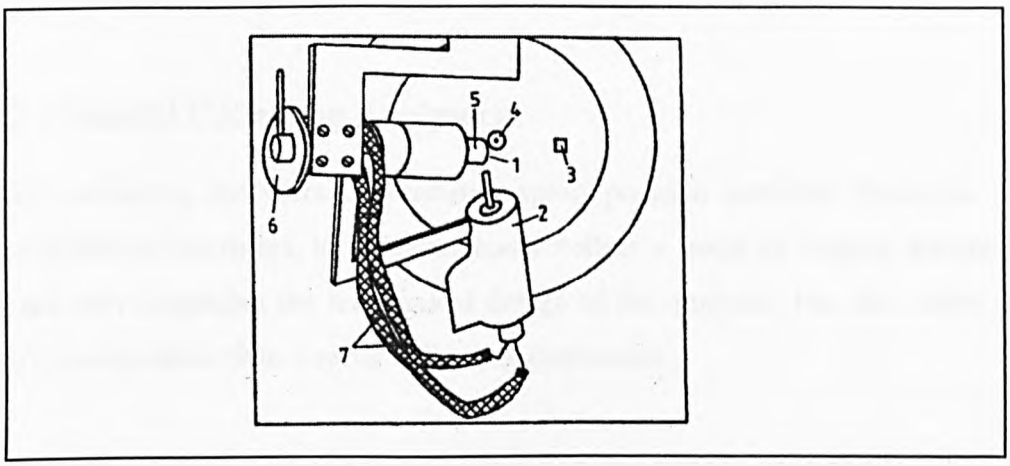

Figure 2.10 : (from Mróz *et al.*, 1988) A simple angle resolved AES system based on LEED optics. 1 – the sample, 2 – the electron gun for AES, 3- the aperture in the LEED optics with the AES collector behind it, 4- LEED electron gun, 5- sample heater, 6- protractor for measuring azimuthal angle, 7- shielded wires to electron gun

Hoflund *et al.* (1987) used an angle resolving CMA based on a double-pass analyser with an angularly resolving mechanical aperture to select a 90° azimuthal segment for analysis. This experimental set up incorporates an external electron gun to provide a grazing angle for the incident electron beam. Varying the acceptance angle of the analyser was shown to effectively control the surface sensitivity of the AES technique, allowing depth profiling of very thin overlayers. Figure 2.11 shows a schematic diagram of this arrangement.

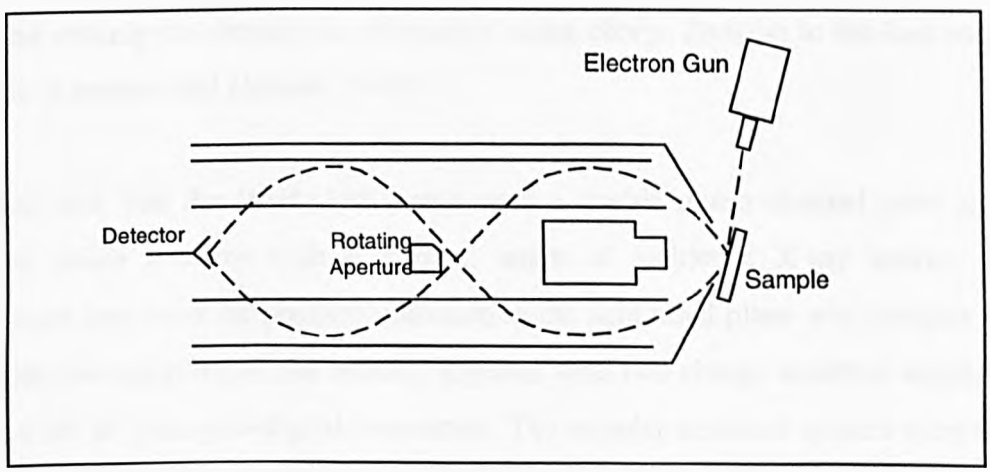

Figure 2.11: The angle resolved CMA as described by Hoflund *et al.* (1987)

2.4.2.1 Parallel Collecting Analysers

Parallel collecting analysers universally exploit position sensitive detectors, rather than mechanical apertures, to simultaneously collect a range of angular information. This not only simplifies the mechanical design of the analyser, but also offers faster spectrum acquisition than a serial collecting instrument.

All the angle resolving spectrometers covered in this section are based on the CMA. This is not surprising since the azimuthal component of the electron trajectory is preserved in this design of analyser, and can be derived simply if a suitable position sensitive detector (PSD) is fitted to the CMA. Position sensitive detectors, together with appropriate electronics to interpret their signals, have been the subject of considerable research (Martin *et al.*, 1981; Lampton and Malina, 1976). Annular divisions on such a detector can give information about the polar emission angle of the electrons (for a given energy), whilst sectioning the detector radially allows determination of the corresponding azimuthal angle.

Bosch *et al.* (1984) used a similar experimental arrangement to that depicted in Figure 2.11 for angle-resolved photoelectron spectrometry. They used a grazing source of incident radiation (in this case X-rays), but instead of using a rotating slit to systematically select azimuthal angles, they used a resistive anode PSD with channel electron multiplier plates. The position of the centre of each charge cloud leaving the plates and striking the detector is determined using charge division to the four output terminals (Lampton and Carlson, 1979).

Van Hoof and Van der Wiel (1980) also used a double micro channel plate and a sectioned anode detector with a grazing angle of incidence X-ray source. The arrangement they used for position detection in the azimuthal plane was complex and used capacitive coupling of the anodes, together with two charge sensitive amplifiers feeding a set of analog-to-digital converters. The angular resolved spectra were then reconstructed in a post-processing operation on a computer.

A video camera can be used as the PSD (Huang *et al.*,1993), again using computer based post-processing to interpret images and produce spectra. This arrangement is

shown in Figure 2.12; an MCP (microchannel plate) electron multiplier is used, followed by a YAG (yttrium-aluminium-garnet) scintillator to give a visible light output to a camera.

With this arrangement, although angle resolved Auger images were successfully obtained, they exhibited poor signal-to-noise ratio (SNR). This problem was attributed to the dynamic range, short integration time and inherently low SNR of the SIT-TV camera used in the experiments, and a modified version using a cooled CCD camera was suggested to overcome these difficulties.

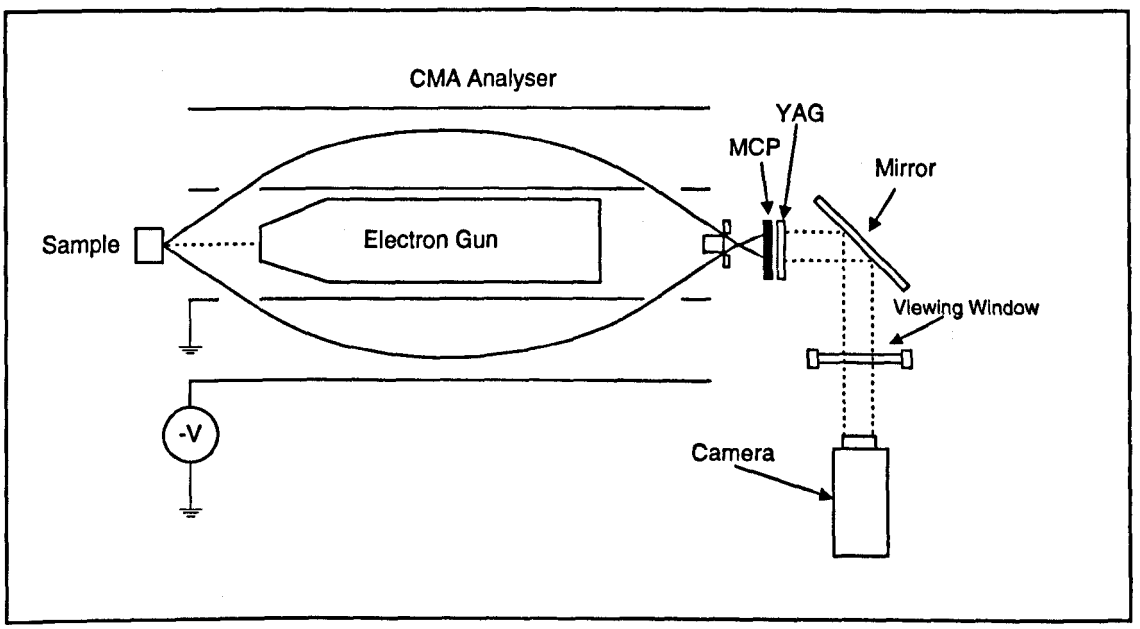

Figure 2.12 : The camera based angle-resolved spectrometer (Huang *et al.*, 1993)

An alternative experimental arrangement for obtaining angle-resolved electron spectra in the azimuthal angle, based on the CMA is described in Chapter six.

2.5 Quantitative Auger Analysis

The use of Auger electron spectroscopy for determination not only of the chemical species present on a surface, but also of their relative or absolute concentration, is now well established in the literature (Seah, 1990). The energy of an Auger peak in an electron spectrum gives us information about the elemental composition of the surface, but if we are able to relate the intensity of the Auger peak to atomic concentration then quantitative analysis is also possible. The process of quantitative analysis can be difficult due to the complex factors introduced by analyser characteristics and experimental conditions.

One method by which quantification can be achieved is by comparison of spectra with results from known samples, often from published sources in the literature (Palmberg *et al.*, 1972; Davis *et al.*, 1976; Sekine *et al.*, 1982). This technique involves comparison of measured Auger peak-to-peak heights (for differential mode spectra) or peak-to-background (for E.N(E) spectra) with the published standards. This method has been shown to be somewhat inaccurate since it fails to take into consideration matrix effects which vary according to experimental conditions (Hall and Morabito, 1979).

Although quantification by comparison with standard results can be improved upon by attempting to match experimental conditions and analyser characteristics as closely as possible, the optimum method of quantification is based upon the determination of atomic concentration from more rigorous calculation (Seah, 1990; Walker *et al.*, 1988; Fitzgerald, 1998). Such methods allow the analyst to take into consideration both experimental factors such as analyser transmission, and sample specific factors including ionisation cross section and backscattering characteristics, all of which influence the Auger signal intensity.

Current quantification techniques calculate the measured Auger signal, $I_{AXYZ}(theor,E_0)$, into a small solid angle $d\Omega$ for a homogeneous sample of element A involving the XYZ Auger transition, as (Seah and Gilmore, 1998b)

$$
\begin{aligned}
I_{AXYZ}(theor, E_0) = \; & I_0 \gamma_{AXYZ} n_{AX} \sigma_{AX}(E_0) \\
& \times \sec\alpha[1 + r_A(E_{AX}, E_0, \alpha)] \\
& \times N_A Q_A(E_{AXYZ}, A) \\
& \times \lambda_A(E_{AXYZ}) \cos\theta d\Omega / 4\pi
\end{aligned}
\qquad (2.14)
$$

where I_0 is the primary electron beam current, γ_{AXYZ} is the probability that the ionised core level X in element A is filled with the ejection of an XYZ Auger electron, $\sigma_{AX}(E_0)$ is the ionisation cross section for electrons of energy E_0 of the core level X in element A with population n_{AX}, α is the angle of incidence of the electron beam with respect to the surface normal, $r_A(E_{AX}, E_0, \alpha)$ is the backscattering factor accounting for additional ionisation of the core level X due to backscattered electrons, N_A is the atomic density of the A atoms, $Q_A(E_{AXYZ}, A)$ is a term to allow for elastic scattering, $\lambda_A(E_{AXYZ})$ is the inelastic mean free path for the XYZ Auger electrons in element A with energy E_{AXYZ} , and θ is the angle of emission of the detected electron with respect to the surface normal.

Historically, the most popular ionisation cross section has been that of Gryzinski (1965), although recent work based on correlation with experimental data shows that use of this cross section can lead to errors of up to 50% (Seah and Gilmore, 1998a). In preference to the Gryzinski cross section, Seah and Gilmore (1998a) propose the use of the form of Casnati *et al.*'s cross section (Casnati *et al.*, 1982) which they show to be accurate to within the uncertainty of the experimental data for values of the overpotential $U(=E_0/E_{AX})$ less than 100.

2.5.1 The Depth Distribution Function

The primary electron beam used in AES experiments typically penetrates several microns into the sample, the precise depth depending on sample material and electron energy. This incident beam stimulates the emission of Auger electrons throughout the entirety of its interaction volume, but it is only the electrons that escape without losing energy (i.e. an energy loss of less than 500meV) which form part of a useful signal. Inelastic processes acting on a sample between its emission and ultimate

detection, limit the depth at which successful analysis can take place; for accurate quantitative analysis to take place, an understanding of this effect is essential.

The information required to describe the electron transport effects leading to the detection of a signal electron is contained in the emission depth distribution function (DDF). As defined by the ASTM E-42 committee for surface analysis, the DDF can be described as the probability $\Phi(z; \theta; \phi)\, d\Omega\, dz$, that an electron emitted between depths z and $z + dz$ leaves the surface in the direction defined by the solid angle $d\Omega$ at polar angle θ, azimuthal angle ϕ with respect to the surface, without significant energy loss.

Early attempts to model this effect were based around the so-called straight line approximation (SLA) in which electrons were assumed to travel from emission to detection without changing direction (Dwyer and Matthew 1984). This method represents the depth distribution function as a simple decaying exponential, giving (Dwyer and Richards, 1992)

$$\phi(z,\theta) = \frac{1}{\cos\theta\lambda_i}\exp\left(-\frac{z}{\cos\theta\lambda_i}\right) \qquad (2.15)$$

where λ_i is the inelastic mean free path, which will be discussed later in this chapter. The SLA has been shown to give reasonably accurate results under certain experimental conditions. Because it neglects the effect of scattering events (in which the electron changes direction), for high atomic number elements and large emission angles the assumptions of the SLA become invalid. Elastic scattering corrections in AES have been the subject of extensive research in recent years (Jablonski and Tougaard, 1998; Cumpson, 1993; Cumpson and Seah, 1997; Cumpson, 1997).

2.5.1.1 Transport Theory

A number of recent publications (Jablonski and Tilinin, 1995; Werner and Tilinin, 1992; Dwyer and Richards, 1992) have been devoted to the development of a theory of electron transport which correctly accounts for elastic scattering. This overcomes one of the major simplifications of the straight line approximation. Transport theory is based around the argument that secondary emission characteristics of electrons in homogeneous samples can be described in terms of the inelastic mean free path, λ_I and the transport mean free path, λ_{tr}. The transport mean free path represents a mean distance between high angle elastic scattering events and will be discussed in the next chapter.

In general, transport theory applies only to uniform samples, making it unsuitable for calculations involving multilayered systems of more than one material. Tilinin *et al.* (1997) presented a method by which both the angular and energy distribution of signal electrons could be found analytically in systems with ultrathin overlayers. Transport theory involves the replacement of the exact differential elastic scattering cross section by an approximate one equal to the corresponding momentum transfer cross-section. It is important to note that application of this method is limited to cases where the ratio of the inelastic to the transport mean free path can be considered constant. This constraint limits the practical application of the technique, and in many experimental situations more accurate modelling techniques are required.

2.5.1.2 Monte Carlo Simulation

Monte Carlo simulations have become widely accepted in many fields of physics, in situations where an analytical solution cannot be found without making assumptions which lead to oversimplification. Modeling the elastic scattering processes influencing a measured spectrum using an analytical approach is an extremely complex task due to the huge number of ways in which an electron can be scattered between emission and detection (Tougaard, 1982). In response to this, the technique of Monte Carlo simulation has become an essential tool in the study of electron solid interaction effects (Joy 1988).

In general, a Monte Carlo simulation represents a problem as a parameter of a hypothetical population, and uses a random sequence of numbers to construct a sample of the population from which statistical estimates of the parameter can be obtained (Shimizu and Ding, 1992). In the context of electron-solid interaction, the hypothetical population refers to a range of possible scattering events which may influence the path of an electron as it moves through a matrix.

Essentially, the Monte Carlo simulation tracks each electron from its emission, until it either leaves the sample or loses energy to the point at which it is no longer of interest. If we are tracking the trajectories of signal electrons (such as Auger electrons), then often we are only interested in electrons that have not lost energy between emission and detection. For this reason, many simulations choose to reject electrons at the first inelastic scattering event, leading to inefficient simulations in which a large proportion of the processing time is wasted tracking electrons which do not form part of the useful signal. A considerable amount of work has been devoted to improving on this situation, and these concepts will be covered in more detail in Chapter four.

Figure 2.13 illustrates the complexity of electron scattering, showing 100 electron trajectories modeled with a Monte Carlo simulation. This example illustrates the interaction volume of a 20keV electron beam at normal incidence on a copper sample.

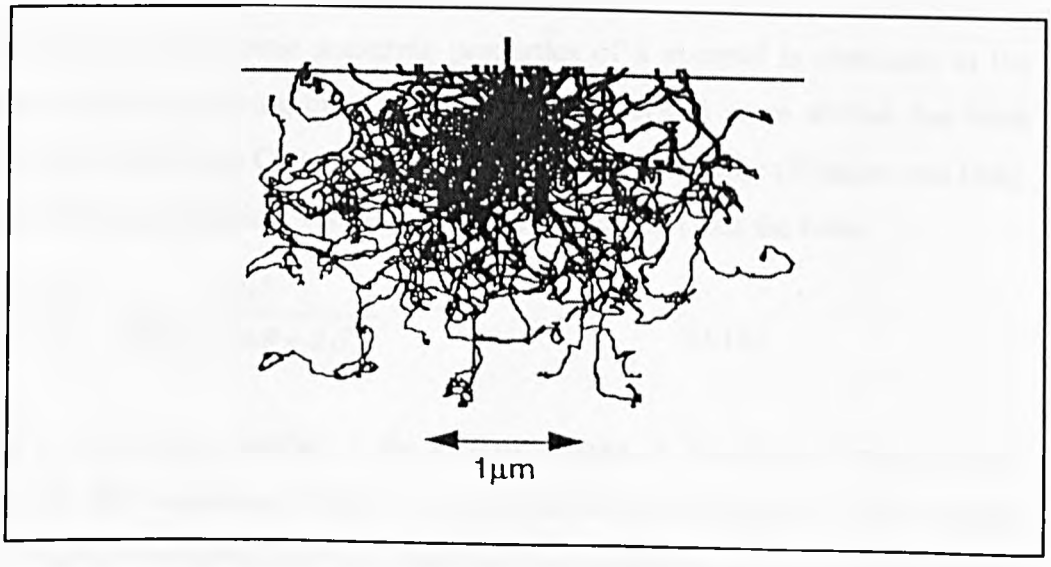

Figure 2.13 : (from Joy, 1988); 100 electron trajectories in bulk copper at 20keV

Scattering events are split into two distinct groups: inelastic and elastic interactions (Joy 1988). Elastic scattering causes an electron to lose no significant amount of energy, but leads to direction changes of between 5° and 180°. Conversely, inelastic events cause the electron to lose energy but result only in a very small direction change (less than 2°). These two types of interaction are assumed to be exclusive, i.e. a scattering event leading to significant energy loss does not result in a direction change and vice-versa.

The path taken by any given electron is determined by the elastic scattering properties of the sample, as described by the elastic scattering cross section. The energy loss between the electron emission and detection must be calculated on the basis of both the inelastic *and* elastic processes. Inelastic interactions actually cause the loss of energy, but elastic processes determine the physical path by which an electron may reach the analyser, and therefore how far an electron travels in the sample between emission and detection.

The ability to carry out accurate simulation of electron trajectories is dependent on the raw parameters used in the algorithm; unrealistic assumptions naturally lead to incorrect results.

2.5.1.3 The Elastic Scattering Cross Section

Information about the elastic scattering properties of a material is contained in the differential elastic scattering cross section. The Rutherford cross section has been used since the first Monte Carlo simulations of electron behaviour (Shimizu and Ding, 1992 and references therein). The Rutherford cross section takes the form

$$\frac{d\sigma}{d\Omega} = \frac{Z^2 e^4}{4E^2(1-\cos\theta+2\beta)^2} \qquad (2.16)$$

where Z is the atomic number, e the electric charge, E the kinetic energy of the electron, θ the scattering angle, and β the screening parameter. This formula, favoured for its simplicity, has been found to be reasonably accurate at high energy and low atomic number. At lower energies, or high atomic number, however, the Rutherford cross section cannot be relied upon, making it unsuitable for calculations

related to AES and XPS since we are often interested in the behaviour of electrons with kinetic energies below 2keV.

The Mott cross section offers a more rigorous approach, and is considerably more accurate than the Rutherford cross section in the low energy regime. Mott (1929) derived the relativistic differential cross section as (Czyzewski *et al.*, 1990)

$$\frac{d\sigma}{d\Omega} = |f(\theta)|^2 + |g(\theta)|^2 + (fg^* - f^*g)\left(\frac{-AB^*e^{i\varphi} + A^*Be^{-i\varphi}}{|A|^2 + |B|^2}\right) \qquad (2.17)$$

with scattering factors

$$f(\theta) = \frac{1}{2iK}\sum_{n=0}^{\infty}\{(n+1)[\exp(2i\eta_{-n-1}) - 1] + n[\exp(2i\eta_n) - 1]\}P_n(\cos\theta) \qquad (2.18)$$

$$g(\theta) = \frac{1}{2iK}\sum_{n=1}^{\infty}[-\exp(2i\eta_{-n-1}) + \exp(2i\eta_n)]P'_n(\cos\theta) \qquad (2.19)$$

where η_{-n} and η_{-n} are the spin 'up' and spin 'down' phase shifts of the n^{th} partial wave, $i = (-1)^{1/2}$, P_n and P'_n are the ordinary and associated Legendre polynomials, respectively, θ is the scattering angle, and A, B and φ describe the degree of spin polarisation of the incoming electron (making the second term in equation 2.17 equal to zero for an unpolarised electron beam). The Mott scattering cross section is used exclusively for the theoretical work in this thesis, full details of these calculations are described elsewhere (Czyzewski *et al.*, 1990).

The elastic mean free path (EMFP), λ_e is the characteristic length relating to the process of elastic scattering. The EMFP is defined as the average distance an electron of a given energy travels between successive elastic scattering events, and can be calculated as follows

$$\lambda_e = \frac{1}{n\sigma_e} \qquad (2.20)$$

where n is the atomic density, and σ_e is the elastic scattering cross section given by

$$\sigma_e = \int_0^{\pi} \frac{d\sigma}{d\Omega} \sin\theta d\theta \qquad (2.21)$$

The transport mean free path λ_{tr} is the scaling length relating to high angle scattering events and is defined as

$$\lambda_{tr} = \frac{1}{n\sigma_{tr}} \qquad (2.22)$$

where n is the atomic density, and σ_{tr} is the transport scattering cross section given by

$$\sigma_{tr} = \int_0^{\pi} \frac{d\sigma}{d\Omega} (1-\cos\theta) \sin\theta d\theta \qquad (2.23)$$

As these results show, calculation of the elastic and transport mean free paths is a simple process of integration, usually carried out numerically from tabulated listings of the differential elastic scattering cross sections.

2.5.1.4 The Inelastic Mean Free Path

The availability of accurate information describing energy loss in electron-solid interaction is essential for accurate quantification. The inelastic mean free path (IMFP), λ_i is defined as the average distance an electron with a given energy travels between two successive inelastic collisions (Powell, 1985, 1986 and 1988). Penn (1976) derived the inelastic mean free path with the following formula

$$\lambda_i(E) = \frac{E}{a(\ln E + b)} \qquad (2.24)$$

where a and b are functions of electron concentration. Penn (1976) presented these two factors in tabulated form for atomic numbers 3-84, allowing calculation of the IMFP in the energy range 200-2400eV.

Tanuma *et al.*(1988, 1990) presented their calculated values of the IMFP based on experimental optical data and a theoretical dielectric function. More recently (Tanuma *et al.*, 1991) presented an updated form of their expression by fitting calculated IMFPs to a modified Bethe loss equation proposed by Inokuti (1971) and Ashley (1988). Their expression for the inelastic mean free path is of the following form

$$\lambda_i = E/\{E_p^{\,2}[\beta \ln(\gamma E) - (C/E) + (D/E^2)]\} \qquad (2.25a)$$

where,

$$\beta = -0.0216 + 0.944/(E_p^{\,2} + E_g^{\,2})^{1/2} + 7.39\times10^{-4}\rho \qquad (2.25b)$$

$$\gamma = 0.191\rho^{-0.5} \qquad (2.25c)$$

$$C = 1.97\text{-}0.91U \qquad (2.25d)$$

$$D = 53.4\text{-}20.8U \qquad (2.25e)$$

$$\mathrm{U} = N_v\rho/\mathrm{M} \qquad (2.25f)$$

where E is the electron energy, E_p is the free-electron plasmon energy, E_g is the band gap energy for non-conductors, ρ is the bulk density, N_v is the number of valence electrons per atom or molecule and M is the atomic or molecular weight. Tanuma *et al.* (1991) presented the inelastic mean free paths calculated by this method for 27 elements and 15 inorganic compounds in the energy range 50-2000eV. The subsequent theoretical work in this thesis is based on these values of λ_i.

2.6 Conclusions

The technique of electron spectroscopy has been described, together with an outline of the Auger process. The three most popular electron energy analysers have been described briefly, together with the instrumentation required for Auger analysis to be carried out. More information about the cylindrical mirror analyser is given in Chapter five.

The technique of angle resolved Auger analysis with a cylindrical mirror analyser has been suggested and existing angle resolving instruments reported in the literature have been reviewed. Suggestions were made for possible experiments which may take advantage of an angle resolving electron energy analyser. This subject is discussed in more detail in Chapter six.

Methods of quantitative analysis have been described, highlighting the importance of the depth distribution function in quantitative analysis. The application of transport theory and Monte Carlo simulation to the modeling of electron-solid interaction has been described. The essential parameters for carrying out such simulations have been identified and reviewed.

CHAPTER THREE

SYSTEMATIC TRENDS IN THE TRANSPORT MEAN FREE PATH WITH ENERGY AND ATOMIC NUMBER

3.0 Introduction

Auger electron spectroscopy (AES) and X-ray photoelectron spectroscopy (XPS) are both highly surface sensitive techniques. This inherent characteristic is due entirely to the very strong inelastic scattering of electrons in the energy range commonly examined under such studies (Powell, 1974; Powell *et al.*, 1994; Seah and Dench, 1979). In the absence of elastic scattering (the so-called straight line approximation) the sampling depth in AES and XPS can be described solely on the basis of the inelastic mean free path λ_i. In this instance the Depth Distribution Function (DDF) is described by a decaying exponential, with λ_i as the decay constant.

Baschenkio and Nefedov (1979) first raised the issue of elastic scattering of photoelectrons in XPS, and since then a large amount of work has been devoted to this topic. In recent studies of electron transport in solids it has been shown that elastic scattering has a significant effect on the angular and depth distribution of signal electrons in both AES and XPS (Jablonski, 1987, 1989; Werner, 1991, 1992). It has been found that a key role is played by the transport mean free path λ_{tr} where

$$\lambda_{tr} = \frac{1}{n\sigma_{tr}} \qquad (3.1)$$

where n is the atomic density (in $Å^{-3}$), and σ_{tr} is the momentum transfer, or transport scattering cross section (in $Å^2$) given by

$$\sigma_{tr} = 2\pi\int_0^{\pi} \frac{d\sigma}{d\Omega}(1-\cos\theta)\sin\theta d\theta \qquad (3.2)$$

where θ is the elastic scattering angle and $d\sigma/d\Omega$ is the differential elastic cross-section (Tougaard and Sigmund, 1982).

As a result of the $(1 - \cos\theta)$ factor in equation 3.2, λ_{tr} represents a mean distance between high angle scattering events proportional to the momentum transfer. In solving the Boltzmann equation, the transport mean free path in conjunction with the inelastic mean free path, λ_i controls electron transport to the first order. Theoretical studies based on this approach led to the development of the so-called generalised radiative field similarity principle by Werner and Tilinin (Werner and Tilinin, 1993; Tilinin and Werner, 1994). This states that electron emission characteristics are not heavily influenced by the shape of the elastic scattering cross section, providing that the angular distribution from the source is sufficiently uniform. Related work produced analytical expressions for the attenuation length and escape probability (Werner and Tilinin, 1992) and the angular and energy distribution of both Auger electrons and photoelectrons (Tilinin and Werner, 1993).

Other studies have exploited transport theory to study the depth distribution function (Dwyer and Richards, 1992; Dwyer, 1994b), to produce inelastic mean free path data from backscattering data (Dwyer, 1994c) and to investigate the effects of elastic backscattering on AES and XPS spectra (Dwyer and Matthew, 1984). In addition to this, Jablonski and Tilinin (1995) used λ_{tr} and χ ($\chi = \lambda_i/\lambda_{tr}$) to generalise results from Monte Carlo approaches to the transport problem.

A recent study by Jablonski and Tougaard (1998) defines a new parameter, the so-called correction factor (CF), to account for the effect of elastic scattering in AES and XPS calculations. This parameter was shown to depend only on the transport and inelastic mean free paths and give good results irrespective of electron energy, atomic number and anisotropy of emission. In general, this relatively simple methodology can only be applied to homogeneous sample structures. Chapter four describes a Monte Carlo method for performing elastic scattering corrections in more complex heterogeneous samples.

From the well known Born approximation, it can be readily shown that $\sigma_{tr} \propto \ln\varepsilon/\varepsilon^2$ where ε is a reduced energy related to the true energy E (in a.u.; 1a.u. = 27.2eV) by $\varepsilon = 0.885\ E a_0/e^2 Z^{4/3}$, where Z is the atomic number, a_0 is the Bohr radius and e the electron charge (Tilinin, 1988). In fact, this approximation is only valid at high energy and Tilinin (1988) identified an alternative medium energy region where the transport cross section for elastic scattering of electrons is a function of the reduced particle energy, giving $\sigma_{tr} \propto \varepsilon^{-1}$ ($\varepsilon < 1$). This work was based on the Thomas-Fermi atomic description and was successfully validated by comparison with experimental data and other established analytical results.

Typical experimental conditions for both AES and XPS include energy ranges which may cover both the high energy Born-approximation region and the Tilinin medium energy regime. This will lead to strongly energy dependant spectral characteristics resulting from variations in electron transport, and subtleties of atomic structure which are not taken into account in the Thomas-Fermi model may play a part.

As mentioned above, χ, the ratio of inelastic to transport mean free path, is a useful parameter in the study of electron transport as a measure of elastic scattering effects. Since inelastic cross sections typically vary as $\varepsilon^{-1}\ln\varepsilon$, at medium energies where $\sigma_{tr} \propto \varepsilon^{-1}$ (Tilinin, 1988) there may be regions where $\chi \propto \ln\varepsilon$, followed by a slow energy variation in some intermediate energy range and $\chi \propto \varepsilon^{-1}$ at high energies where the Born approximation is valid. Werner and Tilinin (1994) found that χ was reasonably insensitive to energy for Ag and Au, but the work presented in this chapter represents a systematic study of the variations in λ_{tr} and χ with atomic number and energy. The following results are based on the well accepted elastic scattering data of Czyzewski et al. (1990), who produced a comprehensive database of differential cross sections for 94 elements across the periodic table and over an energy range of 20eV to 30keV.

The results in this chapter highlight trends in λ_{tr}, χ and λ_{tr}/λ_e at low atomic number and adjacent to the three transition metal series which can be related to the simple scaling ideas of Tilinin. The conclusions drawn from this work support the findings of Cumpson (1997) who presented a simplified quantum mechanical scattering model to explain anomalies in the energy dependence of electron transport behaviour,

particularly at high atomic number and low energy (100-400eV). The results of this chapter complement those of Jablonski and Tilinin (1995), who used λ_{tr} for specific XPS transitions with Al Kα and Mg Kα sources in their study of the effects of elastic scattering on XPS intensities. They presented a comparison of transport theory with Monte Carlo modeling of electron behaviour and found good agreement.

A comparison is also made between the mean free paths used here (obtained from the Mott cross sections of Czyzewski et al.), the recent work of Jablonski (1996) and, where available, results derived from the semi-classical calculations of Tilinin (1992) and Tilinin and Werner (1994). The general trends of these calculations agree well with the available data of Tilinin and Werner, and also with the work of Jablonski at low atomic number. At high Z differences in detail become apparent below 400eV, and significant differences in the variation of λ_{tr} with energy are revealed in the third transition series.

3.1 Trends in Scattering Cross-Sections and Atomic Density

It is well known that the variation of atomic density, n, with atomic number strongly influences the inelastic mean free path, and through the $\lambda_{tr} = 1/n\sigma_{tr}$ relationship, n also directly affects the transport mean free path. Figure 3.1a compares the variation of σ_{tr} and n with atomic number at an energy of 100eV for atomic numbers Z=20 (Ca) to Z=29 (Cu). We find that both these parameters exhibit strong Z dependence at this energy, and although the gradual reduction in σ_{tr} across the series partially compensates for the density effect, we still see a systematic variation in λ_{tr} with atomic number (see Figure 3.2b).

At higher energy (see Figure 3.1b), σ_{tr} does not show a strong Z dependence, and variations in λ_{tr} are largely controlled by atomic density under these conditions. Figure 3.2 shows the variation in n and σ_{tr} over a much wider atomic number range, complementing the results of Figure 3.1. At higher energy and for atomic numbers less than 57, we find that, although σ_{tr} varies monotonically, variations in atomic density will act against a simple Z dependence in λ_{tr}. An understanding of the interplay of n and σ_{tr} is therefore important in explaining trends in λ_{tr} with E and Z.

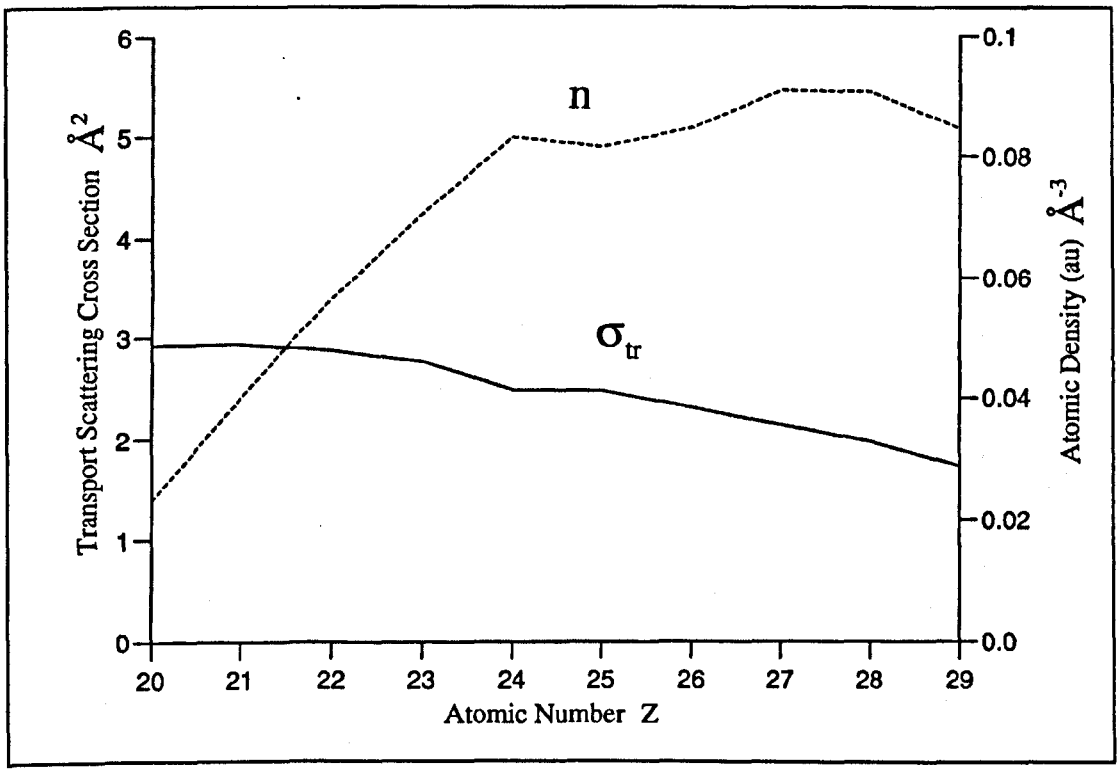

Figure 3.1a : Comparison of transport scattering cross section and atomic density (dashed line) at 100eV, for atomic numbers between 20 and 29

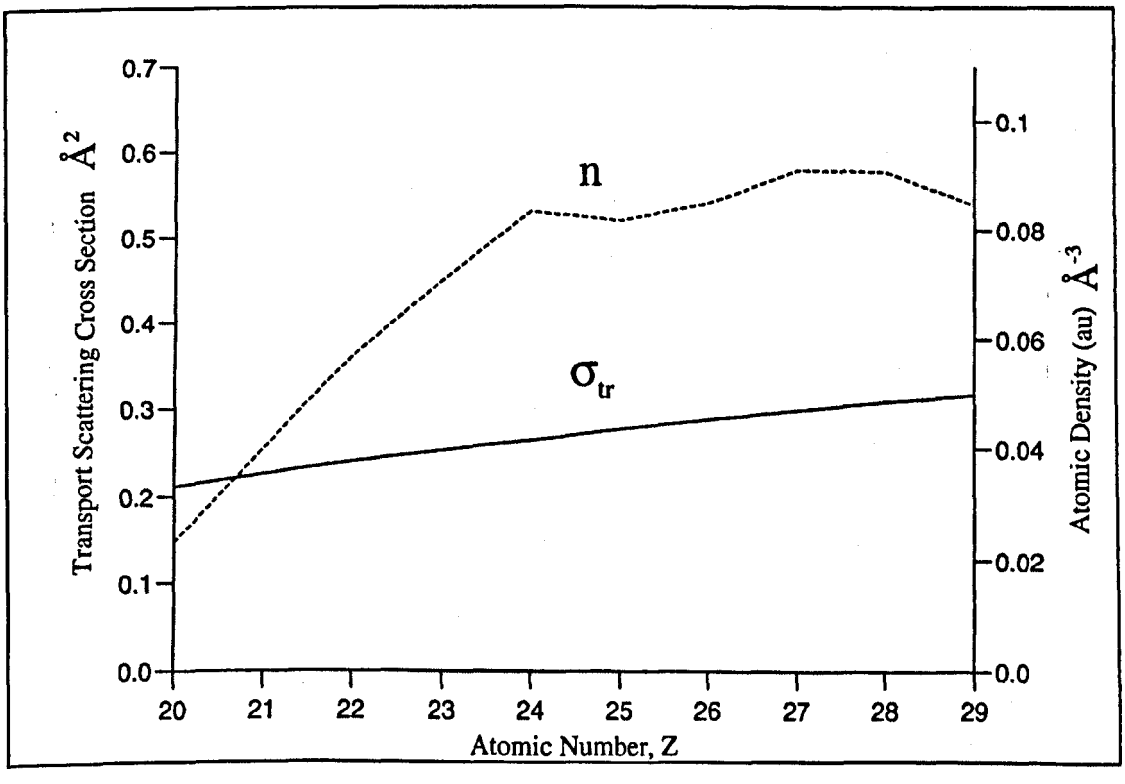

Figure 3.1b : Comparison of the variation in transport scattering cross section and atomic density as shown in 3.1a, but for 1000eV

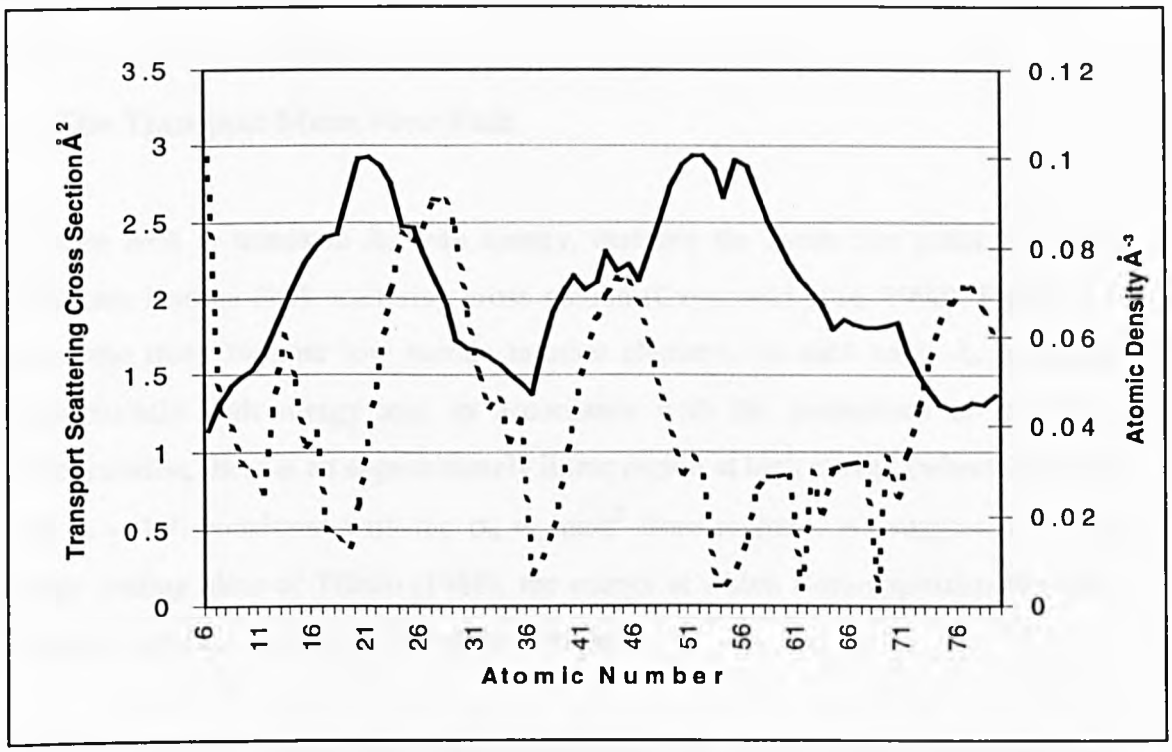

Figure 3.2a : Comparison of the variation in transport scattering cross section and atomic density (dashed line) at 100eV, for atomic numbers between 6 and 79

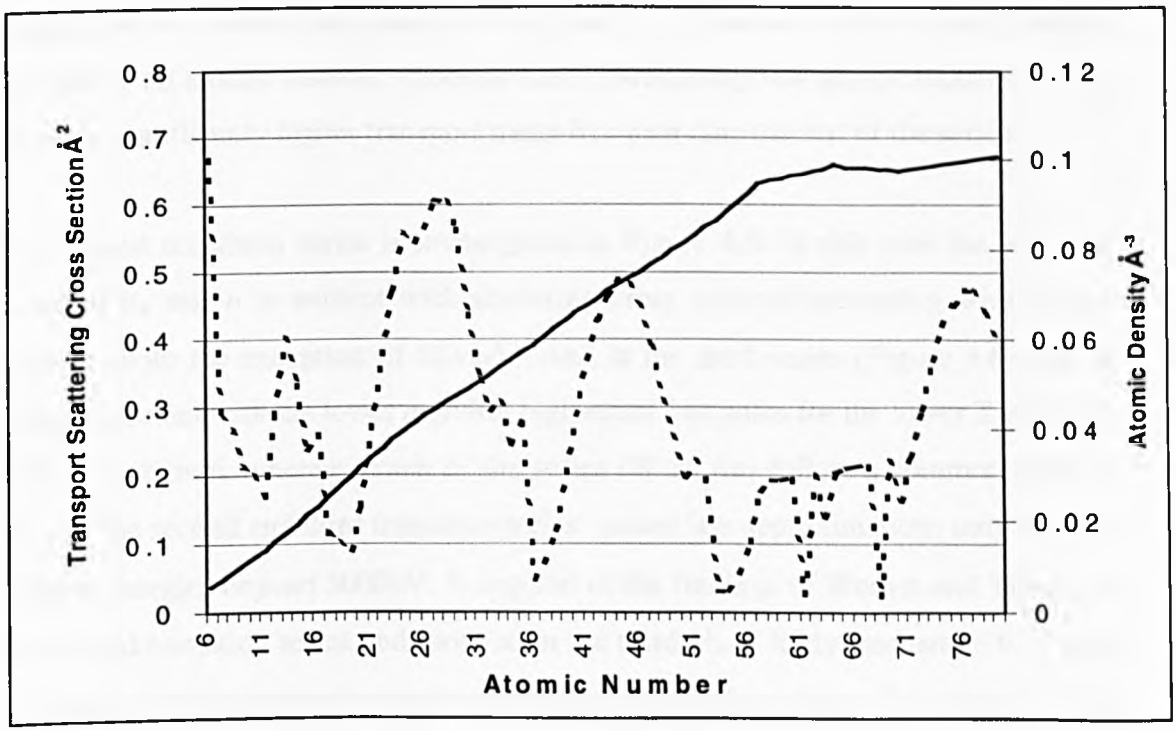

Figure 3.2b : Comparison of the variation in transport scattering cross section and atomic density (dashed line) as shown in 3.2a, but for 1000eV

3.2 The Transport Mean Free Path

We now look at trends in λ_{tr} with energy, deriving the mean free paths from the relativistic Hartree-Fock scattering cross section (Czyzewski et al. 1990). Figure 3.3 shows the trend for four low atomic number elements: in each case, λ_{tr} increases monotonically with energy and, in accordance with the predictions of the Born approximation, there is an approximately linear region at high energy (where $\lambda_{tr} \propto E^m$ with $m \approx 1.7$, consistent with the $\sigma_{tr} \propto \ln\varepsilon/\varepsilon^2$ Born regime). As suggested by the energy scaling ideas of Tilinin (1988), the energy at which Born-approximation-like behaviour begins, increases with atomic number.

Figure 3.4 plots λ_{tr} for the first transition metal series: for this atomic number range the transport scattering cross section is relatively insensitive to atomic number, giving trends in λ_{tr} largely dominated by the atomic density. This is particularly clear at high energy, where power law characteristics set in at between 400eV and 1000eV depending on atomic number. Calcium has a particularly low atomic density, giving rise to a significantly higher transport mean free path than the rest of the series.

The second transition series is investigated in Figure 3.5: in this case the interplay between σ_{tr} and n is evident with scattering cross sections increasing with atomic number (with the exception of Silver). And in the third series (Figure 3.6), this is again important with the lower n giving high mean free paths for the lower Z elements such as Lutetium, whereas much of the series (W to Au) follow a common trend in λ_{tr}. For the second and third transition series' power law approximations only become valid at energies beyond 3000eV. In support of the findings of Werner and Tilinin, in the second transition series and more so in the third, λ_{tr} is fairly insensitive to E over the typical range of energies used in XPS and AES studies ($200 < E < 1000$eV). The maximum, at around 200eV for the higher atomic number elements, can be attributed to multiple inter-atomic elastic scattering in the larger atoms which lead to interference patterns (Cumpson, 1997).

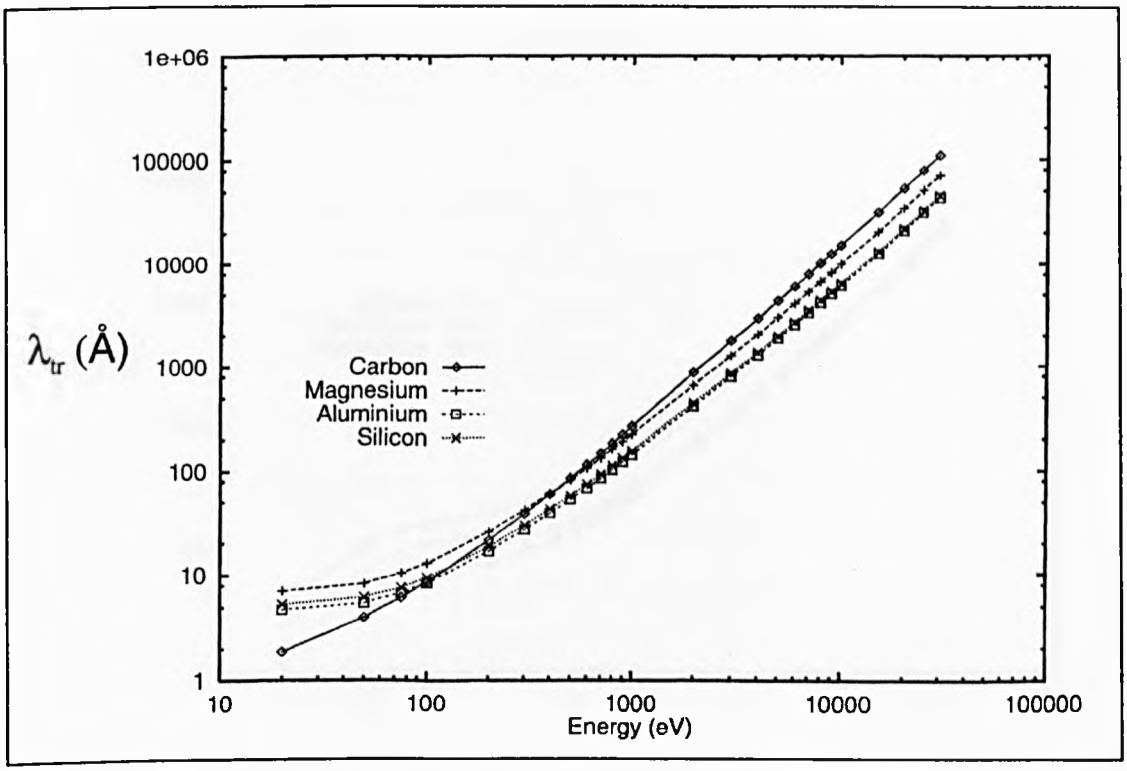

Figure 3.3 : Trends in the transport mean free path with energy for C, Mg, Al and Si, as calculated using a relativistic Hartree-Fock cross-section

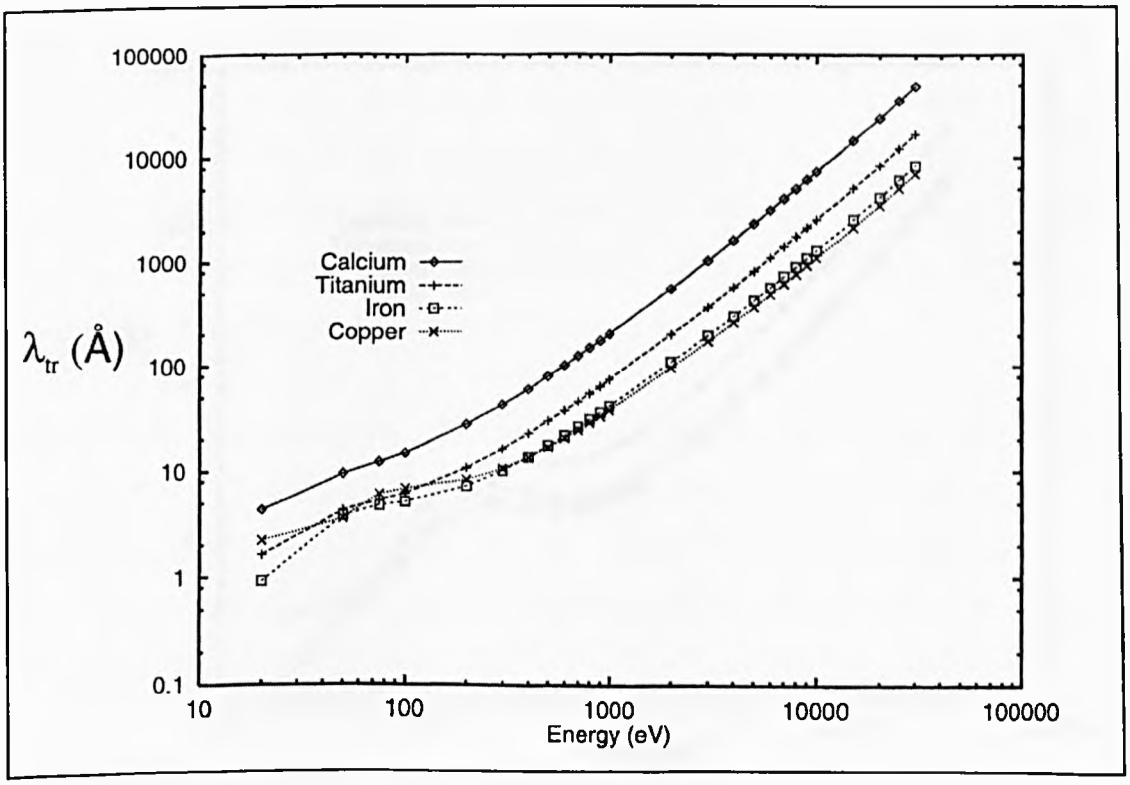

Figure 3.4: Trends in the transport mean free path with energy for C, Ti, Fe, Cu

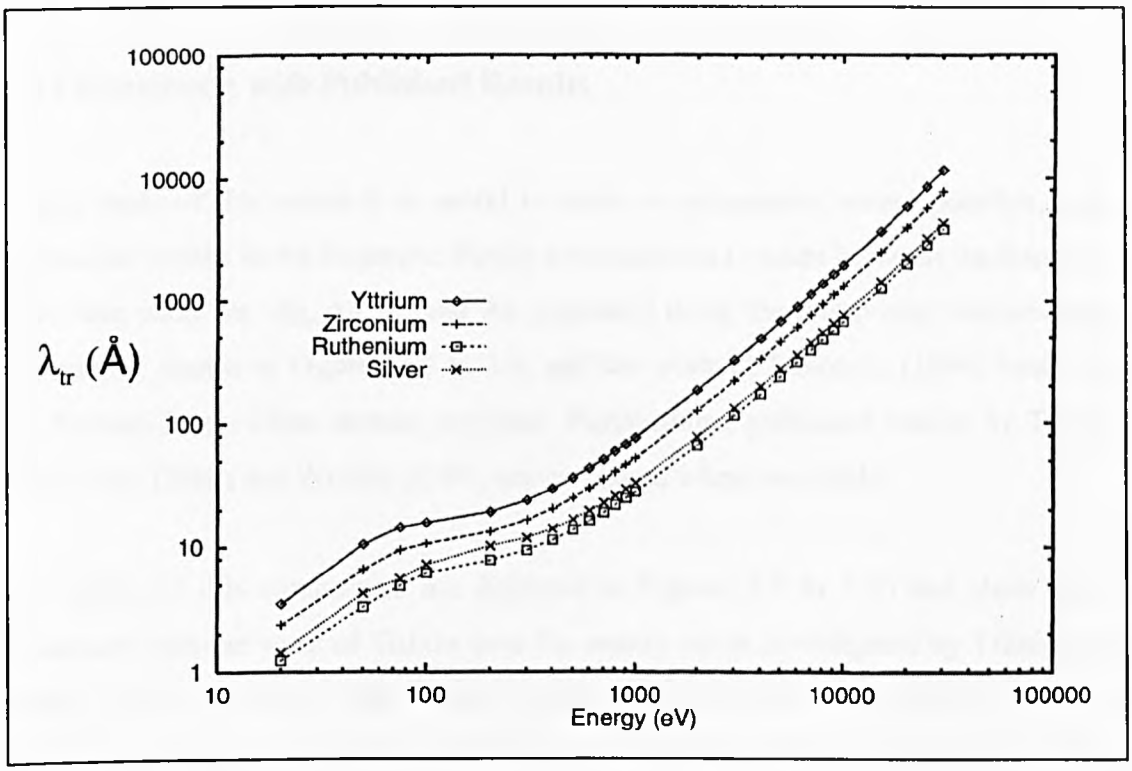

Figure 3.5: Trends in the transport mean free path with energy for Y, Zr, Ru, Ag

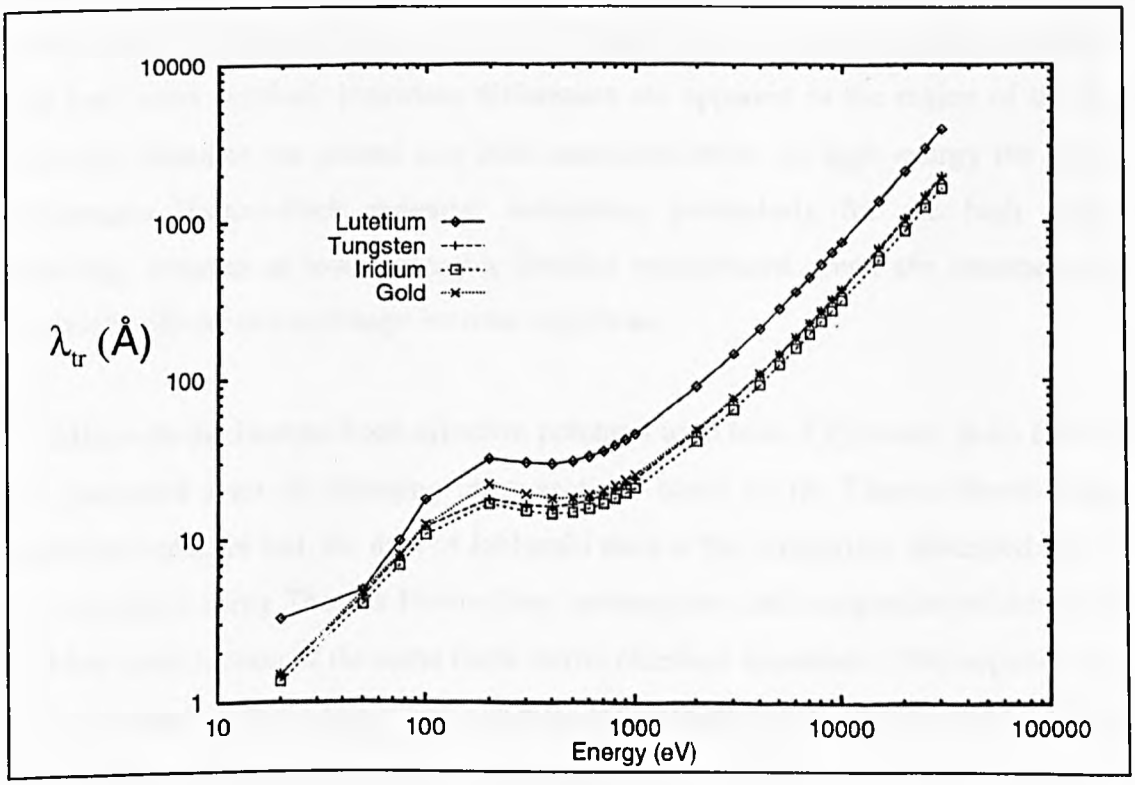

Figure 3.6: Trends in the transport mean free path with energy for Lu, W, Ir, Au

3.3 Comparison with Published Results

In any study of this nature it is useful to make a comparison, where possible, with established results in the literature. Firstly a comparison is made between the transport mean free paths for Mg, Ag, W and Au generated using the relativistic Hartree-Fock potential as shown in Figures 3.3 to 3.6, and the work of Jablonski (1996) based on the Thomas-Fermi-Dirac atomic potential. Furthermore, published results by Tilinin (1992) and Tilinin and Werner (1994) are compared where available.

The results of this comparison are depicted in Figures 3.7 to 3.10 and show good agreement with the work of Tilinin over the energy range investigated by Tilinin and Werner for Ag and Au. The results presented here are in accord with those of Jablonski at low atomic number with Figure 3.7 showing discrepancies of less than one percent in the results for Magnesium. However, at higher atomic number, significant differences become apparent which are particularly evident at low energy (below 400eV). Although the general energy trends are common to λ_{tr} plots generated with both cross sections, important differences are apparent in the region of the λ_{tr} maximum found in the second and third transition series. At high energy the quasi electrostatic Hartree-Fock potential dominates, particularly for the high angle scattering, whereas at lower energies detailed assumptions about the treatment of relativistic effects and exchange become significant.

In addition to the Hartree-Fock effective potential used here, Czyzewski et al. (1990) also presented a set of scattering cross sections based on the Thomas-Fermi-Dirac atomic potential. In fact, the data of Jablonski used in the comparison described above was calculated using Thomas-Fermi-Dirac assumptions and comparison of this with the Mott cross section of the same basis shows excellent agreement. This supports the work presented in this chapter and validates the numerical procedures used to derive the mean free paths from the cross section data.

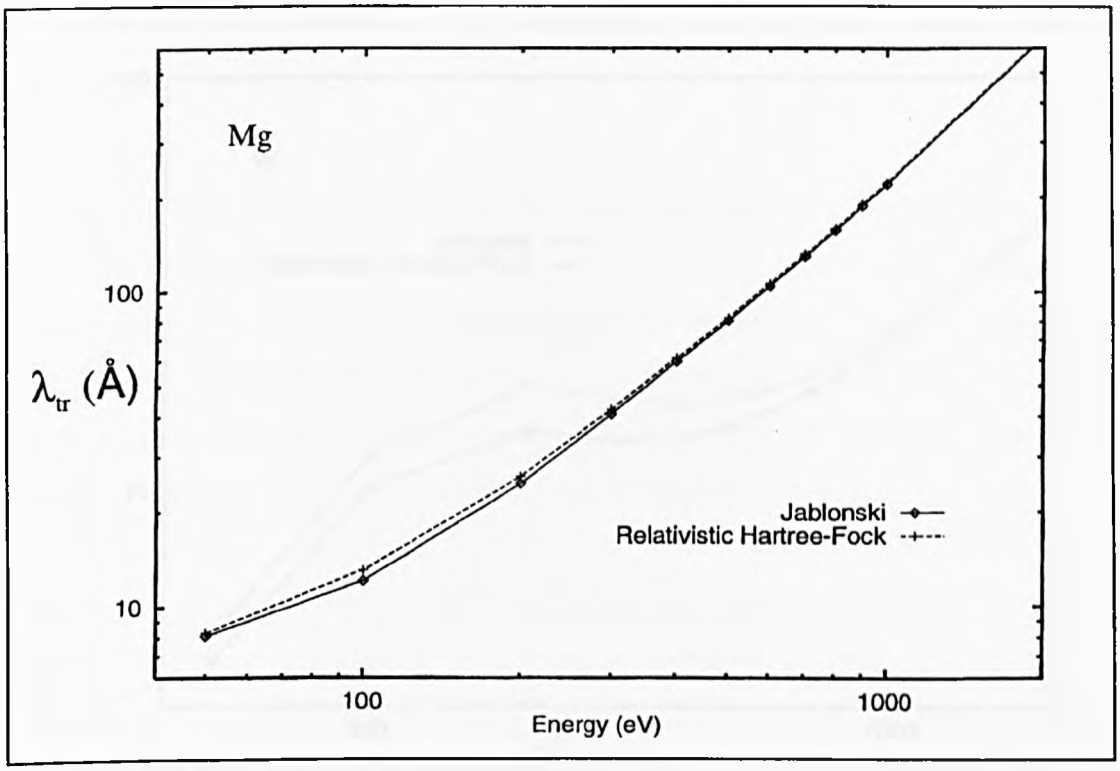

Figure 3.7: Comparison of λ_{tr} based on the relativistic Hartree-Fock potential with results of Jablonski (1996) and Tilinin and Werner (1994) for Mg

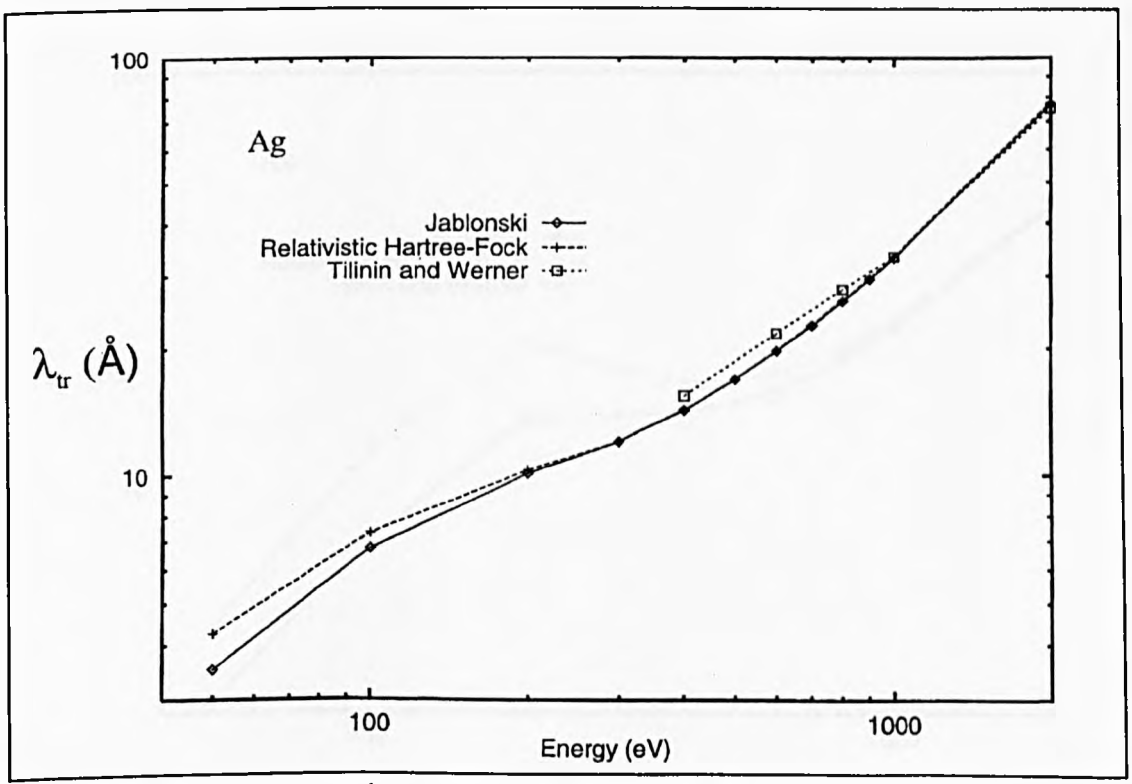

Figure 3.8: Comparison of λ_{tr} based on the relativistic Hartree-Fock potential with results of Jablonski (1996) and Tilinin and Werner (1994) for Ag

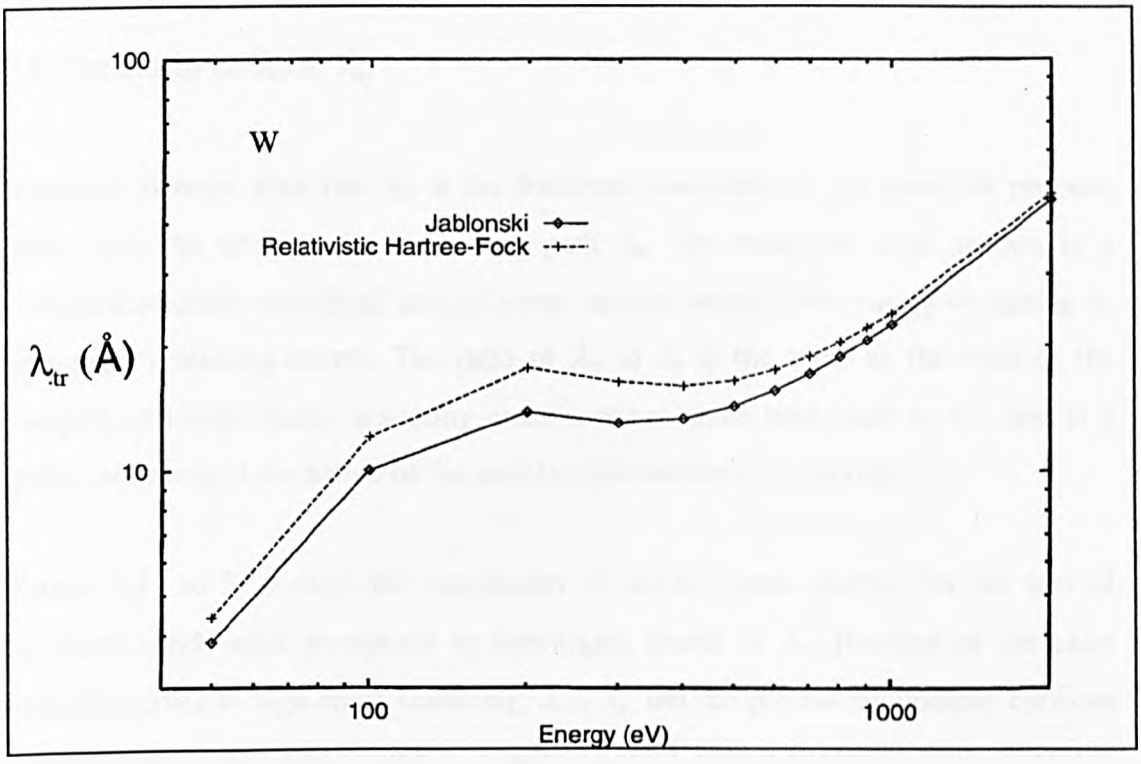

Figure 3.9: Comparison of λ_{tr} based on the relativistic Hartree-Fock potential with results of Jablonski (1996) and Tilinin and Werner (1994) for W

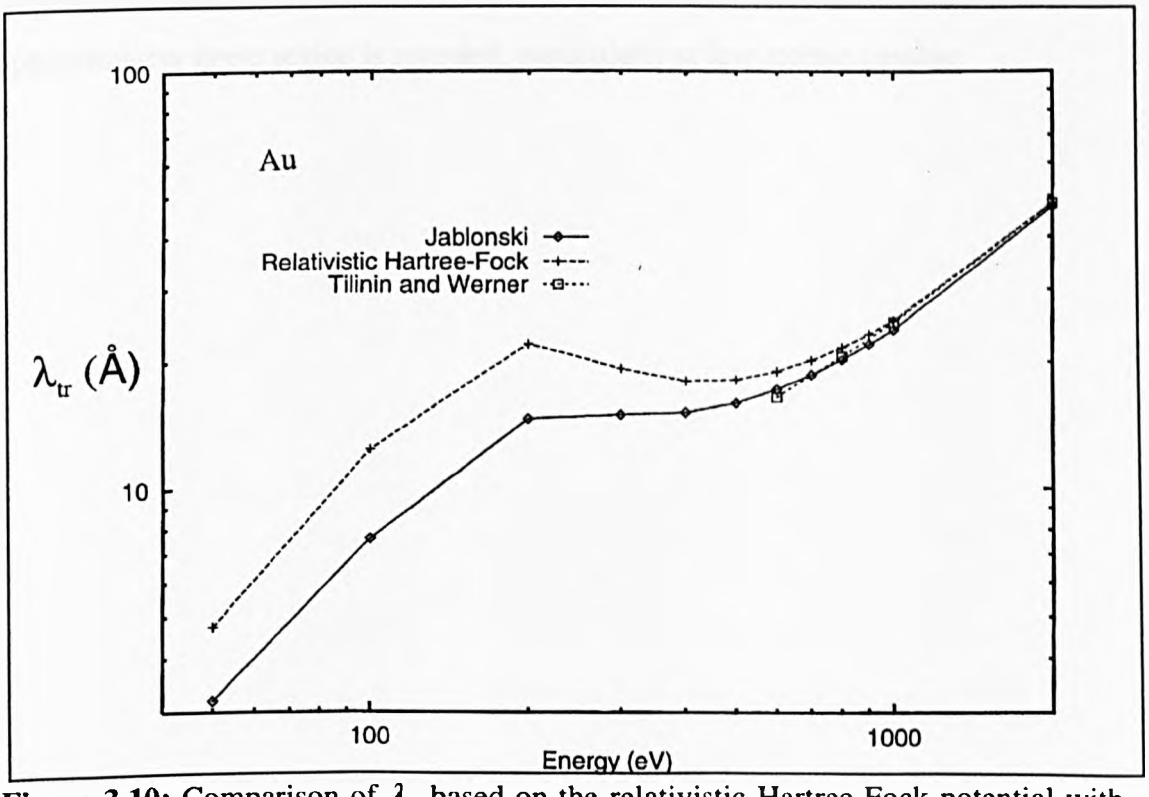

Figure 3.10: Comparison of λ_{tr} based on the relativistic Hartree-Fock potential with results of Jablonski (1996) and Tilinin and Werner (1994) for Ag

3.4 The Ratio of λ_{tr} to λ_e.

Transport theories state that λ_{tr} is the dominant controller of the transport process, rather than the total elastic mean free path λ_e. The transport cross section is a momentum-transfer-weighted integral cross section which gives strong weighting to high-angle scattering events. The ratio of λ_{tr} to λ_e is the same as the ratio of the transport and total elastic scattering cross sections since both scale as n^{-1}, and is a useful indication of the nature of the angular distribution of scattering.

Figures 3.11 to 3.14 show the correlation of λ_{tr}/λ_e versus energy, for the sets of elemental solids used previously to investigate trends in λ_{tr}. Because of the extra weighting given to high angle scattering, $\lambda_{tr} > \lambda_e$, but the precise relationship between the two varies with atomic number in quite a complex way according to the structure of the scattering cross section at a given energy. Furthermore, the relationship between λ_{tr} and λ_e shows a strong energy dependence, although at high energy an approximately linear region is revealed, particularly at low atomic number.

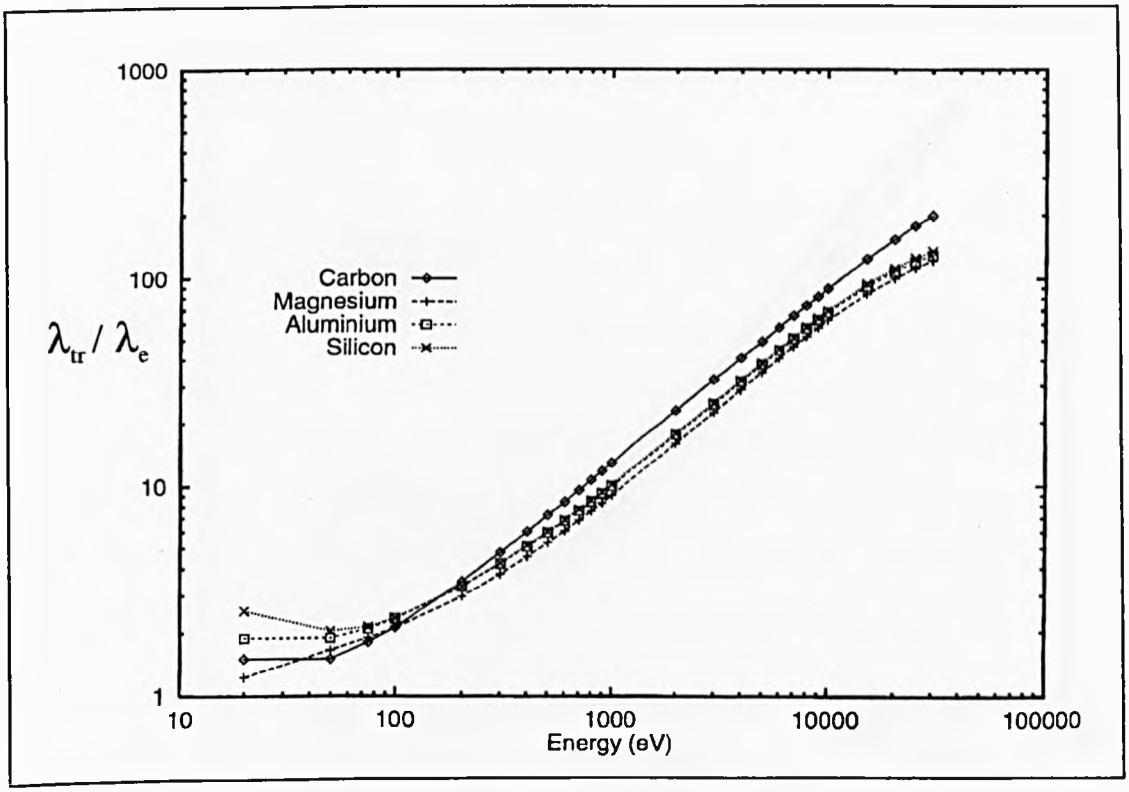

Figure 3.11 : The energy dependence of λ_{tr}/λ_e for C, Mg, Al and Si

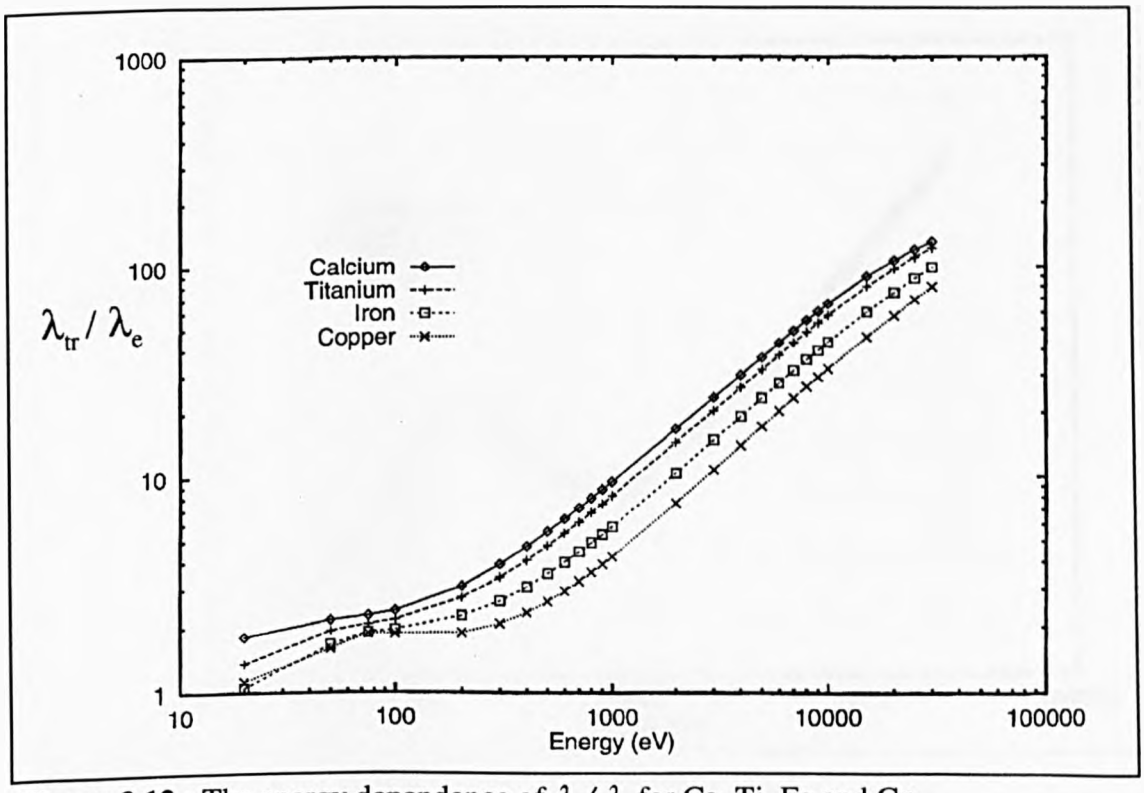

Figure 3.12 : The energy dependence of λ_{tr}/λ_e for Ca, Ti, Fe and Cu

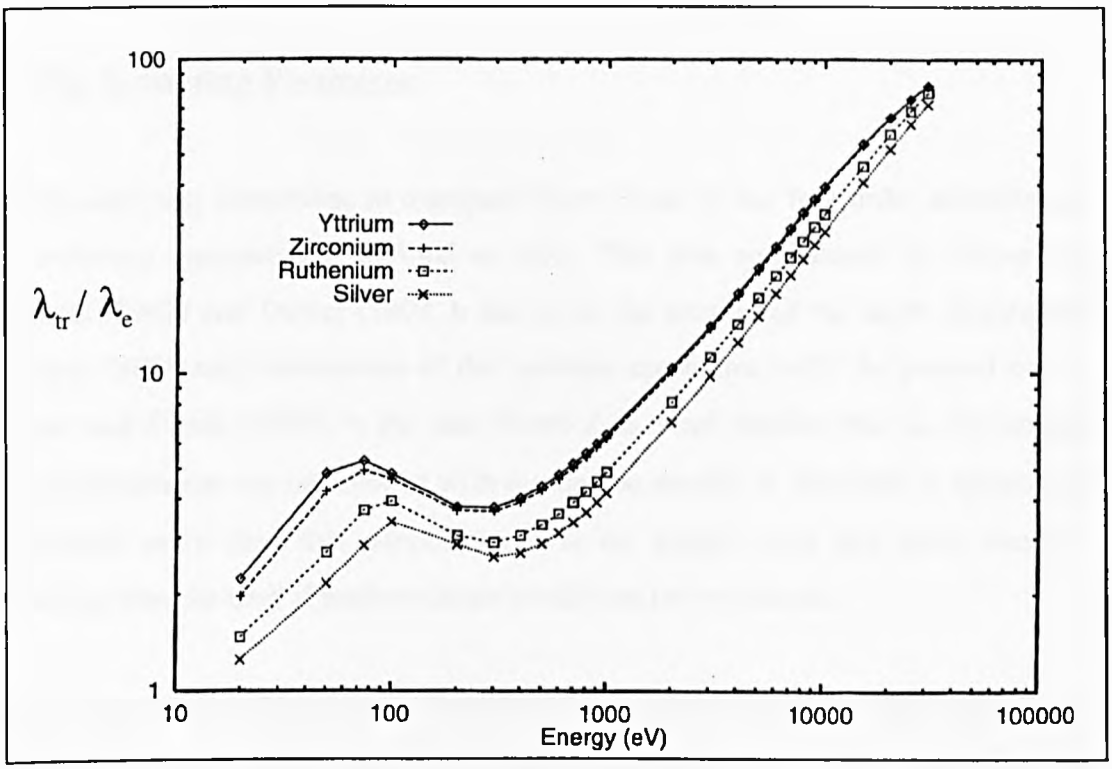

Figure 3.13 : The energy dependence of λ_{tr}/λ_e for Y, Zr, Ru and Ag

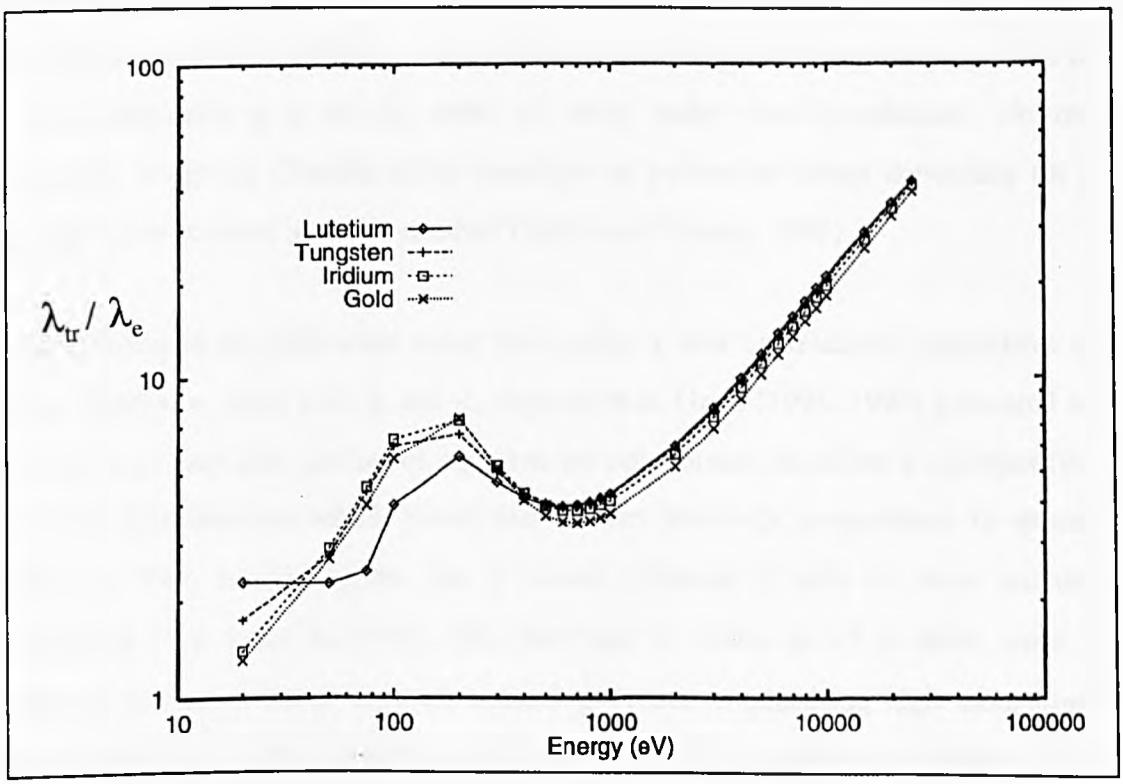

Figure 3.14 : The energy dependence of λ_{tr}/λ_e for Lu, W, Ir and Ag

3.5 The Scattering Parameter.

Elastic scattering corrections in transport theory scale to the first order according to the scattering parameter, χ, defined as λ_i/λ_{tr}. This was emphasised by Dwyer and Richards (1992) and Dwyer (1994, b and c) in the context of the depth distribution function (DDF) and calculations of the inelastic mean free path. As pointed out by Werner and Tilinin (1994), in the case where λ_i is much smaller than λ_{tr} the straight line approximation can be invoked with reasonable results. If, however, χ approaches or exceeds unity then this simplification is no longer valid and more complex modelling must be used if more accurate results are to be obtained.

From the results of Jablonski and Tilinin (1995) it can be shown that for small values of χ, and high exit angles, the XPS yield scaling factor $Q_x \approx 1-\chi/2$, while β_{eff}, the parameter controlling the angular distribution of emission, is related to the true atomic β value by $\beta_{eff}/\beta \approx 1-\chi$. In practice, over the range of energies of relevance for AES and XPS analysis (200-2000eV), the χ values are too high for these simple results to be valid (typically χ is of the order of unity under these conditions). Further corrections involving Chandrasekhar functions at parameter values depending on χ are required to produce accurate results (Tilinin and Werner, 1993).

In comparison to the individual mean free paths, χ will be relatively insensitive to atomic density, n, since both λ_i and λ_{tr} depend on n. Gries (1995, 1996) presented an inelastic mean free path predictive equation for compounds, based on a superposition of atomic contributions which stated that λ_i was inversely proportional to atomic density, n. This would suggest that χ would represent a ratio of cross sections independent of n since $\lambda_{tr}=1/n\sigma_{tr}$. The fact that λ_{tr} scales as n^{-1} is clear, since it represents a mean distance between scattering events, emphasising high momentum transfer, implying atomic scattering processes involving significant penetration into the atomic core. In contrast, a significant contribution to the length λ_i comes from partially collective excitations of delocalised charge distributions. For this reason, λ_i does not necessarily scale as n^{-1} and in fact, the predicted λ_i values of Tanuma *et al.* (1991) are not consistently proportional to the inverse of the atomic density,

suggesting that in practice, χ will show at least a weak dependence on n. More recently, Tanuma *et al.* (1997) have presented an alternative approach to estimating the inelastic mean free path which avoids the need to invoke Gries' simplification.

Figures 3.15 to 3.18 illustrate the energy trends in χ based on the Hartree-Fock elastic scattering cross section (Czyzewski *et al.*, 1992) and the inelastic scattering data of Tanuma *et al.*, (1991). Figure 3.15 reveals a monotonically decreasing trend in χ with increasing energy at low atomic number (with $\chi(E) \approx E^{-0.9}$ close to the prediction of the Born approximation where $\chi(E) \approx E^{-1}$). Al, Mg and Si do not conform to this trend and decrease more slowly with energy without obeying simple power law behaviour.

At low energy (below 100eV), carbon exhibits a larger χ than higher atomic number elements through enhanced elastic scattering effects. As energy increases, however, the increasing nuclear charge leads to enhanced high-angle scattering and a lower χ for C, in qualitative (if not quantitative) agreement with Rutherford-like scattering behaviour. The atomic number dependence of χ between Mg to Si is reasonably constant over the energy range investigated here. Figure 3.16 shows energy trends of χ for the first transition series, revealing an energy range in the region of 100-200eV in which χ varies slowly before an approximate power law trend sets in (with $\chi(E) \approx E^{-0.6}$). It is interesting to note that at energies above approximately 300eV, Cu has consistently higher χ values than the transition metals of lower Z on the same plot.

In the second and third series (as illustrated by Figures 3.17 and 3.18) there is no simple Z dependence across the energy range examined here. In the second series, Ag breaks the trend of increasing χ with atomic number, falling between Zr and Ru at energies above 200eV. Similarly Au in the third series exhibits lower χ values than W or Ir for the same energy range. The ledge behaviour observed at low energy for the higher Z members of the first transition series is further developed across the second series, while in the third series a minimum in χ emerges at approximately 200eV. These minima are followed by a broad maximum in the region of 800eV, although over the energy range 800-2000eV, χ is constant to within a few percent for the W, Ir and Au as shown in Figure 3.18.

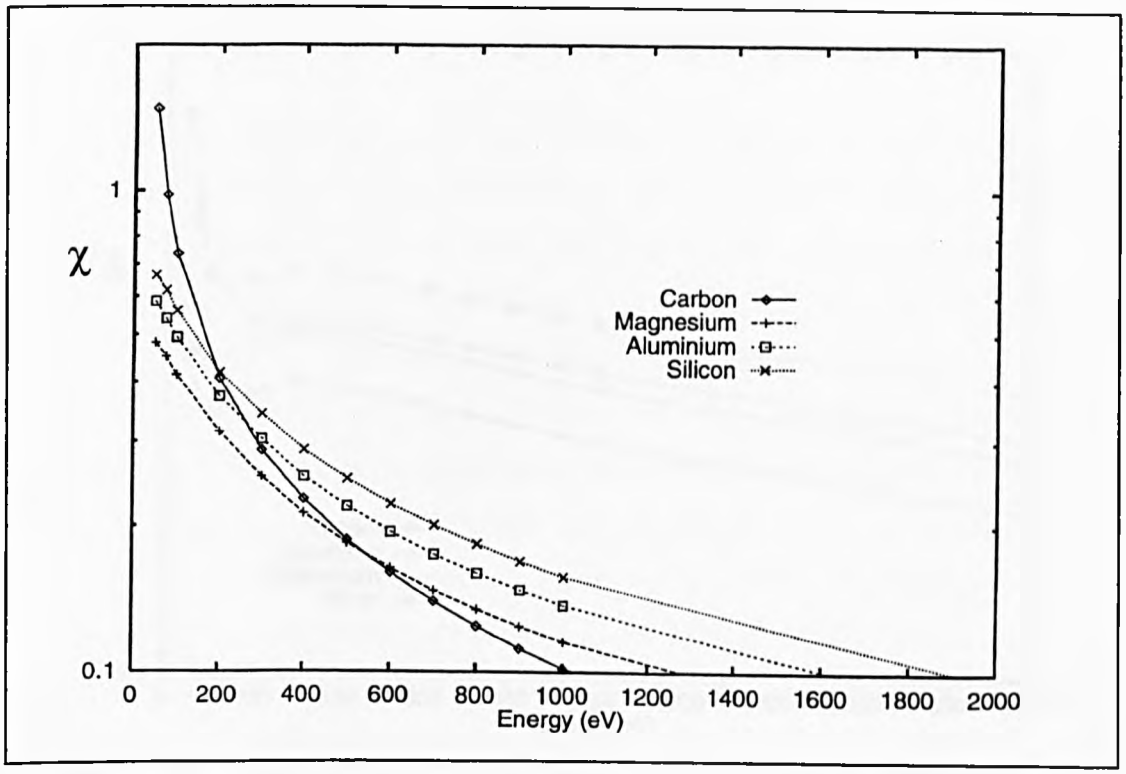

Figure 3.15 : The energy dependence of $\chi=\lambda_i/\lambda_{tr}$, using the inelastic mean free path data of Tanuma et al. (1991) for C, Mg, Al and Si

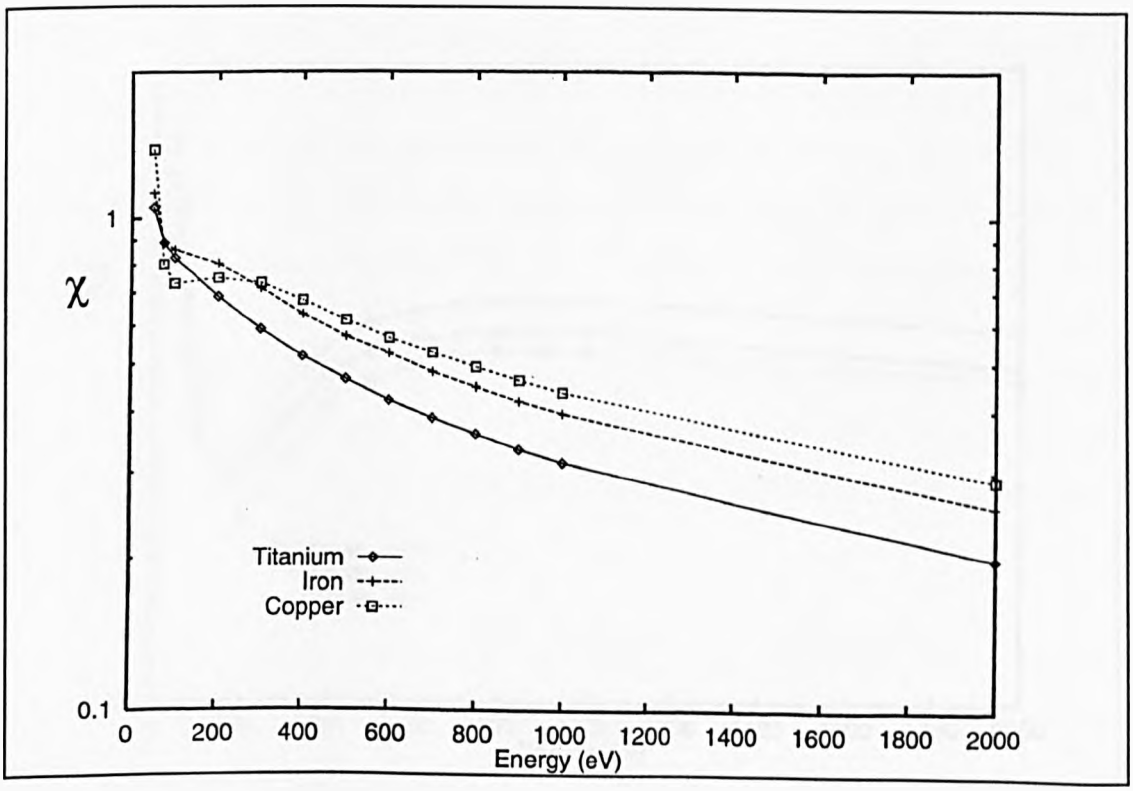

Figure 3.16 : The energy dependence of $\chi=\lambda_i/\lambda_{tr}$, using the inelastic mean free path data of Tanuma et al. (1991) for Ti, Fe and Cu

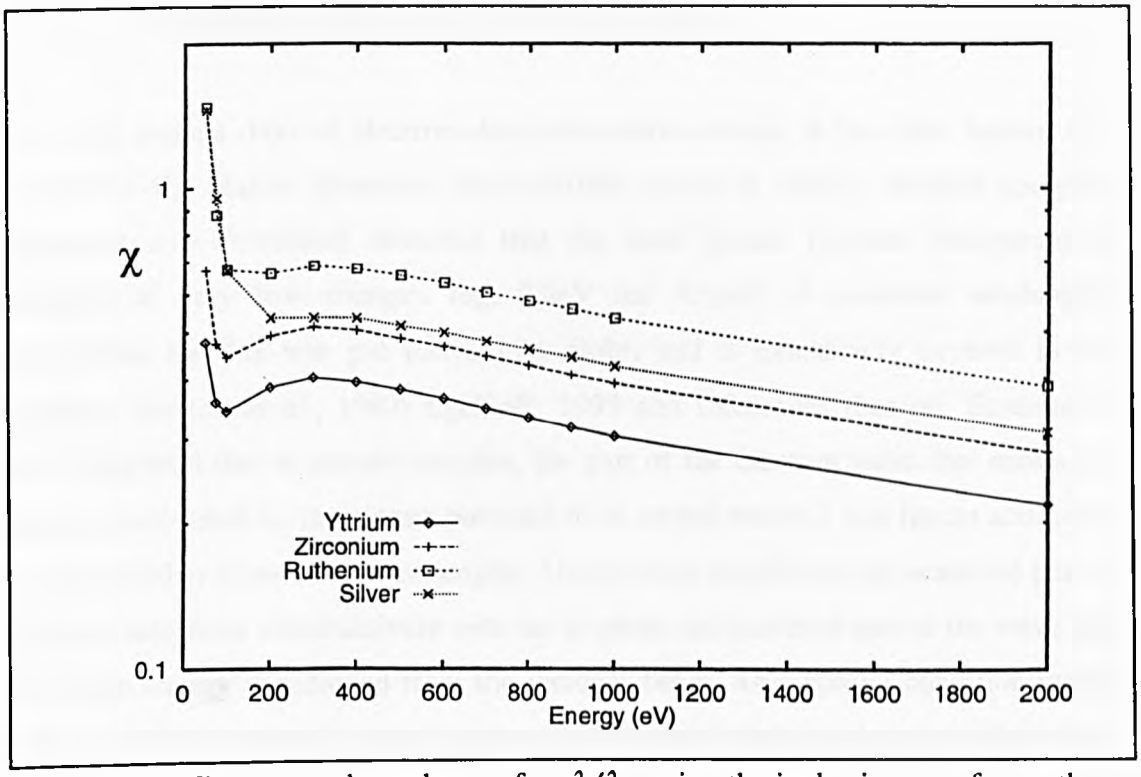

Figure 3.17 : The energy dependence of $\chi=\lambda_i/\lambda_{tr}$, using the inelastic mean free path data of Tanuma et al. (1991) for Y, Zr, Ru and Ag

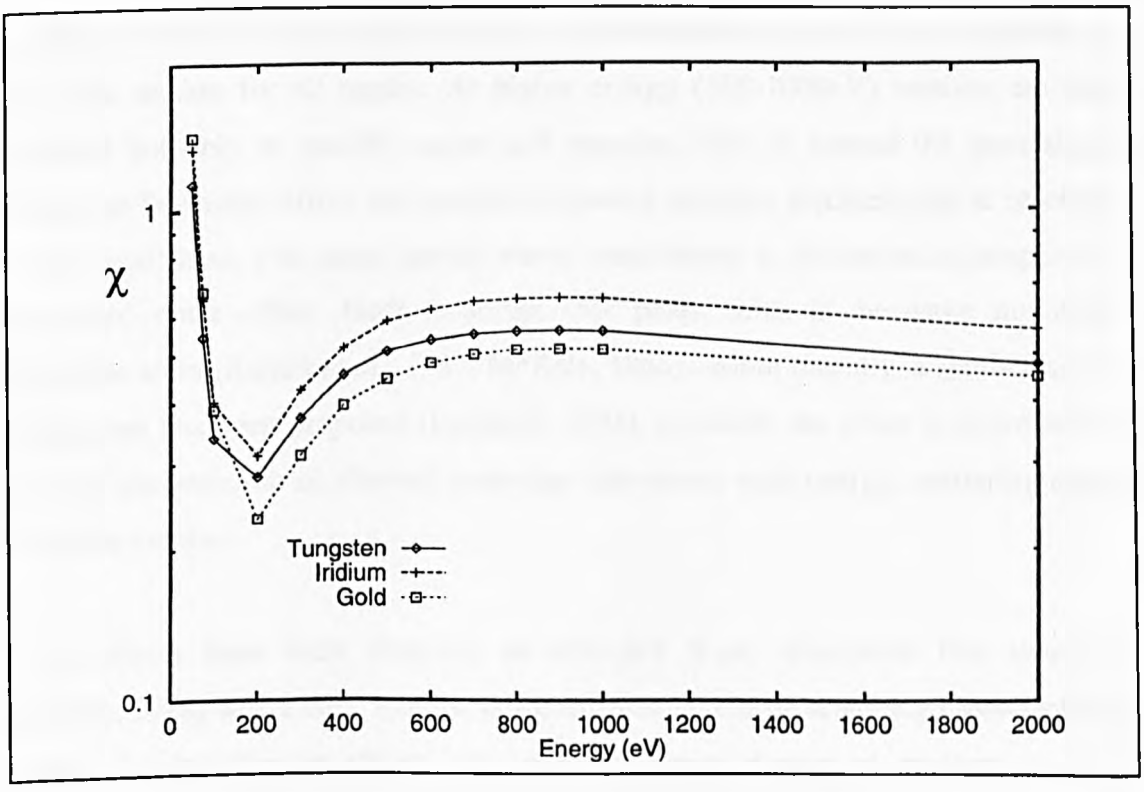

Figure 3.18 : The energy dependence of $\chi=\lambda_i/\lambda_{tr}$, using the inelastic mean free path data of Tanuma et al. (1991) for W, Ir and Au

3.6 The Generalised Ramsauer-Townsend Effect.

Since the earliest days of electron-atom interaction studies, it has been known that minima in the elastic scattering cross-sections occur at certain electron energies. Ramsauer and Townsend observed that the inert gasses become transparent to electrons at very low energies (eg. 0.8eV for Argon). A quantum mechanical explanation for this was put forward by Bohr, and is extensively covered in the literature (Barton *at al.*, 1987; Egelhoff, 1993 and references therein). Essentially, Bohr suggested that at certain energies, the part of the electron wave that enters the atom is accelerated by the atomic potential to an extent where it just fits an additional integral number of electron wavelengths. Under these conditions, the scattered part of the wave interferes constructively with the in-phase un-scattered part of the wave and almost no energy is removed from the incident beam. This special condition would lead to an elastic scattering cross section of almost zero and therefore mean free paths tending towards infinity.

At these very low energies, the Ramsauer-Townsend effect leads to zero amplitude in the cross section for all angles. At higher energy (100-1000eV) minima are also observed but only at specific angles and energies. This is termed the generalised Ramsauer-Townsend effect and can be interpreted quantum mechanically as resonant internal scattering, with many partial waves contributing to the scattering amplitude. This interference effect leads to abrupt 180° phase shifts in the wave amplitude within the atom (Barton *et al.*, 1987; McKale, 1988). More recently, a semi-classical explanation has been proposed (Egelhoff, 1993), in which the effect is described in terms of the variation of allowed scattering trajectories with energy, scattering angle or atomic number.

These effects have been observed in extended X-ray absorption fine structure (McKale, 1988) where deep minima in the differential elastic scattering cross-sections modify the interference effects over specific energy ranges at medium to high energies. Furthermore, Rehr (1994) used an analogy to the Ramsauer-Townsend effect to explain how resonant scattering leading to large oscillations in the atomic X-

ray-absorption fine structure, can produce the dominant contribution to the background fine structure.

The generalised Ramsauer-Townsend effect is of particular significance to the transport scattering cross-section since, as we have already identified, it gives particularly strong weighting to high angle scattering effects. This means that the angles at which the generalised Ramsauer-Townsend minima are most likely to occur will be heavily weighted, exaggerating their influence. To investigate the extent of this weighting, we can refer back to the results of Figures 3.11 to 3.14, which map the trends in λ_{tr}/λ_e with energy for a range of atomic numbers.

Since both mean free paths scale as n^{-1}, the ratio λ_{tr}/λ_e is the same as the ratio of respective scattering cross-sections and therefore gives insight into the angular distribution of scattering. At very low energy, isotropic scattering dominates and we therefore would expect λ_{tr}/λ_e to be approximately unity. At higher energy, forward scattering dominates, which leads to an approximately linear region in λ_{tr}/λ_e on the log-log axes. Examination of Figures 3.11 to 3.14 confirms this explanation but reveals more complex behaviour at medium energies as atomic number increases. The internal atomic structure of the higher atomic number elements leads to characteristic trends in the differential cross section, but with Ramsauer-Townsend minima becoming increasingly dominant at high Z. This leads to a maximum followed by a minimum at medium energies (see Figures 3.13 and 3.14) although the points at which these features occur, depends on atomic number and covers a wide energy range of 50-1000eV. From Y to Ag, for example, the maximum in λ_{tr}/λ_e appears at 100eV or less, whereas for the third transition series the corresponding maximum is evident at around 200eV.

These results support the approximate phase shift analysis of Cumpson (1997), which states that there will be a transport analogue of the generalised Ramsauer-Townsend effect. In fact, the energy dependent trends in λ_{tr}/λ_e described above are also evident in the transport mean-free path and scattering parameter with a similar bias towards high atomic number. Performing the same study with the Mott cross sections based on

a Thomas-Fermi atomic potential reveals qualitative agreement at low atomic number. As Z increases, disparities in λ_{tr} and χ reveal significant model-dependant differences.

In explaining analogies between the generalised Ramsauer-Townsend effect in X-ray absorption fine structure and anomalous features of the energy-dependent trends on electron transport behaviour, it is important to note that quantitative analysis in extended fine structure oscillations, requires curved-wave corrections to the elastic scattering of photoelectrons generated in the photoabsorption process (McKale, 1988). Such effects may also be significant to electron transport in XPS and AES, but these phenomena go beyond the model of transport in an amorphous medium which is fundamental to the role of λ_{tr}.

3.7 Conclusions

Systematic variations in the transport mean free path, $\lambda_{tr}(E,Z)$ have been studied in order to highlight regimes in which elastic scattering effects play a significant role in modifying the straight-line approximation descriptions of electron transport. Plots of the scattering parameter $\chi=\lambda_i/\lambda_{tr}$ as well as the ratio λ_{tr}/λ_e are also studied in order to gain further insight into the effects of elastic scattering. We find that although the extensions of the simple Born approximation proposed by Tilinin are valid at low to medium atomic number and at high energy, the generalised Ramsauer -Townsend effect leads to complex behaviour in the 50-1000eV range. These findings are in accordance with the theoretical analysis of Cumpson (1997).

Transport theory calculations have been shown to give accurate results for homogeneous samples in comparison with Monte Carlo methods (Jablonski and Tougaard, 1998). However, for off-normal angles of incidence and for heterogeneous samples where χ cannot be assumed constant, the transport theory model breaks down. The next chapter is devoted to the development of Monte Carlo methods with the intention of producing a more accurate description of electron behaviour under such conditions.

CHAPTER FOUR

MONTE CARLO SIMULATION OF THE DEPTH DISTRIBUTION FUNCTION IN MULTILAYERED STRUCTURES

4.0 Introduction

In order for successful quantitative analysis of Auger spectra to be carried out, it is important to gain an understanding of the origin of the electrons being collected. It is well understood that Auger electrons allow the analyst to perform chemical characterisation because their energies are specific to the atomic structure from which they originate. In general therefore, if an electron loses energy between its emission within a sample and its detection by the analyser, it no longer conveys useful information about the chemical composition of the surface under examination.

It would be a considerable undertaking to attempt to model the electron behaviour leading to an entire spectrum. Fortunately, we are normally only interested in the electrons that have lost no energy since emission making up the so-called 'no-loss' peaks, since it is these electrons which carry useful information.

The electron transport information required for characterisation of the no-loss peaks is contained in the Depth Distribution Function (DDF), $\Phi(z;\theta,\phi)d\Omega\,dz$, which is defined as the probability that an electron emitted between depths z and dz, will leave the surface of the sample within the solid angle defined by $d\Omega$ at θ, ϕ with little or no energy loss (i.e. a loss of less than 500meV).

In the absence of elastic scattering (the straight-line approximation) the sampling depth of electrons is defined by the inelastic mean free path, λ_i (Dwyer and Matthew, 1984). This gives the simplest approximation to the DDF, as a decaying exponential with the inelastic mean free path as the characteristic length

$$\Phi(z;\theta,\phi)\,d\Omega\,dz = \exp(-Z/\lambda_i \cos\theta)\;\; d\Omega\,dz \qquad (4.1)$$

This approach has been shown to give inaccurate results under certain conditions, due to the fact that it ignores *elastic* scattering events. This is particularly evident for materials of high atomic number and for grazing angles of emission, when elastic scattering through large angles is more likely (particularly at low energy).

4.0.1 Conventional Monte Carlo Simulation

Deriving an analytical technique which takes into account the effects of elastic scattering is a complex problem (Tougaard and Sigmund, 1982). Recently, several attempts have been made to produce approximations to the DDF as a sum of exponentials (Dwyer and Richards, 1992; Werner *et al* 1991), with some success. The work presented here, is based on the iterative process of Monte-Carlo simulation, in which the trajectories of electrons are simulated based on material parameters.

Early Monte-Carlo simulations were almost exclusively a step-by-step reconstruction of electron paths, based on the physical properties of the sample material. A flow diagram of this process is shown in Figure 4.1. Firstly, an electron is generated at a random depth Z, and with a random initial velocity vector selected from an isotropic distribution. Electron trajectories are followed from one point of interaction to the next, with path lengths taken from an exponential distribution based on the inelastic and elastic mean free paths.

After a large number of electron trajectories have been followed, the DDF can be derived as a probability distribution related to the origins of those electrons which were successfully detected without experiencing inelastic interaction with the sample material. The type of scattering event is decided on the basis of the relative total scattering cross sections of the sample material, and the angle of deflection after an elastic event chosen at random from the differential elastic scattering cross section. The energy loss brought about by an inelastic collision is determined by the differential inelastic scattering cross section. However, since electrons which have lost energy cannot contribute to the 'no loss' peak, electron trajectories are not normally simulated further after an inelastic event. Figure 4.2 graphically depicts this sequence of events.

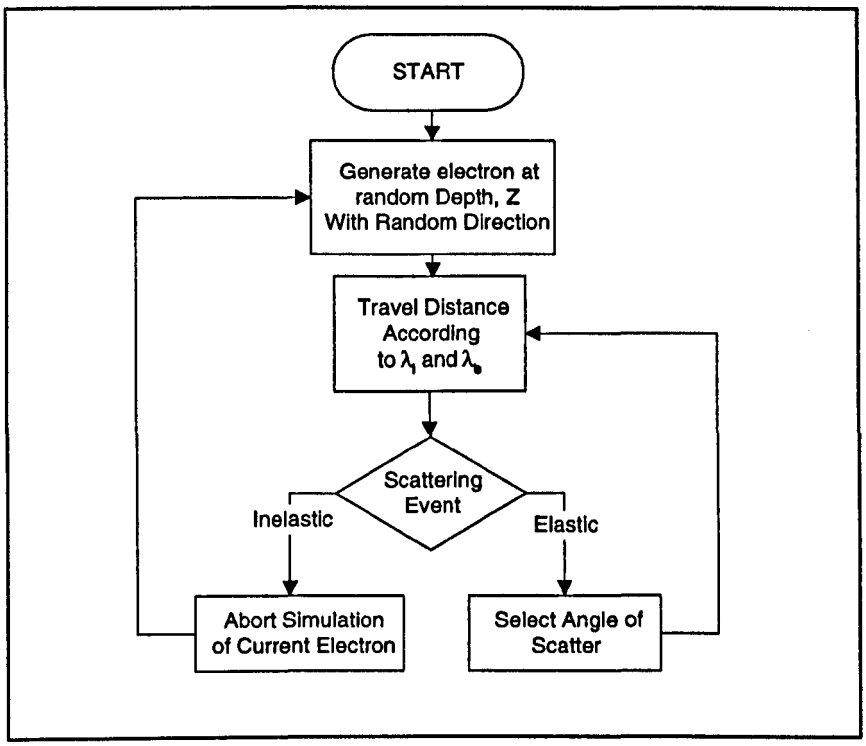

Figure 4.1 : Flow diagram showing the process of conventional Monte Carlo simulation

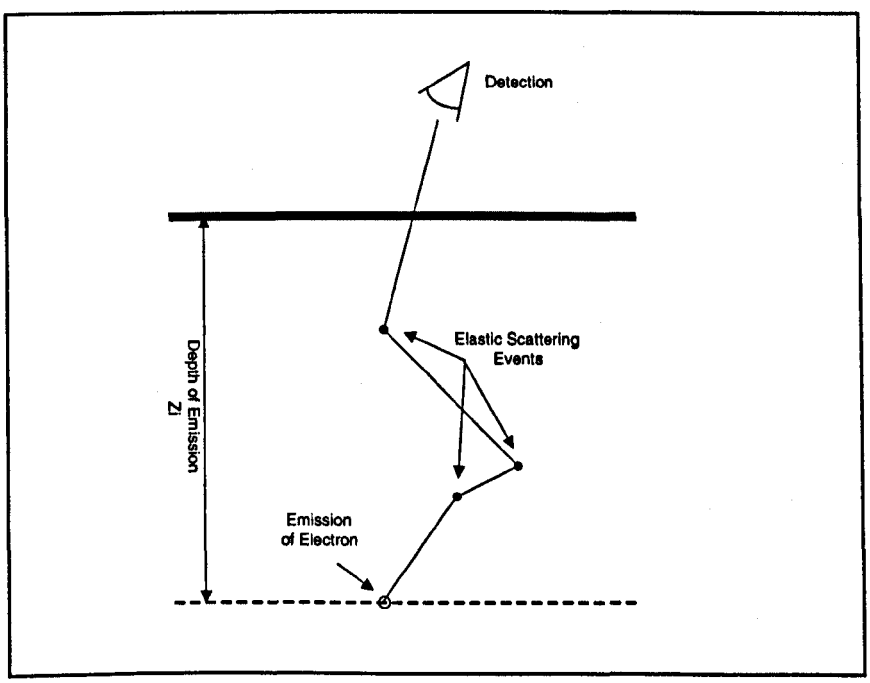

Figure 4.2 : Schematic diagram illustrating the process of conventional Monte Carlo simulation

4.0.2 Reverse Trajectory Monte Carlo Simulation

The conventional approach to Monte Carlo simulation is reasonably efficient if the programmer is attempting to simulate an analyser with a large solid angle of collection such as the retarding field analyser (RFA). If we limit this acceptance angle in an attempt to mimic the performance of an analyser such as the cylindrical mirror analyser described later in this thesis, many of the simulated electrons would fall outside the angle of collection and not contribute to the result ; essentially, the time taken to simulate these electrons is wasted.

Gries and Werner (1990) proposed a technique known as trajectory reversal, by which we can largely avoid the need to trace non-signal electrons. They used the same rules to describe electron behaviour in the sample, but did so in reverse time.

In the work of Gries and Werner, each simulated electron is generated at the position of the analyser and enters the sample with a fixed angle and energy. Electrons are then subjected to elastic collisions and are traced until they either escape from the sample, or collide inelastically. By the trajectory reversal argument, the position of these first inelastic collisions describes the DDF for the sample being simulated. The validity of the trajectory reversal technique in the context of DDF calculation has been explored (Cumpson, 1993) and found to be rigorously correct.

The application of the trajectory reversal simulation method to the description of three dimensional electron behaviour was validated by Werner (1991). He used trajectory reversal to speed up simulations of the DDF looking specifically at the effect on signal electrons from a bulk substrate when a thin overlayer is introduced.

In 1993, Cumpson put forward two new methods for reducing the time taken for Monte Carlo simulations of the DDF to achieve statistically meaningful results. The first of these, the 'statistical weights' method, used a semi-analytical approach to reduce statistical uncertainty of simulation process. The second method, based on the work of Berger and Doggett (1956) used analytical averaging over all possible scattering intervals to further reduce the random element of the simulation.

The work presented here is based around the 'statistical weights' method, since although the 'Berger-Doggett' method produces more rapid results (a factor of ten approximately), the electron trajectories are not explicitly followed, making it difficult to describe electron behaviour in a heterogeneous structure. The simulation software is written in the C programming language for compilation on a UNIX or MS-DOS based computer system (Jackson, 1994).

4.1 The Statistical Weights Method

The conventional technique used for Monte Carlo simulation of electron-solid interaction leading to the calculation of the DDF was outlined earlier in this chapter. Cumpson argued that although it would be a complex task to describe electron trajectories with a purely analytical approach, not all aspects of the calculations need to be handled by a Monte Carlo method.

The 'statistical weights' approach uses a standard Monte Carlo approach to describe the vectoral component of the problem (dealing with direction changes arising from elastic scattering) but deals with the scalar calculations (the inelastic losses) analytically. This method originates from the work of Ebel and Jablonski, (1984) and was found to be particularly powerful when combined with the reverse trajectory approach of Gries and Werner.

Consider the electron trajectory shown in Figure 4.3. In the reverse trajectory scheme, the electron is generated at the analyser (A) and enters the surface of the sample to undergo two elastic scattering events (B and C). At the first inelastic scattering event (D), the electron path terminates; the depth of this inelastic collision is 'binned' to contribute to the final DDF. The choice of path lengths (A-B, B-C etc.) is made according to the formula

$$L_S = -\lambda_e \ln(R) \qquad (4.2)$$

where L_S is the electron path length, R is a random number between 0 and 1, and λ_e is the elastic mean free path.

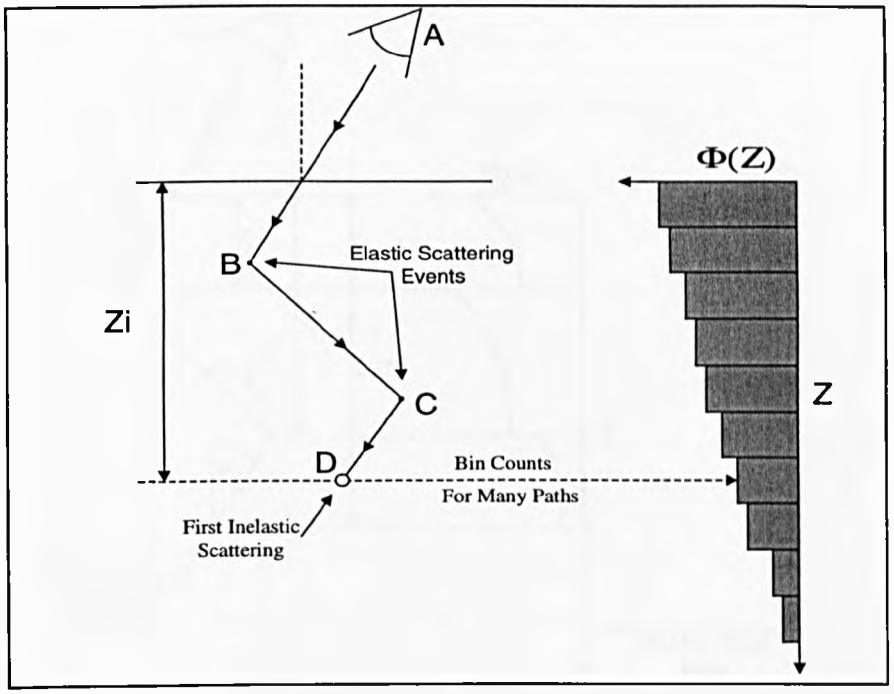

Figure 4.3 : An example of a trajectory reversed electron path, see text for details

We notice that since the choice of path length is made quite independently of the elastic scattering angles, this trajectory can be viewed as just one of many with the same series of elastic scatterings, but with paths that terminate at various points along each straight line segment. Applying the statistical weights method, we can accumulate the probability *distributions* for the final depths, $\varphi_i(z)$, rather than explicitly calculating each depth as it occurs in the simulation.

This process is illustrated in Figure 4.4 in which each of the probability distributions corresponds to a particular set of random elastic scattering events, the intervals and directions of which are calculated on the basis of an exponential attenuation using the IMFP and the length of each straight line segment. The DDF can then be estimated as

$$\Phi(z,\theta,\phi) = N_f \sum_{i=1}^{N} \varphi_i(z) \qquad (4.3)$$

where N is the number of paths and N_f a normalisation factor chosen to ensure that $\Phi(z;\ \theta, \phi) = 1$.

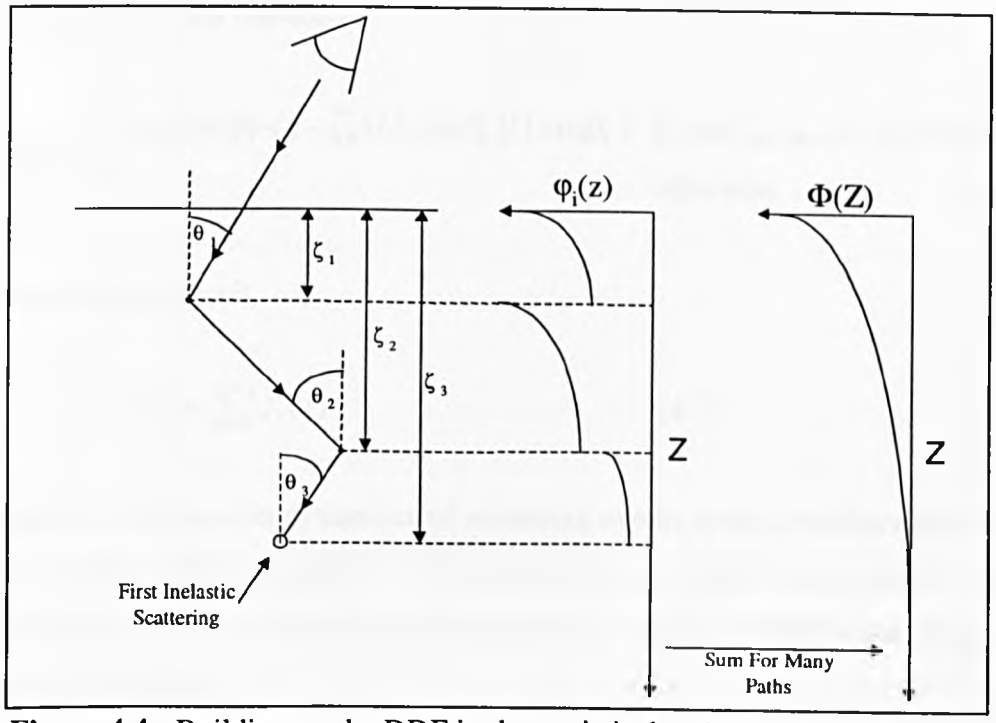

Figure 4.4 : Building up the DDF in the statistical weights method

The method we adopt for producing $\varphi_i(z)$ for any given path is as follows :

The differential elastic scattering cross section gives us a set of depths at which the elastic scattering events occur, $\zeta_1,\zeta_2,\zeta_3, \ldots$, and a set of angles corresponding to each scattering event, $(\theta_1,\phi_1), (\theta_2,\phi_2) \ldots$ (identical to conventional MC simulation). We then calculate electron fluence, $f_k(z)$, between the k^{th} and $(k+1)^{th}$ elastic scattering, for a unit flux incident on the surface. This fluence is proportional to the probability of an Auger electron being successfully detected without significant energy loss, having originated at a depth z and undergone k elastic scatterings, giving

$$f_0(0) = \frac{1}{|\cos\theta_0|} \qquad (4.4)$$

Since all the scattering events we are concerned with are elastic (and therefore involve no loss of energy), we can assume that fluence is conserved between successive line segments in the path, thus

$$\frac{f_k(\zeta_k)}{|\cos\theta_k|} = \frac{f_{k-1}(\zeta_k)}{|\cos\theta_{k-1}|} \qquad (4.5)$$

giving the electron fluence as

$$f_k(z) = \begin{cases} f_k(\zeta_k)\exp[-(z-\zeta_k)/\lambda_i \cos\theta_k]/|\cos\theta_k| & \text{if } \min(\zeta_k, \zeta_{k+1}) < z < \max(\zeta_k, \zeta_{k+1}) \\ 0 & \text{otherwise} \end{cases} \quad (4.6)$$

This yields the result

$$\varphi_i(z) = \sum_0^q f_k(z) \quad (4.7)$$

where q is the maximum number of scattering events to be considered for any given electron path. For the purpose of the calculations presented in this chapter, q was set to 20 since this will almost certainly yield at least one inelastic scattering in each electron trajectory.

Cumpson (1993) calculated that a statistical weights simulation of 4000 electron paths would give comparable results to half a million such trajectories simulated with Gries and Werner's trajectory reversal technique. Although the statistical weights approach demands somewhat more intensive computation, the overall improvement in performance is still considerable, and Cumpson reported that accurate results could be obtained in thirty minutes on a personal computer.

It is worth remembering that at the time Cumpson published his paper, a standard desktop PC would typically be based around an Intel 80486 processor with a clock speed of around 25MHz. Today, even an entry level Pentium based machine intended for light home use and costing well under a thousand pounds offers numerical processing speeds amounting to between a ten and fifty fold increase in performance over the machines of 1993.

4.2 The Bi-layer Model

The main purpose of this study is to extend the capabilities of the statistical weights algorithm, in order to allow simulations of heterogeneous structures and complex three dimensional sample topographies. The first step towards this goal is to simulate the simple case of a single overlayer on a bulk substrate.

In order to model the effects of a heterogeneous sample on electron transport, the material parameters in the simulation must be modified according to the elemental composition of the sample at each step of the simulation. In the context of the statistical weights technique embodied in equations 4.2 to 4.7, the mean free paths used in equations 4.2 and 4.6 must be chosen according to the sample material through which an electron travels for each particular step. Furthermore, the mean free path used in the calculation determining the event of an inelastic scattering will now depend on the electron's history and must be adjusted accordingly.

4.2.1 The Co-ordinate System

In common with many other workers on Monte Carlo algorithms, Cumpson elected to represent the physical position of an electron using the spherical polar co-ordinate system. This consists of two angles, θ and ϕ, and one length, z (which normally refers to the depth into a sample measured from the surface). This may at first appear to be an acceptable criterion on which to base the adjustments mentioned above for simulation of a single overlayer, since transitions from one material to another could be decided on the basis of the depth, z. However, as we will see, although it is indeed possible to detect a boundary crossing from a single linear dimension, the subsequent calculations required to correct for the transition require knowledge of the exact point of intersection in three dimensional space.

Although it is quite possible to handle three dimensional trigonometry in polar co-ordinates, it was thought to be counter intuitive since sample geometry would almost certainly be defined in Cartesian (x,y,z) co-ordinates. For this reason, the present simulation carries out the statistical weights Monte Carlo simulation in polar co-ordinates for consistency with previous work, but converts these to Cartesian co-

ordinates at every step of the simulation for use in the boundary crossing calculations. This is carried out according to the standard transformation (Shimizu and Ding, 1992)

$$x_{n+1}=x_n + s_n(\sin\theta_n\cos\phi_n) \qquad (4.8)$$

$$y_{n+1}=y_n + s_n(\sin\theta_n\sin\phi_n) \qquad (4.9)$$

$$z_{n+1}=z_n + s_n(\cos\theta_n) \qquad (4.10)$$

where x_{n+1}, y_{n+1}, and z_{n+1} represent the new electron co-ordinates, x_n, y_n, and z are the starting co-ordinates, s_n is the current step length, and θ_n, and ϕ_n are the angles given in polar co-ordinates.

4.2.2 Program Structure and Boundary Crossings

The next point to consider is the process involved in dealing with boundary crossings. The bilayer simulation maintains much the same basic structure as the original MATLAB program (Cumpson, 1993; Jackson, 1994), i.e. the first electron step is handled outside the main iterative loop in order to make programming and variable initialisation easier. When a boundary crossing is detected between the substrate and the overlayer (or vice-versa), the electron is repositioned at the point of intersection with the boundary, then proceeds taking into account the new material properties. In simulation terms, the process used to simulate a boundary crossing amounts to an elastic scattering event with no change in angle.

The step length calculations for the segments of the electrons path before and after the boundary crossing are calculated as follows:

$$S_0 = -\lambda_{e0}\ln(R_A) \qquad (4.11)$$

$$S_1 = -\lambda_{e1}\ln(R_B) \qquad (4.12)$$

where R_A and R_B are two independently chosen random numbers, λ_{e0} is the elastic mean free path of the starting material for the transition, and λ_{e1} is the elastic mean

free path for the target material into which the electron travels. S_0 represents the electrons motion from its previous elastic scattering event to the boundary, and S_1 is the linear path length to the next elastic scattering from the boundary.

The flow chart shown in Figure 4.5 outlines the operation of the bilayer simulation. The program structure has been simplified so that the changes involved in the development of the overlayer simulation can be seen clearly. The process of calculating the DDF from the simulated electron trajectories, remains essentially the same as described by Cumpson (1993) in his statistical weights simulation of homogeneous samples.

Data entry to the program is carried out with a simple text driven user interface, and the results are saved as two-column, tab-delimited ASCII text files. These files are suitable for transfer to most data processing and display packages. The results shown in this chapter were generated with a UNIX-based version of the 'gnuplot' graph drawing package.

The design of a graphical user interface which allows mouse-driven parameter entry and graphical display from these simulations is described elsewhere (Jackson, 1994). It is likely that a graphical user environment would be particularly useful if the current simulations were extended to cope with the modelling of more complex sample structures. It is envisaged that the multilayer simulation algorithm could be generalised so that a mouse-driven drawing package could be used to define an arbitrary heterogeneous sample layout.

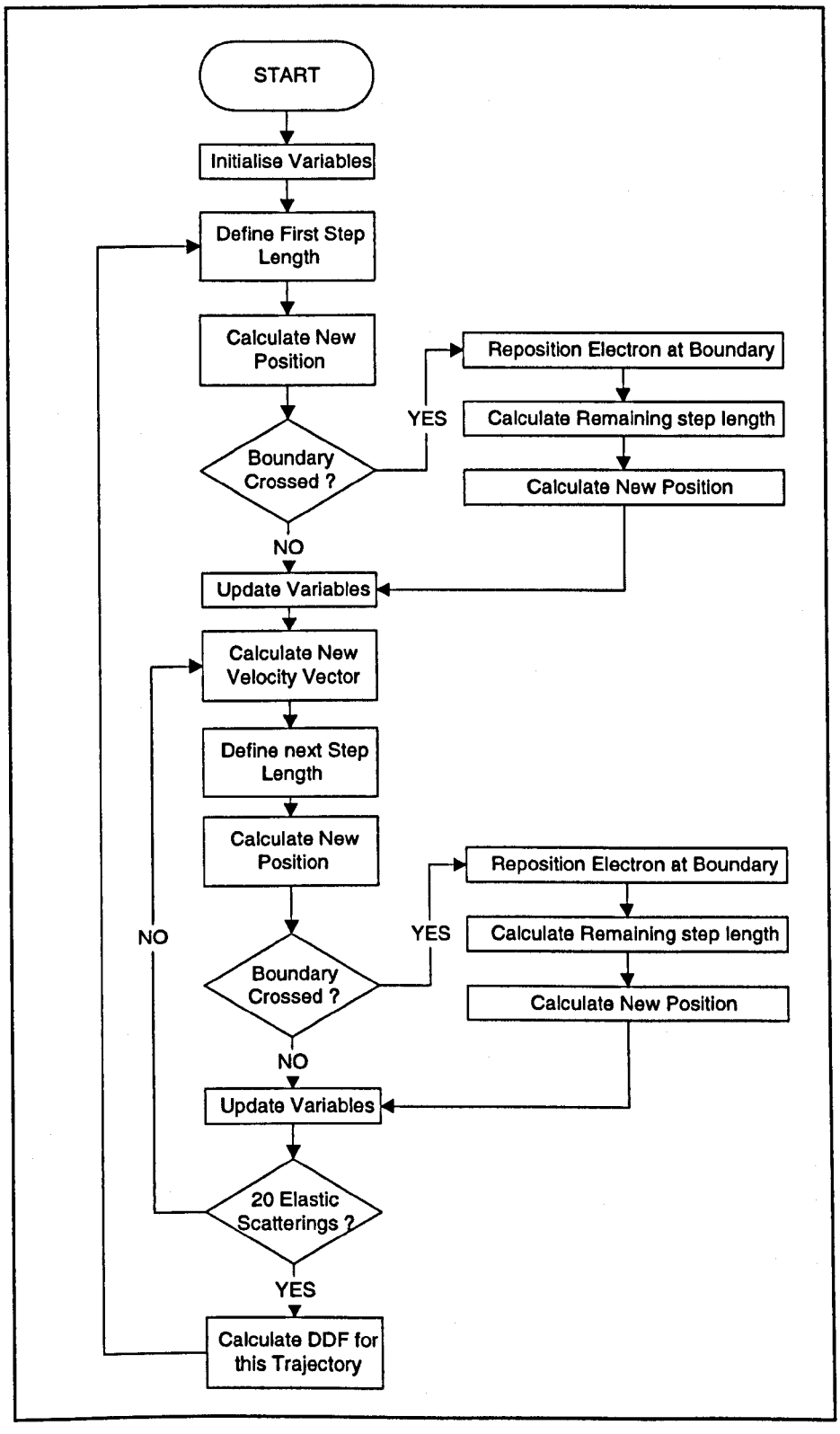

Figure 4.5 : Flow diagram illustrating the bi-layer simulation

Consider a single electron transition from one material to the other in the bi-layer structure, starting at point (x_0, y_0, z_0), ending at (x_1, y_1, z_1), and crossing the boundary between the two layers at point (x_b, y_b, z_b). This is schematically depicted on the three Cartesian axes in Figure 4.6.

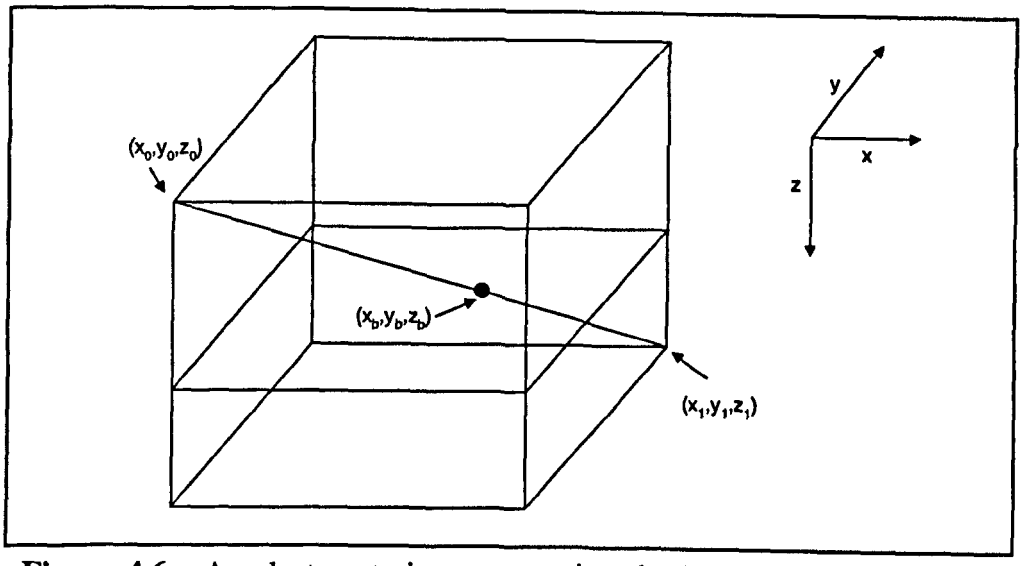

Figure 4.6 : An electron trajectory crossing the boundary, shown in the Cartesian co-ordinate system

We now need to derive a set of equations describing the position of the boundary crossing, and the related step lengths for this transition. Figure 4.7 shows the above electron transition in the (x,z) plane.

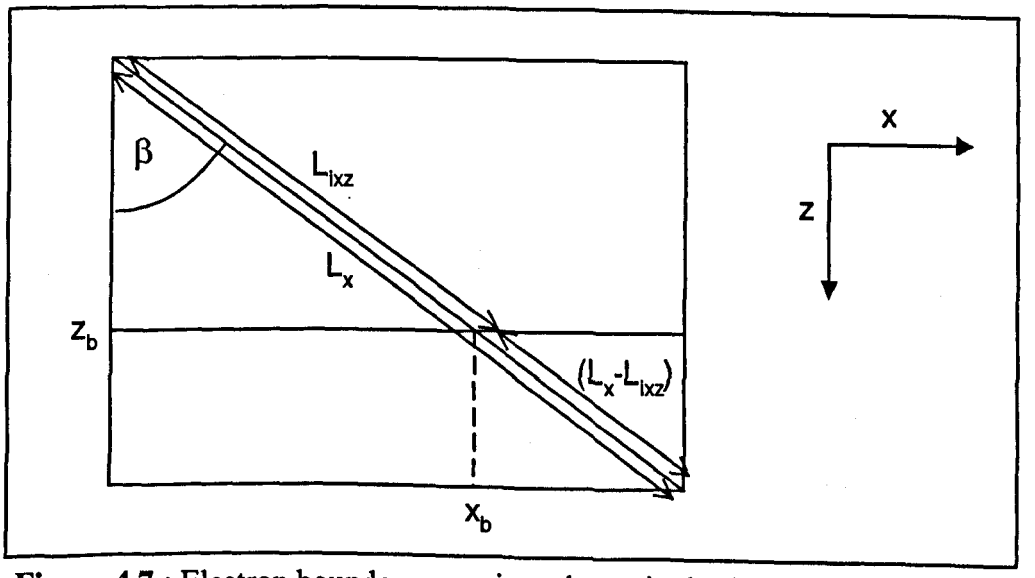

Figure 4.7 : Electron boundary crossing, shown in the (x,z) plane

From Figure 4.7, we can derive the following relationship between the overall step length in the (x,z) plane, L_x and the start and end points

$$L_x = \sqrt{(x_1 - x_0)^2 + (z_1 - z_0)^2} \qquad (4.13)$$

from Figure 4.7 we can deduce that

$$\cos\beta = \frac{(z_1 - z_0)}{L_x} = \frac{(z_b - z_0)}{L_{ixz}} \qquad (4.14)$$

and

$$x_b = \sqrt{L_{ixz}^{\ 2} - z_b^{\ 2}} + x_0 \qquad (4.15)$$

so

$$x_b = \sqrt{(\frac{(z_b - z_0)L_x}{(z_1 - z_0)^2})^2 - z_b^{\ 2}} + x_0 \qquad (4.16)$$

We can perform a similar analysis in the (z,y) plane, as illustrated by Figure 4.8 :

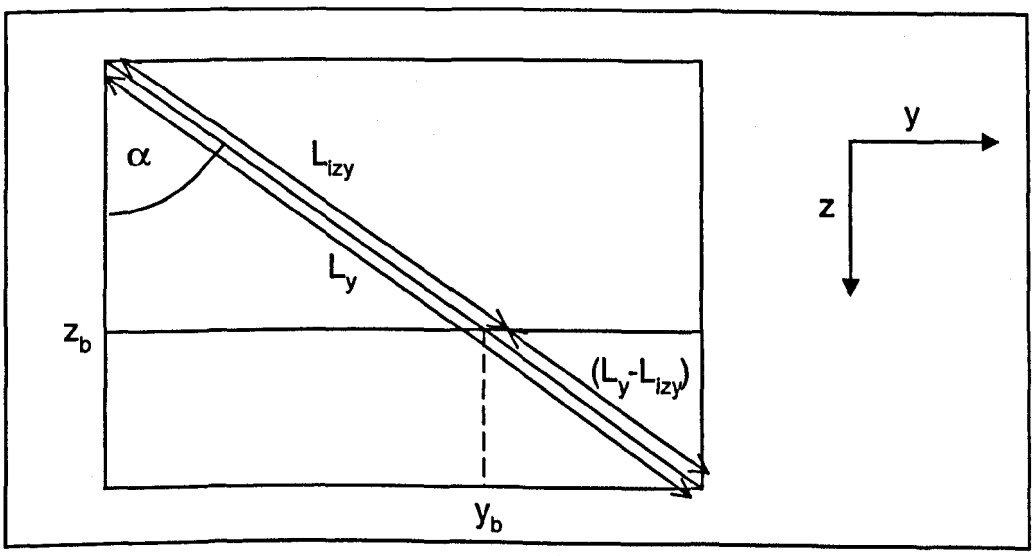

Figure 4.8 : Electron boundary crossing, shown in the (z,y) plane

As before, we can derive equations to describe the position of the electron in this plane

$$L_y = \sqrt{(y_1 - y_0)^2 + (z_1 - z_0)^2} \qquad (4.17)$$

once again, simple trigonometry yields

$$\cos\alpha = \frac{(z_b - z_0)}{L_{izy}} = \frac{(z_1 - z_0)}{L_y} \qquad (4.18)$$

and

$$y_b = \sqrt{L_{izy}^{\ 2} - z_b^{\ 2}} + y_0 \qquad (4.19)$$

so

$$y_b = \sqrt{(\frac{(z_b - z_0)L_y}{(z_1 - z_0)})^2 - z_b^{\ 2}} + y_0 \qquad (4.20)$$

Equations 4.16 and 4.20 provide the position of the boundary crossing in the x and y direction, and setting the position in the z direction is trivial. This gives us all the information required to carry out the simulation described by Figure 4.5.

The process for determining the remaining step after a boundary crossing does not need to make any attempt to consider how far the electron has travelled since its previous elastic scattering event, as the boundary crossing is handled in the simulation as an elastic scattering event with zero angle.

4.2.3 The Tri-layer Model

The functions have now been defined to detect and correct for boundary crossings from one material to another in the reverse trajectory Monte Carlo simulation. The transition from bi-layer to tri-layer simulation is therefore relatively straightforward, although somewhat complicated by the fact that we have to deal with considerably more possible outcomes from a single electron transition. The possible transitions are as follows:

1) No boundary crossing.
2) Crossing of one boundary only.
3) Crossing of boundary (1), but not boundary (2).
4) Crossing of boundary (2), but not (1).
5) Crossing of both boundaries.

Dealing with the eventualities (1-4) above, is basically the same as the single boundary crossings already covered, and is a logical extension to the bi-layer simulation. If both boundaries are crossed in a single transition (this is particularly likely with elements of low atomic number, or when the boundaries are close), we simply set the electron position back to the point at which it made its first crossing, then continue as before.

4.3 Results of the Multi-Layer Simulations

4.3.1 Comparison with Published Results

Cumpson (1993) used two methods to validate the results of the statistical weights Monte Carlo program. Firstly, he took the simple case where the elastic scattering cross section is zero, in an attempt to produce spectra with the 'straight line approximation' form of a decaying exponential with a decay constant of the inelastic mean free path. Secondly, simulation results were compared with the exact analytical solutions of Dwyer and Richards (1992) in certain special cases. Agreement with the analytical results in both cases was excellent, proving the validity of both the statistical weights method and the trajectory reversal approach on which it is based.

The first step towards validation of the multi-layer simulations presented in this chapter is to compare their results with those of Cumpson. To allow comparison with the homogeneous simulation of Cumpson, the bi-layer version of the statistical weights simulation was made to produce DDFs for a sample consisting of bulk carbon with a 3nm carbon overlayer. This was compared with DDFs of bulk carbon generated with the original MATLAB script of Cumpson (1993) which had already been validated as explained above. Both simulations used 965eV electrons and ran for 8000 iterations, the bi-layer simulation producing the result in less than a quarter of the time. This speed difference can be attributed to the compiled nature of the C programming language making it inherently faster than the interpreted MATLAB script (particularly for recursive loops). Figures 4.9 and 4.10 show these results for the single layer and bi-layer simulations respectively.

In order to validate the results under a variety of simulation conditions, comparisons are shown for a range of emission angles between normal incidence and 80°, and also with 1keV electrons in gold (the inelastic mean free path in gold is 1.29nm as opposed to 2.63 for carbon). The results for the gold simulations are shown in Figures 4.11 for the MATLAB script, and 4.12 for the bi-layer simulation.

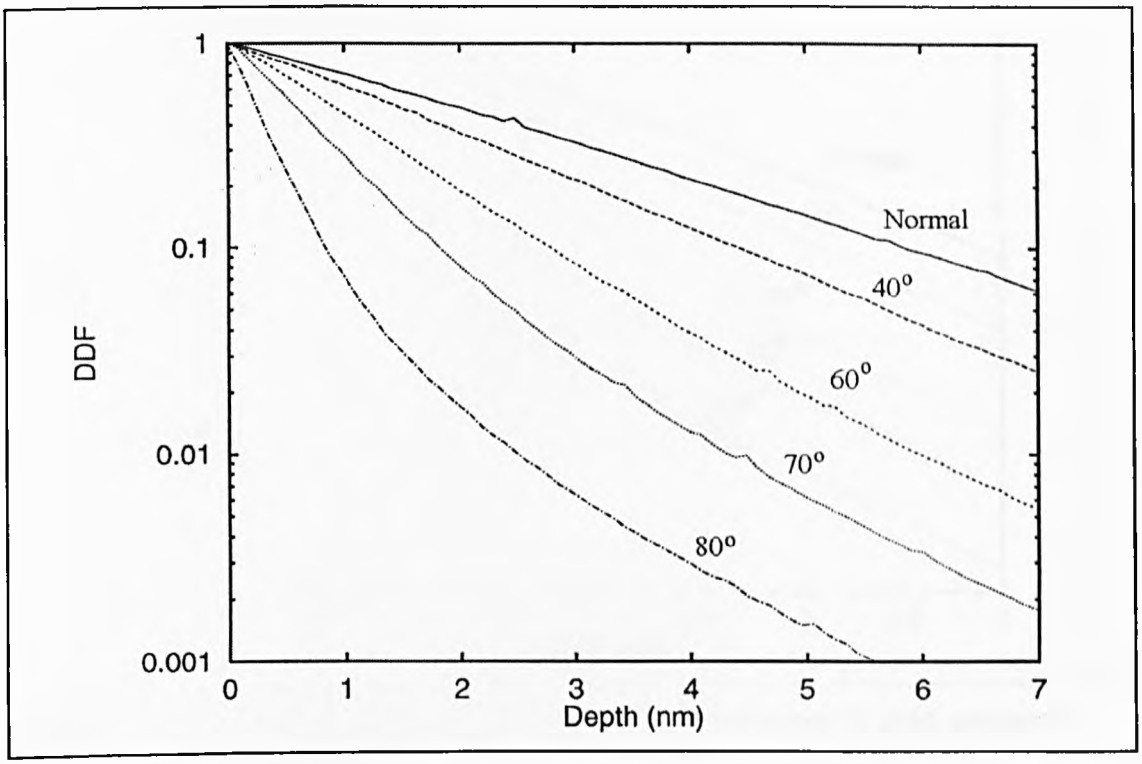

Figure 4.9 : Depth distribution functions for 965eV electrons in carbon produced by the MATLAB script of Cumpson

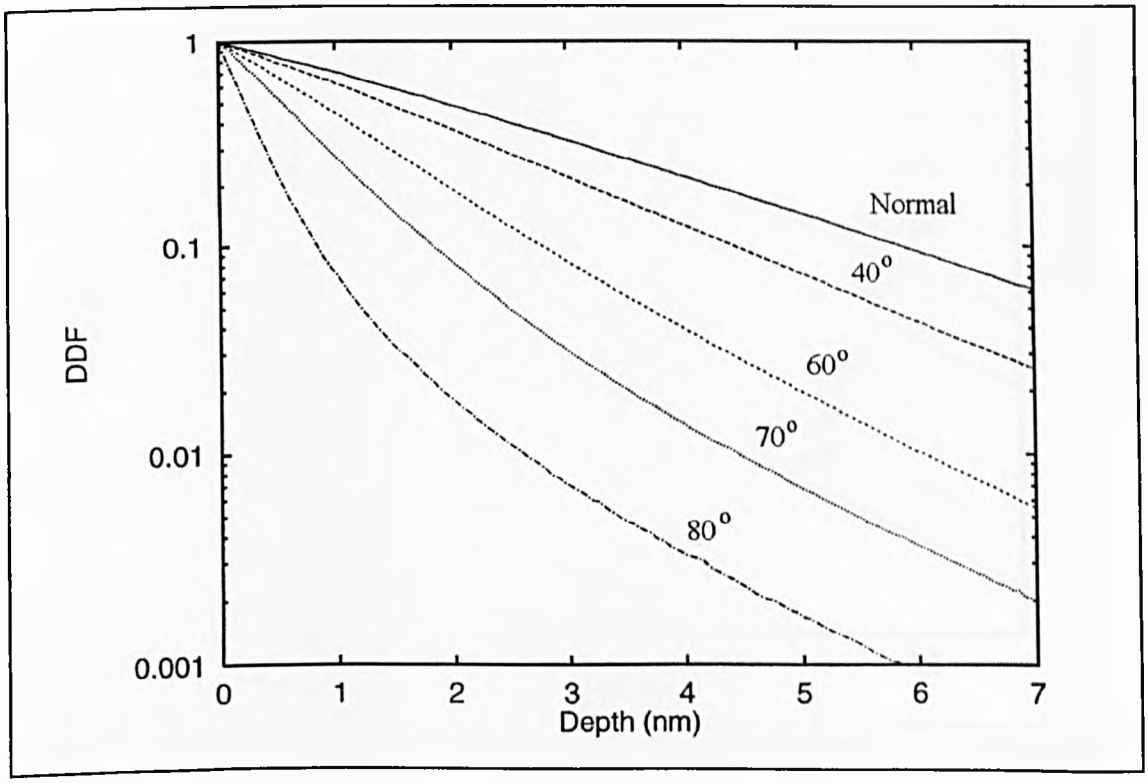

Figure 4.10 : Repeat of the simulation carried out in Figure 4.9, but using the bi-layer simulation for 3nm carbon on a carbon substrate

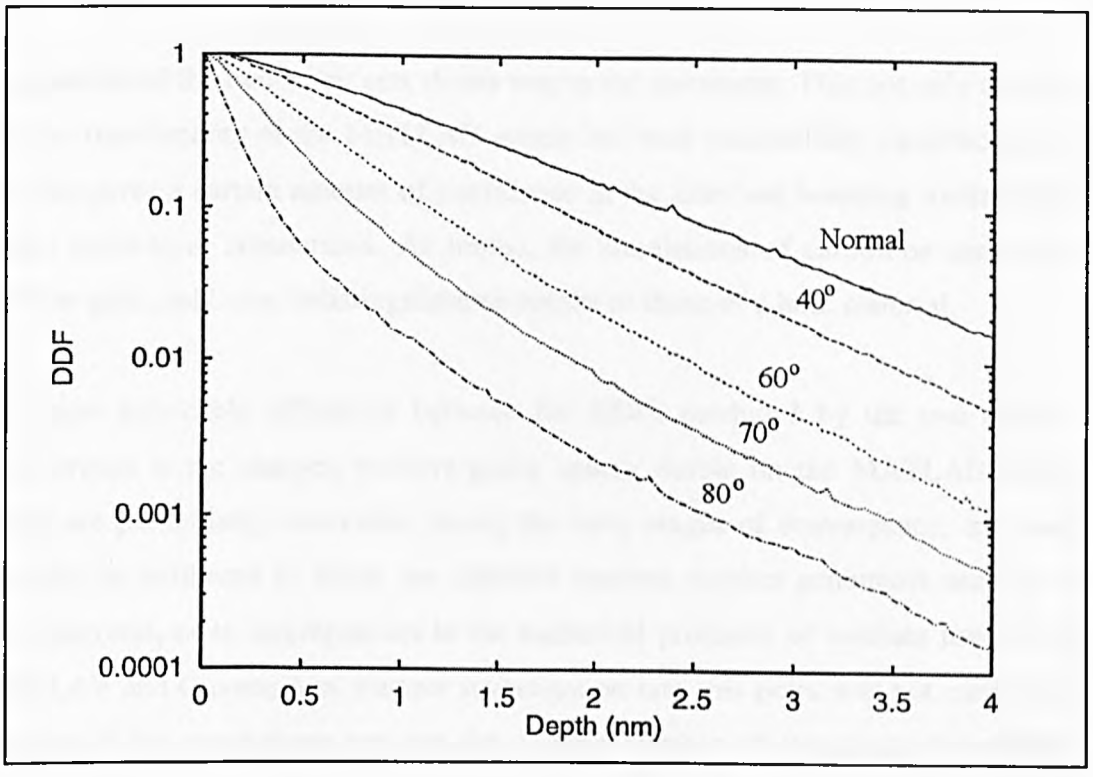

Figure 4.11 : Depth distribution function for 1keV electrons in gold produced by the MATLAB program

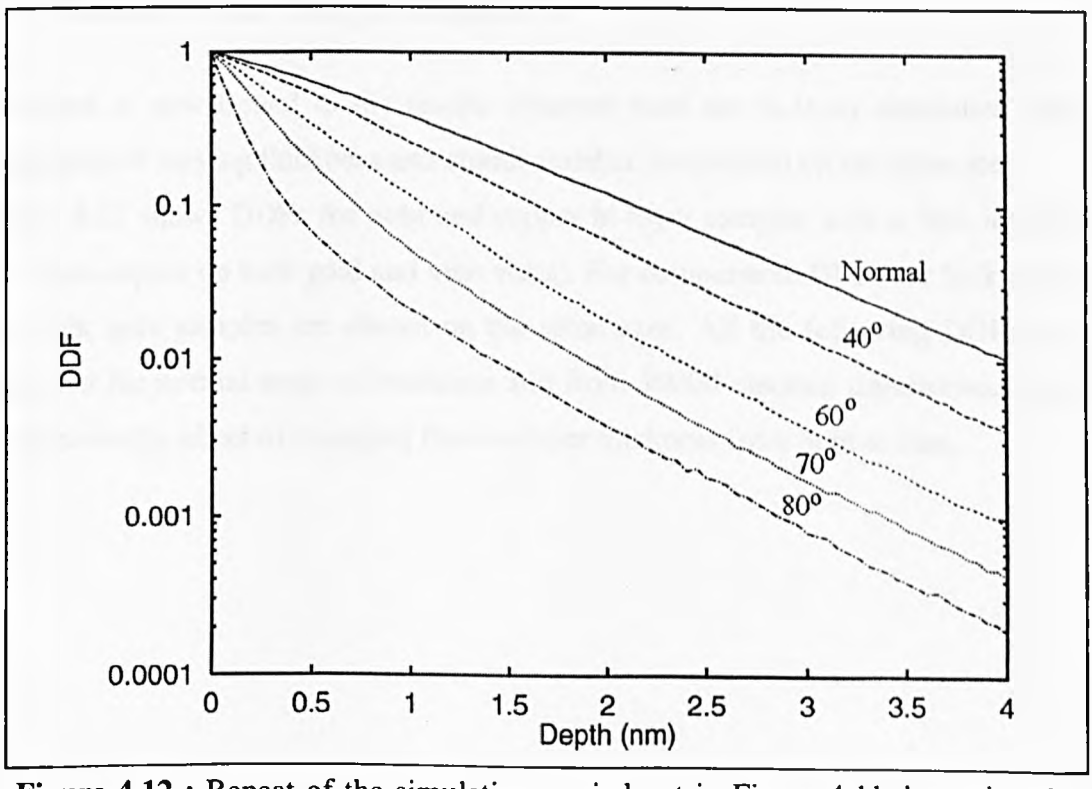

Figure 4.12 : Repeat of the simulation carried out in Figure 4.11, but using the bi-layer simulation for a 3nm gold overlayer on a gold substrate

Comparison of the two result sets shows very good agreement. This not only confirms that the functionality of the MATLAB scripts has been successfully reproduced in C, but also gives a certain amount of confidence in the interface handling routines used in the multi-layer simulations. As hoped, the simulations of carbon-on-carbon and gold-on-gold yield near indistinguishable results to those of a bulk material.

The most noticeable difference between the DDFs produced by the two different programmes is the sharper, positive-going spikes visible on the MATLAB results. These are particularly noticeable during the early stages of convergence, and could possibly be attributed to either the different random number generators used in the two programs, or to discrepancies in the numerical precision of routines used by the MATLAB and C compilers. Further investigation into this point was not carried out, because if the simulations are run for a large number of iterations, the statistics become sufficiently good that these features are no longer visible.

4.3.2 Results of the Bi-layer Simulation

Attention is now turned to the results obtained from the bi-layer simulation when overlayers of varying thickness and atomic number are defined on the substrate.
Figure 4.13 shows DDFs for gold and copper bi-layer samples with a 3nm interface (i.e. 3nm copper on bulk gold and vice-versa). For comparison, DDFs for bulk copper and bulk gold samples are shown on the same axes. All the following DDFs were produced for normal angle of emission and from 10000 electron trajectories. Figure 4.14 shows the effect of changing the overlayer thickness from 3nm to 1nm.

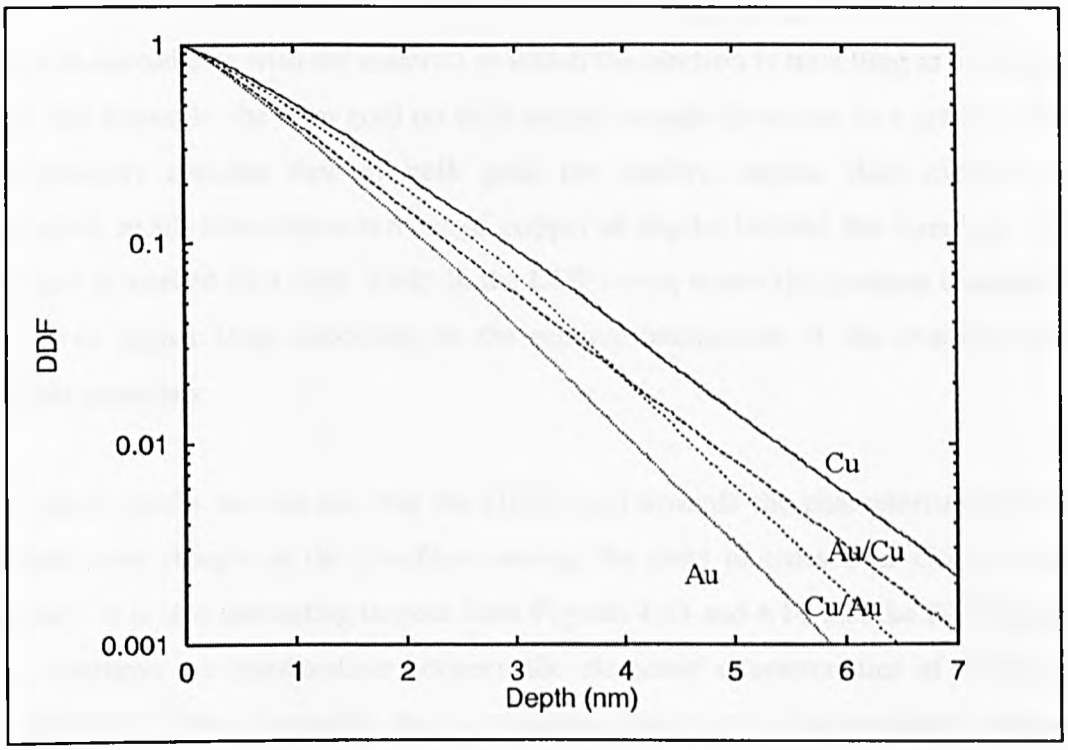

Figure 4.13 : Depth distribution functions for 1keV electrons of copper with a gold overlayer, and gold with a copper overlayer

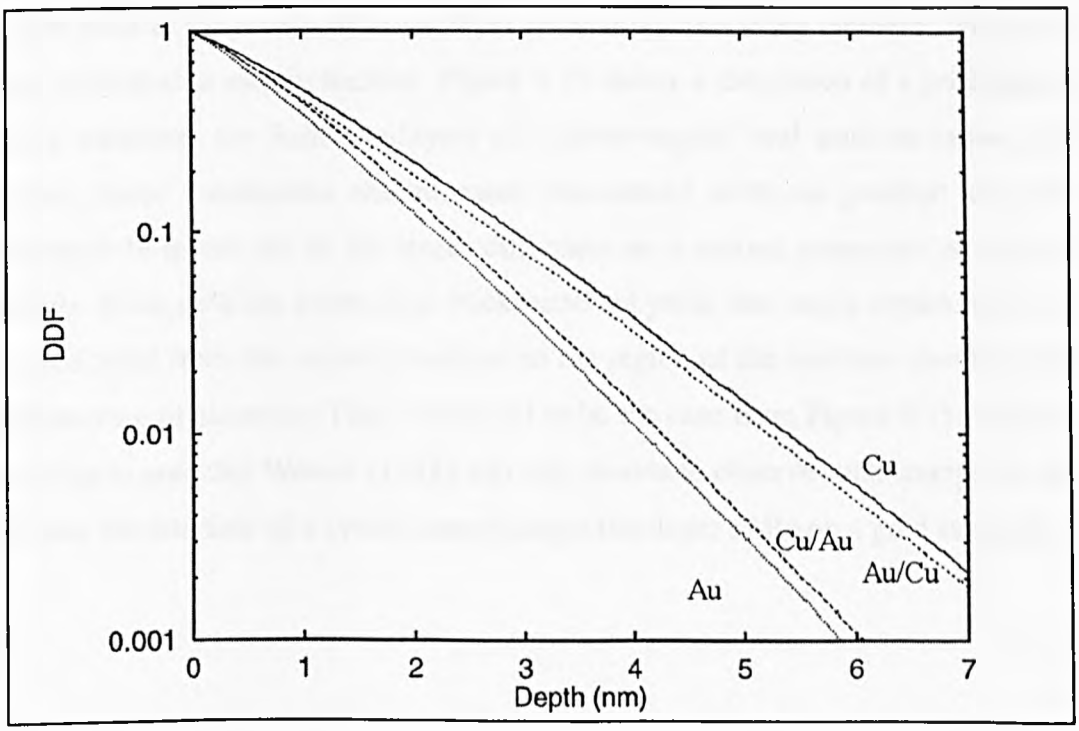

Figure 4.14 : A repeat of the simulation shown in Figure 4.14, showing the effect of moving the interface back to 1nm

As one might expect, and as confirmed by Werner (1991), the gradients of the DDFs change in accordance with the material in which the electron is travelling at any given depth. For example, the 3nm gold on bulk copper sample gives rise to a gradient that approximately matches that of bulk gold for shallow depths, then exhibits an attenuation much more characteristic of copper at depths beyond the interface. The boundary is marked by a clear 'kink' in the DDF curve, where the gradient changes to a lower or higher slope according to the relative attenuation of the overlayer and substrate materials.

From these results we can see that the DDFs tend towards the characteristics of the substrate very sharply at the interface causing the plots to cross over close to this boundary. It is also interesting to note from Figures 4.13 and 4.14 that the DDF slopes in an overlayer are intermediate between the elemental characteristics of overlayer and substrate. This is probably due to electrons that start in the overlayer, but are scattered by the substrate before detection.

We now look at the results obtained from simulations involving elements which are widely separated in atomic number. Figure 4.15 shows a simulation of a gold/carbon bi-layer structure, for 3nm overlayers of carbon-on-gold and gold-on-carbon. As expected, these simulations exhibit more pronounced shifts in gradient than the gold/copper bi-layers due to the larger difference in scattering properties of the two materials. Since gold has a very high backscattering yield, one might expect to see an enhanced yield from the carbon overlayer in the region of the interface due to direct backscattering of electrons. This is seen not to be the case from Figure 4.15, and it is interesting to note that Werner (1991) was also unable to observe such changes in the DDF near the interface of a system comprising a thin layer of Be on a gold substrate.

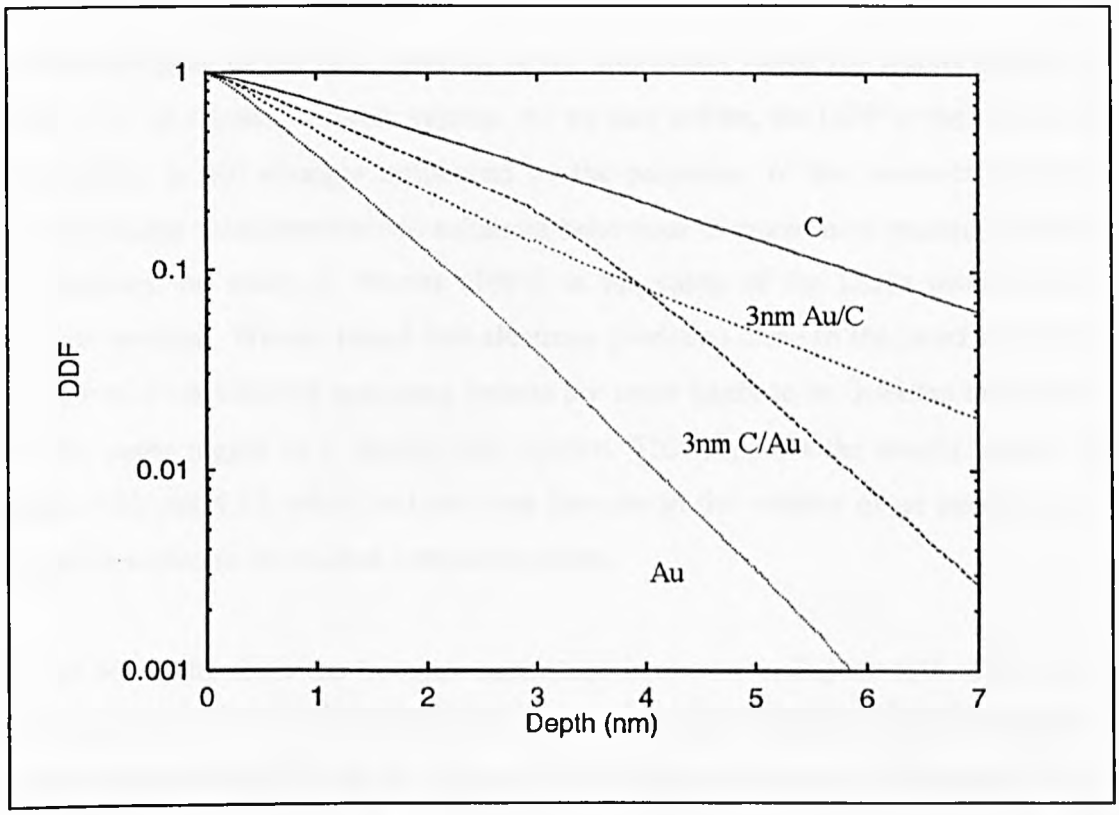

Figure 4.15 : The depth distribution function produced from overlayers of carbon and gold for 1keV electrons at normal exit angle. Results for the bulk substrates are also shown for comparison

Close inspection of the DDFs in Figure 4.15 reveals a small discontinuity in the DDF in the area of the interface. This has been attributed to the numerical precision to which the depth axis is sampled, as demonstrated by increasing the number of samples (or 'bins') per unit depth. When the number of bins per nm was increased from 20 to 40nm a subsequent improvement in the smoothness of the DDFs in this region was observed. All the DDFs presented here have been produced on a scale of 40 bins per nm and it is anticipated that increasing this further would lead to a corresponding smoothing of the DDF if the need became apparent.

From Figure 4.15, we can see that the probability of escape for electrons originating in the carbon overlayer is dramatically reduced by the presence of the bulk gold substrate. Figure 4.16 is a further investigation into this situation, with a simulation of an increasing thickness carbon overlayer on bulk gold. This reveals that as the overlayer thickness increases, the gradient of the DDFs tends towards that of the bulk carbon sample but is still considerably influenced by the gold substrate even for quite thick overlayers (8nm in the last plot).

The interchanging of the two materials in the simulation yields the results shown in Figure 4.17 for a gold overlayer system. As we saw before, the DDF in the region of the overlayer is still strongly influenced by the properties of the substrate, but this time the change from overlayer to substrate behaviour is much more gradual. In fact, this confirms the work of Werner (1991) in his study of the DDFs produced by overlayer systems. Werner found that electrons produced close to the interface in the substrate of a weak/strong scattering system are more likely to be detected than those from the same region in a strong/weak system. This supports the results shown in Figures 4.16 and 4.17, which indicate that features in the vicinity of an interface are more pronounced in the carbon overlayer system.

The set of results from the bi-layer simulation is shown in Figure 4.18. These are DDFs calculated for a 3nm overlayer system of carbon and gold for electrons with a kinetic energy of 200eV. This is a repeat of the simulations shown in Figure 4.15, but with a much reduced electron energy. Comparison of these results reveals a much steeper gradient in all the DDFs, this is due to enhanced elastic scattering and a reduced inelastic mean free path. Perhaps more striking, however, is the observation that the DDFs for the lower energy simulation deviate in gradient from the substrate characteristic at depths much lower than the interface. This is evidence of the importance of redirection of electrons below the overlayer.

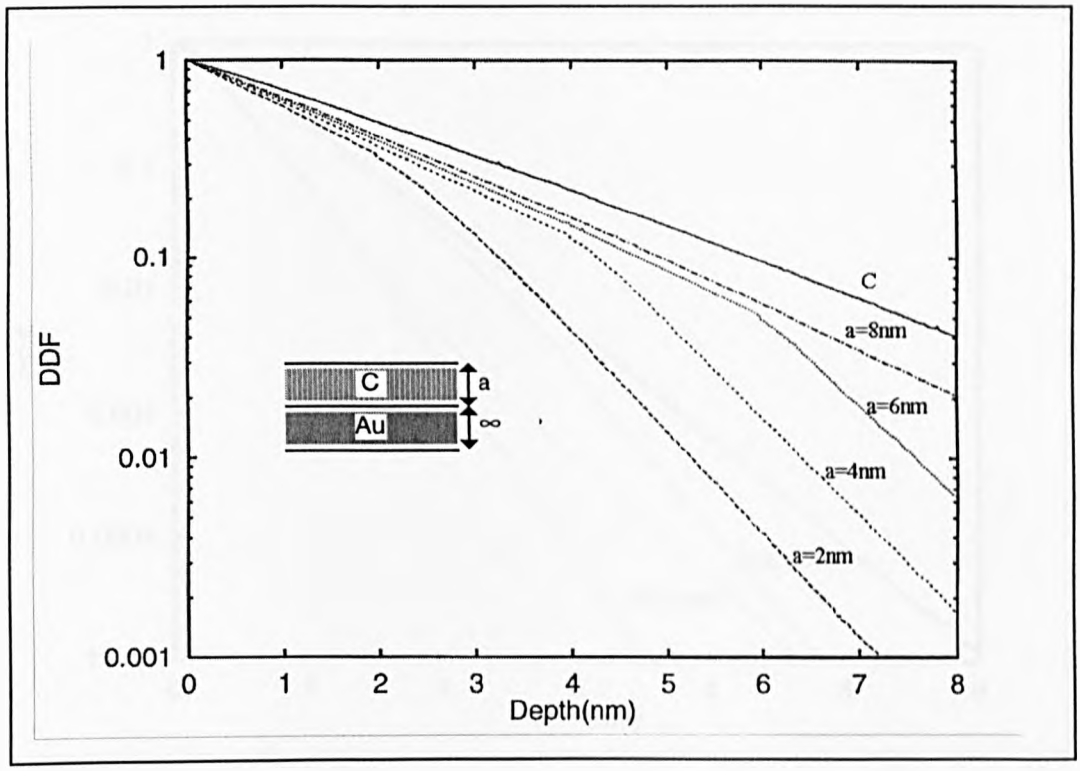

Figure 4.16 : The DDFs produced from varying thickness overlayers of carbon on bulk gold

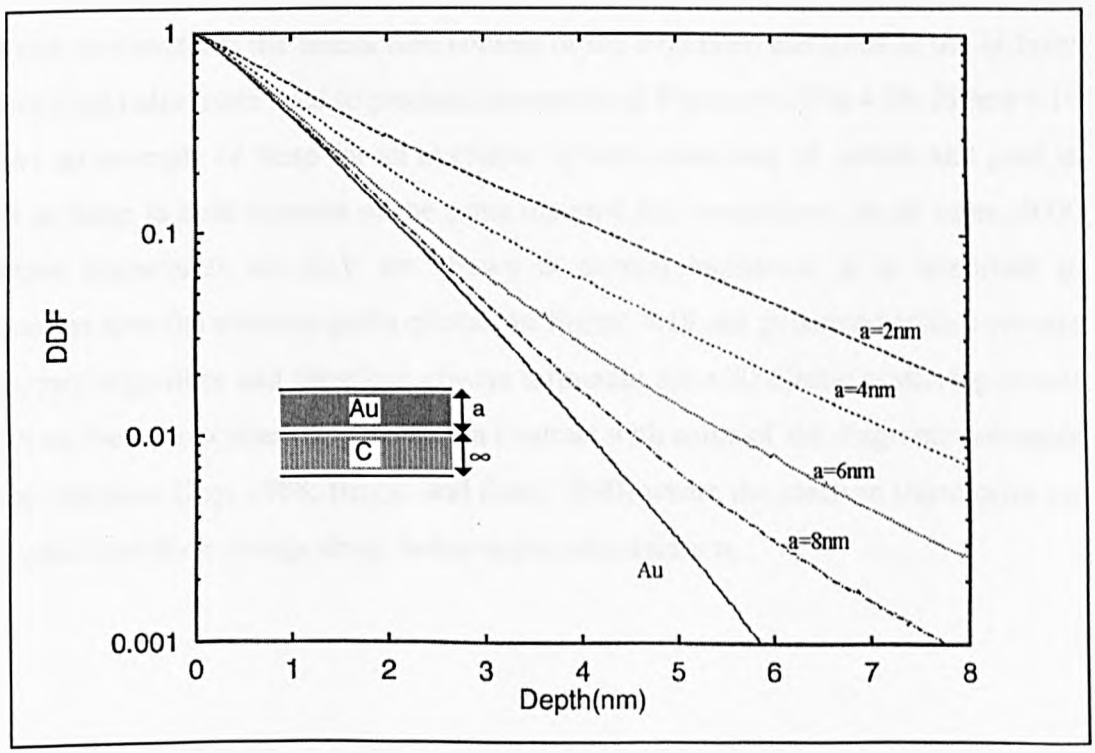

Figure 4.17 : The DDFs produced from varying thickness overlayers of gold on bulk carbon

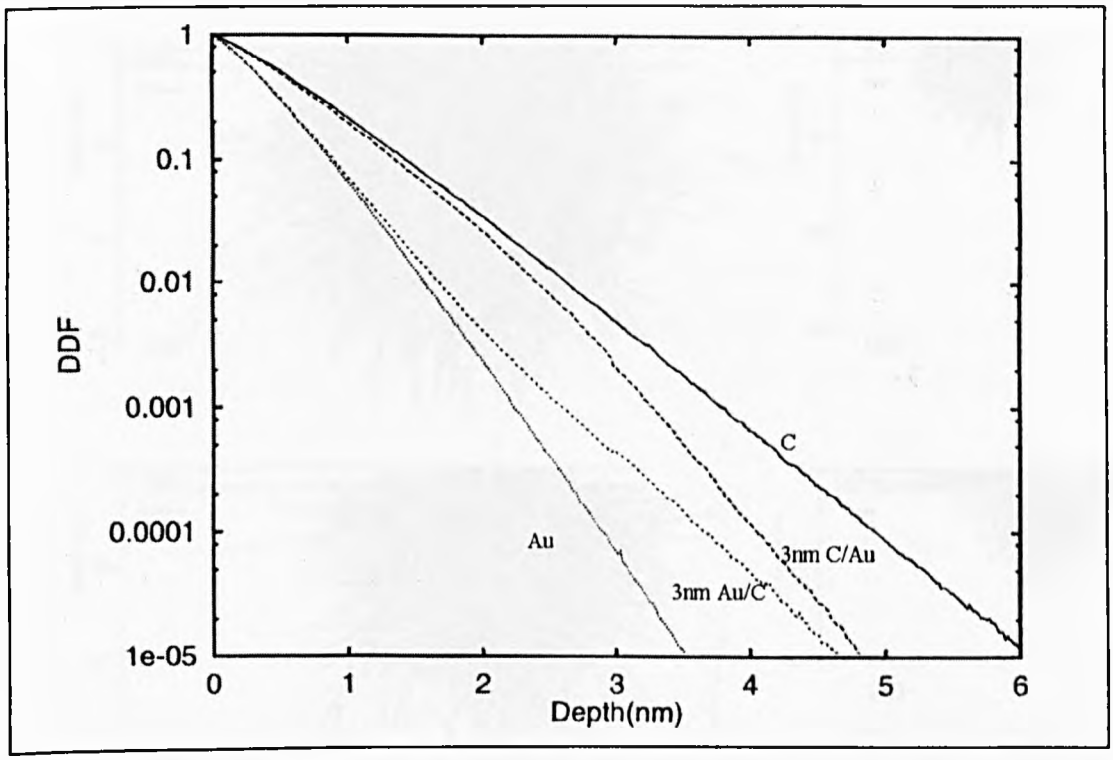

Figure 4.18 : The depth distribution functions produced from overlayers of carbon and gold for 200eV electrons at normal incidence

We can also examine the interaction volume of the simulated electrons in the bi-layer Monte Carlo algorithm used to produce the results of Figures 4.13 to 4.18. Figure 4.19 shows an example of these for an overlayer system consisting of carbon and gold as well as those in bulk samples of the same material for comparison. In all cases, 4000 electron trajectories of 1keV are shown at normal incidence. It is important to remember that the electron paths plotted in Figure 4.19 are generated with a reverse trajectory algorithm and therefore *always* terminate after 20 elastic scattering events (with no inelastic scatterings). This is in contrast with some of the diagrams presented in the literature (Joy, 1988; Briggs and Seah, 1990) where the electron trajectories are followed until their energy drops below a pre-set minimum.

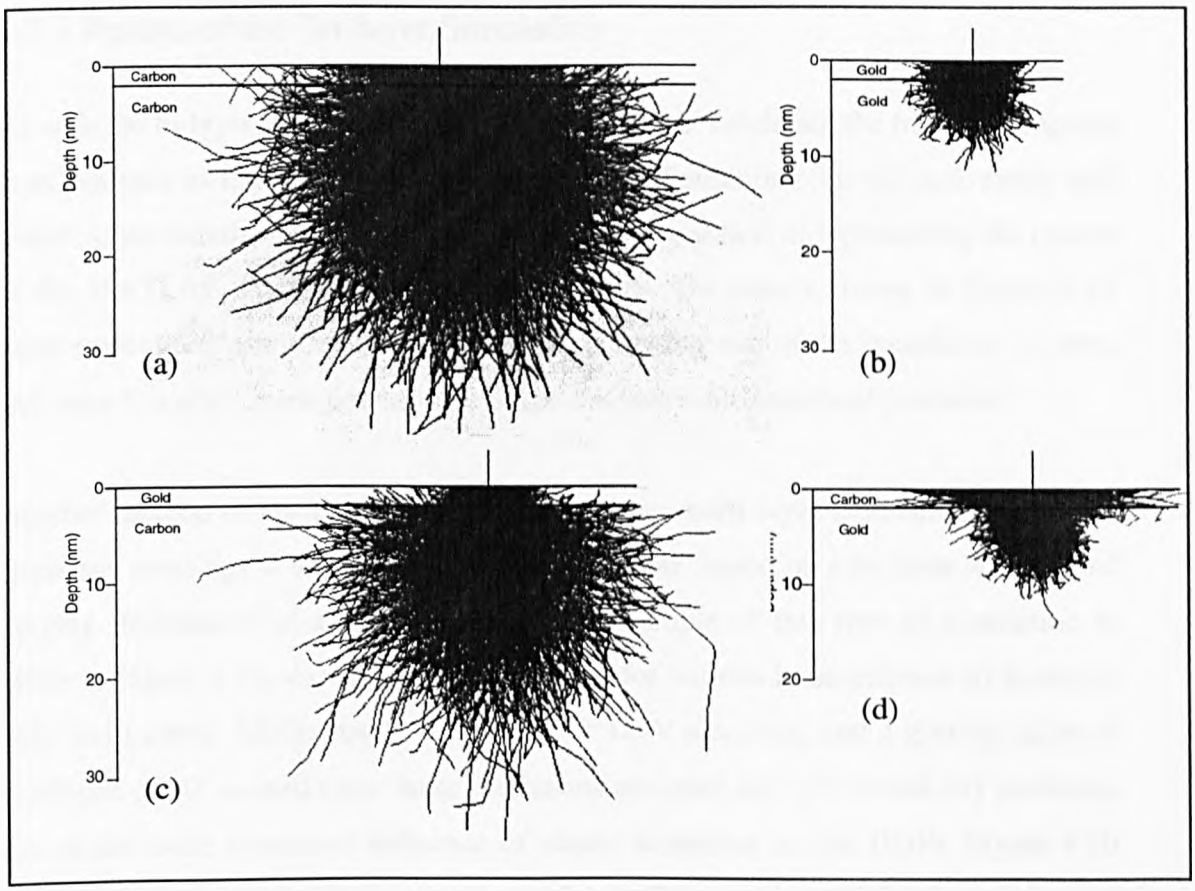

Figure 4.19 : Electron-solid interaction volumes observed in the single overlayer Monte Carlo model at 1keV for 4000 trajectories at normal incidence in (a) bulk carbon, (b) bulk gold, (c) 2nm gold on bulk carbon, (d) 2nm carbon on bulk gold

Although examination of Figure 4.19 gives at best only a qualitative insight into electron behaviour in the simulation, it does explain to some extent the origin of trends in the DDFs produced. Electron scattering patterns in the overlayer systems largely mirror those in the bulk samples according to which region an electron is travelling. Furthermore, in support of the results of Figures 4.16 and 4.17, the boundary between the weak/strong scattering system of carbon on gold appears to give sharper transition in scattering pattern than the gold on carbon interface.

4.3.3 Results of the Tri-layer Simulation

As with the bi-layer simulations, the first step towards validating the tri-layer program is to compare its results with those from previous simulations. To this end, single and double layer simulations were carried out with the intention of reproducing the results of the MATLAB script and the bi-layer program. The results shown in Figure 4.15 were reproduced using the tri-layer simulation setting one of the boundaries to zero, and were found to match perfectly to within the attainable statistical precision.

Another method of testing the way in which these multi layer simulations deal with boundary crossings is to use a homogeneous sample, based on a tri-layer structure of varying thicknesses of a single element. An example of this type of simulation is given in Figure 4.20, showing DDFs generated for various homogeneous tri-layers of gold and carbon. All the results are given for 1keV electrons, and a grazing angle of incidence of 80° is used since these conditions are most likely to reveal any problems due to the more prominent influence of elastic scattering on the DDFs. Figure 4.20 shows DDFs for homogeneous samples with interfaces set at 1 and 5nm, 2 and 3nm, 0 and 6nm. As before, we find that for all the 'sandwich' structures simulated the results are identical; any discrepancy can be attributed to the inherent statistical element of the simulation.

Carbon and gold were chosen as sample materials for the tri-layer simulations due to their difference in atomic number and therefore contrasting elastic scattering characteristics and inelastic mean free path. The first results from the tri-layer simulation are shown in Figure 4.21, for 1keV electrons at normal incidence. The sample geometry for this simulation is shown in the schematic diagram accompanying Figure 4.21 and consists of a tri-layer carbon/gold/carbon structure. Figure 4.22 shows the depth distribution functions for a gold/carbon/gold sample for comparison. Both tri-layer samples consist of a two nanometre overlayer with either a three or five nanometre 'sandwich' layer on a bulk substrate.

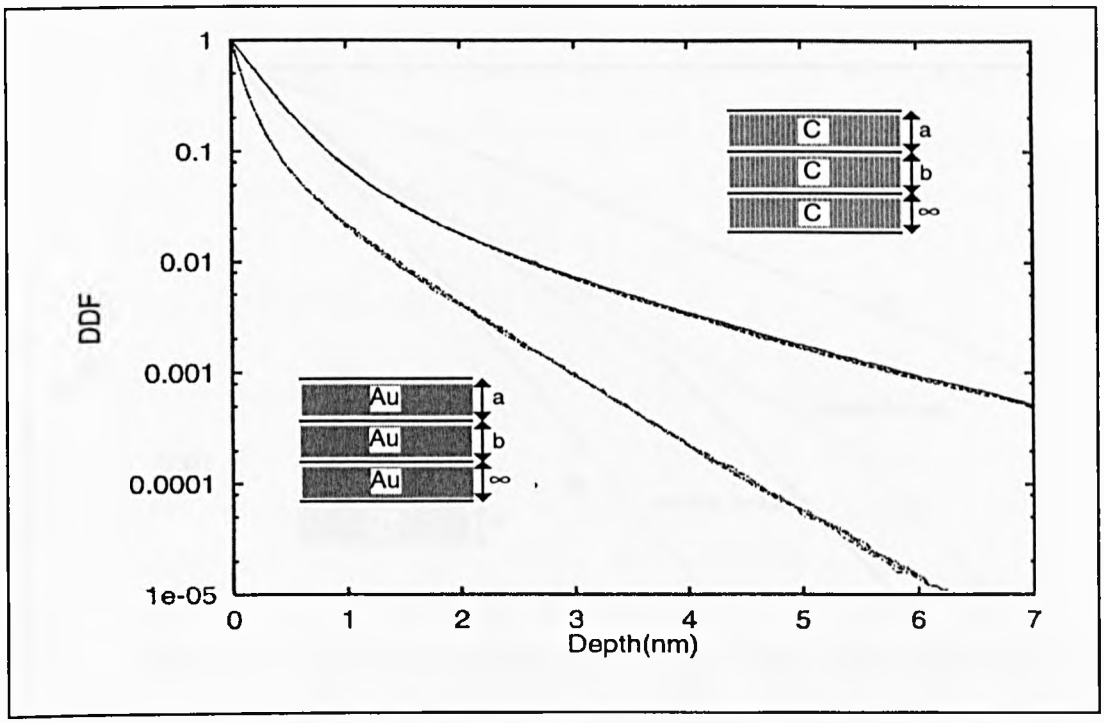

Figure 4.20 : Depth distribution functions for 1keV electrons in homogeneous samples consisting of tri-layer structures of carbon and gold

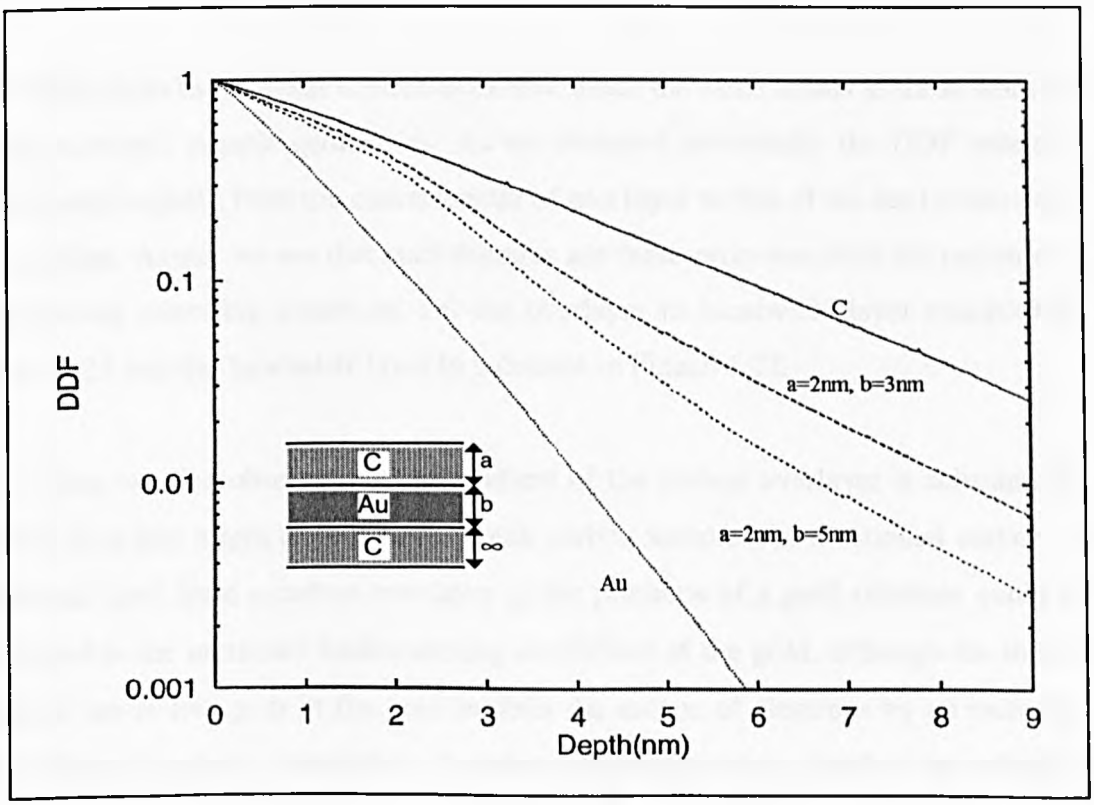

Figure 4.21 : Tri-layer simulation results for 1keV electrons at normal incidence, the schematic diagram shows the structure used in the simulation

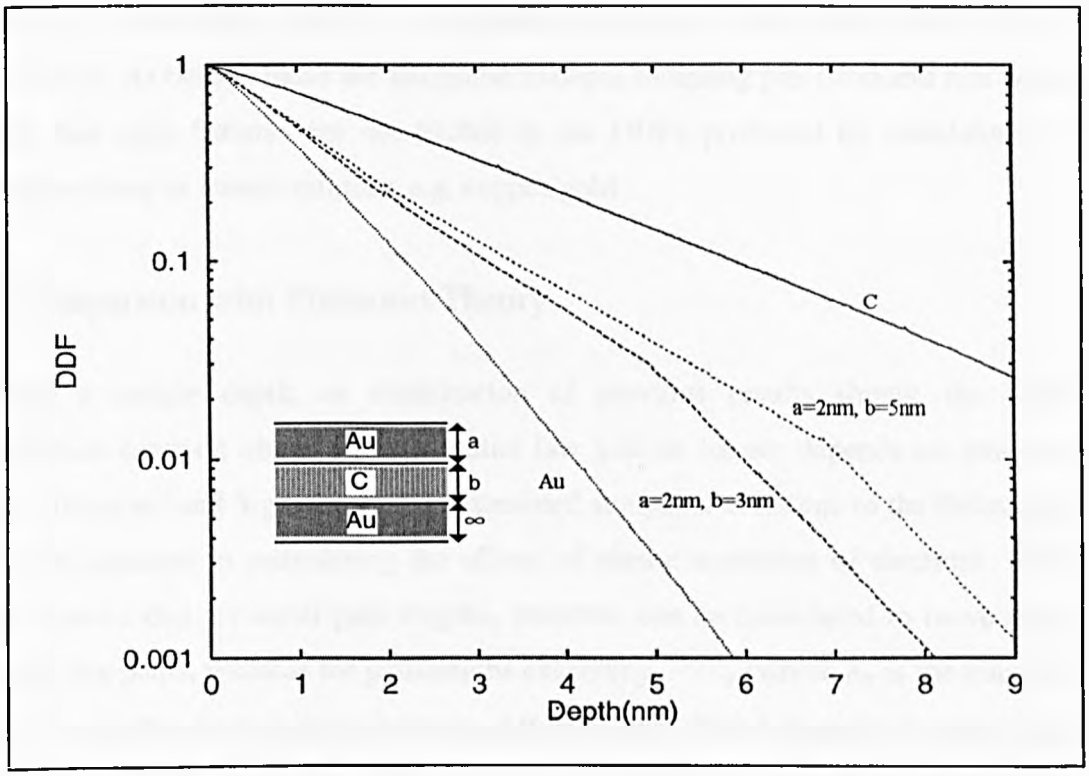

Figure 4.22 : The depth distribution functions produced by the interchanging of the materials simulated for Figure 4.21

The DDFs from the tri-layer structures exhibit much the same trends as those from the single overlayer sample geometries. As we observed previously, the DDF seems to switch quite rapidly from the characteristic of one layer to that of the next at the point of interface. Again, we see that such features are more pronounced in the region of a weak/strong scattering transition, i.e. the overlayer to 'sandwich' layer transition in Figure 4.21 and the 'sandwich' layer to substrate in Figure 4.22.

As before, we also observe that the gradient of the carbon overlayer is substantially greater than one might expect from a bulk carbon sample. As mentioned earlier, an increased yield from a carbon overlayer in the presence of a gold substrate could be attributed to the increased backscattering coefficient of the gold, although the shorter inelastic mean free path in the gold inhibits the escape of electrons by an increased probability of inelastic interaction. A carbon intermediate layer leads to an enhanced DDF for emission in the gold regions in the overlayer, and to some extent also in the substrate.

Small spikes are again visible at the interface on some of the DDFs in the tri-layer simulations. As before, these are attributed to depth sampling precision and it is worth noting that such features are not visible in the DDFs produced by simulations of elements closer in atomic number, e.g. copper/gold.

4.4 Comparison with Transport Theory

Beyond a certain depth, as examination of previous results shows, the depth distribution function obeys an exponential law and no longer depends on emission angle. Tougaard and Sigmund (1982) examined analytical solutions to the Boltzmann transport equation in considering the effects of elastic scattering of electrons. Their work showed that for small path lengths, electrons can be considered to move along straight line paths, whereas for pathlengths exceeding $\sim 2\lambda_{tr}$ (where λ_{tr} is the transport mean free path) electron paths become diffusion-like. This behaviour is termed the no-memory effect since the DDF gradients of electrons travelling from depths exceeding 1-2 λ_{tr} does not depend on their angle of emission.

The no-memory condition is characterised by a straight line on logarithmic DDF plots. For example, from Figure 3.6 in Chapter three, $\lambda_{tr} \approx 2.2$nm for 1keV electrons in gold, examination of Figure 4.12 reveals exponential decay setting in at around 1.5-2nm i.e. approximately λ_{tr}. The characteristic length of this exponential is termed the attenuation parameter (AP) and has been shown by transport theory to be dependant on two parameters: the inelastic mean free path λ_i , and the transport mean free path, λ_{tr}.

Tougaard and Sigmund (1982) also found that when the no-memory condition is satisfied (i.e. $z \geq \lambda_{tr}$), the measured intensity from a tracer decays exponentially with z as

$$\phi^{el} \approx \exp\left(-\frac{z}{\lambda_{diff}}\right) \qquad (4.21)$$

where

$$\lambda_{diff} = \sqrt{\frac{\lambda_i \lambda_{tr}}{3}} \qquad (4.22)$$

In this section, an attempt will be made to validate the Monte Carlo results with the attenuation parameter described by Tougaard and Sigmund (1982), and more recently by Werner and Tilinin (1992) who also used the attenuation parameter to compare Monte Carlo results with transport theory.

Tougaard and Sigmund define the attenuation parameter for no-memory electron paths, λ_a as

$$\lambda_a = \frac{\lambda_i \lambda_{tr}}{\lambda_i + \lambda_{tr}} \nu_0 \qquad (4.23)$$

where ν_0 is the only positive root of the characteristic equation in the case of isotropic scattering

$$1 = \frac{\omega \nu}{2} \ln \frac{\nu + 1}{\nu - 1} \qquad (4.24)$$

and ω is defined as

$$\omega = \frac{\lambda_i}{\lambda_i + \lambda_{tr}} \qquad (4.25)$$

The transport mean free path, λ_{tr}, is calculated from the Mott scattering cross sections by numerical integration as discussed in Chapter three, according to the following integral relationship

$$\lambda_{tr}^{-1} = 2\pi n \int_0^{\pi} \frac{d\sigma}{d\Omega} ((1 - \cos\theta)\sin\theta) d\theta \qquad (4.26)$$

where n is the number of atoms per m^3. The inelastic mean free path data are those of Tanuma, Powell and Penn (1991), which were also used in all the Monte Carlo simulations.

As with all previous comparisons, we start with the simple case of bulk samples. Application of equations 4.21 to 4.24 to carbon and gold yields attenuation parameters of 2.45nm for carbon, and 0.84nm for gold. Examination of Figure 4.20 gives an attenuation parameter of 2.44nm for 1keV electrons in carbon (0.4% discrepancy from the theoretical prediction) and 0.81nm from the simulated DDF of the gold sample (3% discrepancy from the predicted value).

We can now go on to look at the results obtained from a tri-layer simulation. Figure 4.23 shows the depth distribution functions generated for bulk carbon and gold and for tri-layer structures of carbon/gold/carbon and gold/carbon/gold both with interfaces at two and seven nanometres. The areas of interest are as follows: the DDF gradients in the homogeneous samples, the gradients in the buried layers and the gradients of the bulk substrates. Table 4.1 lists the effective attenuation parameters for each of these areas.

Region	λ_a(nm)
Bulk carbon sample	2.43
Carbon substrate of C/Au/C structure	2.06
Bulk gold sample	0.81
Gold substrate in Au/C/Au Structure	0.93
Carbon sandwich layer in Au/C/Au Structure	1.46
Gold sandwich layer in C/Au/C Structure	1.45

Table 4.1 : Effective attenuation parameters from Figure 4.23

Comparison of the values shown in Table 4.1 with the calculated values for λ_a of 2.45nm in carbon and 0.84nm for gold, immediately reveals a very significant discrepancy between the transport theory attenuation parameters for the bulk materials and the effective attenuation parameters for each section of the tri-layer structure.

When examining these results, it is very important to remember that for comparison of the DDF gradient and the attenuation parameter to be valid, we must measure the DDF gradient for sufficiently long electron path lengths so that the so called 'no-

memory' condition is satisfied (i.e. $z \geq 2\lambda_{tr}$) . Only under these circumstances will the probability of electron detection be independent of its history. Unfortunately, in the case of the tri-layer simulation carried out in Figure 4.23, this criterion is unlikely to be met due to the relatively shallow depths involved. For 1keV electrons in carbon, $\lambda_{tr} \approx 10$nm and for gold $\lambda_{tr} \approx 2.2$nm (from Figures 3.3 and 3.6 respectively).

In the case of the bulk materials previously used for comparison with the transport theory results, a normal incidence simulation gives a DDF with a close to uniform gradient at all depths. As the angle of emission increases, so does the depth at which the DDF gradient becomes uniform, although in practice, even for a grazing emission angle this occurs at less than three nanometres for most homogeneous samples. Under these conditions, the memory effect does not play a significant roll in the formation of the DDF.

It is likely that that the no-memory condition is not met, and the history of any given electron has an influence on the probability of its detection in the region of the interfaces in the multi-layer structures studied here. This fact makes it difficult to compare the DDF gradients with theoretical attenuation parameters for sections of the DDF containing overlayers at shallow depths. If, for example, we compare the gradients of the carbon and gold substrates in Figure 4.23 we find a discrepancy of 15% from the theoretical value for carbon, and 10% for gold.

The close agreement between the Monte Carlo simulation and transport theory predictions in the case of homogeneous samples, establishes the validity of the simulated results. Furthermore, the Monte Carlo methods give the flexibility to simulate more complex structures that at present are beyond the capability of the transport theory approach. For the present simulation results, we have only been able to make a comparison with the asymptotic region of the lowest media and it would be useful to compare the results with an idealised multi-layer analytical model. Such analytical models are currently under development but pose considerable theoretical difficulties (Dwyer, 1994a).

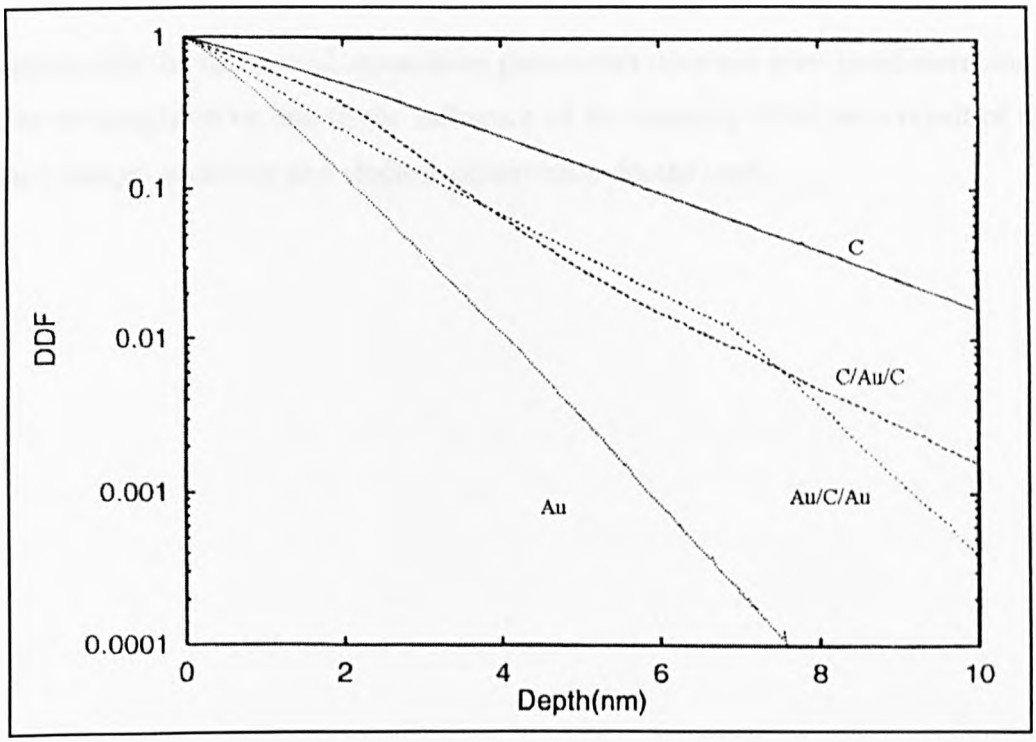

Figure 4.23 : Tri-layer depth distribution functions for 1keV electrons at normal emission angle. The interfaces between the layers are at 2nm and 7nm for both tri-layer samples

4.5 Conclusions

The statistical weights Monte Carlo program of Cumpson has been successfully extended to offer improved performance by the use of a compiled language, and also the simulation of heterogeneous multi-layered structures. Comparison of the multi-layer simulations with the original results of Cumpson shows excellent agreement.

The simulation handles sample geometry in terms of standard Cartesian co-ordinates and could therefore be extended to cope with a wide range of complex structures. Potentially, the software could include a graphical user interface to allow the analyst to define any required sample structure consisting of any material for which the required data can be obtained.

Transport theory equations have been used to produce expected attenuation parameters, which give comparable result to the Monte Carlo simulations in the case of heterogeneous samples. Comparison of DDF gradients produced by the tri-layer

simulation with the theoretical attenuation parameters does not give good correlation, and this is thought to be due to the influence of the memory effect as a result of the complex sample geometry and shallow observation depths used.

CHAPTER FIVE

EXPERIMENTAL APPARATUS

5.0 Introduction

This chapter describes the experimental apparatus forming the basis for the development of an angle-resolved analyser. The initial set-up is described, consisting of a commercially produced cylindrical mirror analyser (CMA) and electron gun, on a diffusion pump based vacuum system. Attention is then turned to the updating of apparatus, including the design and development of a new field emission electron gun, and a compact SEM detector. A commercially available (VG) high performance ion gun has also been added to the system to replace older Varian guns.

These developments pave the way for the angle-resolved spectrometer, and dramatically improve the potential of the instrument in terms of achievable resolution, flexibility and general performance. The designs of the replacement detector head, control electronics and software are substantial subjects, and will therefore be covered in subsequent chapters.

A schematic diagram of the basic components of the system is shown in Figure 5.1. This diagram will be referred to later in this chapter. Figure 5.1 shows the basic system after addition of the VG ion gun and the SEM detector. Details such as routing of the gas handling line, and traps in the vacuum lines have been omitted for clarity.

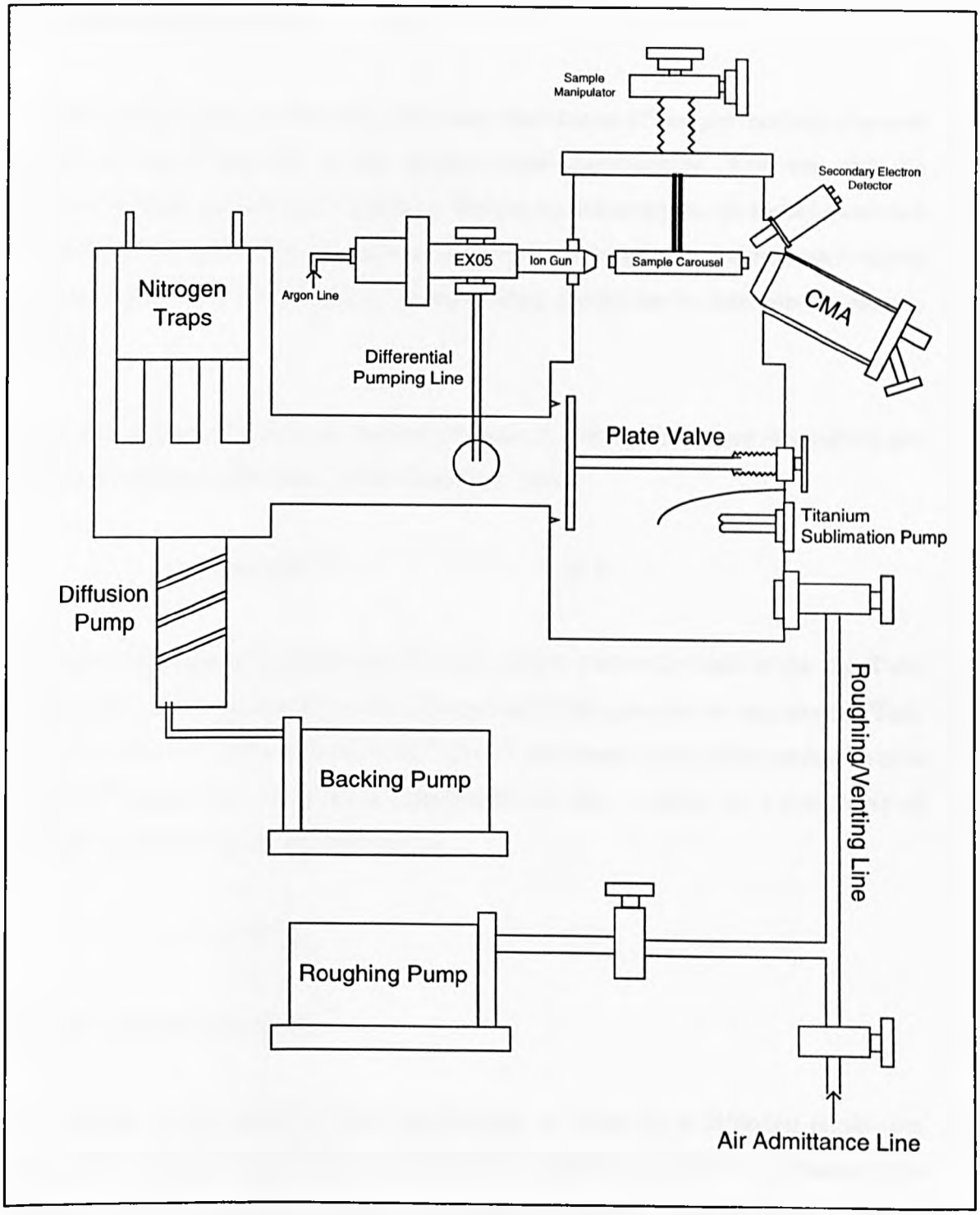

Figure 5.1 : Schematic of the vacuum system supporting Auger analysis

5.1 The Vacuum System

Electron spectroscopy, in common with most other forms of surface analysis, depends heavily on the cleanliness of the surface under examination. This can only be achieved by modern vacuum technology. For successful analysis, we must ensure that the level of vacuum is such that we can carry out a meaningful experiment well within the time taken for a monolayer of contaminating molecules to form on the sample surface.

According to kinetic theory, the number of atoms Z, hitting a unit area of a surface per unit time is given by (Prutton, 1994; Chambers, 1989) :

$$Z = bp(M_tT)^{-1/2}, \qquad (5.1)$$

where p is the ambient gas pressure, M is the relative molecular mass of the gas, T the temperature of the gas and b a universal constant. If the pressure is measured in Torr, then b = 3.51×10^{26} molecules $m^{-2}\ s^{-1}\ K^{1/2}\ Torr^{-1}$. An atomic monolayer corresponds to about 10^{15} atoms cm^{-2}, so at room temperature, the time t, taken for a monolayer of Nitrogen to form is given (in seconds) as

$$t = 3 \times 10^{-6} p^{-1}, \qquad (5.2)$$

where p is measured in Torr.

The vacuum system used in these experiments is based on a diffusion pump (see Figure 5.1), which if unaided could reliably be expected to achieve a pressure in the region of 10^{-7} Torr. At this pressure, as equation 5.2 shows, a monolayer will form on the surface in 30 seconds; not even long enough to transfer a sample from the position in which it is cleaned to the spectrometer in the present system.

In order to carry out a typical Auger experiment (which may take several hours), vacuums in the region of 10^{-9} Torr are required. To achieve this, the diffusion pump is

used in conjunction with both a titanium sublimation pump and a nitrogen cold trap. This regime of operation is referred to as Ultra High Vacuum (UHV).

As well as sample cleanliness, there is another very important requirement for UHV; two of the major developments described in this thesis, namely the field emission electron gun and channel plate electron detector, depend on a good vacuum for reliable operation. If channel plates are operated in a poor vacuum, an internal plasma can cause bombardment of the internal surface of the channel, reducing its secondary electron yield. Furthermore, in contrast to a thermionic electron gun which would operate quite reliably at 10^{-4} Torr, field emission guns need a much better vacuum of at least 10^{-8} Torr to avoid the risk of internal damage due to electrostatic discharge.

The main vacuum system components are shown schematically in Figure 5.1. The system is split into two separate chambers which can be isolated by a manually operated plate valve. This feature allows the diffusion pump to be in constant operation, even when the main chamber is vented to allow samples to be loaded or other work to take place.

When the samples have been loaded, the chamber can initially be pumped down via the roughing line (see Figure 5.1) to high vacuum ($1x10^{3}$ mbar approximately) before opening the plate valve to allow the diffusion pump to continue evacuation of the chamber. When the diffusion pump has achieved a vacuum in the region of 10^{-7} mbar, further improvements in vacuum can be achieved by use of either the titanium sublimation pump (TSP) or by baking the system. The chamber is baked to a temperature of 180^{O}C for 12 hours to 24 hours depending on the level of internal contamination. This procedure allows vacuums of $2x10^{-10}$ mbar to be routinely achieved and maintained. Vacuum levels are continuously monitored by an ion gauge in the main chamber (used for reading pressures lower than 10^{-3} mbar) and pirani gauges in the roughing and backing lines (for pressures from atmospheric down to 10^{-3}mbar).

5.2 Sample Insertion and Positioning

As mentioned previously, sample insertion involves venting the main analysis chamber and therefore requires the main analytical instruments to be shut down. Samples are fixed to a stub, usually with silver paste, which is then loaded into a carousel capable of holding 12 such stubs.

The carousel is in turn attached to a sample manipulator, allowing movement in three orthogonal directions (x,y,and z) so that different areas of the samples can be brought into the field of view of either the analyser, or the ion gun. Lateral motion in the x and y directions, and translation in the z direction can be accomplished over a range of about 25mm, whilst axial rotation is possible through a full 360^{o} to allow any sample to be selected from the carousel.

5.3 The Electron Energy Analyser

A more detailed description of the electron energy analyser used in the present work will now be presented.

5.3.1 General Description

The development of the angle resolved analyser is based on a single pass Cylindrical Mirror Analyser (CMA) made by Varian Associates, probably in the 1970s. The position of the CMA in the analysis chamber is shown in Figure 5.1. The CMA was introduced in chapter two, but before further design work is described, a more rigorous examination of its properties will be carried out.

Figure 5.2 shows a schematic cross section of the CMA analyser. The principle of operation is as follows: The inner cylinder of the analyser is grounded, and the outer is held at a potential –V, this potential difference sets up an electrostatic field between the two cylinders (radii $\mathbf{r_1}$ and $\mathbf{r_2}$ respectively). An electron gun, positioned co-axially with the inner and outer cylinder, produces a beam of electrons with sufficient kinetic

energy to induce Auger emission at the desired energy (see chapter 2 for a description of Auger emission).

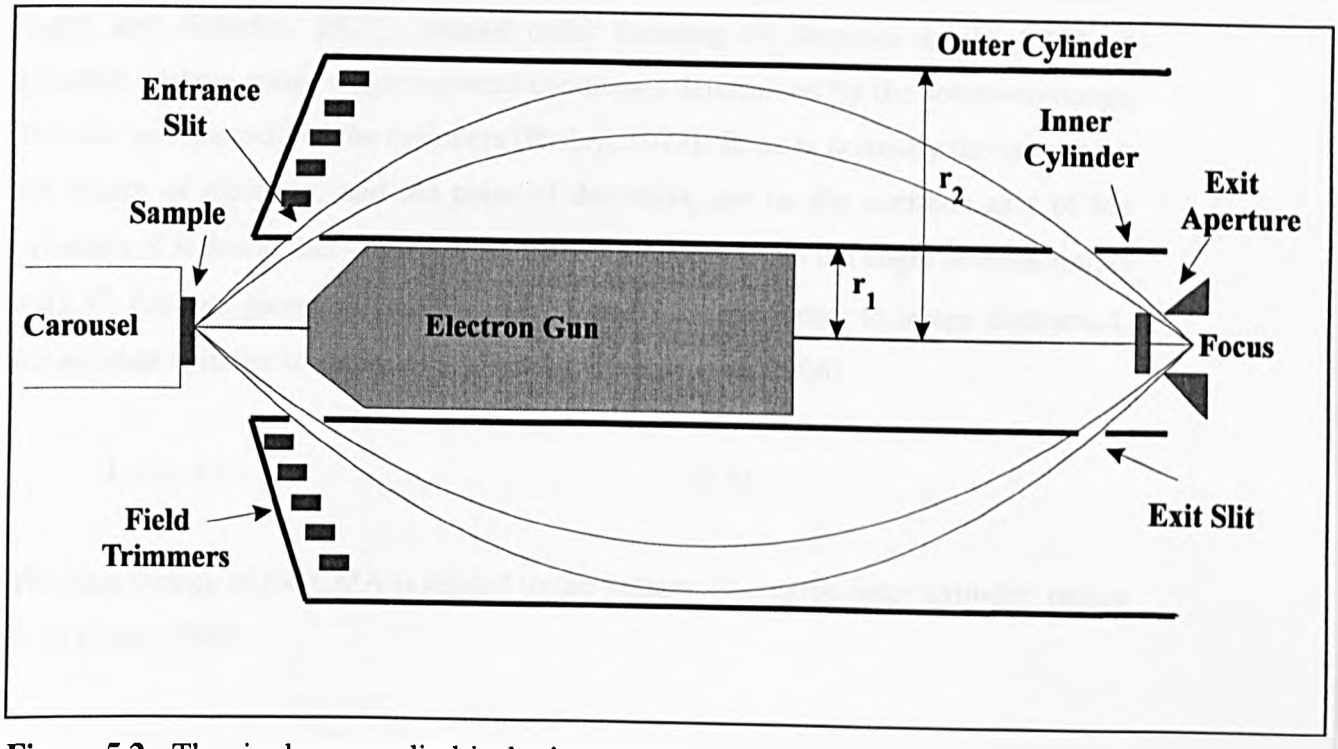

Figure 5.2 : The single pass cylindrical mirror analyser

Electrons from the point source of excitation leave the sample and move in field free space until they enter the space between the inner and outer cylinders through the entrance slit (width **w**). The field the electrons are subjected to, forces them to adopt parabolic trajectories, the paths of which depend on the relationship between the electron kinetic energy and the voltage on the outer cylinder. Electrons of an energy corresponding to the pass energy of the analyser are deflected in such a way that they pass through the exit aperture of the analyser, and are brought to a focus at a point. Electrons with an energy much greater than the pass energy will be deflected less by the field and hit the outer cylinder, similarly, electrons of lower energy than the pass energy will fall short of the exit slit and strike the inner cylinder.

5.3.2 Focusing Conditions

Much work has been done to characterise the CMA in general (Sar El, 1967/1968; Eagen and Sickafus, 1977). Second order focusing of electrons in the CMA is achieved under a range of geometrical conditions determined by the source-to-image distance and the radii of the cylinders (Risley, 1972). If, as is normally the case, both the source of electrons, and the point of detection, are on the common axis of the cylinders, it is found that second order focusing occurs when the angle of emission, $\alpha = 42.3^{\circ}$. For this geometry (see Figures 2.5 and 5.2), the source to image distance, L for an inner cylinder of radius r_1 is given by (Hafner *et al*, 1968)

$$L = r_1\, 6.13 \qquad (5.3)$$

The pass energy of the CMA is related to the voltage, V_0 on the outer cylinder, radius r_2 by (Seah, 1989)

$$E = 1.3e(\ln r_2/r_1)^{-1}.V_0 \qquad (5.4)$$

For this analyser, $r_1 = 27$mm, $r_2 = 61$mm, yielding

$$E = 1.6eV_0 \qquad (5.5)$$

as the relationship between outer cylinder voltage and pass energy.

The CMA accepts electrons with emission angles over the range $\Delta\alpha = 42.3^{\circ} \pm 6^{\circ}$, this produces a minimum trace width which is not on the axis of symmetry, but occurs a distance, r, before it at (Bishop *et al*, 1972; Sar El 1970)

$$r = 5.28r_1(\Delta\alpha)^2 \qquad (5.6)$$

where $\Delta\alpha$ is the divergence of the incident electrons. The minimum trace width, ω is then given by

$$\omega = 7.76 r_1 (\Delta\alpha)^3 \qquad (5.7)$$

This leads to the important conclusion that the energy resolution of the CMA can be improved by placing an annular slit of width ω coincidentally with the position of the minimum trace width (a distance r from the focus).

5.3.3 Energy Resolution of the CMA

The energy resolution of a spectrometer is defined as either relative or absolute. Absolute resolution is given as either ΔE, the full width at half maximum (FWHM), or ΔE_b, the base width of such a peak (or base resolution). Relative resolution is defined as R, the ratio of the electron energy spread to their kinetic energy, E_0. For an ideal anlyser $\Delta E = \Delta E_b/2$, where for small values of δα, ΔE_b is given by (Sar El, 1967; Sar El , 1970)

$$\frac{\Delta E_b}{E_0} = 5.50(\delta\alpha)^3 \qquad (5.8)$$

if an annular slit of width W is placed at the position of the minimum trace width, the resolution becomes (Rivière, 1990)

$$\frac{\Delta E_b}{E_0} = \frac{0.18W}{r_1} + 1.375(\delta\alpha)^3 \qquad (5.9)$$

A great deal of work has been carried out in order to improve the CMA analyser and develop it for a wide variety of applications (Vasina *et al*, 1979; Bosch *et al.*, 1984; Huang *et al.*, 1993).

5.3.4 The Varian CMA

The CMA used in this study has been characterised by El-Bakush (D.Phil. thesis 1994). Instrumental factors affecting the signal were investigated, including the detector efficiency, magnetic fields and internal scattering.

The geometry of the Varian CMA used in this study is limited by certain physical factors, perhaps the most important of which is the length of the cylinders themselves. In order to allow access to the sample by other instruments, such as secondary electron detectors and ion guns, the CMA cylinders end just in front of the entrance slit. Although this provides a convenient geometry for experiments, allowing the sample to be moved and observed freely, it has very important implications on the analyser design.

Figure 5.3 shows a simulation of the electrostatic fields within the CMA carried out with the SIMION electrostatic lens simulation package (to be described in Chapter six). The simulation shows the contours of the electrostatic field within the analyser, and although only a crude model, illustrates that the fields are not uniform in the area of the entrance slit.

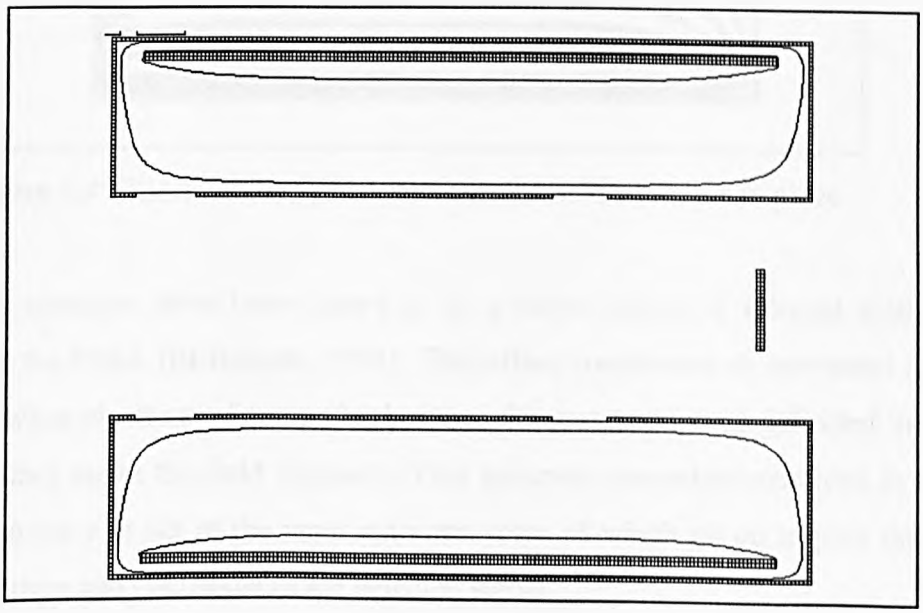

Figure 5.3 : Electrostatic field contours in the CMA without field trimmers

Clearly this distortion will have a detrimental effect on the performance of the analyser, producing energy and emission angle dependent defects in spectra. The problem is solved by the introduction of extra electrodes known as field trimmers (see Figure 5.2). In the Varian CMA, there are five field trimmers, positioned in front of the entrance slit and equally spaced between the inner and outer cylinders. The field trimmers are biased with increasing potentials between ground and the voltage on the outer cylinder, and have the effect of drawing the contour lines forwards, making them approximately uniform in the area of the entrance slit. This effect is shown by simulation in Figure 5.4; again, the geometry of the simulation is crude, but serves to illustrate the technique.

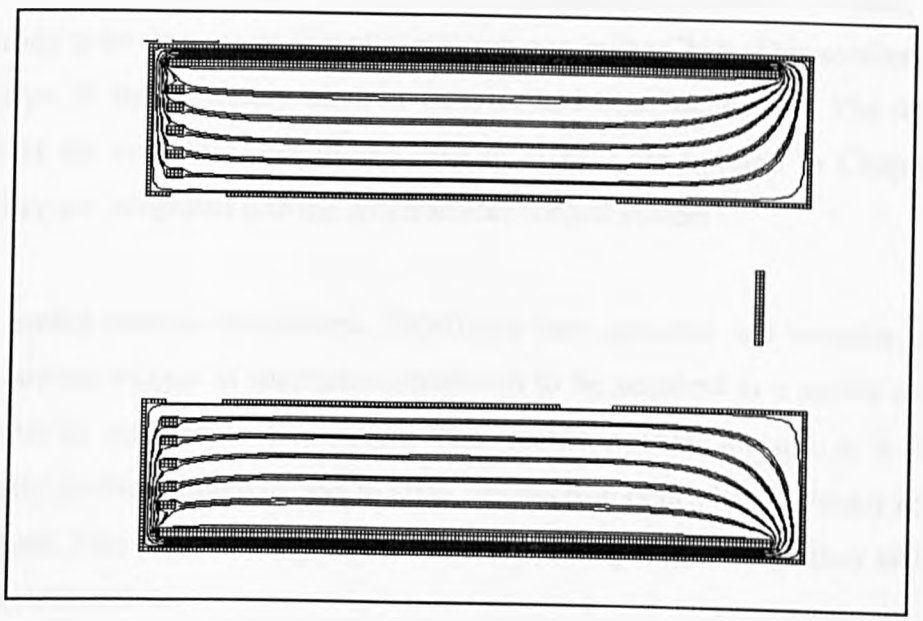

Figure 5.4 : Electrostatic field contours with field trimmers in place

These field trimmers have been shown to be a major source of internal scattering signal from the CMA (El-Bakush, 1994). This effect contributes an unwanted signal to spectra, when electrons of energy higher than the pass energy are deflected in such a way that they strike the field trimmers. This generates secondary electrons in close proximity to the exit slit of the inner cylinder, some of which go on to pass through the exit aperture and contribute to the detected signal.

Although the field trimmers are an important feature of the analyser, the simulations carried out in subsequent investigations will assume an ideal CMA (i.e. with extended cylinders). Furthermore, the exit aperture which has also been shown to contribute to

the internal scattering signal, will not be included here. This will simplify the simulations, yet have no significant effect on the results within the limited accuracy of the current SIMION software. If more detailed examination of a particular area of the analyser was required, a large scale simulation of that region would allow more accurate results to be produced (on the current scale, one pixel relates to 1mm).

5.4 Secondary Electron Detector

5.4.1 Background and Theory

A secondary electron detector has been designed to allow scanning electron microcopy to be carried out using the electron gun in the CMA. This section describes the design of the secondary electron detector and head electronics. The design and testing of the computer control and display aspects are covered in Chapter seven, since they are integrated into the spectrometer control system.

The scanning electron microscope (SEM) is a very powerful and versatile analytical tool, allowing images of submicron resolution to be acquired in a matter of seconds with little or no sample preparation. This makes it ideal for use as a means to accurately position a sample, and to align the electron optics before Auger analysis is carried out. This is particularly useful since a scanning Auger image may take several hours to accumulate.

Although the term SEM refers to a family of analytical methods, in this instance we are only using one mode of operation; the emissive mode. In this mode electrons emitted from the sample as a result of electron bombardment are collected to form an image. The electrons of particular importance for this measurement are those of low energy (<50eV) since they provide information about surface topography and can be used to form images resembling those from an optical microscope, but with much better resolution.

The design upon which the detector described in this section is based (the so-called Everhart-Thornley detector) is widely used, and numerous commercially available units are based on the same basic principle. Figure 5.5 shows a schematic diagram of the complete detector head.

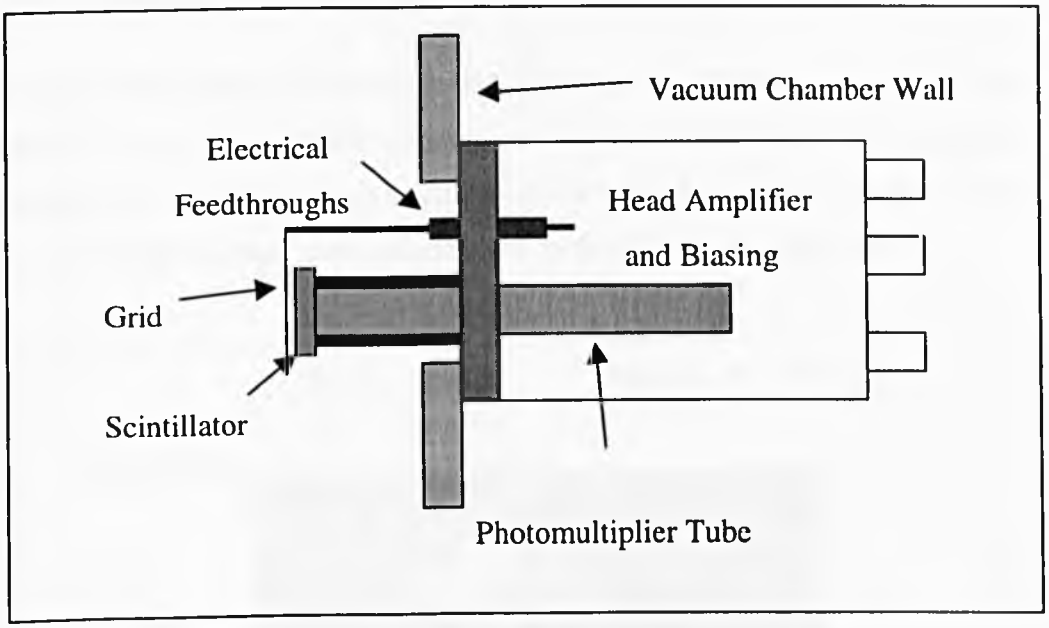

Figure 5.5 : The secondary electron detector

A positively biased grid (forming a 'Faraday cage') accelerates electrons towards a scintillator consisting of a phosphor screen formed on a glass plate with a thin film of aluminium. A bias voltage is applied to the aluminium film, effectively accelerating the electrons towards the phosphor so that they strike with sufficient kinetic energy to emit photons. The photons pass through a glass window and are collected by a photomultiplier tube which, in conjunction with a sensitive pre-amplifier, produces a voltage proportional to the incident energy of the electrons. This voltage is passed onto the signal processing electronics, which can be used to form an image if the electron beam is scanned in a raster pattern over the sample.

The major difference between the design presented here, and all currently available commercial equivalents lies in the transmission of the photons between the scintillator and the photomultiplier. Conventionally, photons are conducted to the photomultiplier by total internal reflection in a light guide (similar to transmission in a fibre-optic cable). This allows the scintillator to be placed inside the vacuum chamber conveniently close to the sample under examination. This design differs from

commercially available equivalents in that the photomultiplier tube protrudes into the vacuum chamber surrounded by a glass guide tube to provide the required seal between the vacuum and atmospheric pressure.

5.4.2 The Scintillator/Photomultiplier

The detector head consists of two parts, the scintillator and Faraday cage which are permanently fixed to the vacuum system, and the removable section housing the photomultiplier tube, bias network and head amplifier. Figure 5.6 shows the scintillator and Faraday cage, mounted on a 2¾ inch UHV compatible flange.

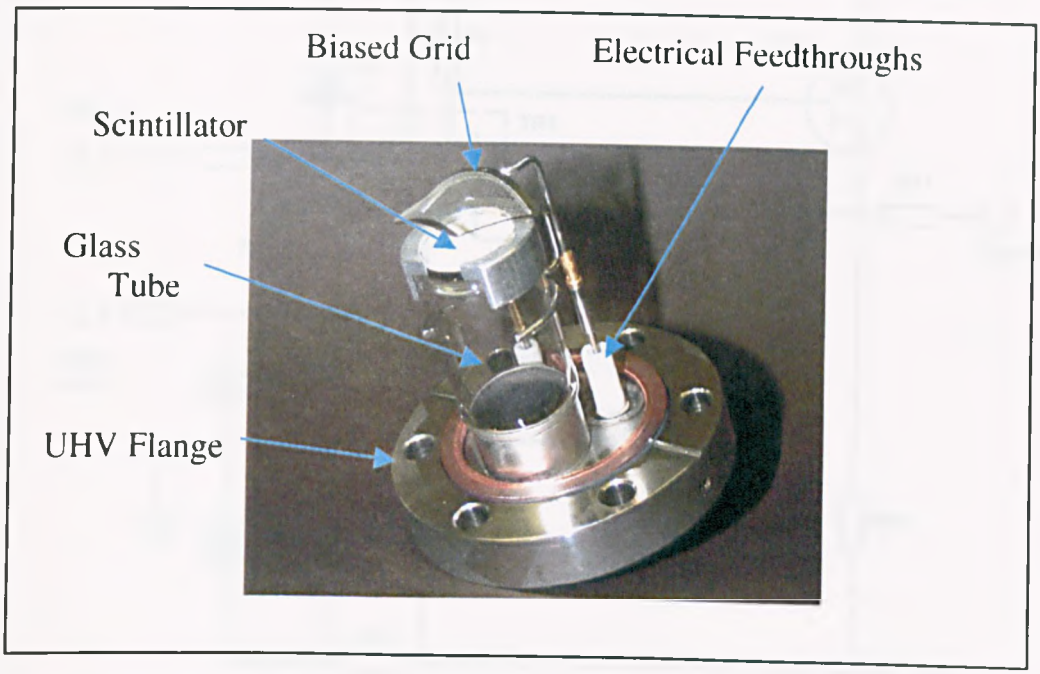

Figure 5.6 : The scintillator and Faraday cage

The scintillator was fabricated on a 1mm thick 20mm diameter glass disk onto one side of which is deposited a semi-transparent layer, by evaporation of 0.01g of aluminium for 30 seconds (the evaporation apparatus will be discussed in Chapter six).

5.4.3 The Head Amplifier

The output current from the photomultiplier tube must be amplified in order to give a voltage signal suitable for driving either a CRT display, or a digital to analogue converter if computer control is used. Figure 5.7 shows the circuit diagram for a suitable amplifier, based on a JFET device. Note that both power supplies are decoupled with 20μF 40V electrolytic capacitors for low frequency ripple rejection, and 100nF ceramic capacitors for high frequency decoupling (these are omitted from the circuit diagram for clarity).

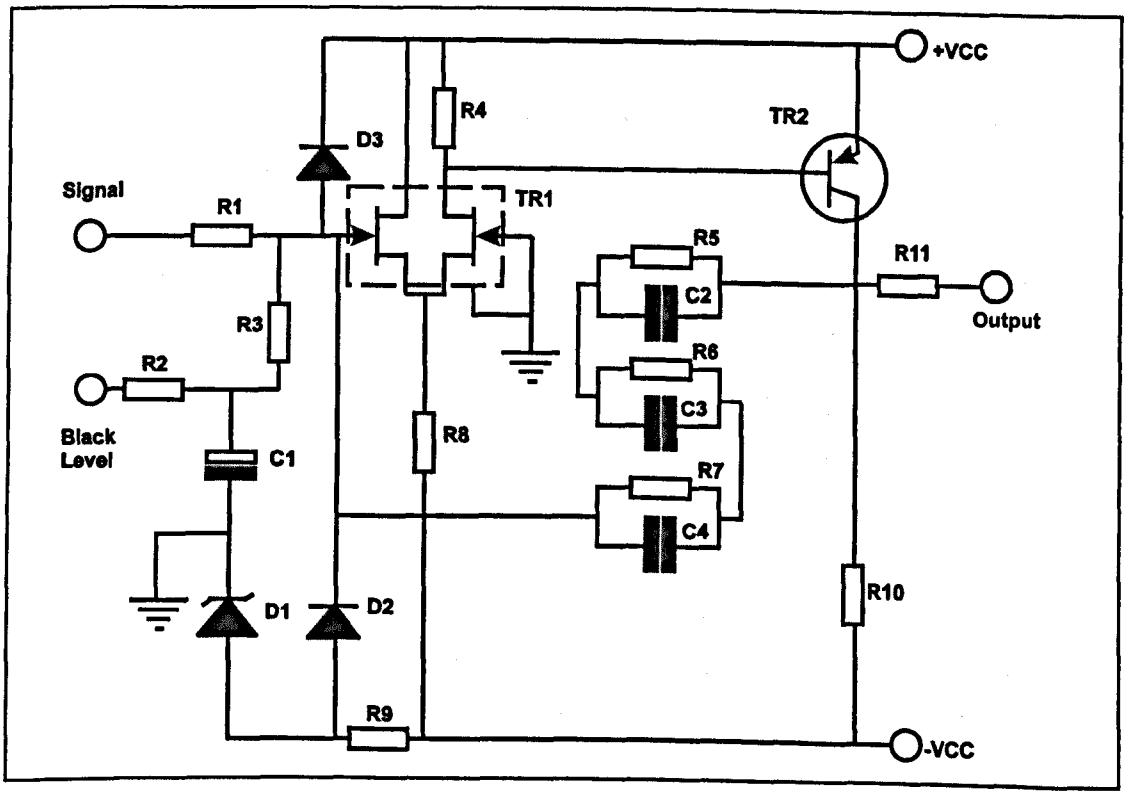

Figure 5.7 : Photomultiplier head amplifier

Component values are as follows : D3, D2 JPAD50; D1 BZX79C5V6; C1 20μF, 40V; C2, C3, C4 2.2pF; R1 2.7kΩ; R2 1.2kΩ, R3 100kΩ; R4 560Ω; R5, R6, R7 10kΩ; R8 4.7kΩ; R9 10kΩ; R10 1.2kΩ; R11 100Ω; TR1 2N5912, TR2 2N3906.

The head amplifier circuit consists of a long tailed pair amplifier based on a single package dual J-FET transistor, with an emitter follower stage to provide current drive to the video output. The input to the amplifier is protected via a series resistor (R1) and two diodes D3 and D2, which prevent damage to the J-FET by limiting the

maximum gate voltage to this device. The J-FET used in this design can withstand a larger forward gate voltage then reverse. For this reason a zenor diode (D1) is used to limit the maximum negative swing to –5.6V.

The negative feedback network formed by R5,R6,R7 and C2,C3,C4 essentially gives a low pass response, but as frequency increases the phase shift produced by this network approaches 180^{o} producing a positive feedback effect. Since the gain of the amplifier is less than one at these frequencies, the positive feedback does not lead to oscillation, but instead just produces a peak in the frequency response. In practice this leads to sharper edges when the signal is used to form an image.

A capacitively smoothed black level input provides a DC offset to the signal. This is useful for manual adjustment to bring the signal into the required voltage range for driving an oscilloscope, or to ensure that the signal falls evenly within the dynamic range of the analogue to digital converter in the case of computer imaging. At the moment this level is set manually, but it is envisaged that computer control would be more suitable in a fully automated imaging system, in which case a digital to analogue converter would be used to drive this input either directly or via a voltage controlled regulator.

The resistive network used to provide bias voltages to the photomultiplier tube is shown in Figure 5.8. The circuit is straightforward, and is constructed by soldering devices directly to the pins on the socket. All the resistors are the same, giving a linearly increasing potential across the dynodes. The current flowing through the bias network determines the maximum current that can be drawn by the dynodes and therefore the maximum output of the photomultiplier. The completed head amplifier and bias network is shown in Figure 5.9.

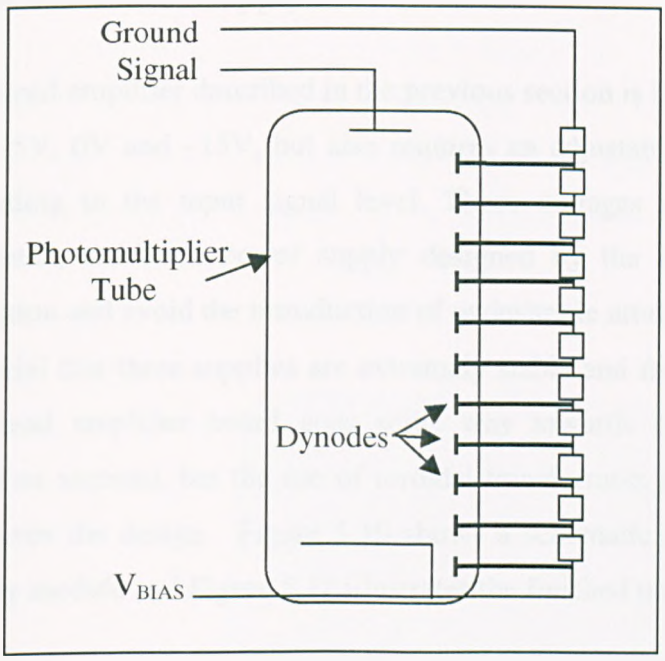

Figure 5.8 : Photomultiplier bias network – All resistors are 200 Ohm ½ Watt

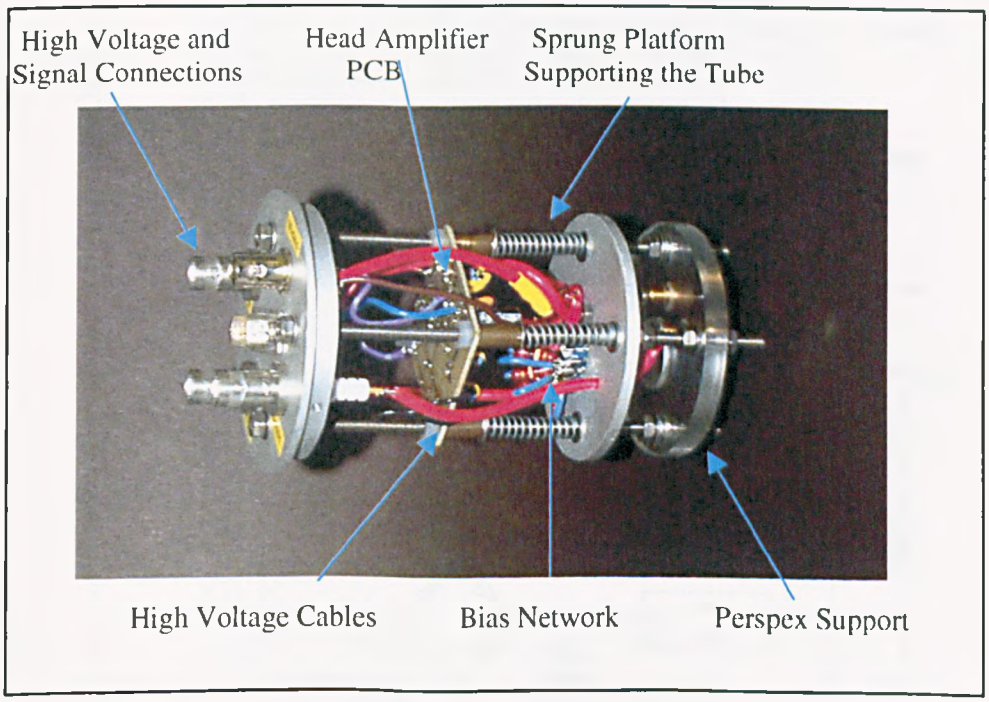

Figure 5.9 : The complete SEM detector head

5.4.4 The Power Supply

The head amplifier described in the previous section is based around a dual rail supply of +15V, 0V and −15V, but also requires an adjustable input to set the black level according to the input signal level. These voltages are provided from a voltage regulated, stabilised power supply designed by the author. To ensure low noise operation and avoid the introduction of undesirable artefacts into the video output it is essential that these supplies are extremely stable and free from noise. Decoupling on the head amplifier board goes some way towards achieving this condition (see previous section), but the use of toroidal transformers and voltage regulators further improves the design. Figure 5.10 shows a schematic circuit diagram of the power supply module and Figure 5.11 illustrates the finished unit in the rack case.

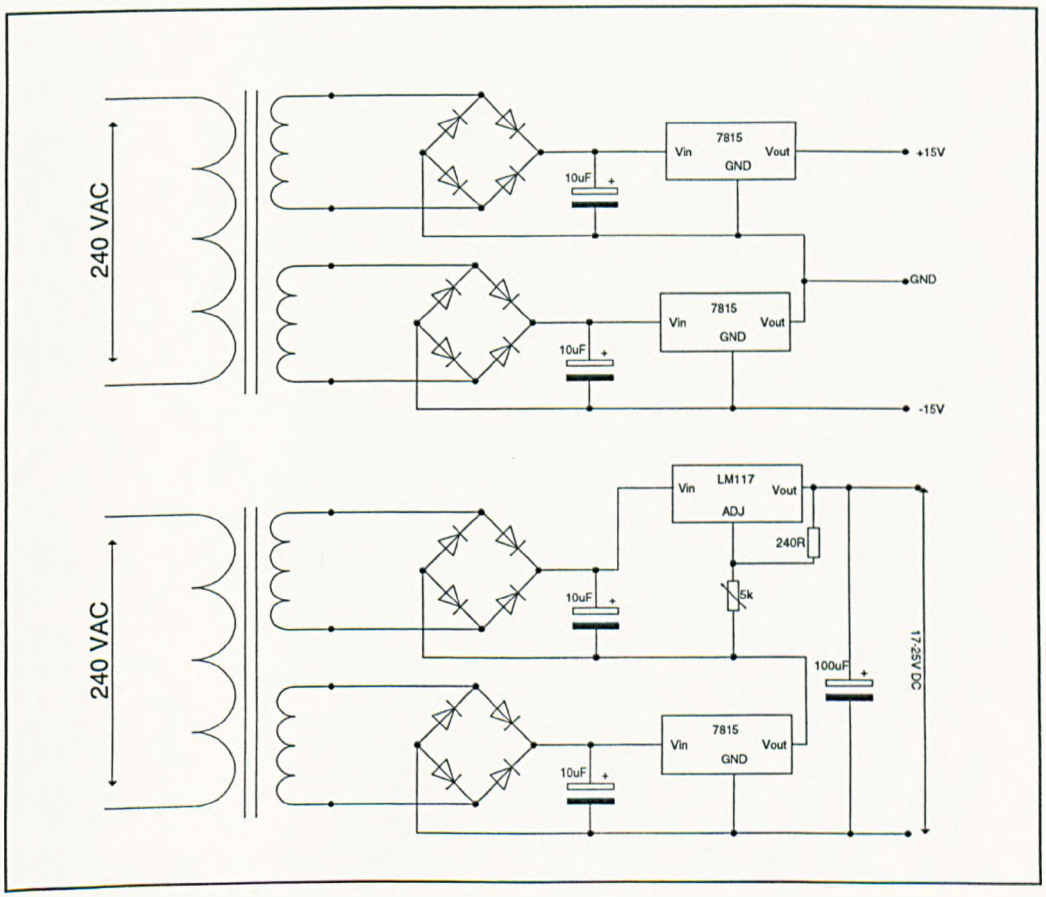

Figure 5.10 : Circuit diagram of the low voltage power supply to the SEM head amplifier

The secondary electron detector requires three high voltage supplies to bias the grid, scintillator and photomultiplier respectively. These are provided by standard bench supplies manufactured by Fluke. Unlike the low voltage supplies which are capacitively filtered to reduce noise and ripple, these bias voltages are left unfiltered and we therefore rely on a highly stable output from the supply itself.

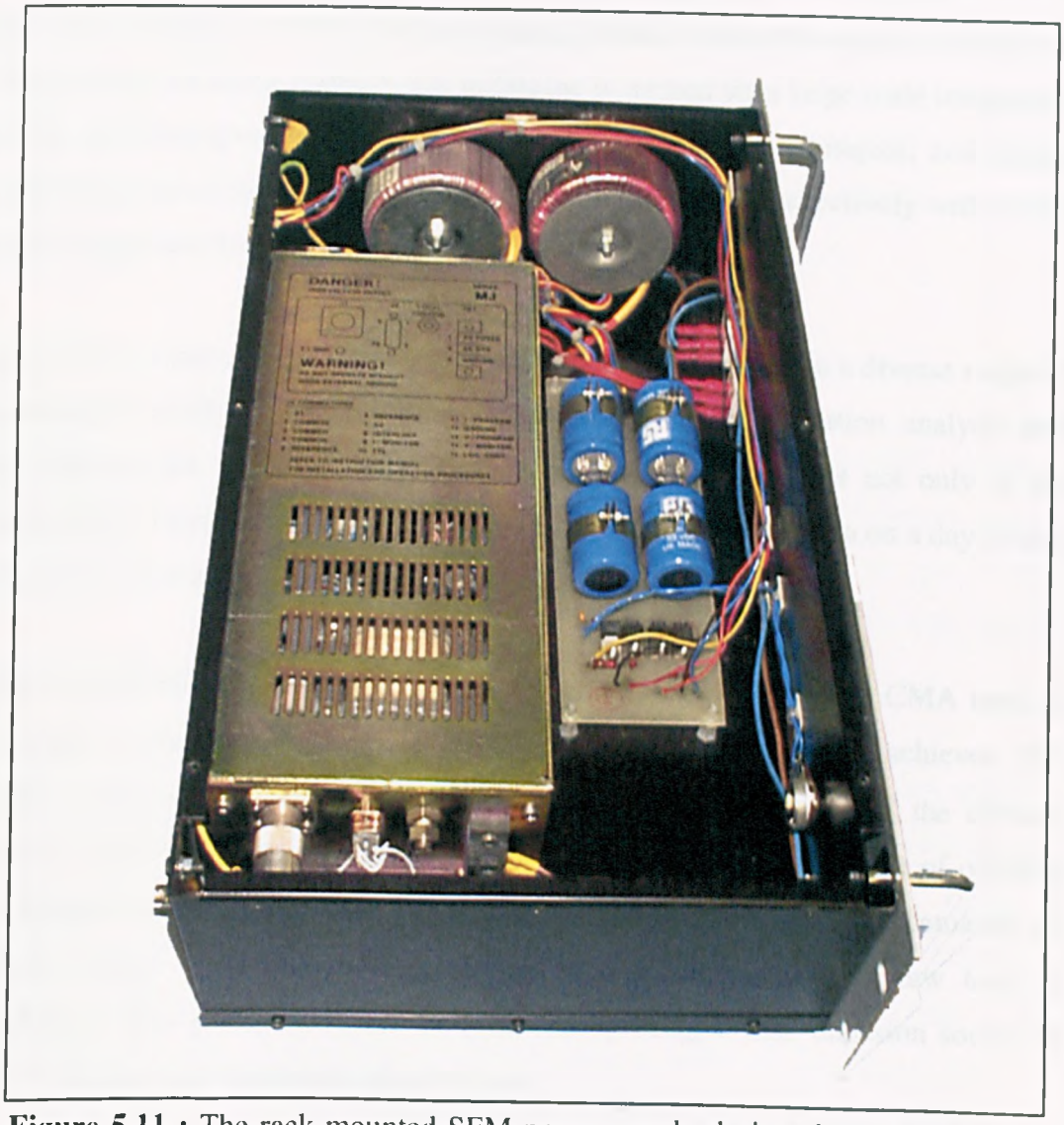

Figure 5.11 : The rack-mounted SEM power supply designed and built by the author

5.5 The Field Emission Gun

5.5.1 Introduction

The process of miniaturisation is of huge importance to semiconductor designers and therefore surface analysis as a whole. In 1990, the minimum feature size available using the conventional fabrication techniques was about 1μm this is known as very large scale integration (VLSI). At the moment, feature sizes in the region of hundreds of nanometres are being realised, this technique is termed ultra large scale integration (ULSI). As fabrication processes improve, so must analysis techniques, and indeed the development of the new processes could not be carried out effectively without the facility to appraise the results.

Auger electron spectroscopy (AES) is used routinely by analysts in a diverse range of applications including semiconductor failure analysis, contamination analysis and bond pad and die attach analysis. These techniques are important not only in the development of new fabrication methods, as described above, but also on a day-to-day basis as part of a quality control system.

Using a thermionic electron gun, such as the gun fitted to the Varian CMA used in this study, electron beam spot sizes in the region of 5μm could be achieved (El-Bakush, 1994). Obviously this would not be suitable for analysis of the devices currently under production. In fact, the design of a spectrometer capable of offering the required spatial resolution for resolving defects in submicron ICs, demands an electron source which can provide beamwidths in the order of a few tens of nanometres. This performance can be achieved by using a field emission source in place of the existing thermionic electron gun.

Field emission guns (FEGs) have been commercially available since VG launched their HB200 FEG in 1973, and at the same time Walter and Cotes introduced a similar FEG in the USA. However, field emission based columns were not in common use in analytical microscopy until the early 1990s. This was partly because the early FEGs relied on cold field emission, which produced a small spot size but very low beam currents (typically 1 to 5nA). More recent FEGs operate on the principle of Schottky

field emission, allowing much higher beam currents than the cold field emission guns (up to 200nA).

In this study, a field emission electron gun has been designed to replace the tungsten filament thermionic gun in the Varian CMA. In essence, a FEG has two main advantages over a thermionic gun: higher brightness and lower energy spread. In practice, FEGs are capable of producing spot sizes as low as 1nm, making them suitable for analysis of ULSI devices. Although a variety of FEGs are available commercially, in order to be integrated into the CMA, the gun must be designed to exact dimensions to fit into the inner cylinder of the analyser, and with an appropriate socketing arrangement to provide high voltage electrical connections.

5.5.2 The Miniature FEG for the CMA

The present field emission column is a high-resolution all-electrostatic design to replace the Varian thermionic column. The main physical features of this column are identical to those of the Varian design, measuring approximately 95 mm in total length, and 42 mm in diameter. The focal point of the column is 12 mm from the end nose of the column.

The column comprises a novel pre-aligned Schottky emitter and extractor, which is easily fitted and does not require outside alignment. The focusing action is achieved by an 'einzel' lens as illustrated in Figure 5.12 (Orloff and Swanson, 1979). A schematic of the whole column is shown in Figure 5.13, and a photograph of the assembled column is given in Figure 5.14. The high voltage feedthrough of the original column has been re-designed to take account of the extra electrical connections of the emitter and stigmator. Figure 5.15 depicts the calculated electron beam diameter versus energy for a beam current of 20nA. These calculations are obtained with the help of a software package (SHOP) developed by York Electron Optics Ltd. One property of this column is the ability to focus at low energy, down to 150V. This is useful for carrying out studies of electron loss spectroscopy, and may also prove useful for choosing the appropriate energy for the study of semi-insulating materials (El-Gomati, 1998).

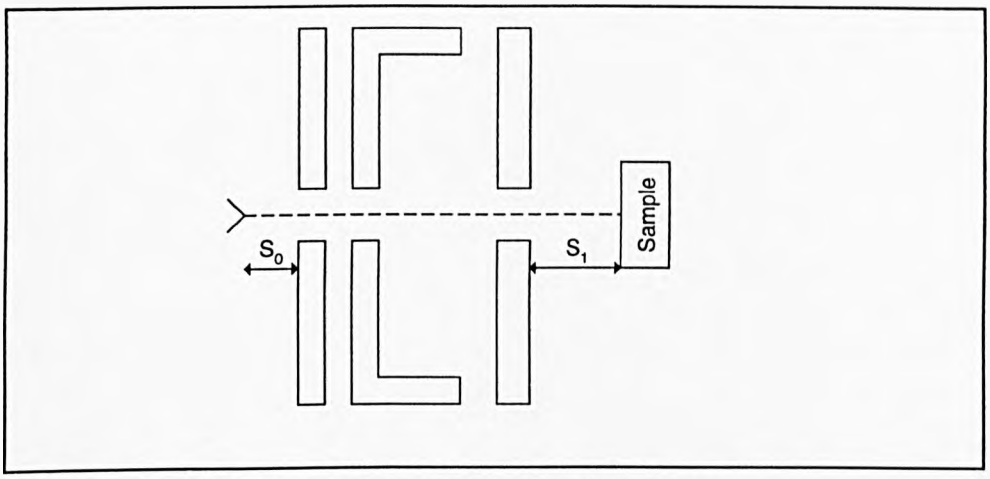

Figure 5.12 : The 'einzel' lens of Orloff and Swanson

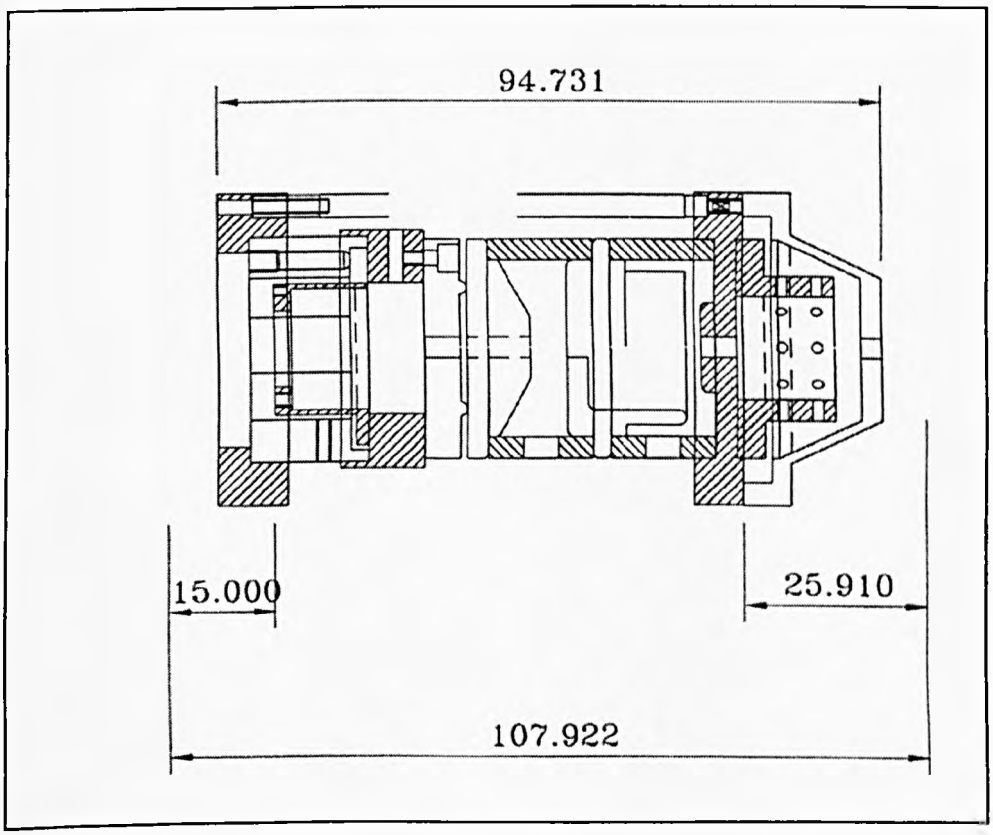

Figure 5.13 : Schematic diagram of the electron column (dimensions in mm)

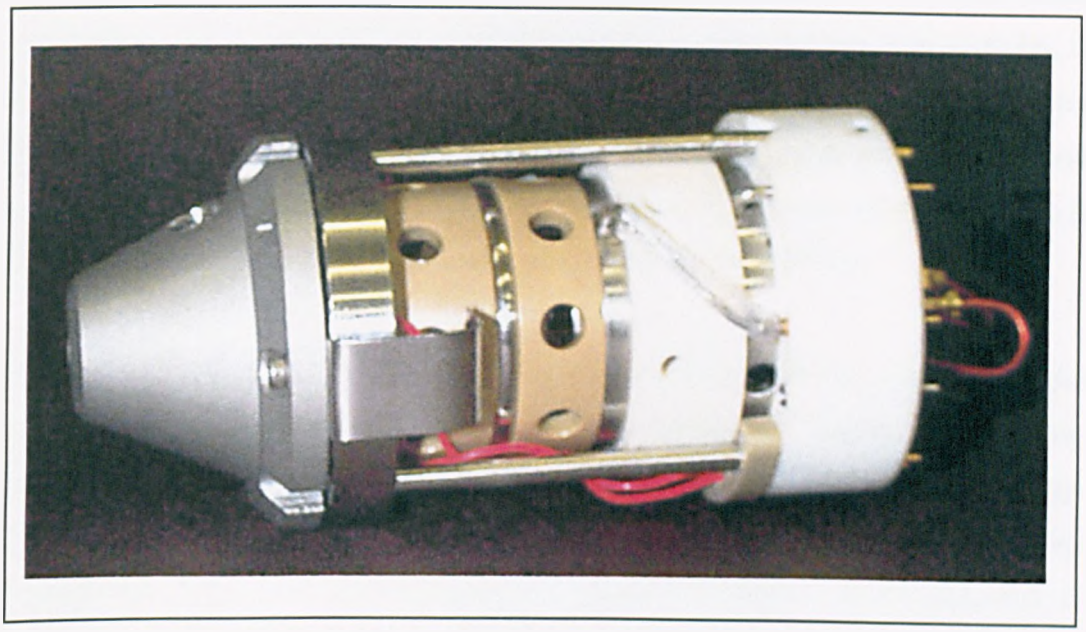

Figure 5.14 : Photograph of the assembled electron gun as used in the present analyser

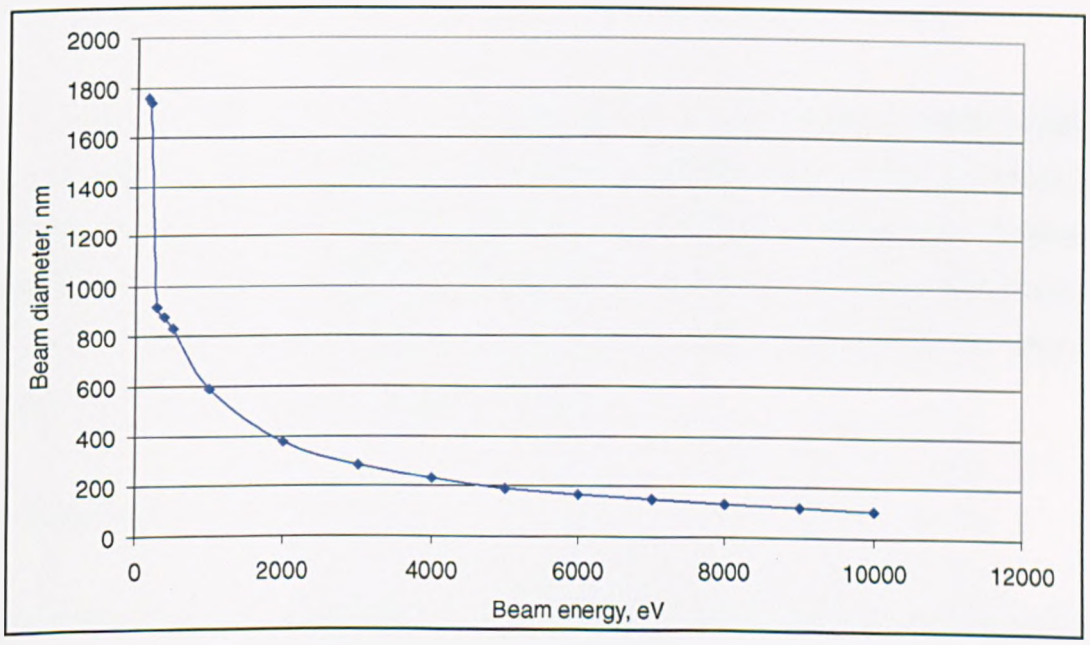

Figure 5.15 : Calculated graph of the variation in electron beam diameter of the field emission gun with electron energy

The high voltage power supply unit is a commercial design by ACCEL Power Supplies Ltd., UK. This unit supplies the Schottky emitter heating current 0-3A, the extractor voltage 0-10keV, the lens voltage 0-25kV and the accelerating electron energy 0-25keV. Although the supply is capable of 25keV generation, the insulation of the CMA limits operation to a maximum of 10keV. This covers the useful range for the maximum Auger electron signal as suggested by Bishop (1983).

The whole column is made of non-magnetic stainless steel LN316, machinable glass ceramic (MACOR) and PEEK (ICI Advanced Materials), another high voltage insulator. It has been fabricated in the departmental mechanical workshop. The pre-aligned Schottky emitter comprises a suppresser electrode, which houses the ceramic assembly carrying the hairpin filament on which the single crystal tungsten 100 emitter is mounted. The emitter is doped with Zr to lower its work function from 4.6eV to 2.8eV, and this also acts to confine the angular emission from 30° to 12°, increasing the electron beam brightness. A pre-aligned extractor is added to the suppresser, and the emitter is aligned with an aperture in the extractor measuring 300μm. The University of York are in the process of patenting this design.

The pre-aligned extractor makes the replacement of the emitter a much simpler exercise than comparable guns, which require a separate UHV system in which to perform this operation, or those equipped with an alignment mechanism. External alignment would be extremely difficult to achieve in a CMA due to the geometry of the gun housing. The author has replaced the emitter successfully on several occasions as part of the initial testing of the gun.

5.5.3 Scanning Unit

Control of electron beam position is essential both to allow accurate positioning of the beam, and to facilitate the collection of scanned Auger and secondary electron images. To achieve this, a dedicated control unit has been developed with which the electron beam can be swept in two directions by the application of variable voltages to an octopole deflector at the end of the electron column (El-Gomati, 1993). A photograph of the scanning unit is shown in Figure 5.16.

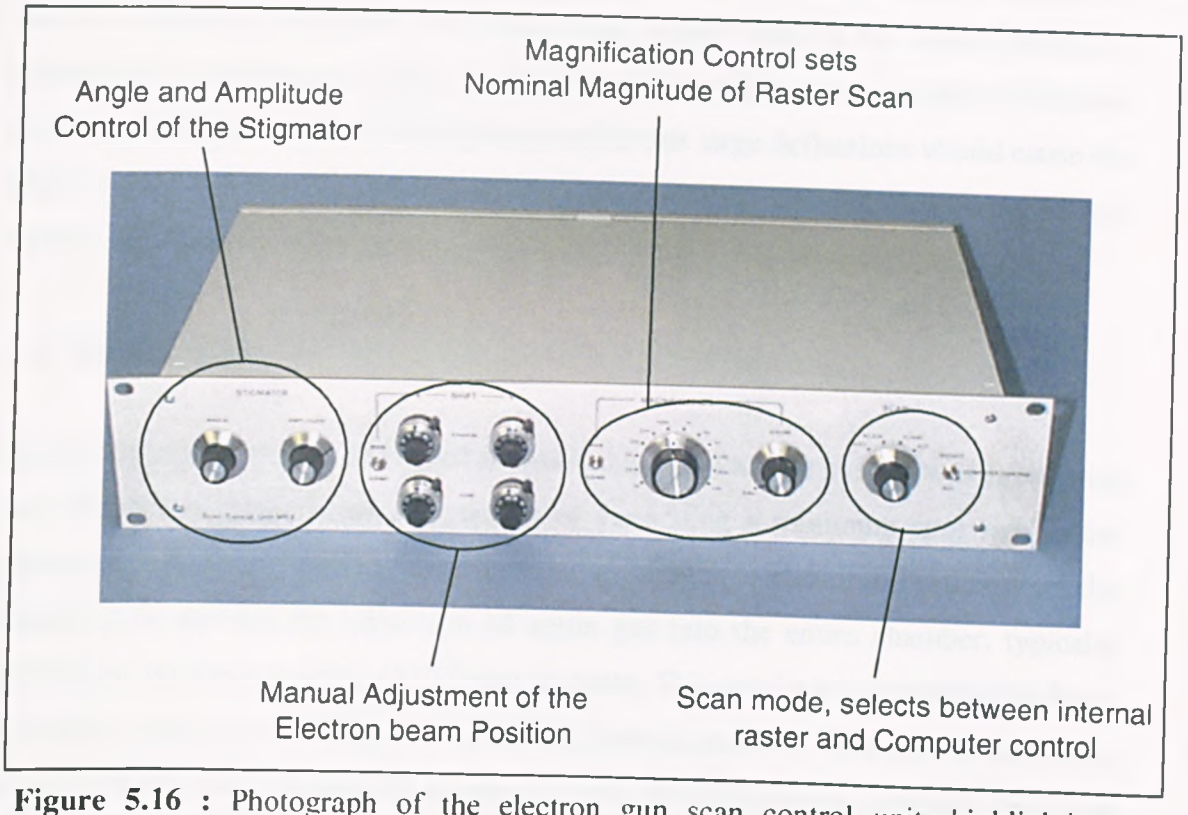

Figure 5.16 : Photograph of the electron gun scan control unit, highlighting principal features

The scanning unit can be made to produce a fixed offset from the normal beam position in any given direction. The offset can be controlled either by front panel potentiometers or by externally applied voltages. In addition to this, the scan unit has in-built waveform generators which can be set to generate raster scans, at two fixed rates selectable by the analyst (at approximately 4Hz and 50Hz respectively).

The scan unit also has a stigmator control, which can be used to correct for the effect of distortion in the shape of the beam by applying a compensating bias of adjustable amplitude in a single direction (El-Gomati 1983). The octopole arrangement is biased in such a way as to produce two orthogonal astigmatic components, in a plane normal to the electron path. These two components act to cancel lens astigmatism by elongating the electron path in one direction and compressing it in another.

The extent to which the electron beam moves from its natural position under the influence of a given deflecting voltage depends on the kinetic energy of the electrons. The maximum deflection for a given beam energy is therefore determined by the

voltage of power supply feeding the scan unit which is currently +/- 100V (assuming that the deflecting amplifiers can sweep their output close to the supply rails). At particularly low electron energies, the beam position will be very strongly influenced by the deflecting voltage and it may be possible that large deflections would cause the beam to strike the outer housing of the gun assembly, causing shadowing in any images obtained.

5.6 The Ion Gun

At the outset of the work described in this thesis, the vacuum system was fitted with two Varian ion guns. These are capable of generating a minimum spot size in the region of 5 to 7 mm, limiting their application to sample cleaning. Furthermore, the Varian guns demand the admission of argon gas into the entire chamber, typically raising the pressure to some $1x10^{-5}$mbar or more. This results in a considerable delay between cleaning and analysis whilst the chamber pressure is restored, especially when a field emission electron source is used, which demands pressures less than $1x10^{-9}$ mbar for reliable operation.

5.6.1 Sample Cleaning and Preparation

In order for optimal analysis of a sample to be carried out, the concentration of any contaminants present must be as low as possible and ideally below the detectable level of about 0.1% for AES. There are several ways in which this can be achieved. Typical methods for sample cleaning include rinsing with a solvent, ultrasonic cleaning and heating. In many cases, however, *in situ* cleaning is highly desirable due to the reduced risk of contamination between cleaning and analysis (particularly from oxide layers which may form very quickly under atmospheric exposure).

Ion bombardment is one of the most commonly used methods of *in situ* sample preparation, where the sample is exposed to a beam of energetic ions of an inert gas such as argon. This has the effect of removing the top layer of the sample to be examined, exposing a clean surface for subsequent analysis.

5.6.2 Depth Profiling and Bevelling

The investigation of chemical composition in a direction perpendicular to the surface of a sample is a natural extension of Auger analysis. This is carried out by systematic analysis of thin sections defined on a depth scale (Hofmann, 1990) and can be achieved either by non-destructive, or more commonly destructive techniques.

One of the most widely used methods of depth profiling is the iterative process of Auger analysis followed by removal of a thin surface layer by ion beam bombardment. This gradually builds up a depth profile of the sample under examination with the information of interest being represented as a series of spectra corresponding to each point on the depth scale. This technique can be readily applied when the Auger analysis is carried out with a thermionic electron probe, since the same chamber pressure can typically be used for operation of both the ion source and the electron gun. In the case of a field emission electron gun, however, much lower chamber pressures are required for Auger analysis and this can make the process of depth profiling extremely slow.

Bevelling techniques can also be used for determining depth information and are often more suitable for field emission based analysers. For this technique, a bevel is prepared either chemically, mechanically or by ion bombardment and is then subsequently examined by laterally resolved analysis to produce a depth profile of the sample. Ion beam bevelling techniques are limited in their accuracy by the spot size of the ion source (often more than 1mm). With the intention of allowing more accurate bevelling to be carried out, a high performance ion gun has been installed in the spectrometer system.

5.6.3 The EX05S Ion Gun

The EX05S is a differentially pumped argon ion gun manufactured by VG and supplied by Fisons Instruments. It has a number of features which make it superior to the previously used Varian ion guns. Figure 5.17 shows a schematic diagram of the ion gun.

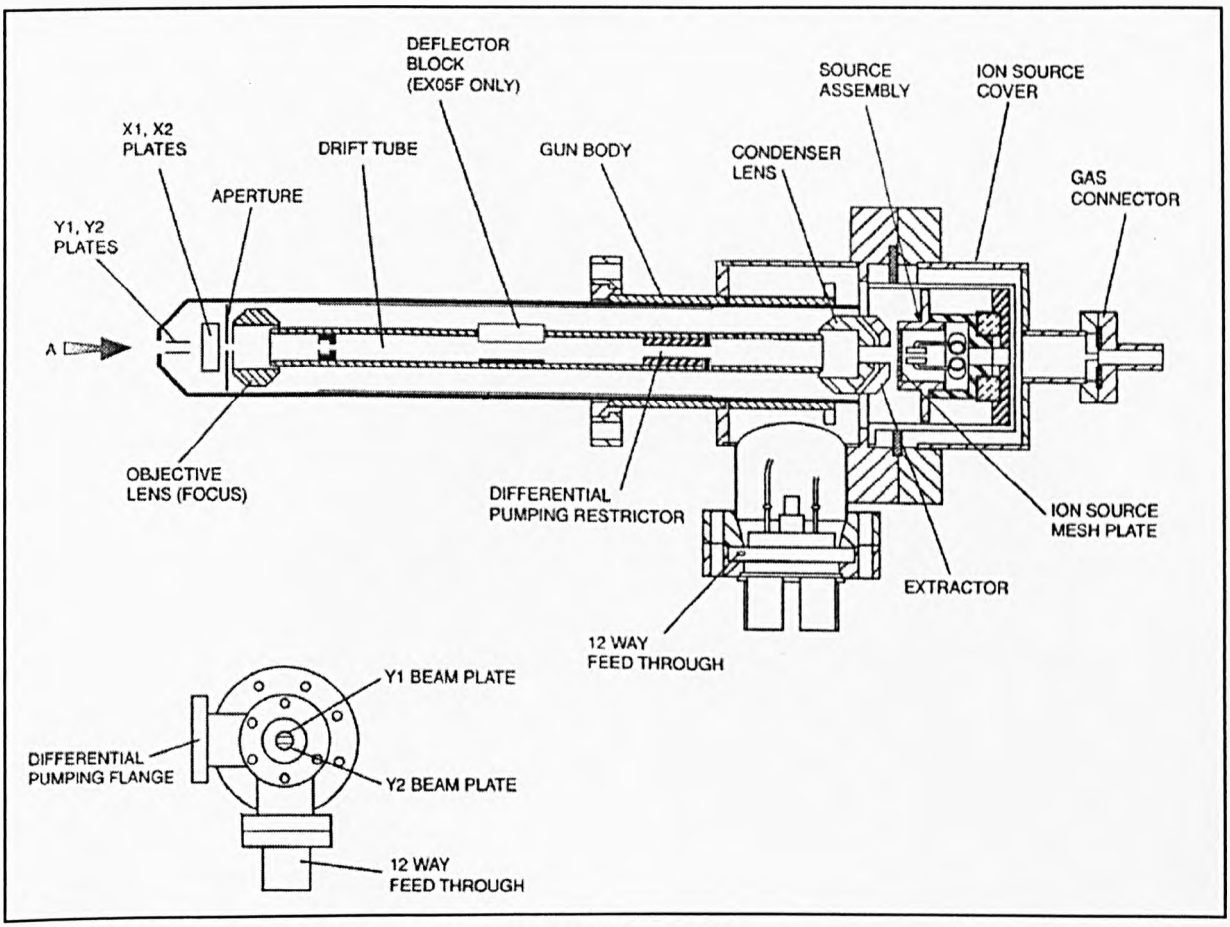

Figure 5.17 : Schematic diagram of the EXO5S ion gun

The argon gas (of at least 99.999% purity) is introduced to the source region where it is ionised by electrons emitted from the heated filament as they are accelerated towards the source mesh plate by a positive potential. The output of the ion source section is a mixture of positive ions and gas is ejected into the optical column. Excess gas is differentially pumped away, allowing a much lower chamber pressure during operation compared to the conventional Varian guns (typically less than $1x10^{-7}$ mbar). The extractor voltage accelerates the electrons into the drift tube where they are focused by the condenser lens, then subsequently brought to a focus on the sample by the objective lens. The EXO5S is capable of producing a minimum spot size of nominally 20µm. A quadrupole deflector arrangement allows electronic control of the ion beam position on the sample.

5.6.3.1 Ion Beam Control and Bevelling

The deflector plates of the EXO5S are controlled by the 346SAX physical imaging unit which is designed to sweep the ion beam in a raster scan. In this mode, using a sample current amplifier and a CRT display it is possible to successfully image with the ion gun to allow for accurate alignment with a sample.

Although the 346SAX imaging unit is a somewhat dated design with an output stage based on thermionic valves, the author was able to modify the controller with the addition of low voltage control inputs suitable for the output of a digital-to-analogue converter. This allows the computer control of the ion beam position, and the hardware interfacing required to achieve this will be covered in more detail in Chapter seven. Figure 5.18 shows the user interface of software developed to allow the analyst to design and cut bevels under computer control.

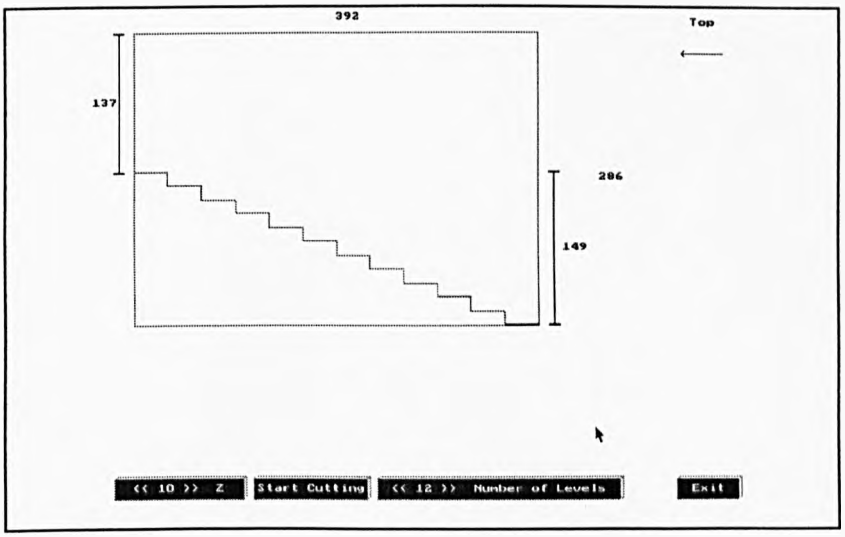

Figure 5.18 : Prototype software for ion beam bevelling under computer control

The user interface depicted in Figure 5.18 is entirely mouse driven, the geometry of the bevel is set by 'dragging' the sides of the picture with the mouse pointer. Selecting the 'start cutting' option begins a digitally controlled raster scan of the sample. Etch rates are determined by the specific characteristics of the ion beam and extensive calibration will be required for reliable quantitative analysis to be carried out.

5.6.3.2 Preliminary Testing of the EXO5S Ion Gun

Before the ion gun can be used for any from of depth profiling or bevelling, characterisation of its properties is essential. Although time constraints prevented a full systematic study of etch rates and beam diameters with the new gun, Figure 5.19 shows the preliminary results of a crater etched with the EXO5S.

Figure 5.19 was collected with the 'Alpha step' instrument, which passes a moving stylus over the sample, plotting the resulting contour. The full scale of the graph is 25kÅ. The crater was etched onto a silicon sample, using a 3kV ion beam at 3μA sample current for 30mins at x25 magnification, followed by 30mins at x50 magnification (forming a two-tier crater).

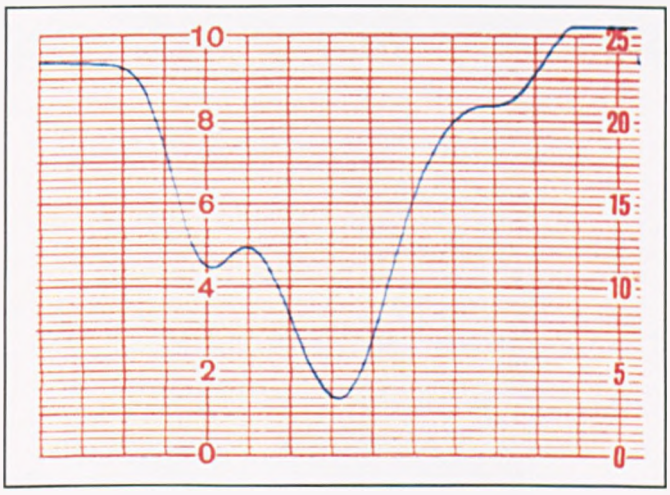

Figure 5.19 : Preliminary test result from the EXO5S ion gun

The computer controlled bevelling hardware and software has been subsequently tested and found to give reasonable results in preliminary experiments (Gelsthorpe, 1997).

5.7 Conclusions

In this chapter, the experimental apparatus forming the basis of the Auger spectrometer has been described. This included a description of the vacuum system together with an explanation of the need for such an environment and common procedures for sample insertion and positioning. The electron energy analyser at the heart of the spectrometer was then described, including electrostatic simulations to illustrate some features of the specific analyser used here.

The design of a compact secondary electron detector head was then discussed, together with the head amplifier and power supply to support such an instrument. A novel compact field emission electron gun was described, which allows much higher spatial resolution to be achieved than with conventional thermionic sources. Finally, we describe the installation and initial testing of a commercially available high performance ion gun with additional modifications to allow computer control of beam position.

CHAPTER SIX

THE ANGLE-RESOLVED DETECTOR HEAD

6.0 Introduction

The design, implementation and testing of the major modifications to the CMA detector head are covered in this chapter. These include the addition of a segmented channel plate assembly in place of the existing single channel analyser, together with the analogue and digital electronics required to support this arrangement.

6.1 The Multi-Channel Detector (MCD)

The aim of the developments described in this section is to design and implement a Multi-Channel electron Detector (MCD) to replace the channeltron in the CMA, allowing information to be gathered not only about the number of electrons passing through the analyser, but also about their angular distribution. It is hoped that this development will allow correction of topographical artefacts in high resolution Auger electron spectroscopy and microscopy.

6.1.1 Microchannel Plates

Microchannel plates (MCPs) provide an alternative method of secondary electron multiplication to channeltron based detectors. Their use is adopted in the design of this detector in order to provide a means of recovering spatial information from electron energy spectra. The use of an MCP detection system is chosen in preference to an array of channeltron detectors to allow for future developments to take place with a minimum of modification to the existing hardware.

The basic principle of operation of the channel plate is illustrated in Figure 6.1.

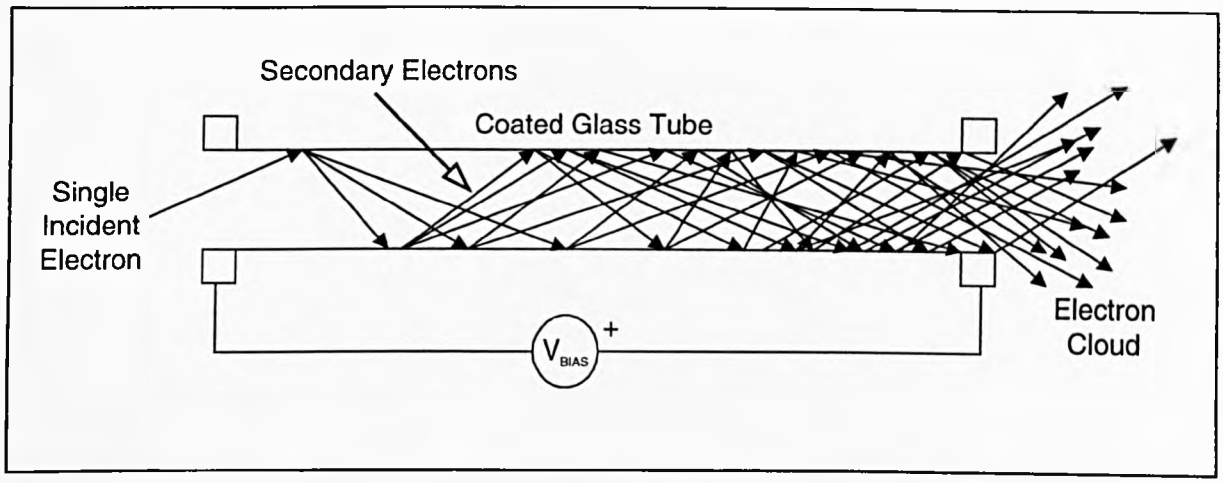

Figure 6.1: Electron multiplication in a single channel of the channel plate

The channel plate itself is made up from a plate of lead glass through which a large number of channels pass (like the one shown in Figure 6.1) each with an internal diameter between about 10 and 20μm. The walls of the channels are coated with a semiconductor material, with a coefficient of secondary emission much greater than one (Galileo, 1996).

When an incident electron strikes the wall of a channel, several secondary electrons are produced and go on to strike the opposite wall. The resulting avalanche produces a large burst of electrons at the exit of the channel, corresponding to each incident electron. The channels are set at an off normal angle with respect to the face of the plate, in order to ensure that any electrons arriving at normal incidence to the plate are sure to strike the wall of a channel (Alrad Instruments Limited).

The gain, g, of the MCP is represented as

$$g = \exp(G.L/d) \qquad (6.1)$$

Where L/d is the length-to-diameter ratio of the channel, and G is a gain factor, dependant on the electric field strength (defined as voltage drop per unit length) and the secondary electron emission characteristics of the semiconductor material lining the channel wall.

Consideration of equation 6.1 leads to certain criteria which must to be taken into account when designing a detector using channel electron multiplier plates.

The electron gain of the plate is very high (typically about 10^3) and is a strong function of bias voltage across the plate (see eq. 6.1), so a very stable supply must be used in order to minimise noise. Also due to the complex relationship between fabrication tolerances and gain of the individual channels, the gain of a channel plate may vary over its area. This is an important consideration, as we will see, since in the CMA analyser the detected electrons strike a different part of the plates according to both their angle of emission and energy.

Saturation of the plates occurs at a gain of about 10^4 electrons, and after this point the gain cannot increase further. This is caused by the space charge at the output end of the channel repelling secondary electrons, causing them to return to the wall without generating further electrons. The gain limitation imposed by this ion feedback can be overcome by cascading channel plates to produce a two-stage system. This technique is shown schematically in Figure 6.2.

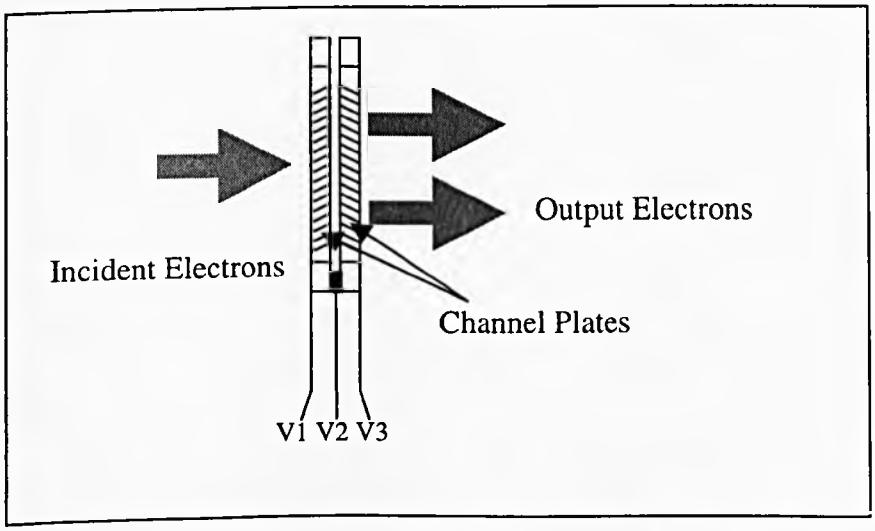

Figure 6.2 : Cascaded channel plate arrangement; V3 > V2 > V1

This arrangement will be adopted in the present detector since the nominal threshold of the charge sensitive amplifier currently proposed for use in the analyser is at best

$5x10^4$ electrons. To ensure reliable counting of electron events, therefore, a high channel plate gain is desirable.

In the two stage arrangement, incrementally increasing voltages are applied to the plates to provide an accelerating potential across the plates. These will be provided by a stabilised potential divider network.

6.1.2 MCD Detector Head Design

It is proposed that the acquisition of angular information is achieved by replacing the existing channeltron assembly, with the series MCPs depicted in Figure 6.2, followed by a sectioned gold anode for collection of the resulting electrons. These will form the multi-channel electron detector (MCD).

By dividing the target into geometric regions we can extract information about the origin of the electrons. This is possible because the azimuthal component of an electron's trajectory is preserved as it passes through the CMA. Furthermore, it would also be possible to improve the energy resolution of the analyser by splitting the detector into a set of concentric rings, since lower energy electrons would fall closer to the centre of the target (for a given polar emission angle). Theoretically, the resolution of both these methods is limited only by the resolution of the technique used to fabricate the anode collector, and the accuracy with which the detector is aligned with the CMA optical axis.

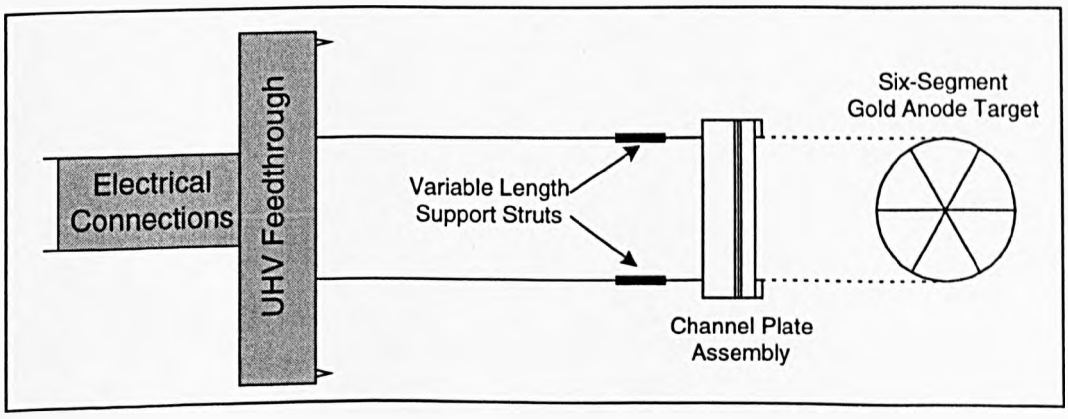

Figure 6.3 : Schematic diagram of the channel plate detector

The MCD assembly provides a means by which to position the two series MCPs at the focal point of the CMA, with a suitable segmented target behind them to collect the resulting electrons. The supports are variable in length, in order to allow the optimum position of the detector plates with respect to the focal point to be attained. If the detector were positioned precisely at the focal point of the analyser, little or no angular information would be obtained, since all electrons would fall in the same region at the centre of the plates. Conversely, if the detector was placed too far off the focal point, electrons would fall outside the region of collection provided by the plates. For this reason, detailed simulations of the analyser were carried out using the SIMION electrostatic lens simulation package; these will be described subsequently.

The target is split into six segments in the current MCD design, in order to coincide with areas of shade arising from features of the internal geometry of the CMA. Gold anodes were fabricated by vacuum evaporation onto a PEEK (ICI Advanced Materials) cylinder forming the basis of the detector head. The evaporation process is straightforward; the apparatus used in this process is shown in Figure 6.4.

Firstly strips of insulation tape were used to mask the three lines of symmetry on the PEEK assembly, which is then suspended over a titanium carrier containing a small amount of gold. Passing a high current through the titanium carrier produces sufficient heat to evaporate the gold. Initially, a screen prevents the evaporated material from reaching the target, but this is moved away to allow gold to condense on the unmasked area on the surface of the PEEK. This screening process allows the thickness of the deposited layer to be accurately controlled independently of the time taken for the gold to reach its melting point, by varying the time of exposure of the sample.

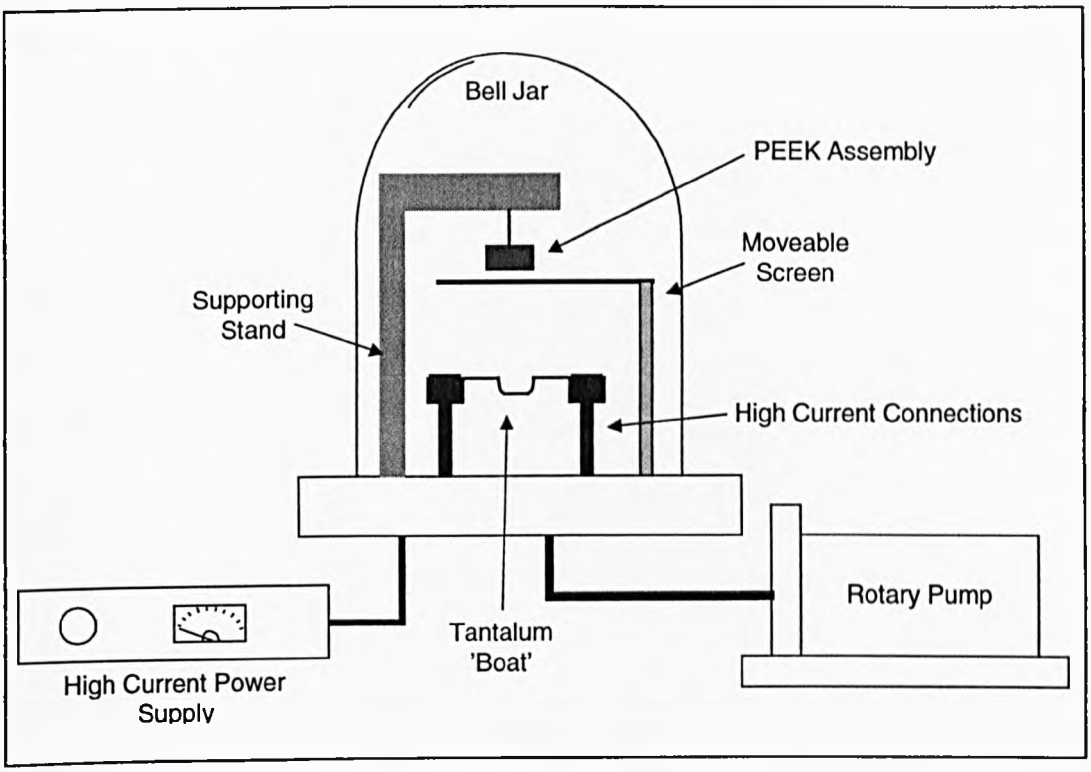

Figure 6.4 : The process of vacuum evaporation

6.1.3 Electrical Connections

The assembly providing the bias voltages to the plates is shown schematically in Figure 6.5.

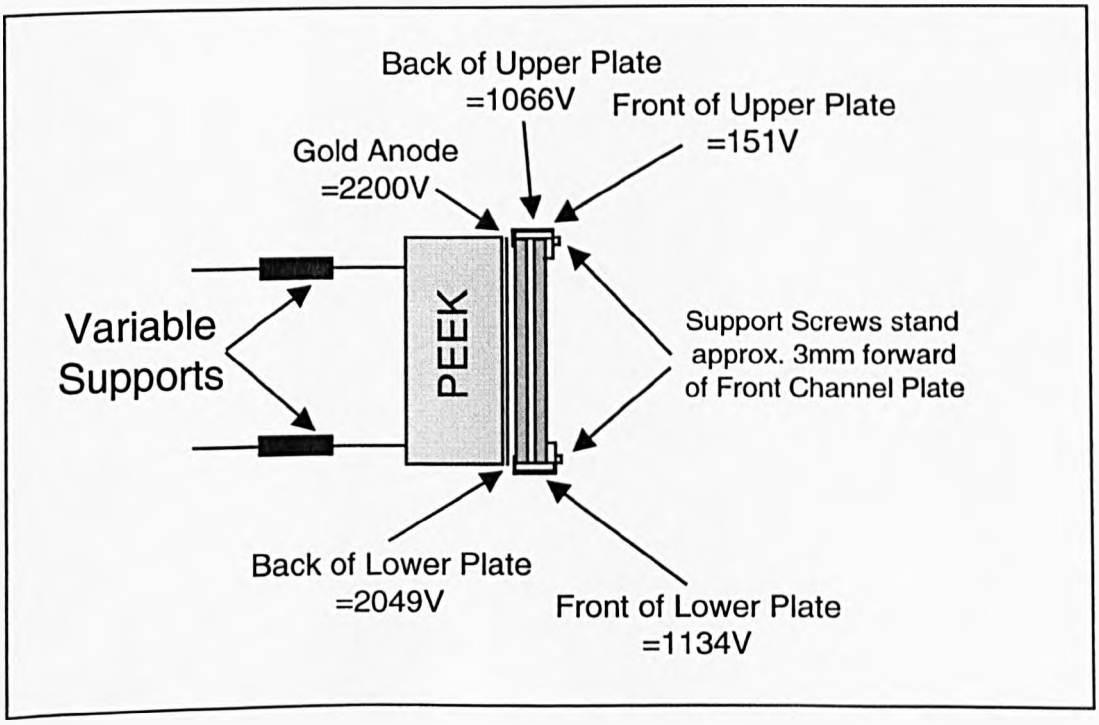

Figure 6.5 : Schematic of channel plate assembly, showing applied voltages

The MCD assembly requires a total of 10 electrical connections, four of these are DC biasing voltages to supply the channel plates, and the other six are the signal connections to the segments of the gold detector anode (also DC biased).

In the present design, the detector is mounted onto a 11pin UHV feedthrough on a 2¾" flange. Internal electrical connection can be made to seven of the available pins via UHV compatible kapton wire with push fitting, crimp connectors. However, four of the electrical feedthroughs are used to form the support struts shown in Figures 6.3 and 6.5, and do not have free ends to mate with the push fit connectors. Electrical connection is made to these struts by spot welding of short lengths of kapton wire. Because high voltages are present in the detector head assembly, electrical isolation of the support struts is essential. This is provided by the PEEK assembly on which the channel plates are based.

6.1.4 The Complete Detector Head

Figure 6.6 shows the finished channel plate detector head, supported on a UHV flange. The length of the support struts shown in the diagram is 122mm.

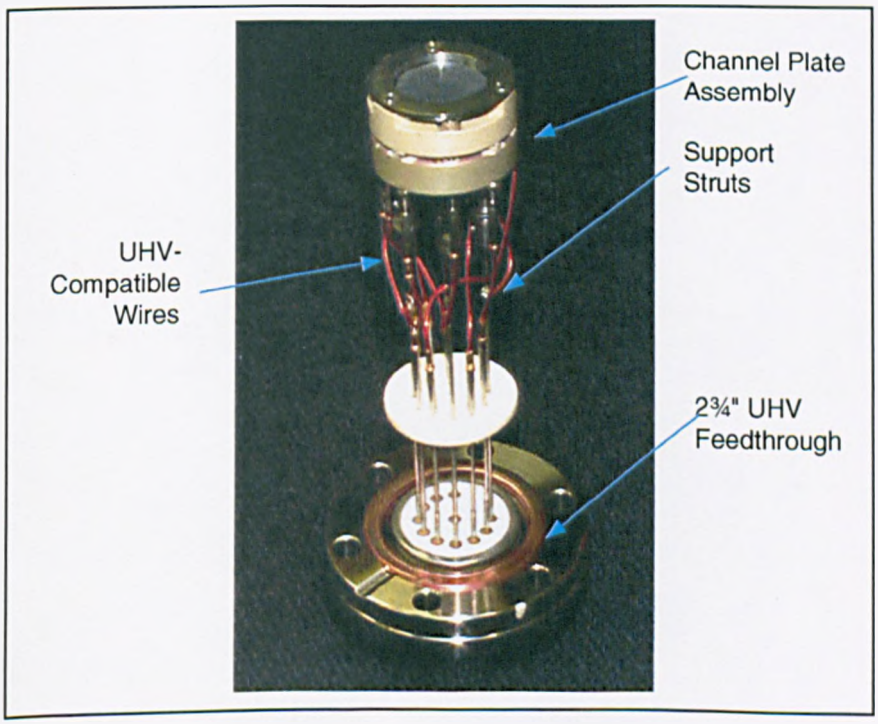

Figure 6.6 : The completed detector head assembly

6.2 SIMION Simulation

SIMION is an electrostatic lens analysis and design program which computes the trajectory of ions (electrons in this case) through an arbitrary lens arrangement defined by the analyst (Dahl and Delmore, 1988). The lens (in this case the CMA) is defined as a 2 dimensional system, but cylindrical symmetry assumptions allow 3 dimensional ion trajectory calculations.

Because of the focussing properties of the CMA, the position of the multichannel detector head in relation to the focal point of the instrument is critical. The next step, therefore, is to simulate the trajectory of electrons as they pass through the analyser, to gain an understanding of the effect of detector position on the expected signal.

SIMION version 4.0, as used in this work, bases its simulations on a two dimensional potential array of up to 16000 points in total, each pixel in the array relating to the voltage at that point in space. Because of the dimensions of the CMA, this allows for a maximum resolution of just over 1mm, and for simplicity a resolution of 1mm per pixel was chosen. Although this represents a rather coarse scale in terms of the CMA features (the annular exit slit is less than 1mm in width), it was considered to offer sufficient accuracy for the present work.

6.2.1 Ideal Detector Position

Figure 6.7 shows a schematic diagram of the sectioned gold anode which forms the basis of the angle resolved MCD head. For optimum angular resolution to be achieved, it is important to consider the following points

- The detector head must be aligned such that the etched lines between the gold anode target segments correspond to the shadowed areas of the CMA.
- The distance between the detector and the focal point of the CMA must be selected to optimise illumination of the plates over the electron energy range commonly examined.

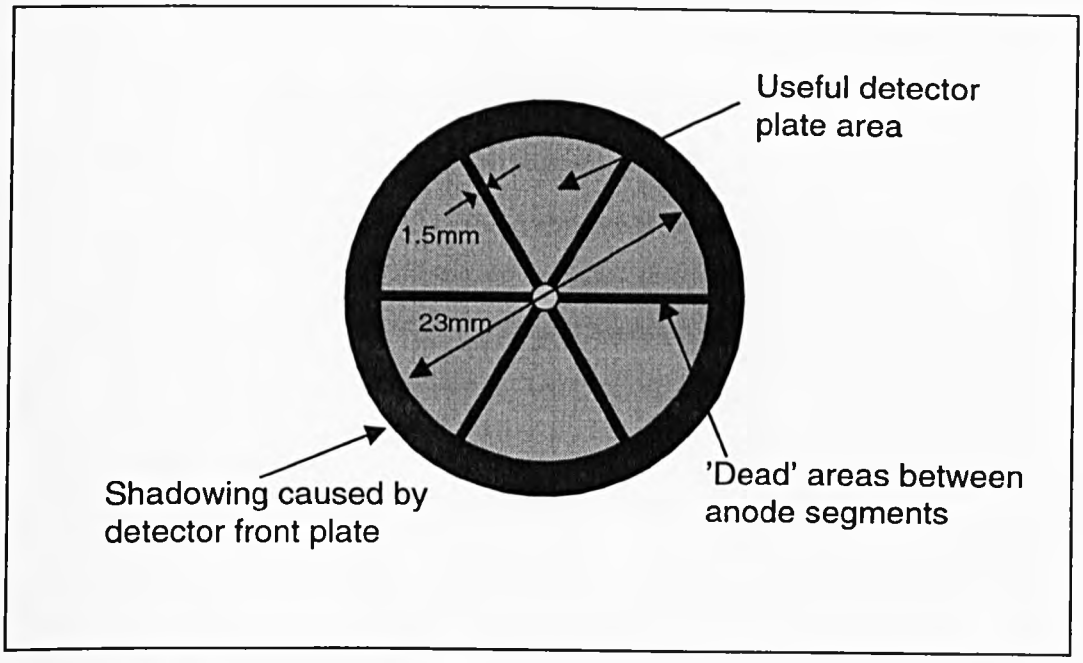

Figure 6.7 : Schematic diagram of the 6-segment detector plate arrangement

It can be seen from Figure 6.7 that as well as the 'dead' areas between the detector segments where the gold has been etched away, there is an area in the centre of the detector where the insulating PEEK base is exposed. This means that in order to be detected, electrons must fall within an annulus between this centre 'dead' area and the metallic detector front plate which clamps the channel plates in place.

With a view to determining this ideal detector position, we simulate electrons at two representative energies entering the analyser over an angular range of 36.3^{o} to 48.3^{o}, i.e. representing the complete range of accepted angles. An attempt has been made to place the detector in what may be considered to be an ideal position for a pass energy range up to 3000eV. Figure 6.8 shows the electron trajectories for an electron energy of 1keV and Figure 6.9 illustrates the same simulation but with a pass energy of 100eV.

The simulations are based on the dimensions of the Varian CMA, with inner and outer cylinder radii of 27mm and 61mm respectively.

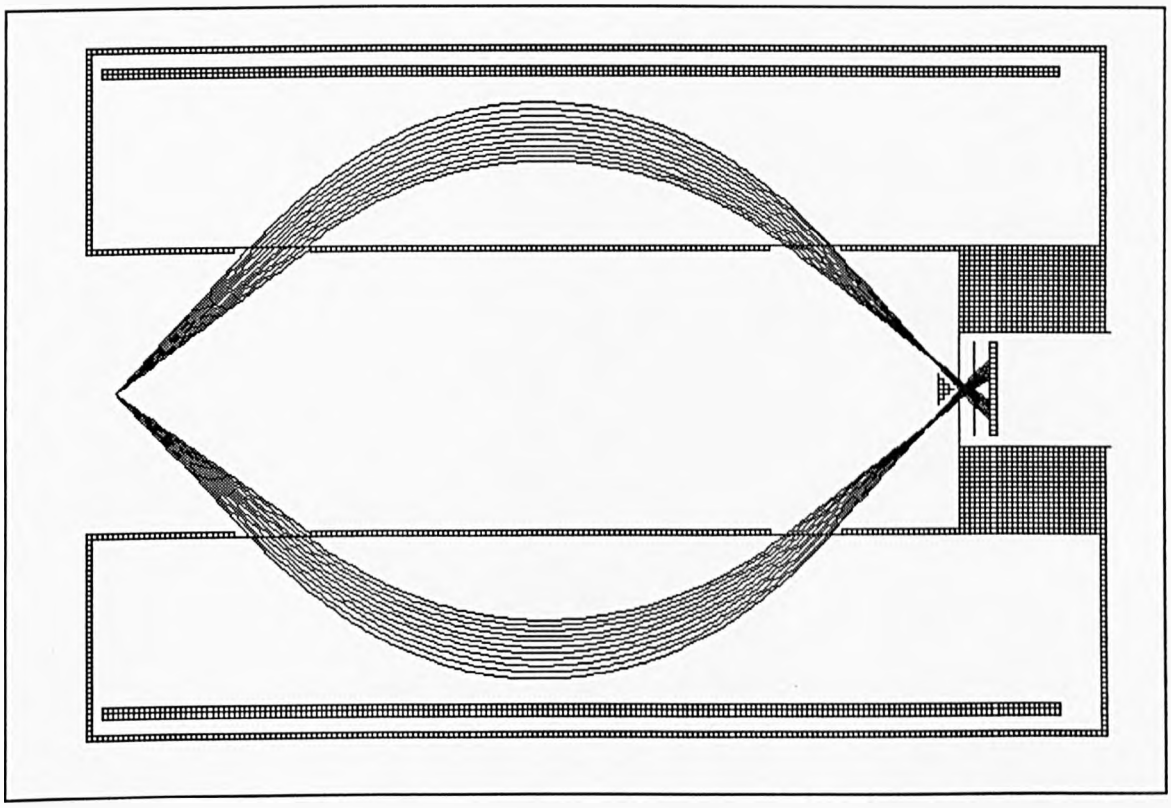

Figure 6.8: Simulation of the CMA with the channel plate detector in place in the ideal position for high angular resolution. The electron energy is 3000eV

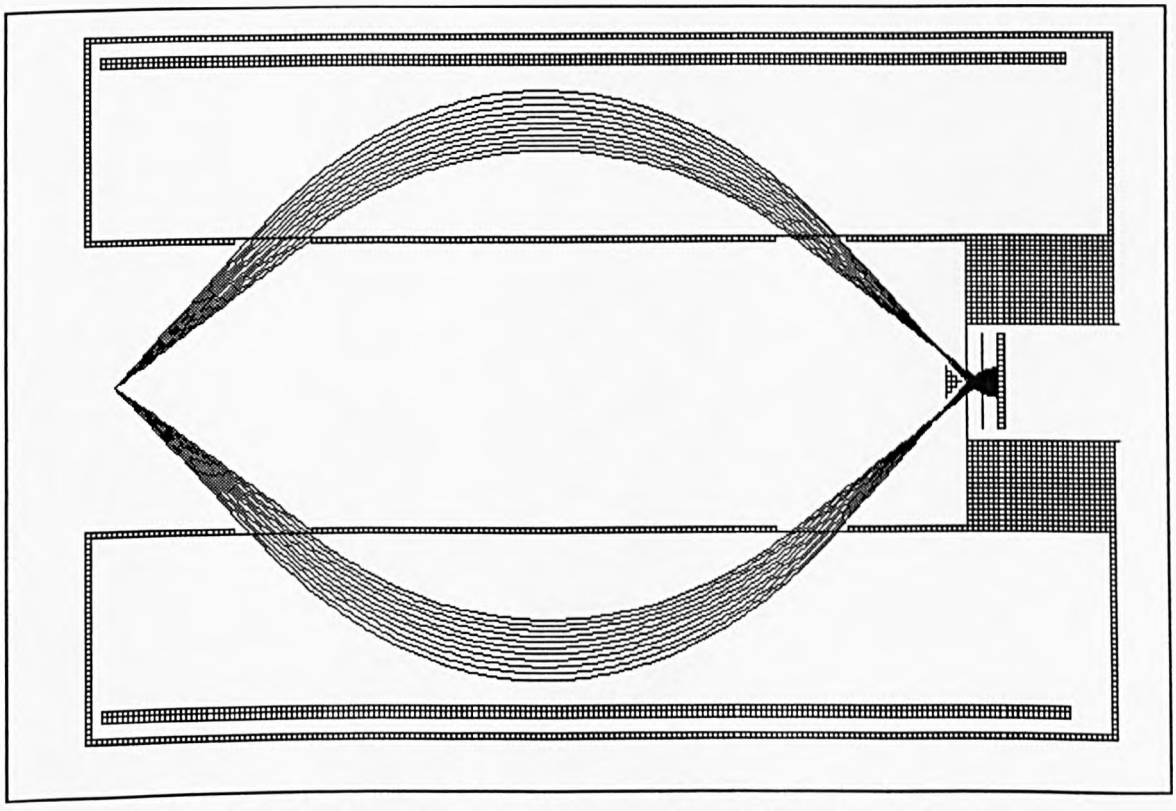

Figure 6.9: Simulation of the CMA with the channel plate detector in place in the ideal position for high angular resolution. The electron energy is 100eV

As Figure 6.8 reveals, the spread of electrons over the detector is such that no electrons fall outside the collected range (even at high energy), but good angular resolution should be possible at low energy, since a large proportion of the plates are covered. The position of the detector in this case is approximately 3mm from the exit slit, which may be difficult to engineer due to the protrusion of the fixing screws on the detector head. As one would expect, at lower electron energies the voltage on the channel plate detector has a more significant effect on electron trajectories, causing them to illuminate a smaller area of the plates. This effect may result in an energy dependant shift in the gain of the analyser if the channel plates do not have uniform gain over their area.

6.2.2 Actual Detector Position

Unfortunately, physical limitations of the current detector head design do not allow the detector to be placed at the position thought to be ideal as calculated by the simulations (the PEEK assembly and mounting screws protrude more than 3mm in front of the top channel plate). Due to the high voltage present on the channel plate assembly and the limited accuracy with which the detector can be positioned at this stage (allowing for compression of copper gasket etc.), it is considered desirable for the detector to be positioned with the top channel plate 6mm from the exit aperture. To investigate the implications of this compromise, two further simulations have been carried out. These are shown in Figures 6.10 and 6.11, where the detector is positioned 6mm from the exit slit. Figure 6.10 is for high energy (3keV) electrons, and the simulation shown in Figure 6.11 is based on low energy (100 eV) electrons.

Examination of Figure 6.10 shows that with the 3keV electron beam, the electrons which are incident on the analyser at the upper limit of its acceptance angle, strike the rim of the target and may be lost (into the channel plate assembly). The lower energy electrons, however, do not exhibit such behaviour, since as mentioned previously their trajectories are more easily changed by the electrostatic attraction of the detector plates and are 'towed in' towards the axis of the CMA, allowing them to strike the channel plate. Because low energy electrons illuminate a larger plate area than those in Figure 6.9, this detector position may be advantageous.

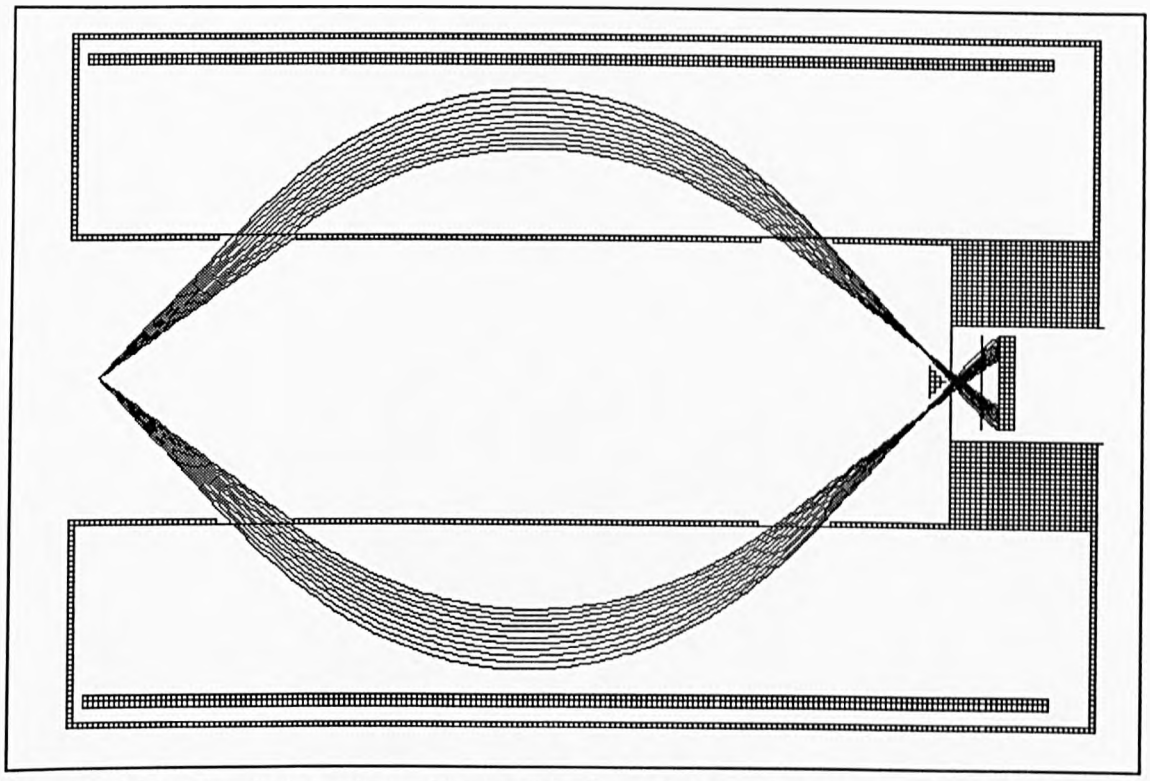

Figure 6.10: Simulation of the CMA with channel plate detectors, with the plates in the actual position used in the present experimental set-up (6mm from the exit slit). 3keV electrons are used in this simulation

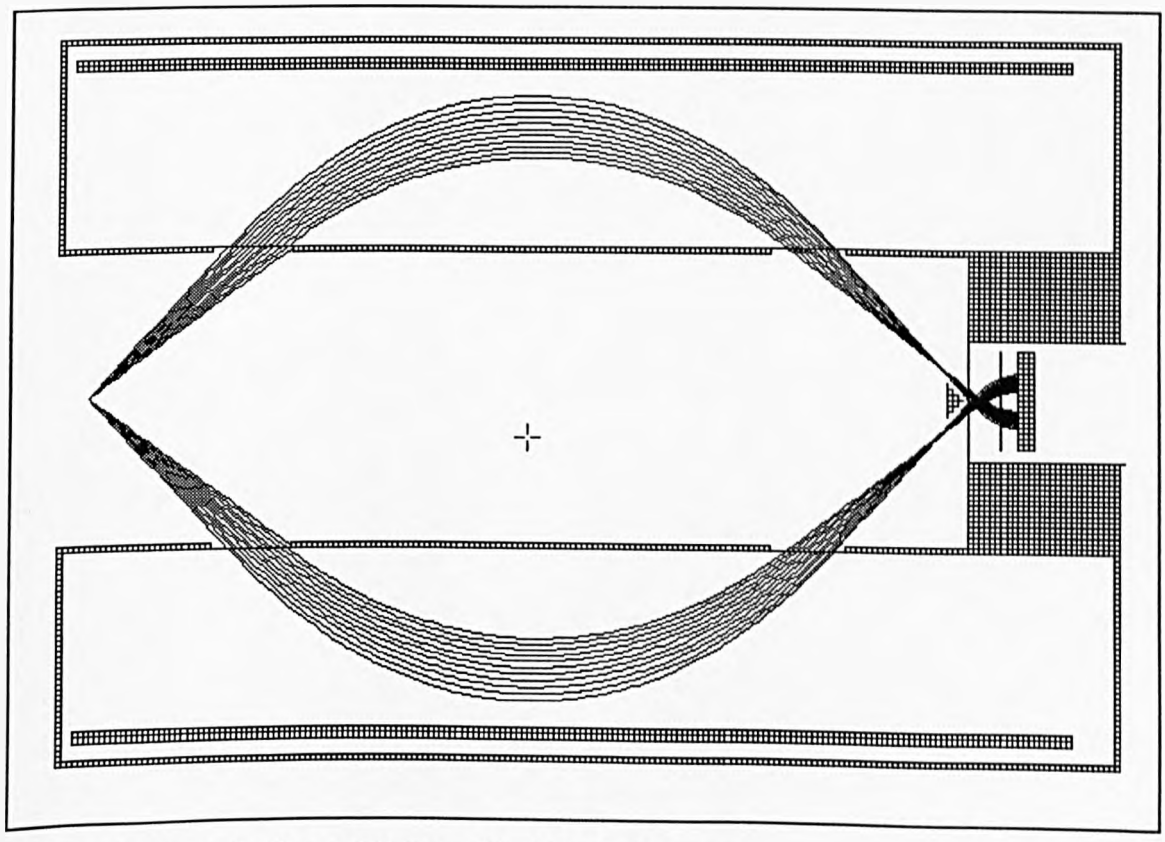

Figure 6.11: As in figure 6.8, but with 100eV electrons

6.2.3 Channel Plate Gain Variation

The gain of a channel plate can vary over its area, and according to arrival angle of electrons. The extent of this gain variation is determined by a number of factors including fabrication tolerances and the condition of the plate.

Electrons arriving at different angles and energies strike different areas of the plates, and are therefore multiplied by different gains. This in turn may lead to inaccuracies in the spectra acquired if the gain variation spreads the electron energy distribution either side of the detector threshold. The possible effects of channel plate gain variations will be discussed further in Chapter eight.

6.3 Detector Head Electronics

The electronics required to bias the channel plates, and process the resulting signals from the detector head will be housed in a die-cast box which directly mates with the UHV feedthrough. This arrangement ensures minimum electrical path lengths and therefore minimum signal-to-noise ratio.

6.3.1 Biasing the Channel Plates

The voltages used in this initial study were chosen to give a gain of approximately 2×10^3 electrons from each plate (i.e. total gain of 4×10^6). The biasing circuit shown in Figure 6.12 was designed to produce these voltages. This arrangement gives 915V across each plate, achieving the required gain specification. For protection purposes, manufacturers recommend that power supplies to the channel plates must be only capable of delivering a maximum current of 1mA. This is achieved by the use of series resistance, R where the minimum value of R is given by operating voltage $\times 10^3$.

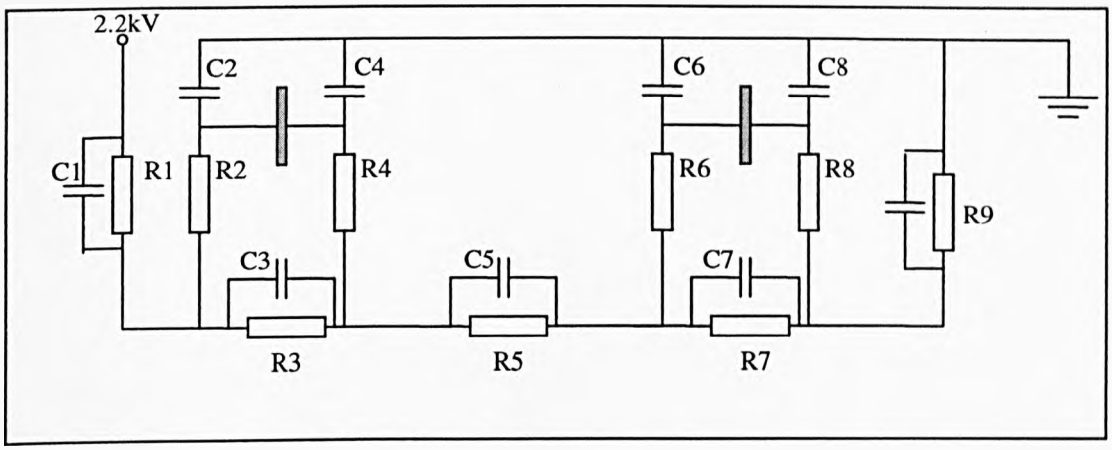

Figure 6.12: The channel plate biasing circuit used in the present detector

Component values for the circuit in Figure 6.12 are as follows : R1 8M3; R2, R4, R6, R8 1M; R3, R7 20M; R5 1M5; R9 3M3. C1, C3, C5, C7, C9 0.01 μF; C2, C4, C6, C8 470pF.

6.3.2 The Head Amplifier

The output from a channel plate detector is in the form of very small charge pulses. To convert these into a signal that can be read by a computer, they must first be amplified, then a discriminator stage is used to produce digital pulses suitable for driving a computer interface.

AMPTEK Inc. (see refs.) is an American company, that manufactures single package charge sensitive preamplifier-discriminator devices. These are ideal for use in this type of analyser. One such device (the A111F) is designed especially for instrumentation employing microchannel plates and other low capacitance charge producing detectors. The unit was originally developed for NASA's deep space probes, and is therefore designed to an extremely high specification. The unit is very compact, runs from a single rail power supply, and is compatible with CMOS and TTL logic. The circuit used for one of the four channels of detection is shown in Figure 6.13

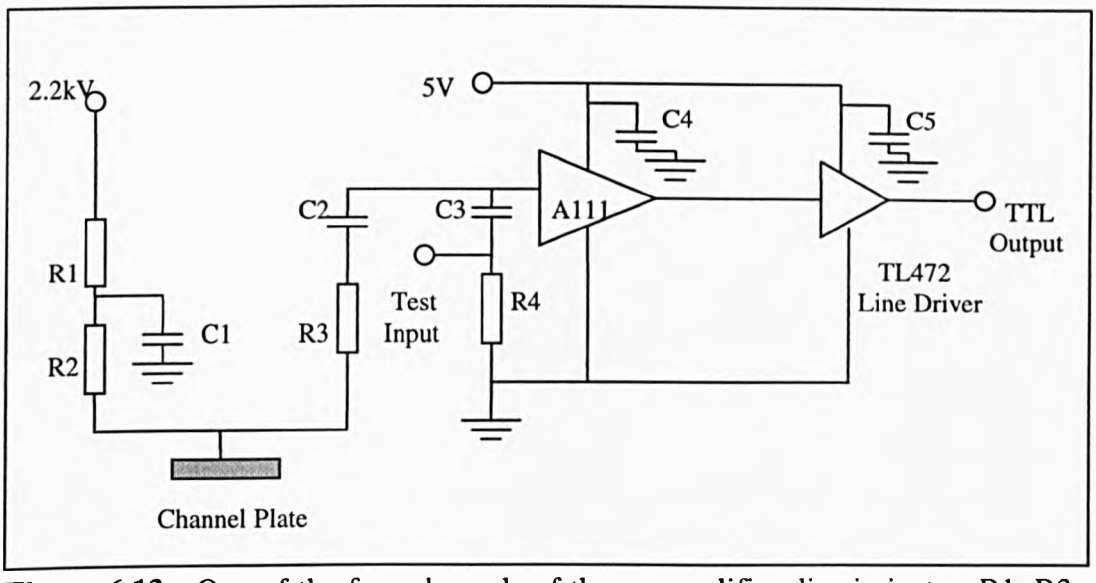

Figure 6.13 : One of the four channels of the preamplifier-discriminator. R1, R2, R3 1M; R4 50Ω; C1, C2 0.01μF; C3 2.2pF; C4, C5 0.1μF

The network produced by R1 and C1 provides decoupling of the high voltage supply. R2, R3 and C2 are used to bias the detector plate, decouple the high voltage from the amplifier, and also protect the circuit should a short circuit to ground occur in the detector plate (in which case 1.1mA will flow, which is sufficiently low to pose no risk of damage to the circuit).

The threshold of the A111 device can be set by the addition of a simple feedback network. Nominally this is set to $5x10^4$ electrons, but in this design will be set to $5x10^5$ (since the channel plate gain is approximately $1x10^6$); this is achieved by shorting pins 7 and 8 of the device. Provision will be made on the PCB for extra components to be added so that the threshold can be changed if the need becomes apparent in future.

An input point is provided to test that the circuit is operating correctly. When a pulsed voltage is fed into the test input, the 2.2pF capacitor will produce a charge at the input to the amplifier, simulating the arrival of a signal from the detector plate.

Although the output of the A111 is at TTL level, it is fed into a TL427 line driver to give slightly more current drive and to give some buffering to reduce the risk of

damage in the event of static discharge. This is desirable since the A111 is such an expensive device compared to the line driver.

Figure 6.14 shows a photograph of the completed head amplifier, including the high voltage bias network and the charge sensitive amplifiers. The external connections are made via a standard 9-Way 'D' connector, carrying both the low voltage supply to the amplifiers and line drivers, and the data output to the computer interface. The voltage required for biasing the detector head is supplied using RG58 co-axial cable to a high voltage BNC connector (rated at 7kV).

The PCB layout was designed in such a way as to provide a ground plane over the low voltage (signal) area, but bare circuit board in the high voltage area. The completed board was sprayed with silicon compound to provide extra protection from high voltage breakdown. All internal cables carrying high voltages are double insulated and rated at 10kV.

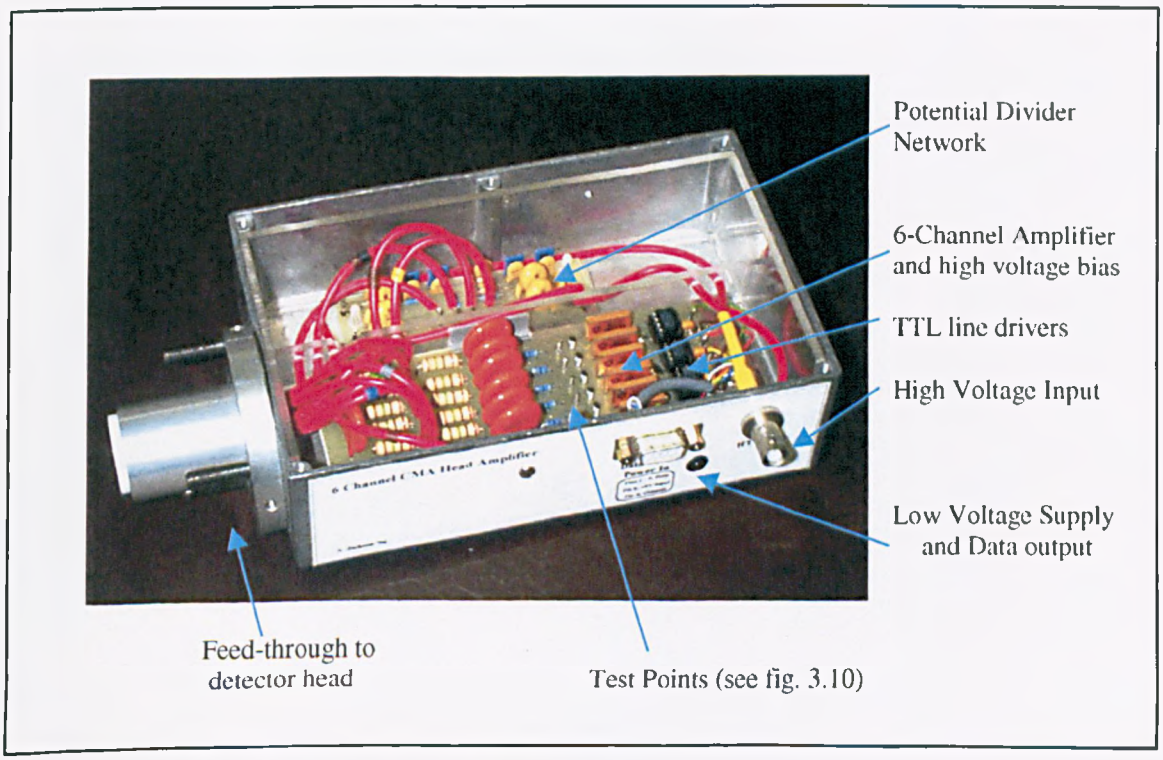

Figure 6.14: The six channel CMA head amplifier

6.4 Initial Testing of the Detector Head

6.4.1 Channel Plate Testing

The channel plates must be treated with great caution when being powered up for the first time, since any small design error in the detector assembly, power supply, channel plates, or head amplifier may cause costly damage, particularly if discharge occurs in the area of the plates. Before any voltages were applied to the plates, the biasing circuit was tested. The measured voltages are shown below in table 6.1, together with the ideal voltages at the various points.

	Ideal Voltage	Actual Voltage
Lower side of Bottom plate	2049	2033
Upper side of Bottom plate	1134	1118
Lower side of Top plate	1066	1050
Upper side of Top plate	151	153

Table 6.1 : Comparison of ideal and real voltages from the channel plate biasing circuit

The above voltages result in a potential of 897V across the top plate and 915V across the lower plate. The variation from design specification is therefore less than 2%, and almost certainly due to component tolerances. This discrepancy should not have a noticeable effect on performance as it will result in only a small change in channel plate gain.

The first test on the plates themselves was to measure their resistance. According to the data supplied with the plates, under normal conditions this should be in the region of 80 - 300 MΩ on a 10V range. The resistance of the plates was tested at 150MΩ for the top plate and 160MΩ for the lower plate, which is well within the expected limits. The resistance between the upper side of the lower plate and the lower side of the top

plate was tested to ensure that there was no internal short circuit between these two and found to be > 100GΩ which is acceptable.

The final part of the initial testing of the channel plates, was to power them up gradually in the vacuum system and monitor the pressure. As the voltage across the plates was increased, there was no appreciable change in pressure, indicating that no discharge was occurring.

6.4.2 Testing the Head Amplifier

Initial testing of the preamplifier/discriminator uses the test port shown in Figure 6.13. The principle of operation is that the capacitor integrates the input voltage pulses to give a charge at the input to the amplifier-discriminator circuit. The test signal takes the form of a short pulse with a sharp rise time and slow decay (to simulate the characteristics of real signal). This input pulse is shown in Figure 6.15, along with the resulting measured output (the same for all channels).

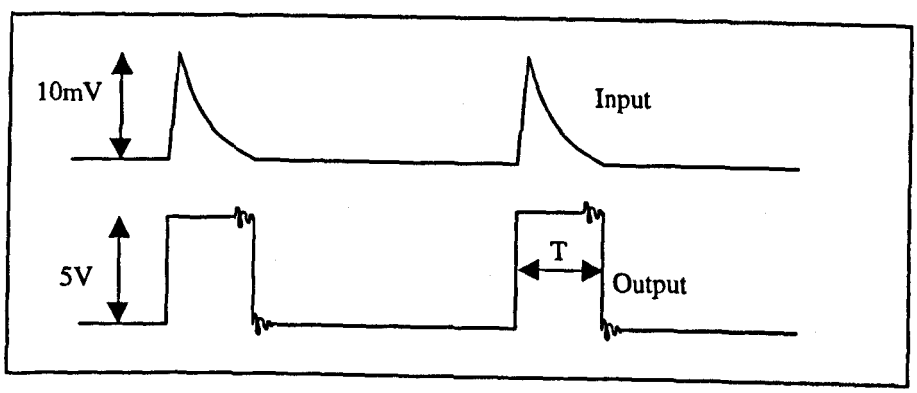

Figure 6.15 : Test input voltage, together with resulting TTL output

The duration of the output pulse, T, is measured as approximately 400ns for the test signal described above. More experimentation, with a variety of test signals, would be required to test the absolute limit of this pulse width. The current results suggest a limit to the maximum possible output frequency of 2.5MHz, since successive pulses merge into one another at higher frequencies.

Channel plates exhibit a 'dead time' after each incoming event, during which they are unable to respond to subsequent events. The duration of this 'dead time' can be

roughly equated to the resistance/capacitance (RC) time constant. This effect, in conjunction with other RC time constants (such as those inherent to the amplifier/discriminator circuit) limits the maximum count rate of the overall system.

The small amount of ringing visible on the output waveform may be due to poor impedance matching between the output of the line driver and the transmission line. This problem can occur when the small output inductance of the last stage in the line driver forms a tuned circuit, producing a resonance due to the presence of complex poles in the transfer function of the output equivalent circuit. This can be remedied by the introduction of a small series resistance with the output, effectively reducing the 'Q' of the circuit and forcing both poles of the second order system to be on the real axis of the pole zero plot. At this stage, no attempt was made to reduce this ringing, since the logic devices used in the computer interface have thresholds at 2.4V for '1' and 0.8V for '0'. The magnitude of the ringing is measured at less than 0.25V, so it is very unlikely that multiple counting of events will occur.

6.5 Conclusions

The design and implementation of an angle resolving Multi-Channel Detector head (MCD) for the CMA electron energy analyser has been described. The present MCD design is based on a six-segment gold anode detector with a dual channel plate arrangement to provide the required gain. The mechanical design of the MCD has been described, together with schematic diagrams and photographs of the current experimental detector head. Simulations of the MCD head in the CMA analyser have been carried out with the SIMION electrostatic lens modeling software. These have allowed a preliminary investigation of the effect of detector position to be performed.

Finally, the power supply and head amplifier required to support the channel plate detector have been described. The current design is based on a commercially produced charge sensitive pre-amplifier and discriminator, which has made circuit design reasonably simple. Preliminary tests of the finished head amplifier were able to confirm its correct operation.

CHAPTER SEVEN

THE CMA CONTROL SYSTEM

7.0 Introduction

Initial tests of the angle resolved detector were carried out using control software and hardware designed for the single channel CMA (El-Bakush, 1994). This made it necessary to collect spectra sequentially since the software was unable to support more than one channel of data acquisition. Any variation in electron beam conditions or sample position over the course of spectral acquisition may give rise to undesirable features, and this coupled with the time constraint of sequential operation over the six channels, highlights the importance of an appropriately designed control system to gather data in parallel from all segments of the angle resolved detector head.

The hardware and software of an integrated control system are described in this chapter, with the intention of bringing together all the major control functions of the spectrometer in a user-friendly environment. The main features of the control system are as follows:

- Spectrometer tuning system to allow positioning of the sample at the focal point of the analyser. This is a critical part of the analysis procedure since the CMA is inherently sensitive to sample position.
- Parallel acquisition of spectra from all six channels of the segmented detector.
- Full computer control of the electron beam position in two dimensions to allow scanning Auger microscopy (SAM) to be carried out either along a line or over an area.
- The implementation of an entirely computer controlled scanning electron microscope (SEM). This will allow accurate positioning of the electron beam on the sample before Auger analysis is carried out and also make it possible to accurately select areas for SAM studies before potentially time consuming experimental measurements.

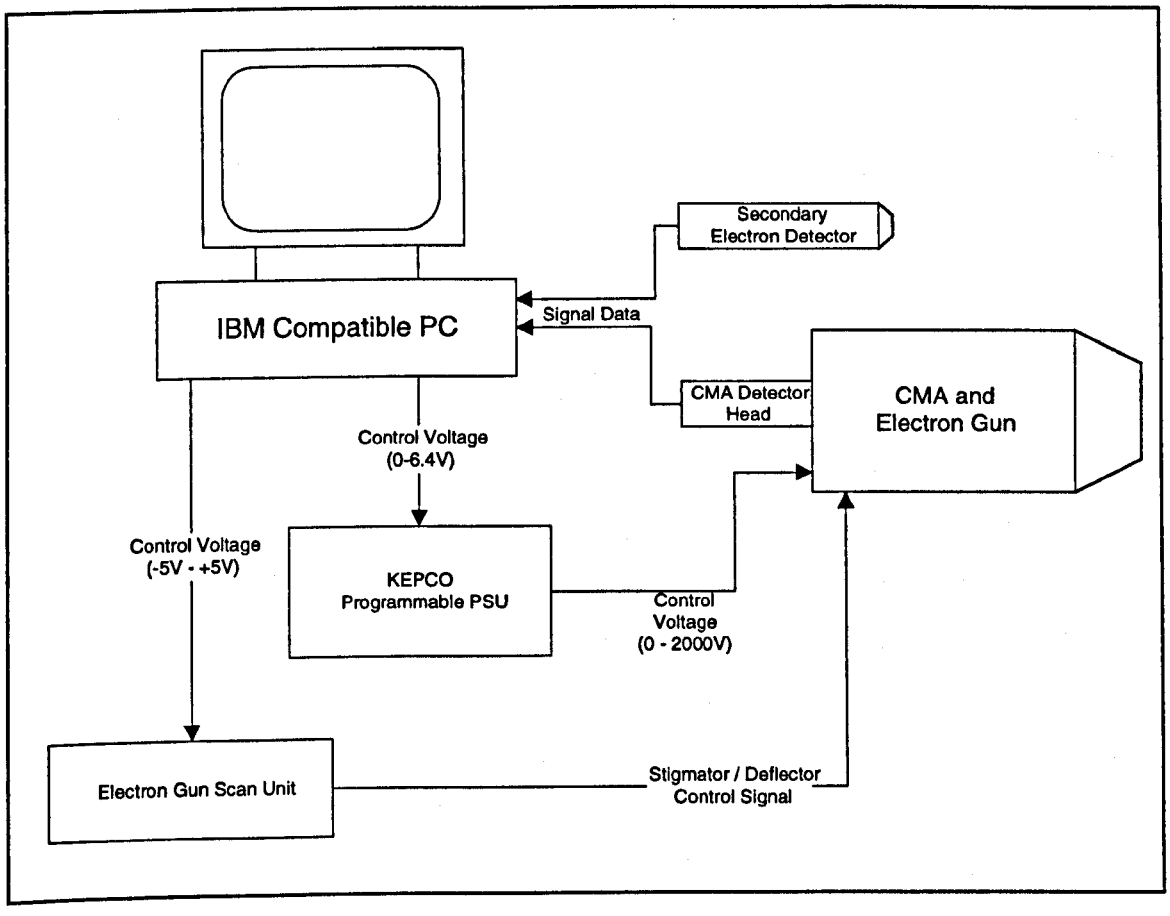

Figure 7.1 : Overview of the spectrometer control system

A block diagram representation showing the basic elements of the CMA control system is shown in Figure 7.1. The heart of the system is a 'Pentium' based PC, supplying all the required control signals and interpreting the data from the CMA detector head (digital pulses), and secondary electron detector (analogue voltages). Control of the electron beam is achieved thorough a purpose-designed scan unit which allows computer control with analogue inputs for deflection in the X and Y directions, whilst a programmable power supply provides the high voltage bias to the outer cylinder of the CMA; again this is controlled with an analogue signal.

7.1 Control Hardware

The control system for the angle resolved CMA is based around a Burr Brown interfacing system, which although it can be tailored to the requirements of a specific application, has been supplemented with two purpose built control boards to provide an accurate processor-independent timing system.

7.1.1 The Burr Brown PCI 20000 System

The hardware control system must be capable of simultaneously generating, synchronising and interpreting signals in both analogue and digital form. The Burr-Brown PCI 20000 provides a suitable solution, with a highly versatile modular architecture. The system consists of **Carriers** to provide the interface to the PC bus and **Modules** for data acquisition and control. The carrier boards are full length ISA bus style cards which accommodate up to two or three modules (depending on specification), but may also have on-board interfacing functions. The modules are small (3.9" square) cards, each with a specific set of capabilities which plug onto the carrier board. Because the carrier boards are memory mapped with assignable base addresses, a single PC can control several, and expansion is limited only by the number of available slots on the PC's motherboard.

7.1.1.1 The Carrier Board

Two carriers are used in the present control system, the PCI-20098C which provides various interfacing functions including provision for two extra modules, and the PCI-200001C which has considerably less functionality (only having the capability of digital I/O), but provides docking for up to three modules. The functions provided by the carriers are as follows:

- 16 channels of digital I/O with handshaking; these will be used for off board timing functions.
- A multiplexed analogue-to-digital converter providing up to sixteen channels of analogue input.
- A programmable burst generator to provide a crystal controlled digital clock.
- Two sixteen bit counters for data collection from the CMA.

7.1.1.2 Analogue Input and Control

Three analogue control signals are required from the computer. One of these provides the control voltage to the KEPCO programmable power supply, setting the pass energy of the analyser. A further two analogue signals are required to allow computer control of the electron beam position in two dimensions via the scan unit. The PCI-20006M digital-to-analogue converter (DAC) modules give two channels of analogue output each with sixteen bit analogue-to-digital converters (ADC), so two such modules will be used in this system.

Due to its very nature, any digital system imposes a degree of quantisation. In order to ensure that the energy resolution of the spectrometer is not limited by this, it is important to choose a DAC of sufficient resolution to accurately represent the desired voltage. The Burr Brown PCI20006M sixteen bit DAC modules can be configured in hardware for operation between various voltage ranges by the setting of jumpers on the PCB. Since the sixteen bit DAC provides a possible 2^{16} (= 65536) voltages between the lower and upper limit of its range, it is sensible to select a range which matches as closely as possible the requirements of the instrumentation it is controlling in order to minimise quantisation noise. The available ranges are as follows:

- -10V to +10V with 300μV resolution.
- -5V to +5V with 150μV resolution.
- 0 to +5V with 76.3μV resolution.
- 0 to +10V with 150μV resolution.

Examination of the signal levels shown in Figure 7.1 reveals that one of the available DAC voltage ranges (-5V to +5V) is optimally suited to control the electron gun scan unit. In this case, the available resolution in physical terms is just 1 / 65536 of the current scan range in each dimension as determined by the magnification setting. The implications of this quantisation become less and less significant as we scan over smaller areas since the discrete steps imposed by quantisation become increasingly small in comparison with the electron beam spot diameter.

In the case of the CMA pass energy, however, we need a control voltage between 0 and +6.4V to sweep the output of the KEPCO power supply between 0 and 2000V. This in turn, leads to an energy range of 0 to 3200eV for the spectrometer. The most suitable DAC voltage range for this application is 0 to 10V, giving 41943 discrete voltage levels over the 3200eV pass energy range. This imposes an absolute limit on the energy resolution of the CMA of 76meV, and since Auger linewidths are typically in the order of several eV, and spectra are normally collected in steps of 1eV, this quantisation is perfectly acceptable.

The Burr-Brown system includes a range of termination panels to facilitate bringing 'real-world' signals to and from the Modules. An analogue termination panel is used to interface the ribbon cable from the digital-to-analogue converter module to a set of shielded BNC sockets feeding the control equipment. The sending of a series of control data segments to the appropriate memory address carries out control of the DAC modules. It is therefore principally a software issue and will be covered in detail subsequently.

The analogue input to the computer will be used for interfacing signals from the secondary electron detector and head amplifier. This is based on a twelve bit analogue-to-digital converter (ADC) and similarly to the case of the DAC above, the issue of quantisation is a concern. As before, optimisation is performed by matching the voltage range to the specific application, in this case by a series of software commands. In the twelve bit case, there are 2^{12} (i.e. 4096) possible voltage levels between the lower and upper voltage limit.

The graphics capabilities of the compiler used for the control software only allow a 'palette' of sixteen user definable colors at any time, and therefore assuming that at least this resolution can be met over the operating range of the head amplifier (measured as 2 – 3V peak-to-peak typically), ADC quantisation will not degrade performance. After consideration of this, the –10V to +10V range was chosen (the maximum for this ADC), since this allows for a wide range of operating conditions, and DC offsets to be accommodated with minimum risk of saturating the ADC input.

7.1.1.3 Event Counting

By design, the angle resolved CMA produces six independent digital data signals, which are to be measured in terms of their average frequency over a given time interval, in order to build up an energy spectrum. Two sixteen bit counters are provided by the main carrier board, if necessary these can also be configured as a single 32 bit counter in order to reduce the risk of overflow. In this application, it is envisaged that counter overflow is monitored and managed by a continuously running piece of dedicated control software, eliminating the requirement for the larger counting capability, and allowing us to use the counters in sixteen bit mode without the risk of false data due to counter rollover.

To satisfy the control requirement therefore, an extra four event counters are required. These are provided by the PCI-20007M counter/timer module which occupies one of the expansion slots on the first carrier board. The inputs to this board (gating and clocking) are fed from a 40 way IDC ribbon cable, the counters can be controlled and read via the carrier board.

The PCI-200007M is based around two 82C53 programmable interval timer devices, each with three independent sixteen-bit counters (only two of which can be accessed in the required mode because of the particular hardware setup on the module). The counters are capable of handling clock inputs of up to 10MHz (Harris 1996), and have six programmable modes.

The digital connections to the carrier board include clocking and gating signals for the event counters, but also the sixteen bits of digital I/O and handshaking lines, the application of which will be discussed later in this chapter. These signals are interfaced via a Burr-Brown termination panel to allow connection between the high density (0.25mm pitch) ribbon cable feed to the carrier board and a standard 0.5mm pitch IDC connection to other components of the control system.

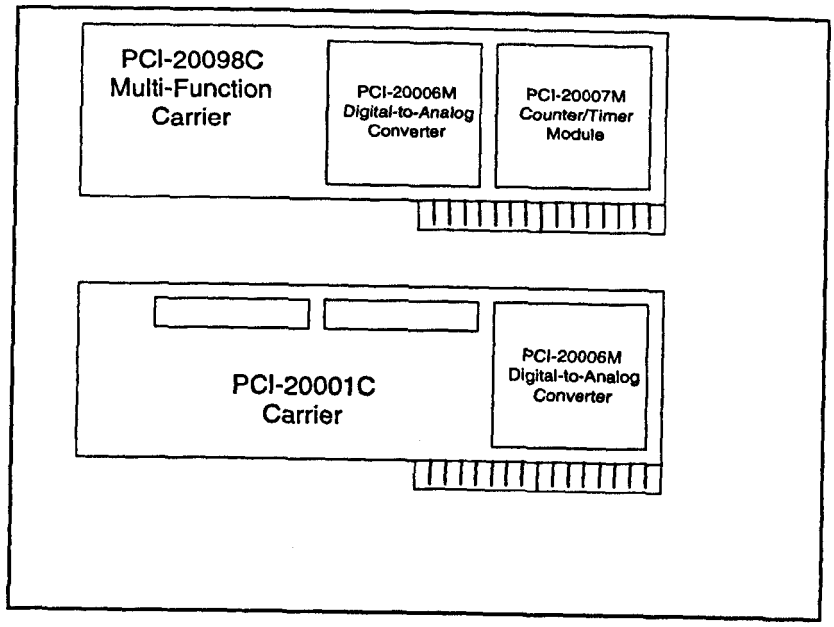

Figure 7.2: Configuration of the Burr-Brown instrumentation system

The configuration of Burr Brown carrier boards and modules used in the current control system is shown in Figure 7.2. The diagram shows the two carrier boards and three extra modules providing the basic functionality to meet the interfacing requirements set out in the preceding section of this chapter.

7.1.2 Timing and Gating

The principle function of the spectrometer is of course to collect electron energy spectra, represented as the number of counts in a given dwell time at various energy increments over a selected range. Figure 7.3 shows the basic steps involved in spectrum acquisition from a control system perspective. An essential aspect of the process depicted in Figure 7.3 is the accurate measurement of dwell time and synchronisation of event counter gating. Of course, if there is a significant variation in the time over which a signal is collected from one energy to another, or indeed from one spectrum to another, then the results will be degraded accordingly.

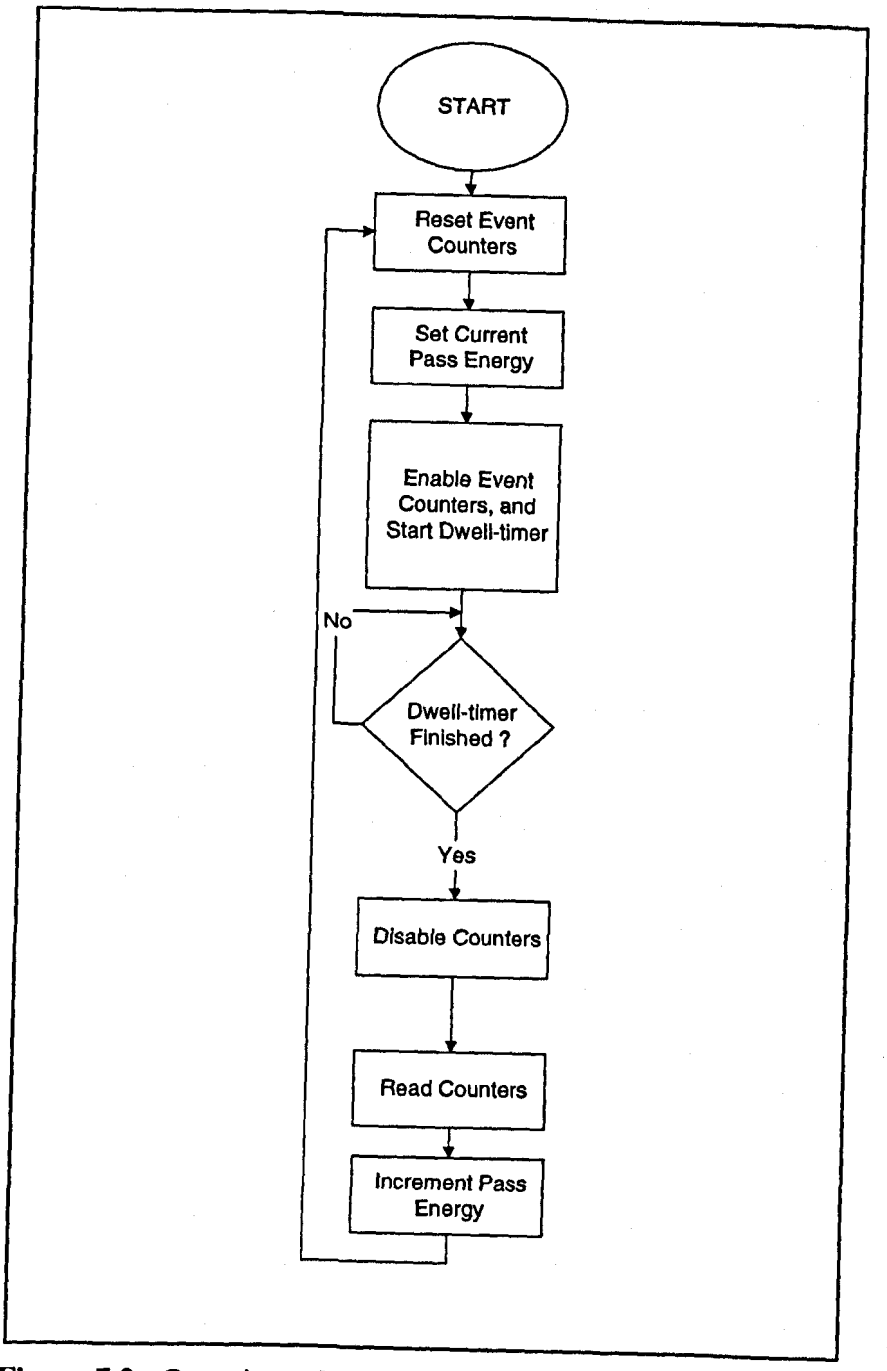

Figure 7.3 : Overview of the spectrum acquisition process

Any computer system has an internal clock which can usually be accessed from a high level programming language such as C. This would notionally provide a reliable timebase by which to judge dwell timing, since typical dwell times are in the order of at least a few hundred milliseconds and high level functions such as the 'delay' command in C allow timing to the nearest millisecond. However, there are two fundamental problems with timing with software, and these are outlined as follows:

1. Many elements of a computer system, such as a mouse for example, rely on interrupts which may stall the processor at unpredictable and undefined times. This would cause a random fluctuation in the effective dwell time if the event counters were gated in software, since it is likely that interrupts from peripheral devices would prevent the computer from enabling or disabling the counters at precisely the same instant that the timing function reads the computer's clock.
2. In the present design there are six event counters. Without the use of a true parallel programming language (such as OCCAM), there is no method of simultaneously latching and reading all counters. Sequential languages such as C do not offer this capability.

7.1.2.1 Timing System Overview

In the case of the single channel CMA system, the event counter was gated by the control software, which in turn was triggered by interrupt signals from an offboard counter. This provided a solution to the problem outlined in (1) above, but the requirement to cope with six event counters gives rise to the need for a hardware solution. This would allow the event counters to be accurately gated under software control, but without a dependency on processor intervention for accurate start and end gating. The current design for a suitable hardware based timing and gating system is shown in the context of the overall control architecture in Figure 7.4.

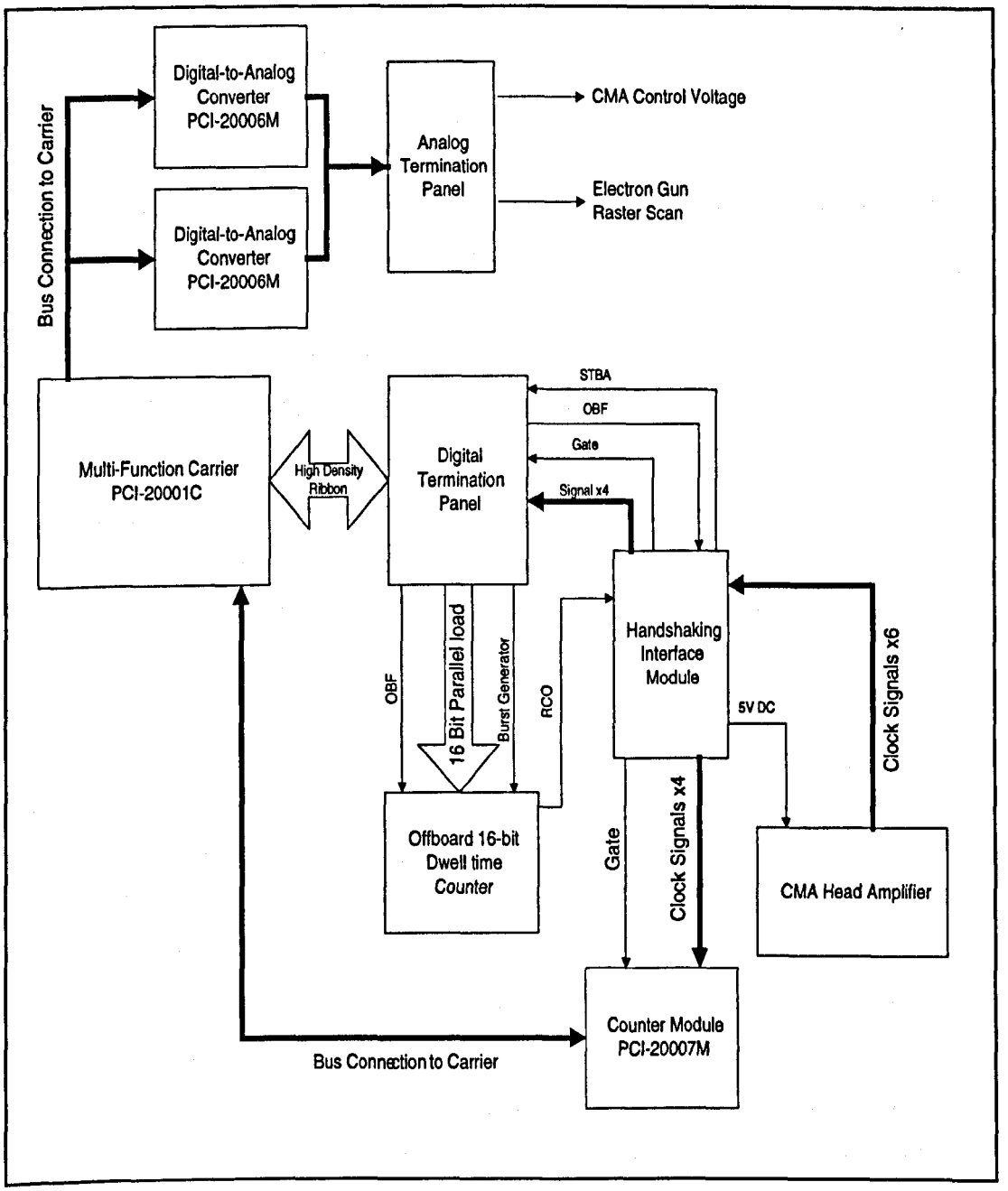

Figure 7.4 : The off-board timing system

Two main components have been added to the initial interfacing setup, an offboard sixteen bit counter, and a handshaking and interface module. The principle of operation is as follows:

- A number corresponding to the dwell time required is loaded into the offboard counter (the number must be calculated according to the clocking frequency of the counter).
- The crystal-controlled burst generator on the Burr Brown carrier board clocks the sixteen bit counter, the frequency of this is set by software.
- On the first rising edge of the burst generator clock, the event counters are enabled; this ensures that the gating and counting operations are synchronous.
- At the instant that the dwell time counter reaches zero, the counters are asynchronously disabled and can then be read and reset as appropriate.

7.1.2.2 Handshaking and Control Signals

The status of the offboard counter is monitored in software by the use of handshaking signals integral to the Burr Brown system. The handshaking signals, and the purpose they serve is explained below:

- **OBF :** This signal is automatically set by the carrier board when valid data appears on the digital I/O ports. This is used as a trigger to load the offboard counters, and indicates the start of the dwell time counting operation.
- **BG :** The burst generator clock signal is started and stopped in software, and it is the task of the hardware system to monitor the first rising edge of the BG clock.
- **RCO :** The ripple-carry-out of the sixteen bit counters. This signals the end of the dwell time.
- **STBA :** An interrupt signal which can be interrogated by the software to monitor the status of the timing signals, but is also used as a gating signal to the event counters. STBA is set by the first rising edge of the burst generator after the OBF signal and is cleared asynchronously by the RCO line from the counters. An asynchronous clear removes the risk of digital 'hazards' occurring at the end of the dwell time as a result of the precise synchronisation between the RCO signal and the burst generator clock, ensuring repeatable and accurate time measurement.

The timing diagram for the handshaking signals is shown in Figure 7.5 below. Note that the number of burst generator pulses between the OBF and RCO is determined by the dwell time and burst generator frequency.

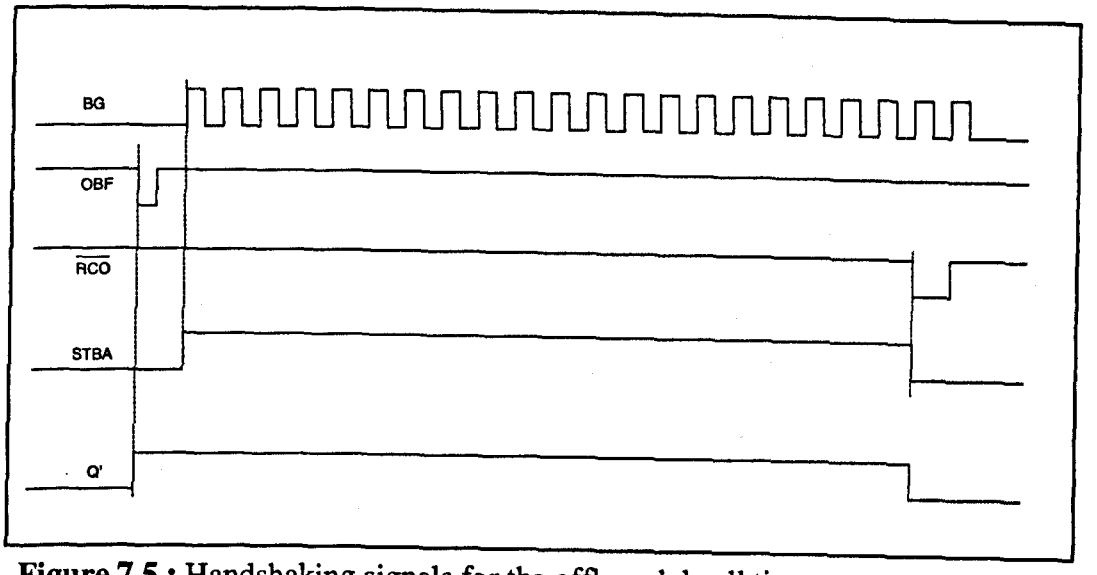

Figure 7.5 : Handshaking signals for the offboard dwell timer

7.1.2.3 The Off-board Counter

Figure 7.6 shows the circuit diagram for the dwell time counter used in the present design. The circuit consists of four 74169 synchronous four-bit up/down counters, each of which is enabled by the ripple-carry-out of its neighbour to produce the full sixteen bit counter. Synchronous loading is provided by the 'LOAD' inputs which are all tied to the OBF line from the Burr Brown carrier board. Both the 'ENP' and $U/\overline{D}$ are tied low, as the second enable input is fixed and the counter is required to count down.

The burst generator frequency is set to 1kHz, this gives a range of dwell times between 1ms and 65.6s which is ample for most Auger experiments. The 74169 counters are rated at a maximum of 35MHz which gives a considerable performance margin. Conceivably, by altering the burst generator frequency accordingly, dwell times outside the range specified above could be accommodated although it is not intended that this facility is included at the current stage in the design.

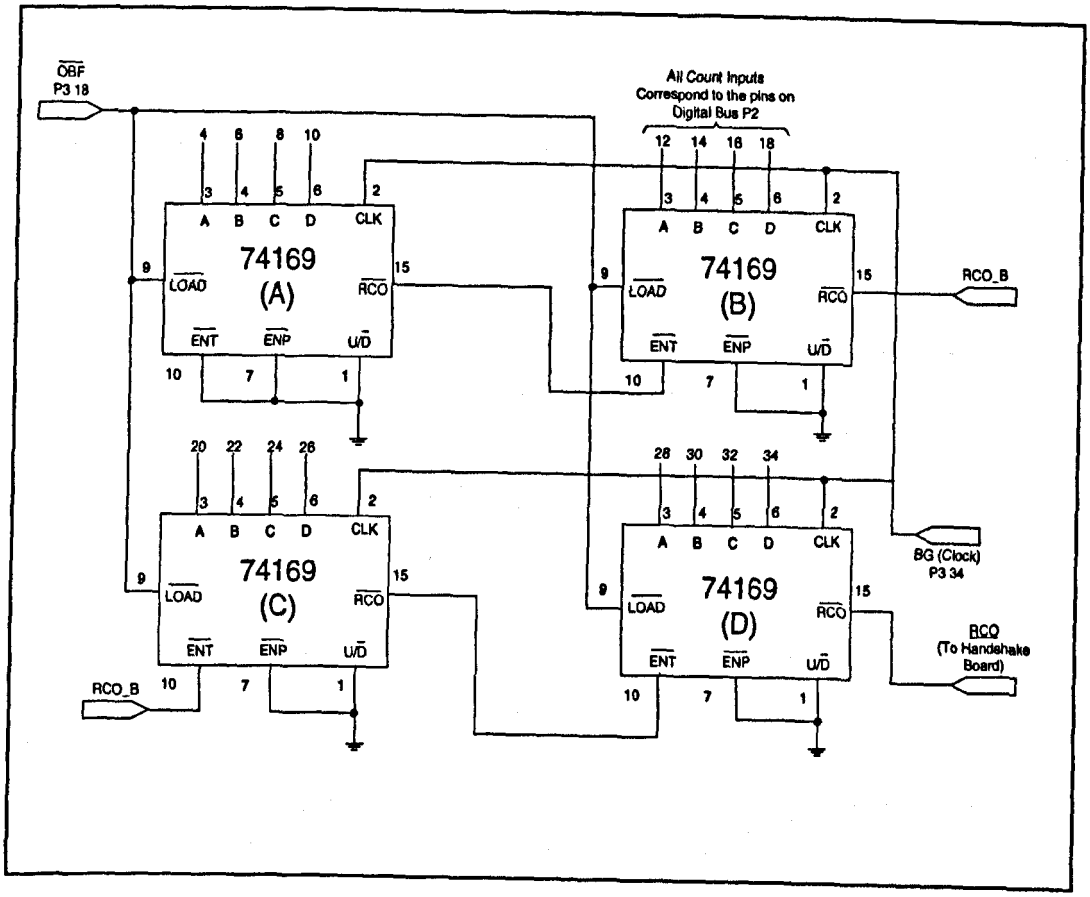

Figure 7.6 : The sixteen bit off-board dwell time counter

7.1.2.4 The Handshaking/Interface Board

The second of the two boards added to the Burr Brown control system performs two tasks: the generation of the handshaking signals, and the interfacing of control and data signals between the CMA head amplifier, the external counter module and the Burr Brown carrier and counter module.

Figure 7.7 is a schematic diagram of this add-on board. The interfacing between the head amplifier and the two sets of event counters requires no extra logic, but a dual flip-flop is used to generate the handshaking and gating signals to synchronise the data collection process. Connections to the carrier board are made with single wires from a terminal block on the interface board to the Burr Brown digital termination panel, but this could conceivably be replaced with a ribbon cable in future.

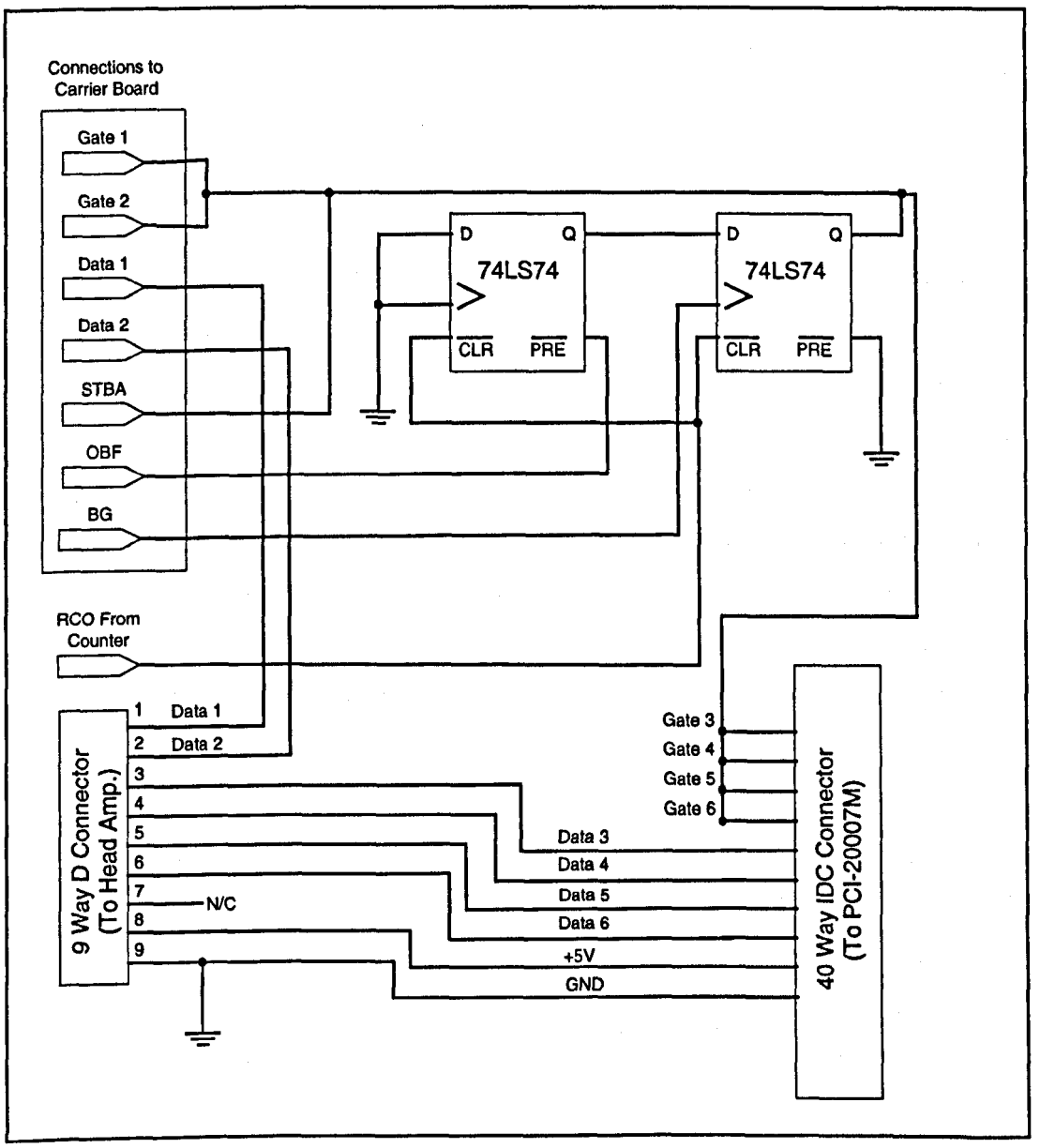

Figure 7.7 : Circuit diagram of the handshaking and interface board

7.2 Control Software

This section describes the design of a complete software package to collect, store and display data from the spectrometer system. The main functions of the control system were outlined in section 7.0, this leads to a number of design considerations:

- Although it must perform a wide range of quite complex low–level interfacing functions, these must be transparent to the user. The final program must be as user-friendly as possible with a self-explanatory graphical user interface.
- The package is intended to bring together all aspects of spectrum and image acquisition so that, for example, a computer generated SEM image can be used to accurately position the electron beam for Auger studies to take place.
- Reliability is an essential aspect of any program; this will be taken into account with a number of measures to eliminate the possibility of errors as a result of incorrect user input. Also, the data collection routines will be designed to cope with a wide range of possible conditions, to reduce the risk of inaccurate data being generated.

The description of the overall system is carried out with 'bottom-up' design approach, first covering the low level interfacing routines used to communicate with the control hardware, then building on this with the design of the functional modules which carry out the control operations. Finally the design of the graphical user-interface is explained, this section also shows how the various features of the analyser are integrated.

Previous generations of CMA control software used on this instrument were written with a combination of the FORTRAN and C programming languages. This allowed the use of existing user interface software written in FORTRAN, whilst the low level control was carried out with C. The software described here is written exclusively in Borland Turbo C++. This allows direct control of hardware devices but also provides a range of graphical display routines which can be exploited in the design of a user interface.

7.2.1 Low Level Control Software

The Burr-Brown PCI-20000 instrumentation system is accessed by a technique known as memory mapping. This means that the programmable devices on the interfacing carrier board and the add-on modules can be controlled by writing eight bit numbers to appropriate memory locations of the host computer. The specific base address of this memory block can be selected by setting the switches on the carrier's 10-position DIP switch. The functions and registers of the carrier and modules can then be written to or read from various offset addresses relative to this base address. In this case the base address of the first carrier was chosen as CE00, and the second as CE40.

Figure 7.8 shows the memory map of the carrier boards. For the PCI-200098, addresses 0300 to 03FF are reserved as it only has support for two modules.

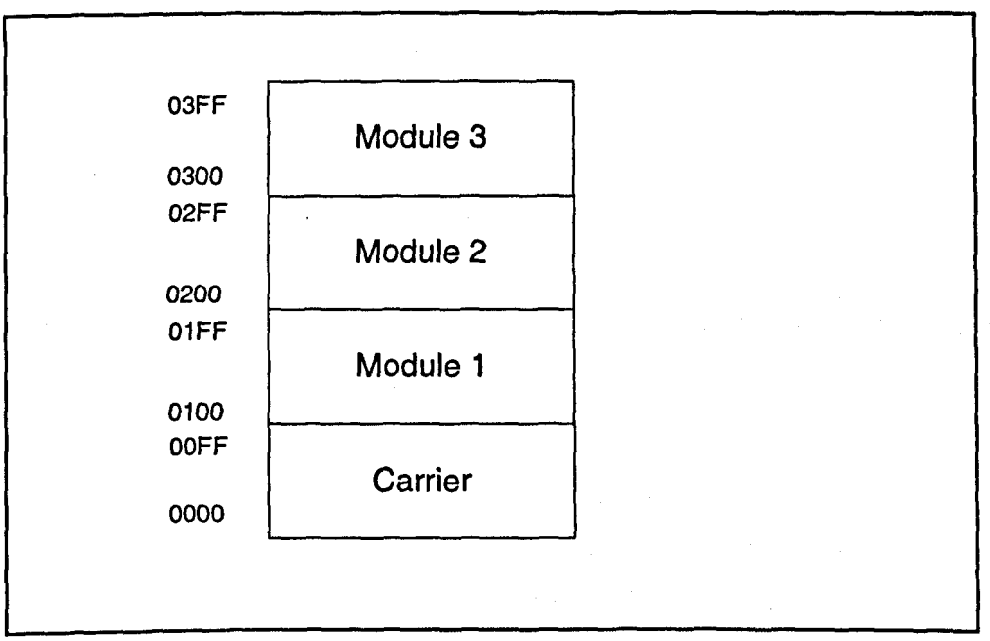

Figure 7.8 : Carrier board memory map relative to base address (HEX)

The header files described in this section have been designed to make the hardware control operations as efficient as possible, so that higher level functions can access hardware devices with simple function calls. Some of the hardware interfacing commands are embedded into the higher level software; these will be described in the context of their host functions.

7.2.1.1 Checking and Initialisation

Each of the components in the Burr Brown system has an identification number which can be accessed by reading its base address. The function 'bb_check' reads the base addresses of the various carriers and modules and compares them with the identification numbers given in the documentation. This function uses the 'peekb' command in Borland C++, which operates on the following syntax:

result = peekb(base_address, offset),

where 'result' is an eight-bit number, corresponding to the current value at the given address in memory. This syntax is particularly convenient for reading from the Burr Brown instrumentation system since the base address is the same for all functions related to a given carrier.

An initialisation routine has been written to set up some of the parameters of the hardware which only need to be defined once; i.e. each time the program is run. This applies mainly to the six counters, two on the main carrier, and four in the PCI-20007M module. The flow diagram shown in Figure 7.9 outlines the operation of the function 'bb_init'. Firstly, the four 82C54 counters are set into mode 0; this is the binary event counting mode. The counters are then reset, and a similar process is carried out on the two counters on the carrier board.

It is important to note that there is very little similarity between the four counters on the module and the two on the carrier board, for example, the counters on the carrier board count up, whereas the 82C54 devices on the module count down. These effects will be considered in more detail when the design of the high level counter control software is discussed.

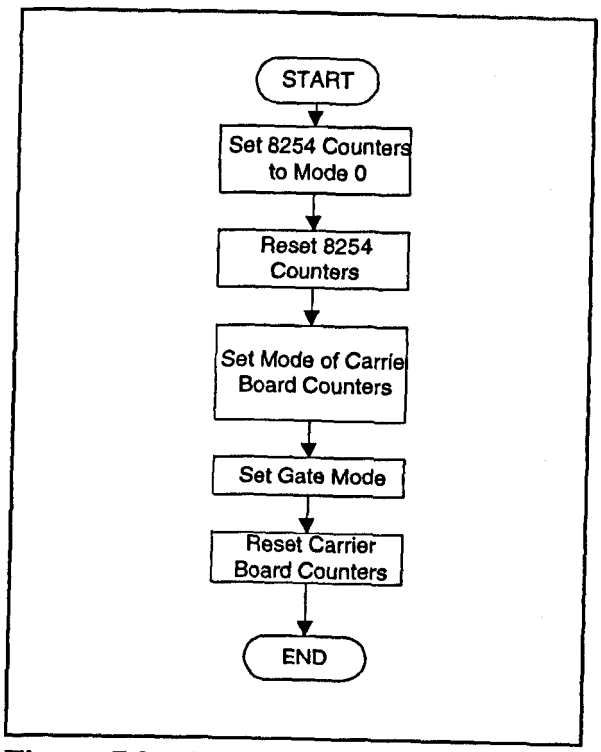

Figure 7.9 : Flow diagram of the control system initialisation process

7.2.1.2 Control of the Digital-to-Analogue Converters

As explained already, data is sent to the hardware by writing to memory addresses. The command for this in Borland C++ is 'pokeb' with the syntax of:

pokeb(base_address, offset, value),

where value is an eight-bit number. For this reason, it becomes the task of the software to split sixteen-bit numbers into two eight-bit bytes so they can be sent to the hardware (which is almost exclusively based on sixteen-bit devices).

The process of accessing the digital-to-analogue converters is no exception to this, as outlined in Figure 7.10. The function 'dac_out' converts the voltage required to an appropriate sixteen-bit number (from 0 to 65535) calculated according to the range setting of the DAC. It then strips the high and low byte from the sixteen-bit number and writes them to the appropriate addresses as given in the Burr Brown documentation. The DAC devices must then be given a 'strobe' command to latch out the new voltage.

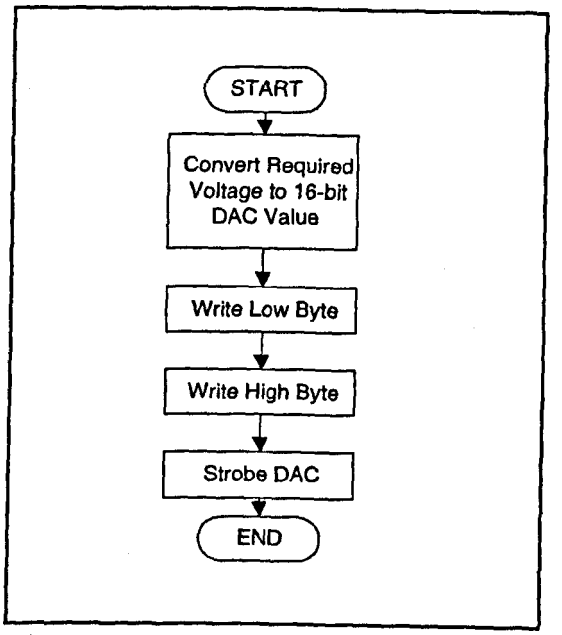

Figure 7.10 : The process of writing to the DACs

7.2.1.3 The Burst Generator

The multi-function carrier provides a programmable burst generator which generates a series of bursts of pulses, referenced to the carrier board's clock. Figure 7.11 shows the required programming steps for configuring the burst generator.

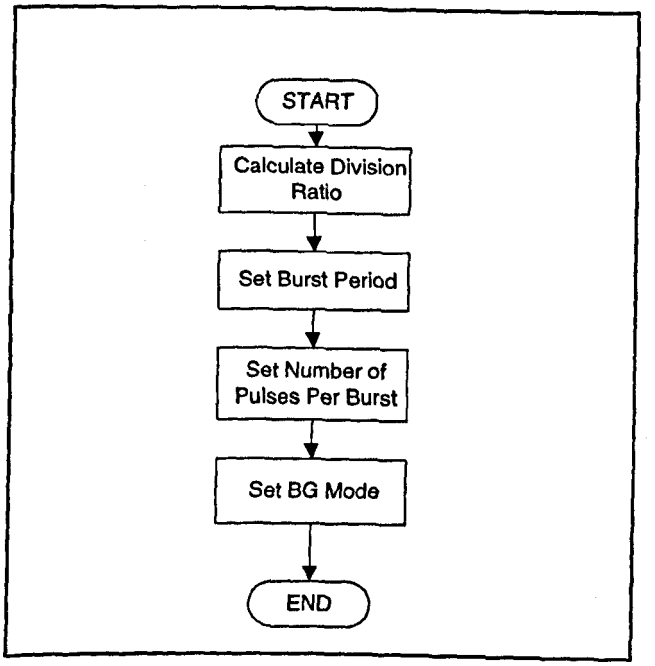

Figure 7.11 : Configuring the burst generator

As we see from Figure 7.11, the burst period is set according to the 8MHz internal clock of the carrier board and is written as a 32-bit number to a register on the carrier board. Although it is possible to produce bursts of high frequency pulses with a programmable gap between them, in this case we are aiming to provide a continuous clock signal. For this reason, we set the number of pulses per burst to one, and the resulting clock frequency is determined only by the burst period (rather than the pulse period as one might expect). The mode of the burst generator is set to continuous, and it is then a simple operation of writing single numbers to the status register to start and stop the clock signal.

7.2.1.4 Digital Input and Output

The process of sending digital signals to the off-board timing hardware is central to the operation of the spectrometer control system. In the current design, all digital interfacing is carried out by one of the two carrier boards, which provides 16 channels of buffered, TTL compatible I/O, plus various handshaking lines. Although the Burr Brown interfacing system offers a diverse range of configuration options, setting up a 16-bit digital output is relatively straightforward and can be done with a single write operation to the mode register. The process is covered in Figure 7.12, and the software to implement this function can be found in the header file DIO.H.

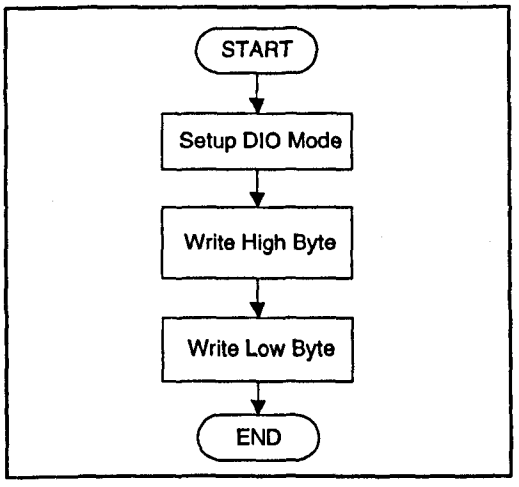

Figure 7.12 : The process of 16-bit digital output

7.2.1.5 Handshaking Signals

Many of the handshaking signals involved in the digital interfacing operation are handled automatically by the hardware. For example, the OBF line is automatically set by the digital output hardware when valid data is first written to the port. The only handshaking signal required for synchronising the software is the STBA line, which indicates the start and end of the data collection process. Figure 7.13 shows a flow diagram for the process of reading the STBA handshaking signal, returning a '1' if it is set, and a '0' otherwise. The file 'HANDIO.H' contains the software to realise this function.

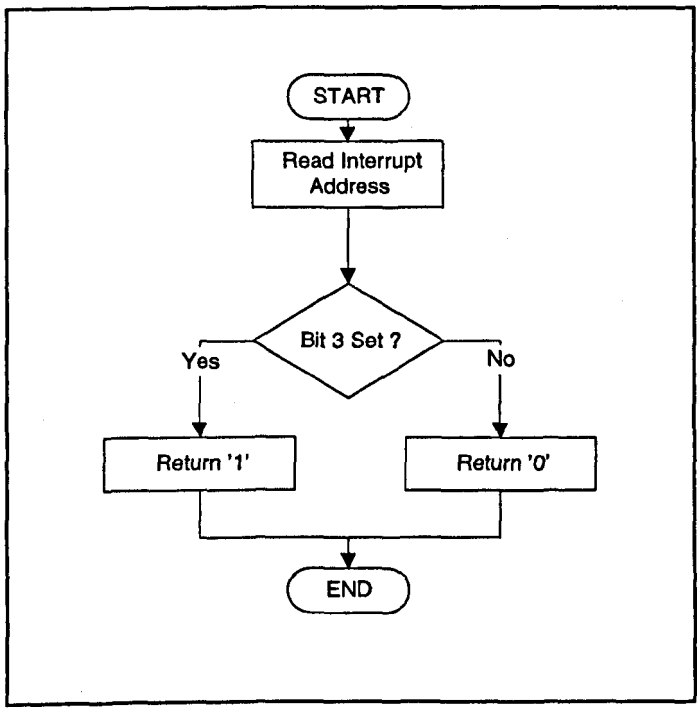

Figure 7.13 : Reading the STBA handshaking line

7.2.2 The Main CMA Control Program

Design of the interfacing software has been covered in detail in the previous section, attention will now be turned to the higher level functions of the program. This includes the various functional software modules which acquire and manipulate data from the spectrometer, and also a complete integrated graphical user interface.

The software design described here was an extensive undertaking, and represents several thousand lines of source code. As a result, not all of the functions involved in the program are described in intimate detail and the reader is referred to the commented source code listings in the appendices for further information. As before, flow diagrams are used as a means of depicting program structure, Figure 7.14 shows the overall operation of the control program.

To some extent, the functional elements shown in Figure 7.14 are reflected in the program structure as individual C++ functions. The remainder of this section is devoted to breaking down these functional units and describing how they are implemented in software. The design of the graphical environment will not be separated from the rest of the high level software since some of the control functions are closely tied in with parts of the user interface and the interplay between the two can be complex.

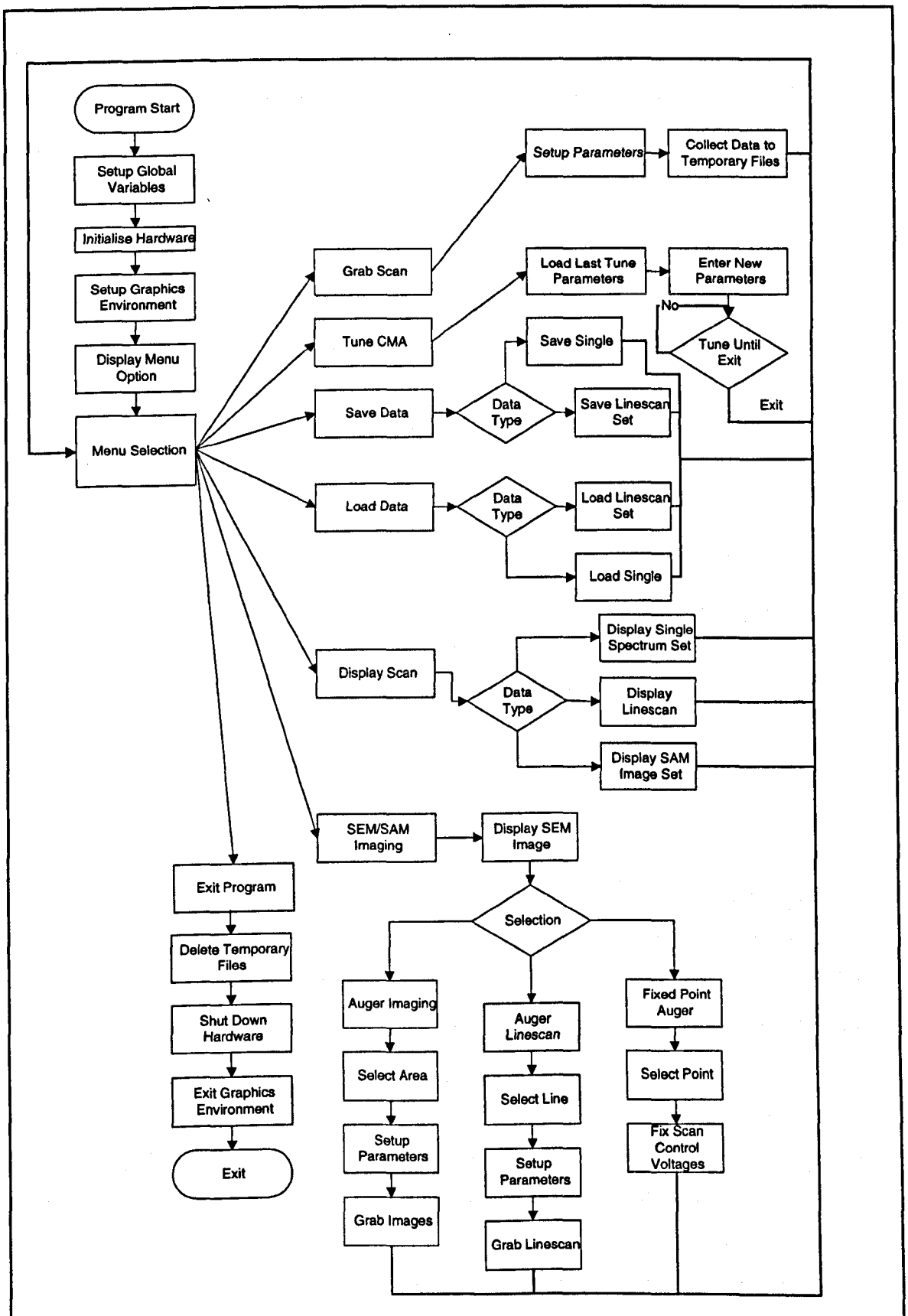

Figure 7.14 : Overview of the CMA control software package

7.2.2.1 Program Initialisation

The section of the program before the menu loop is exclusively devoted to setting up both hardware and software options which are required throughout the operation of the program. The setting up procedure involves the following steps:

- **Setting Global Variables :**

  This is carried out as part of the header files, *HEADER.H* and *GUIHEAD.H*, where carrier base addresses and constants used in the user interface are defined. Further definitions are included at the start of *CMA6.CPP* to set up parameters which are stored and modified by several functions. These essentially define the starting conditions for the program.

- **Hardware Initialisation :**

  The function *bb_init* is called to setup the interfacing hardware (see 7.2.1.1)

- **Graphics Setup :**

  The function *main_screen_setup* and *initialise_graphics_environment* execute commands to detect graphics hardware and select graphics mode for the user interface. Also global parameters such as background colour and text colour are set in this routine.

7.2.2.2 The Main Menu

All the functions of the software package are accessed from a central menu loop which runs continuously until the user chooses to exit the program. For design of the menu loop a Warnier-Orr diagram (Kirk, 1992) is used. This allows a function to be broken down from a general statement to a step-by-step description of its operation on a pseudo command line level.

Figure 7.15 is the first Warnier-Orr diagram, showing the operation of the Main-Menu routine, this process is embodied in the functions '***draw_menu***' and '***menu_input***'. As well as providing access to the various control functions of the software package, the main menu also allows the user to switch between the currently available scan sets.

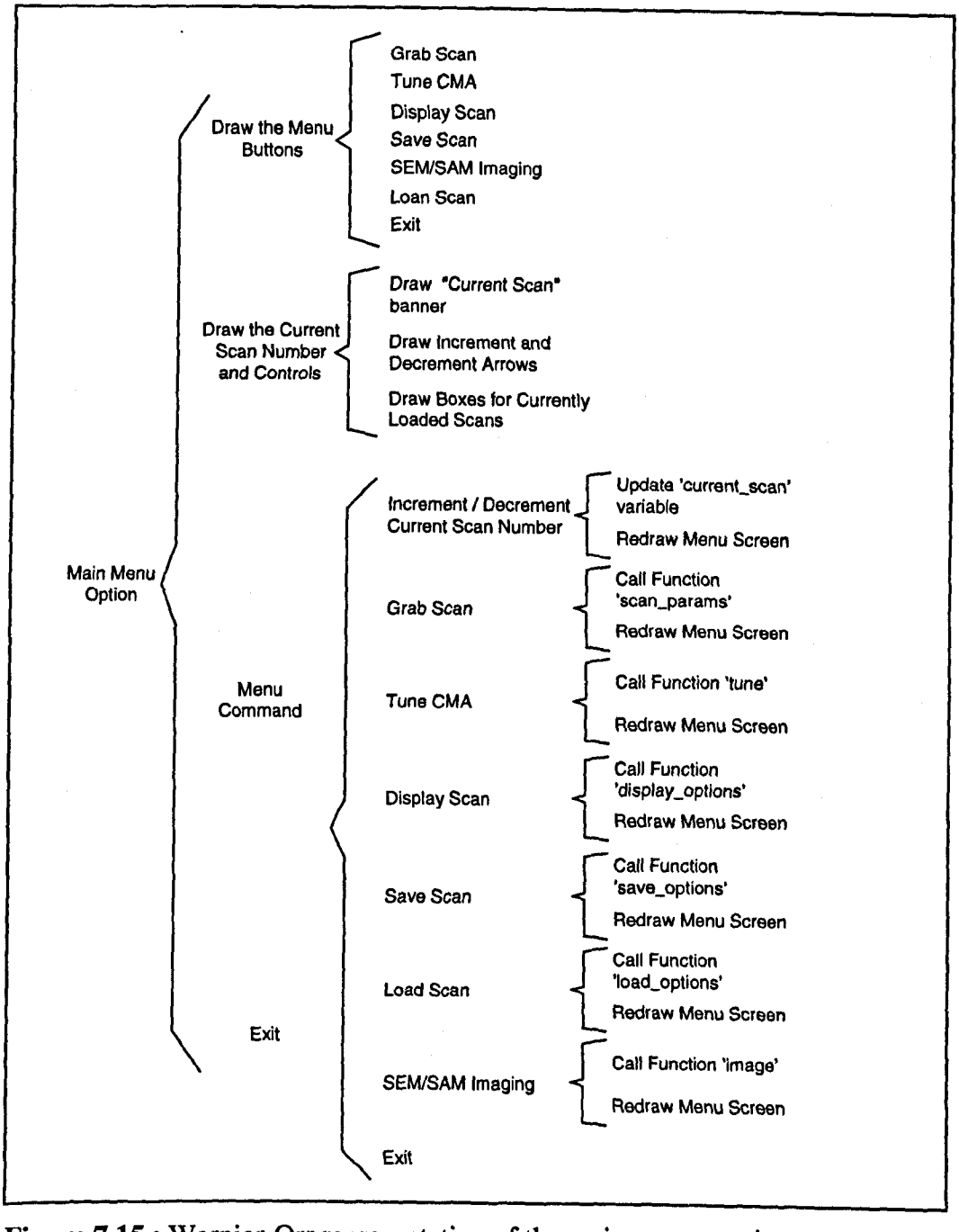

Figure 7.15 : Warnier-Orr representation of the main-menu routine

In order to allow large amounts of data to be handled by the control software, sets of spectra are never actually loaded into the computer's physical memory. Instead, when the spectrum sets (each of six spectra) are 'loaded' into the virtual workspace, they are still held as temporary data files on the hard drive. The temporary files can then be accessed to be re-saved, displayed or processed. This mechanism will be described in more detail when the 'save' and 'load' operations are covered later in this chapter.

The main menu graphics functions make use of the Borland C++ built-in graphics functions, which allow the programmer to generate graphical environments with a

library of pre-written functions (similar to API calls used in Microsoft Windows programming). In this development, some functions, such as the '***button***' function which draws a text-filled box on the screen, have been added to speed up user interface implementation. All the graphics functions use the unit of pixels to place objects on the screen. This software uses the standard 640x480 pixel VGA mode, supported by most modern PCs.

In common with all the subsequent user interface functions, the '***menu_input***' function uses calls to the mouse driver routines in the user interface C++ file 'GUI.CPP'. These routines communicate with the mouse hardware via interrupts set up by a standard Microsoft compatible mouse driver, and have been written to simplify user interface design (QUE, 1989). Function calls such as 'mouse_on' and 'mouse_off' set the status of the mouse cursor, while 'mouse_y' and 'mouse_x' return the current x,y co-ordinates of the mouse cursor on the screen. The units of pixels are used for all the mouse functions, making it straightforward to relate mouse position to the placement of graphical objects on the screen. Figure 7.16 shows the resulting menu screen, with nine spectrum sets currently loaded into the workspace (the screen-dump has been cropped to remove the blank sections of the display).

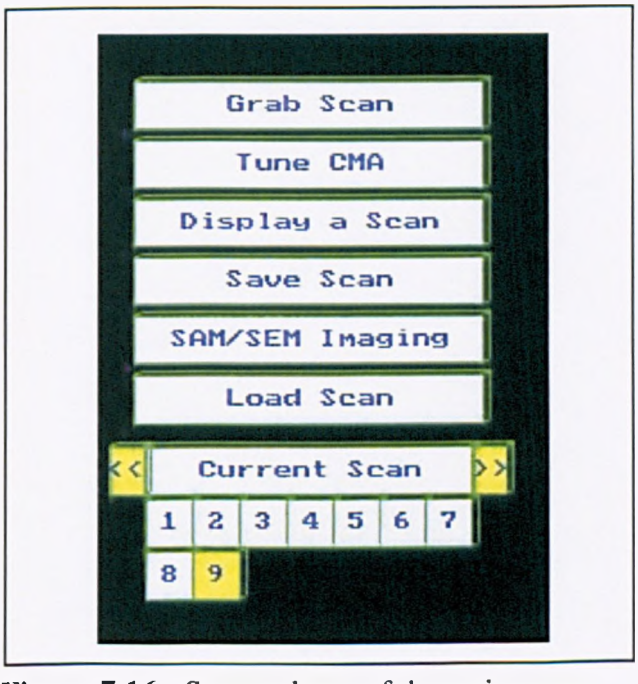

Figure 7.16 : Screen-dump of the main menu

7.2.2.3 The 'Grab Scan' Option

The 'Grab Scan' option from the main menu allows the user to simultaneously collect spectra from all six channels of the analyser. This process is shown in Figure 7.17 as a Warnier-Orr diagram.

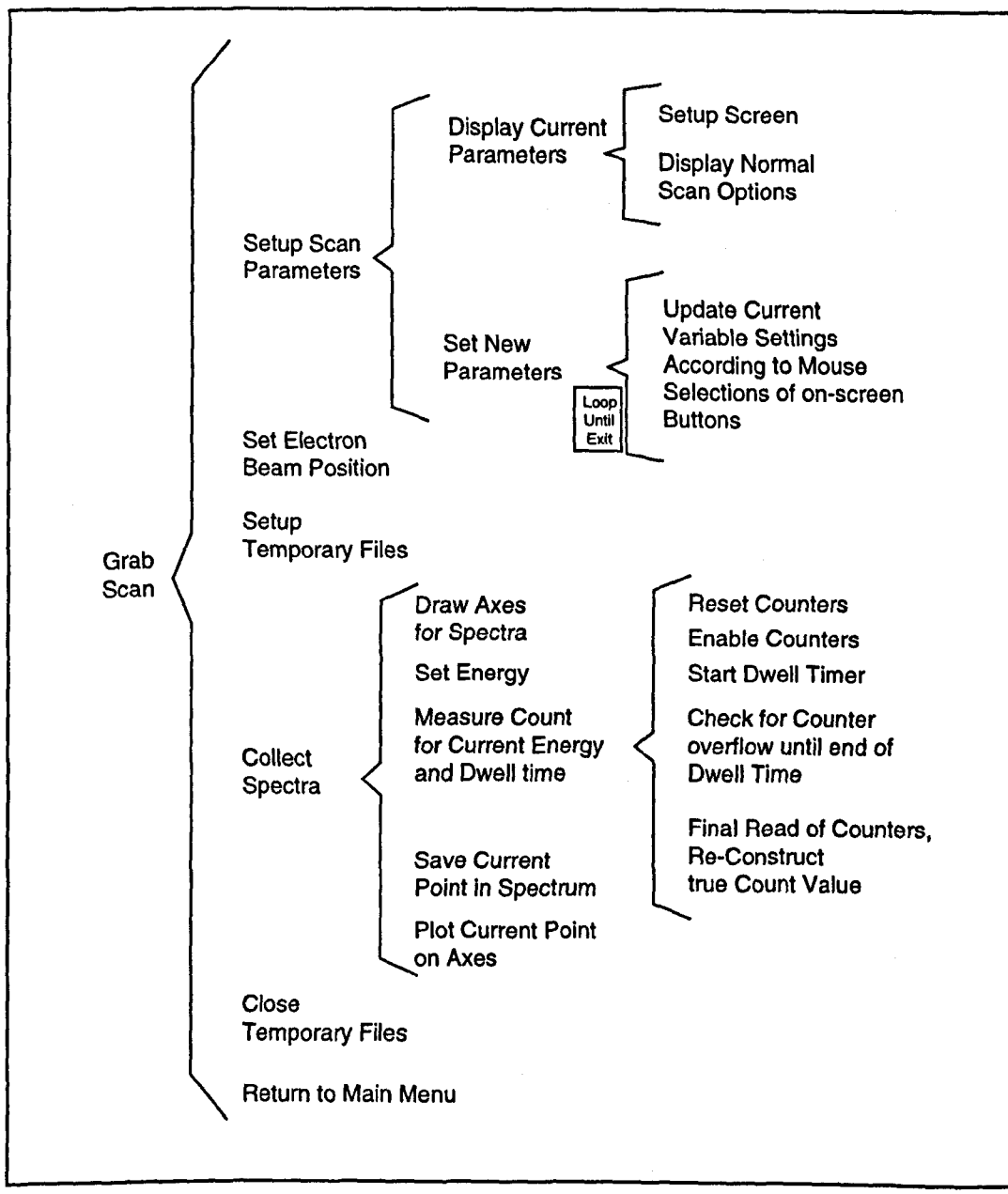

Figure 7.17 : Warnier-Orr diagram of the 'Grab Scan' process

From the main menu, the *'scan_params'* function is called, this is a harness function, which calls *'display_params'*, and *'set_params'* to allow the user to select the parameters for spectrum collection. These are as follows:

- **Energy Range** : This can be set to anything between zero and 3000eV. The theoretical upper limit of this is set by the performance of the computer controlled power supply connected to the outer cylinder of the CMA.
- **Dwell Time** : This can be set at anything from 1ms to 65s. The range of this is set by the burst generator frequency and the resolution of the off board counter (see 7.1.2.3)
- **Number of Points** : The number of points at which a count is measured, the upper limit is currently set to 5000, but could be increased if required.
- **Maximum Count** : This defines the maximum expected count for any given point in the spectrum, and must be estimated by the analyst. It is used to scale the real time display of spectra during the collection process.

Figure 7.18 shows the parameter adjustment screen. Values are adjusted by positioning the mouse cursor over the arrows either side of the numbers and either clicking the mouse button (to change by a pre-defined increment) or holding down the button (to rapidly step through possible values).

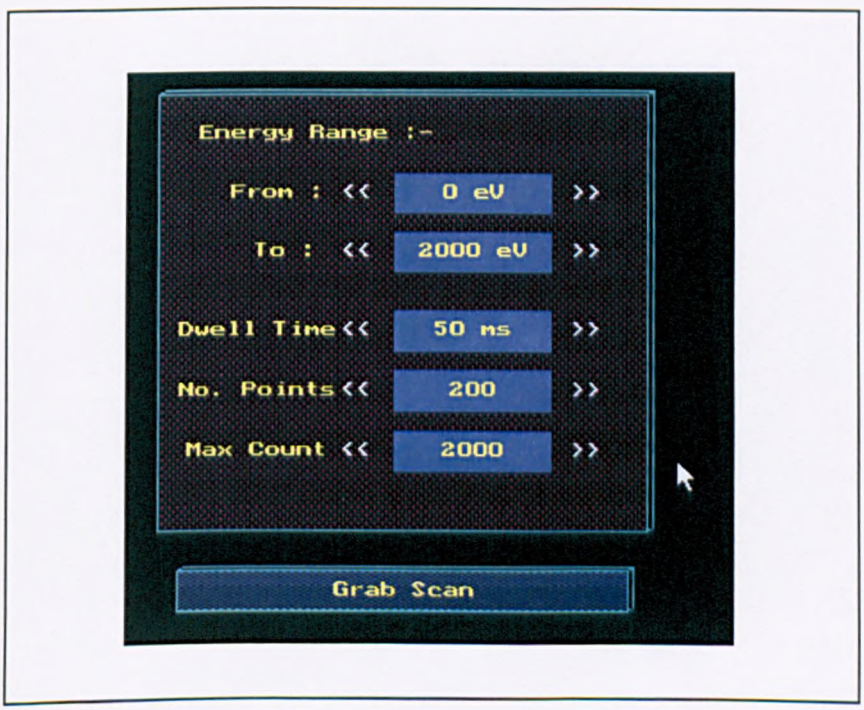

Figure 7.18 : Setting up the parameters for the grab scan operation

The method by which a single, or sustained click is detected and processed is shown as a flow diagram in Figure 7.19.

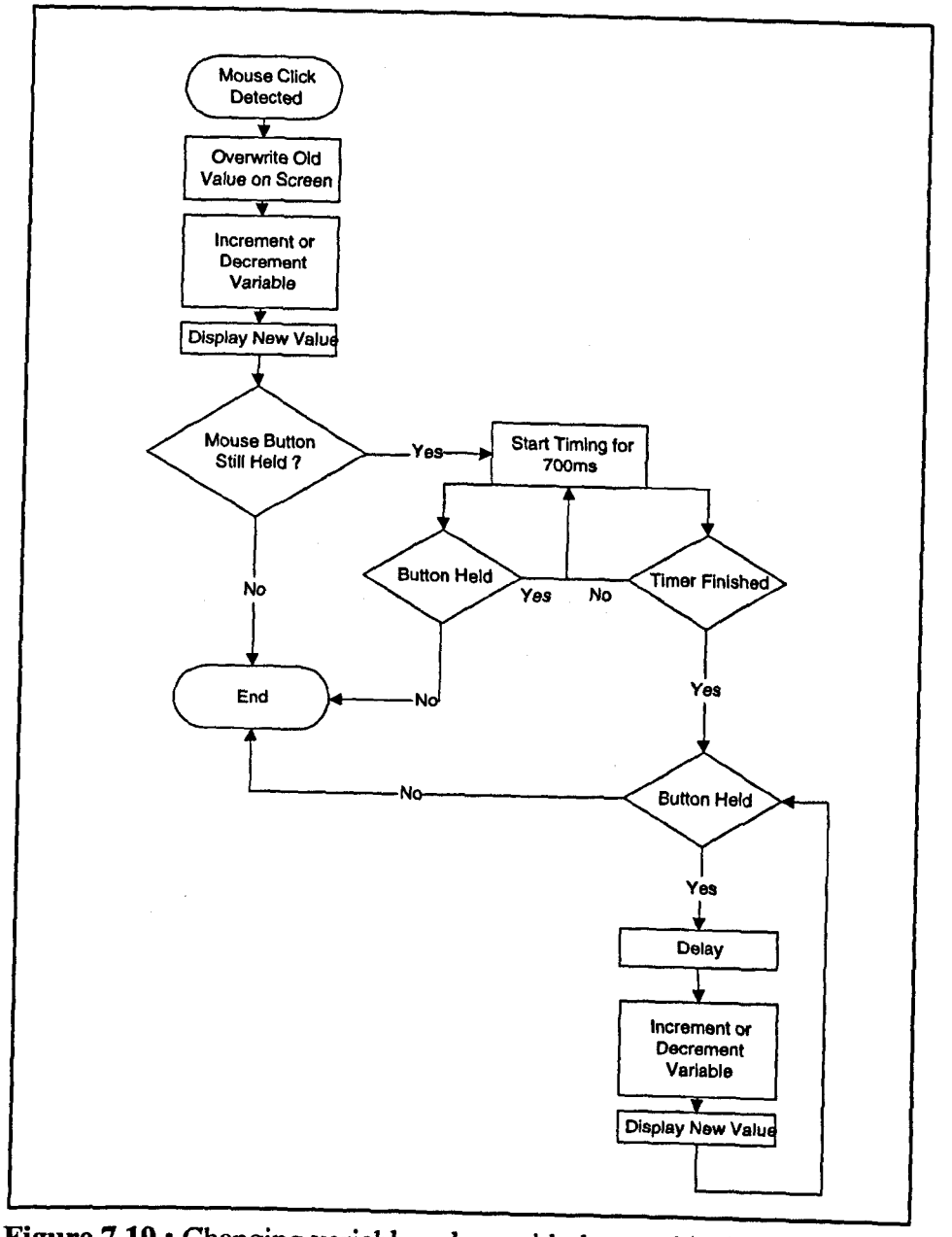

Figure 7.19 : Changing variable values with the graphical user interface

The value is first altered according to which 'button' has been selected on screen, then if the mouse button is held for a further 700ms the value is rapidly updated in an iterative loop until the button is released. The off-board dwell time counter is used to time the 700ms delay (see 7.2.1), allowing the software to continuously check for release of the mouse button during this period, so that single clicks in rapid succession are possible without forcing a 700ms delay between each.

Once the parameters are set, the spectra can be collected. Three main functions are involved in this operation:

- ***grab_scan*** : The main harness function containing the iterative loop to sweep through the spectrum, acquire the count results and display them.
- ***draw_axes*** : sets up the screen for the real time display of spectra during collection. Both axis limits are user defined so higher than expected counts may not be displayed.
- ***get_point*** : This function returns the count at a given energy for all six channels. It calls the low-level hardware control functions explained in 7.2.1 to accurately measure the dwell time and to correct for counter rollover. The latter of these operations is carried out by repeatedly reading all six counters during the dwell time, and incrementing a register every time a rollover is detected. The actual counter value can then be reconstructed. Using this technique, the maximum count value is limited only by the data type used to store the final number.

Spectra from the six detector channels are differentiated by six colours which can be easily defined in software. An example screen-dump of the collection process is shown in Figure 7.20, the background colour has been inverted for clarity.

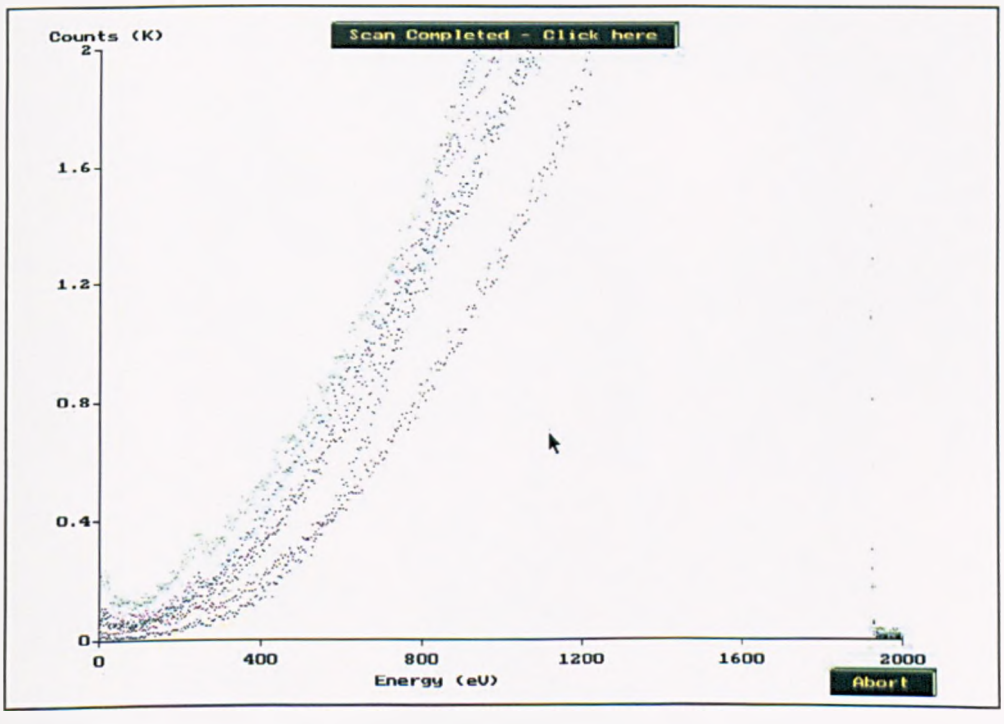

Figure 7.20 : The real-time spectrum display during collection

7.2.2.4 The 'Tune' Option

With any electron spectrometer, the physical position of the sample under examination, along with the beam conditions must be optimised in order to ensure that valid results are obtained. As explained in Chapter 5 the CMA is particularly sensitive to sample position, making this optimisation all the more critical.

A tuning function is incorporated into the control software package to allow the analyst to observe spectra over a relatively small energy range and with short dwell times. Normally this optimisation is carried out over a range of about 100eV in the region of an elastic peak. The sample position and electron beam conditions can then be adjusted for maximum peak intensity.

Figure 7.21 is a Warnier-Orr diagram outlining the main software elements of the tuning operation. The relevant C functions are '***tune***' and '***tune_ch***' which are unique to this process, and '***display_params***', '***set_params***', '***draw_axis***' and '***get_point***' which are also used in the 'grab scan' routine. As indicated by the number of common functions, the tuning process is very similar to normal spectrum collection, except that it rapidly scans in energy over the required range until the user aborts the operation with dwell times that would be too short to collect spectra with a high signal to noise ratio.

To help with the tuning process, a bar-graph at the top of the screen displays the current maximum count over the selected energy range, as well as a highest maximum count so far. It is intended that this maximum count figure relates to the elastic peak, although this metric relies on the channel plates having sufficient dynamic range to cope with the large electron flux associated with an elastic peak without going into saturation.

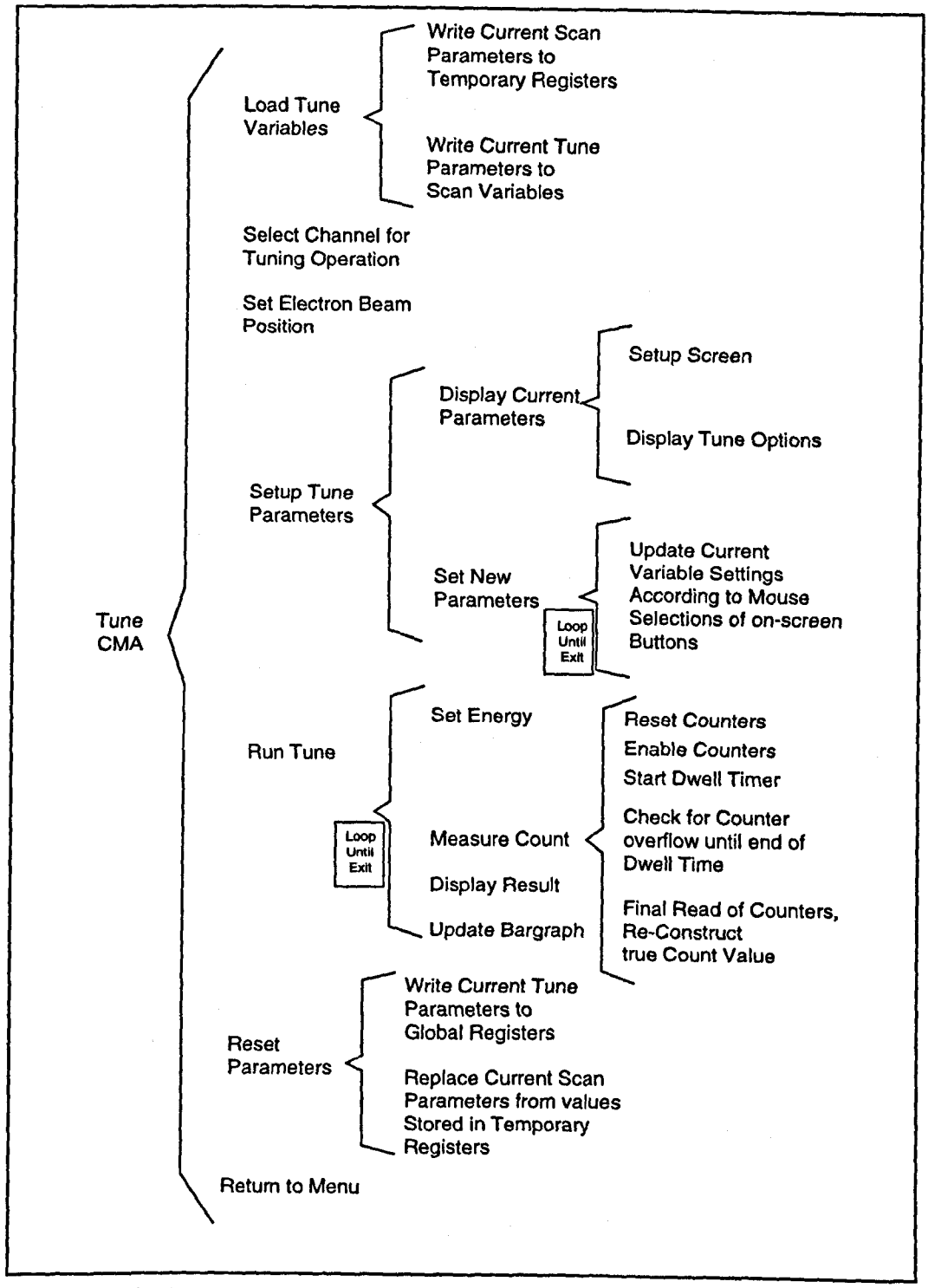

Figure 7.21 : Warnier-Orr diagram of the CMA tuning software

As the diagram illustrates, the tuning parameters are stored separately from the normal scan parameters, and written in place of the scan parameters during the tuning operation. Both sets of parameters are stored in global registers and therefore initially take the values of any previous runs. This saves time if the analyst wishes to repeatedly collect data with similar analyser settings.

The tune function is able to display data either from any one of the six single channels, or the sum of all channels. Although for a conventional CMA, there is only one ideal focal point at which the sample must be positioned, the angle resolved detector head is intentionally placed slightly behind the focal plane. This gives rise to a complicated relationship between sample position and electron yield.

Under certain conditions, it is anticipated that the optimum sample position may not be the same for all six of the channels. If spectra are to be acquired from the full 360° range, it would probably be most appropriate to tune the analyser such as to optimise the sum of all six signals, but it may be desirable to tune over a more limited angular range if data is only required from a subset of channels. Figure 7.22 is a screen dump illustrating the tune function.

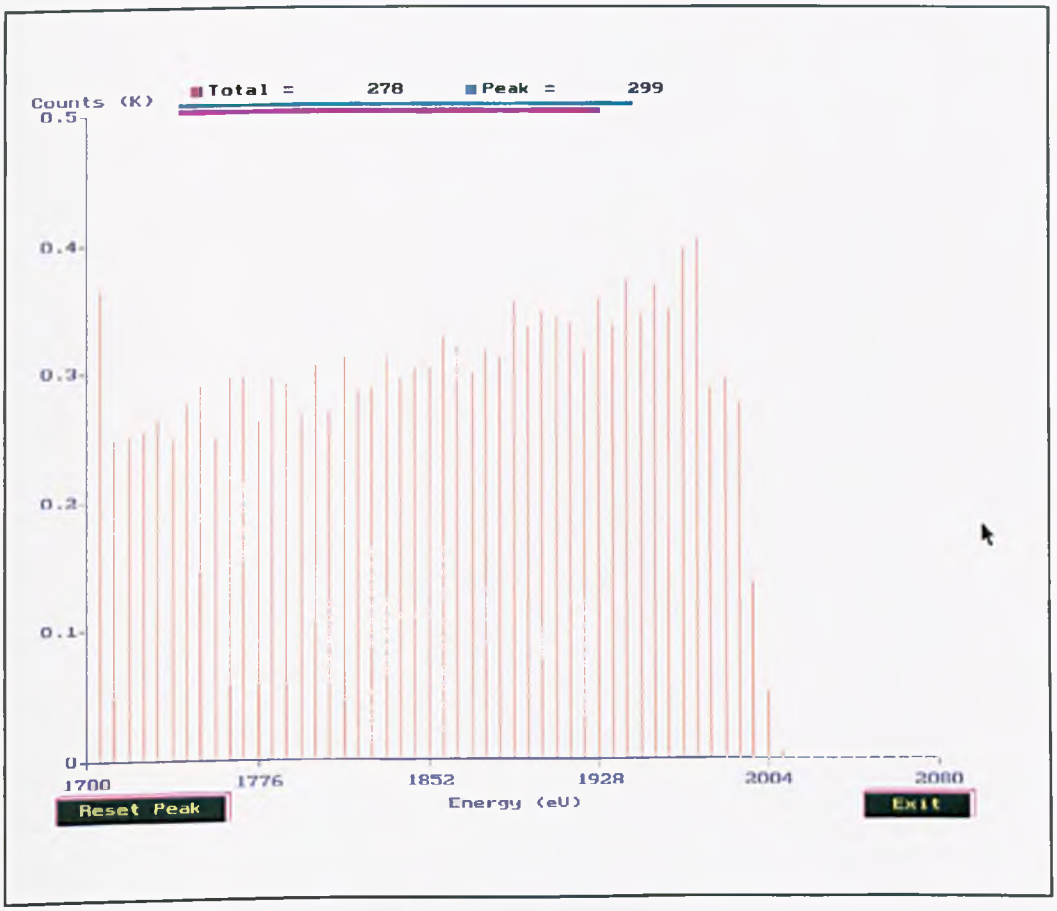

Figure 7.22 : Screen dump of the tuning process

7.2.2.5 Electron Beam Imaging and Linescans

The control software package provides the analyst with a fully integrated environment in which secondary electron microscopy (SEM), Auger linescans and scanning Auger microscopy (SAM) can be performed. In the present software, this is carried out under the 'SEM/SAM Imaging' option. With reference to Figure 7.14, the analyst is provided with an SEM image of the sample on the computer screen which they can then use to select either an area over which to take a SAM image, a line along which to perform an Auger linescan, or a single point at which to acquire spectra.

7.2.2.5.1 The Computer Controlled SEM

In order for Auger linescans and images to be carried out under computer control it is useful to have the facility to relate sample features to the electron beam scan voltages supplied by the computer's interfacing hardware. This has been achieved by the implementation of a fully computer controlled SEM, based on the secondary electron detector head described in Chapter six.

The computer controlled SEM relies on the 12-bit Analogue-to-Digital Converter (ADC) on the Burr-Brown interfacing board to process the signal from the secondary electron detector's head amplifier, giving a maximum of 4096 different levels over the selected voltage range (currently –10 - +10V). The DC offset (black level) and gain of the secondary electron detector must be set to optimise the signal so that it occupies as much as possible of this voltage range in order to give the maximum contrast in the SEM image. Because the ADC is only used in this part of the software, the ADC initialisation software is embedded into the SEM imaging code and not implemented in a separate function. The function ***'get_adc_value'*** returns the current voltage on the ADC once initialisation has been carried out.

Figure 7.23 shows the process involved in setting up the ADC, and 7.24 outlines the steps required to read an instantaneous voltage. Note that since the ADC is 12 bit, only the four least significant bits of the high byte are used.

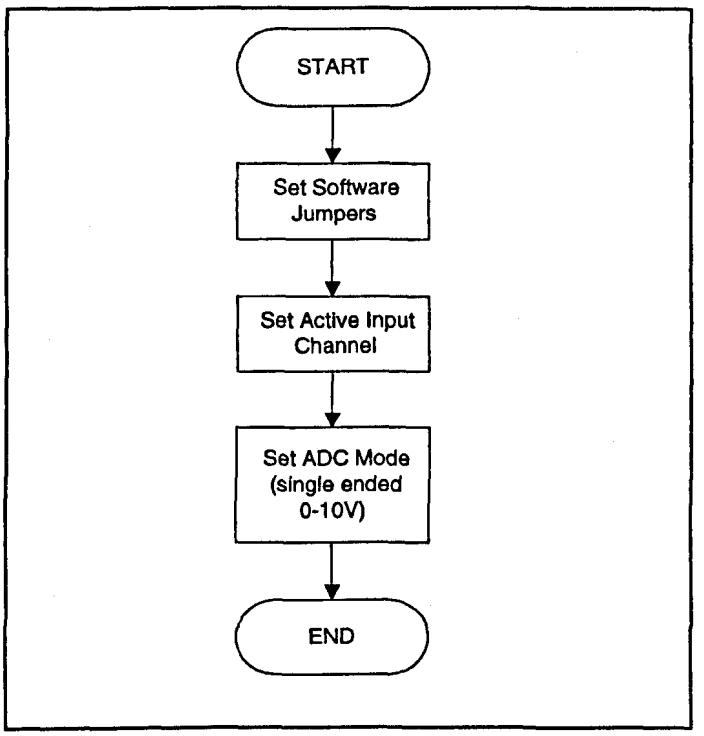

Figure 7.23 : Initialisation of the Analogue-to-Digital converter

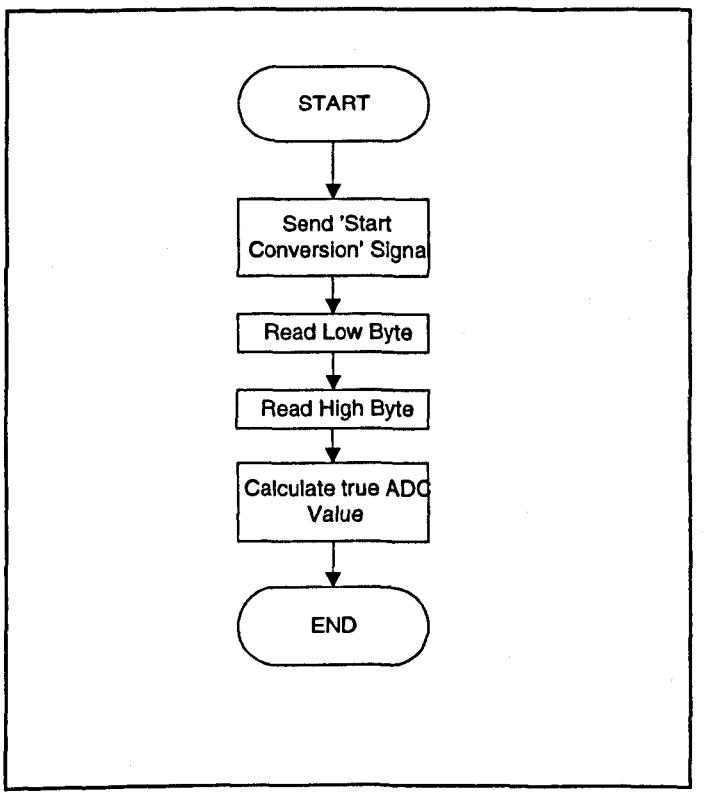

Figure 7.24 : Reading from the Analogue-to-Digital converter

An overview of the complete SEM display routine is given in Figure 7.25. Not surprisingly, the time taken to display an image depends on its size and resolution. For this reason, provision is made to display smaller low resolution images for rough sample positioning, or larger high resolution images for accurate sample positioning, and defining areas for Auger studies.

The Borland C++ compiler used in this study supports a maximum palette of 16 colours at any one time, although these 16 can be selected from a range of 256. The software allows the user to select from three sets of colours as follows:

- **Thermal :** This gives a range of red, orange and yellow colours which result in a very bright image with good contrast.
- **Greyscale :** Very close to the results one might expect from the monochrome CRT display often found on conventional analogue SEMs.
- **Colour :** A multi-coloured palette containing blues, reds, greens and yellows; because adjacent colours in this palette can be quite different it may reveal features more clearly in certain circumstances where high contrast is desirable.

Once the image settings have been chosen, the SEM image is displayed as a conventional raster scan, with X and Y values being written to the DACs to control electron beam position, and the ADC voltage readings being displayed as pixels on a corresponding scan over the screen. Figures 7.26, 7.27 and 7.28 illustrate the computer controlled SEM software in operation for thermal, greyscale and colour palettes respectively. Once the sample is positioned as required, the user can select one of the following options:

- **Scanning Auger image**, in which case an area is chosen by 'dragging' a box with the mouse and the 'sam' function is launched.
- **Fixed Point Auger**, this simply involves selecting a single point with the mouse, global variables are then set which define the electron position during 'grab scan' and 'tune' operations.
- **Auger Linescan**, the user specifies the start and end of the line, the 'linescan' function is then launched to collect spectra along the selected line.

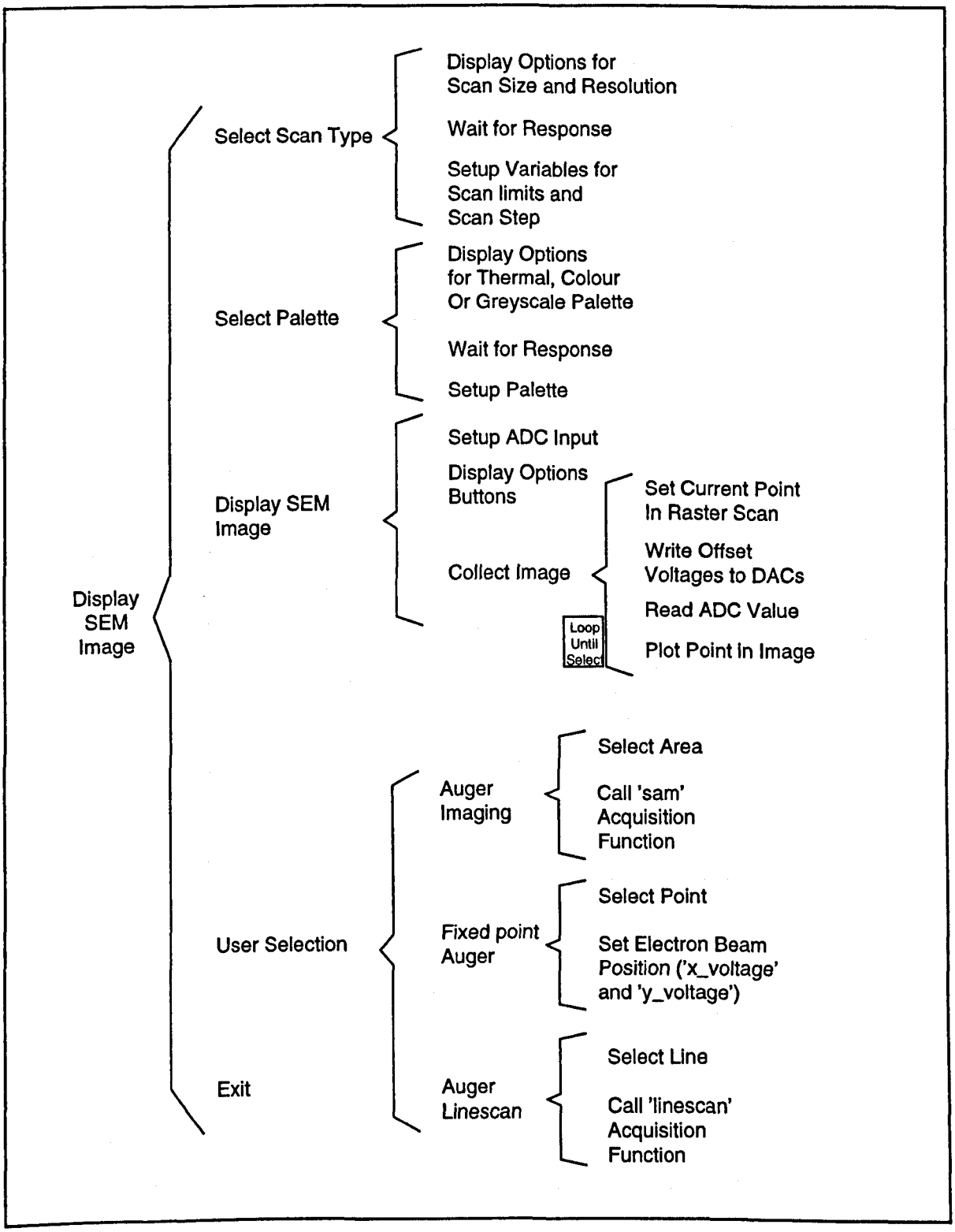

Figure 7.25 : Warnier-Orr diagram illustrating the SEM imaging software

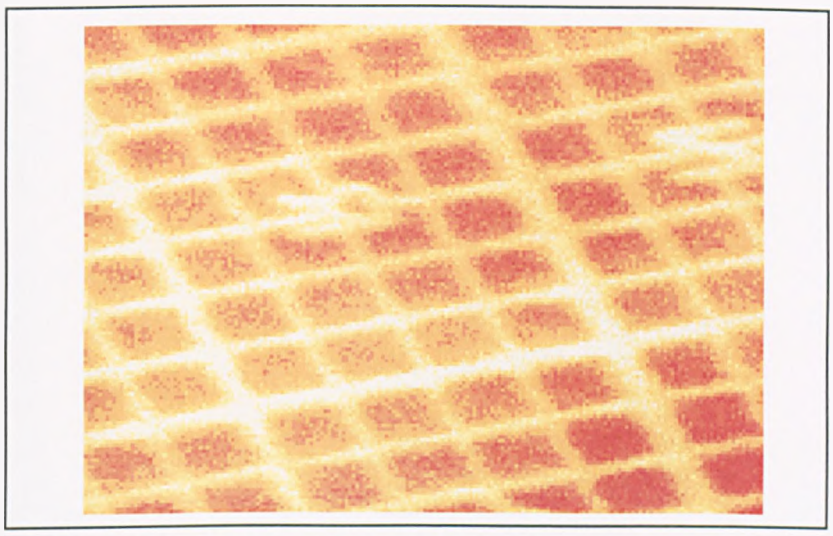

Figure 7.26 : Example of an SEM image of a 30 micron Gold grid acquired with the thermal palette

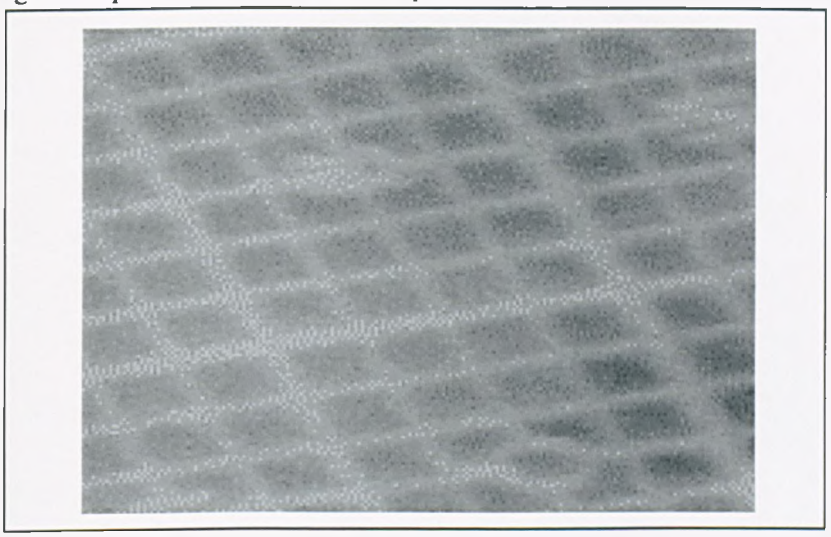

Figure 7.27 : Example of an SEM image of a 30 micron Gold grid acquired with the greyscale palette

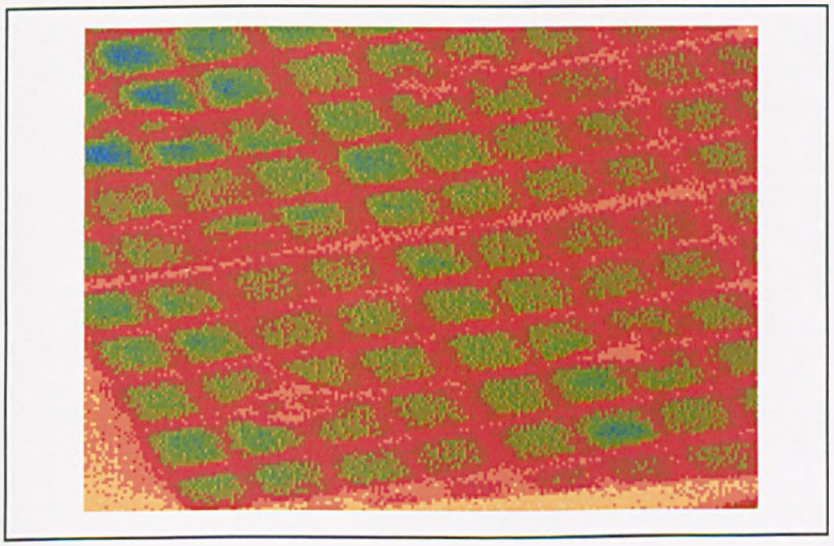

Figure 7.28 : Example of an SEM image of a 30 micron Gold grid acquired with the colour palette

The screen dump shown in Figure 7.29 illustrates the process of defining a line on the SEM image along which to perform a linescan. The analyst selects the start and end points for the line, and is given the option to either accept them, or change them if a mistake has been made. Note that the full range of colours in the current palette is always displayed on the right-hand side of the screen, although this was not shown on previous figures.

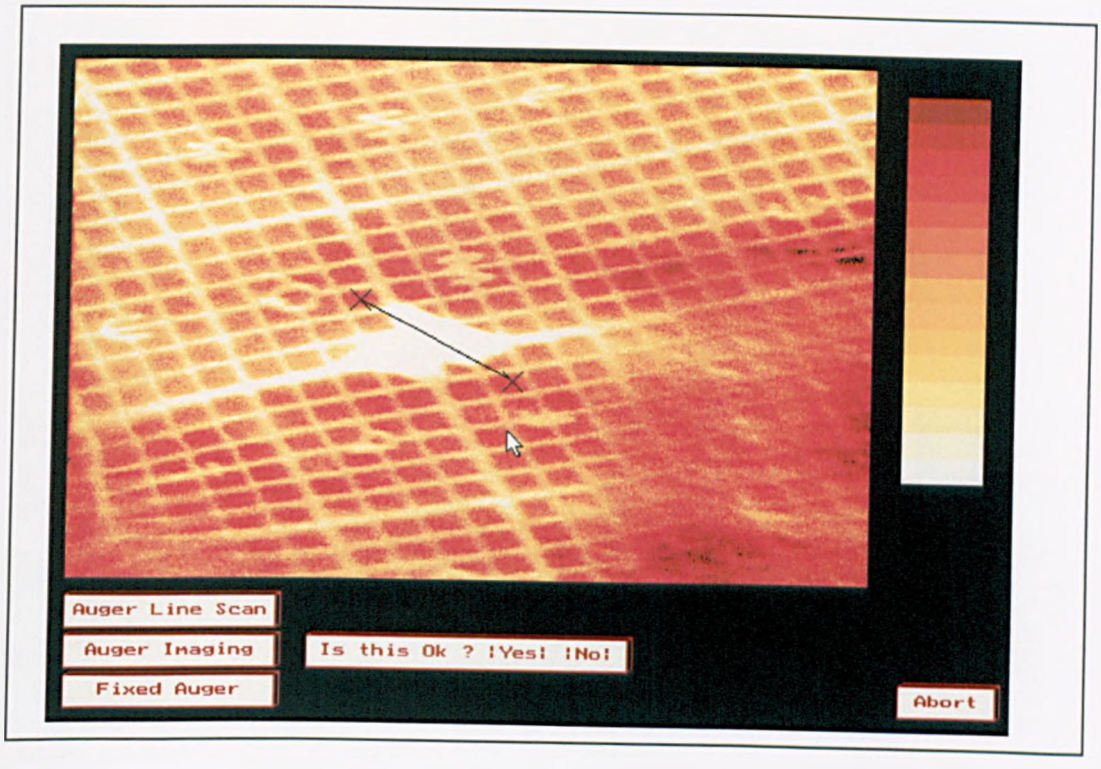

Figure 7.29 : Defining the linescan with the mouse-driven user interface

The mouse can also be used for defining the area for a Auger image to be collected, this process is shown in Figure 7.30, again with the thermal palette. When the user clicks the mouse at any point on the image, this defines the top left corner of the box, and whilst the right mouse button is held the box can be dragged out to any size. It is important to remember that the CMA has a field of view of only a few tens of microns and it is therefore essential to take into account the current magnification when choosing an area for SAM imaging.

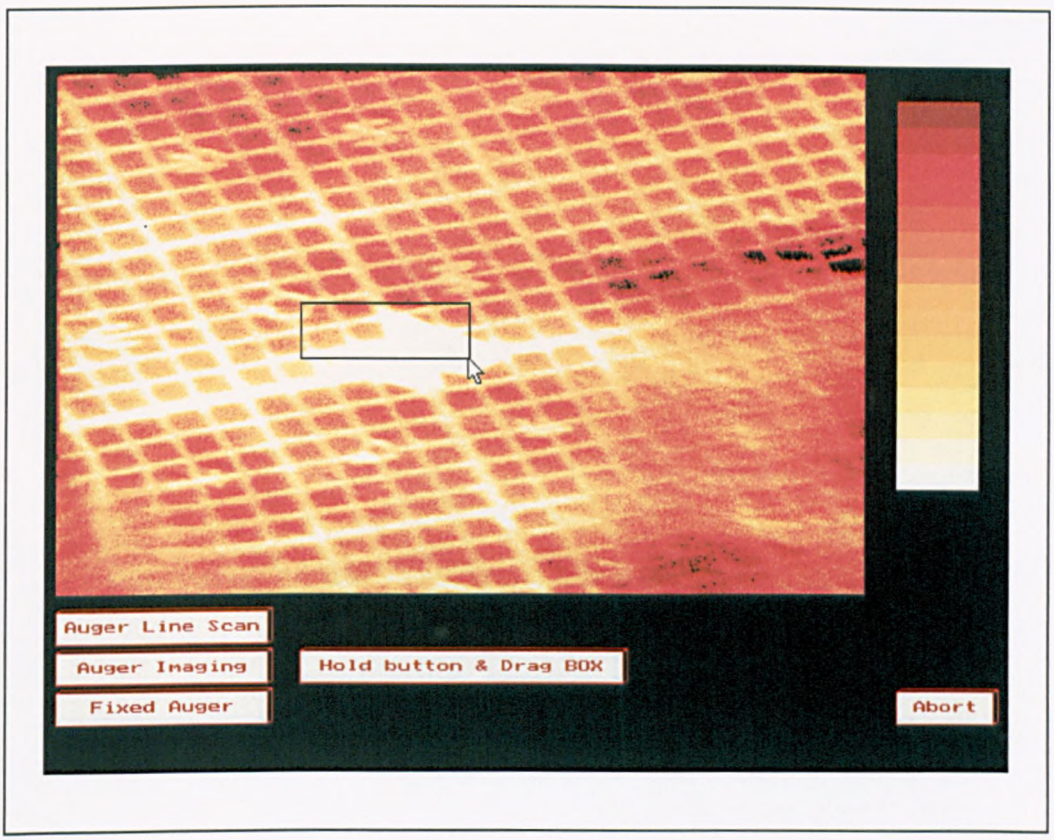

Figure 7.30 : Dragging out a box for collection of an Auger image

In practice, a full size high resolution image such as the one shown in figure 7.30 above takes about 10 seconds per frame with the present hardware. This could possibly be improved by using a faster processor, but is certainly too slow for coarse sample alignment. As mentioned earlier, one of the available options is a larger, lower resolution SEM image for use when only a rough idea of the current view is required. Figure 7.31 is an example of such an image, shown with a colour palette.

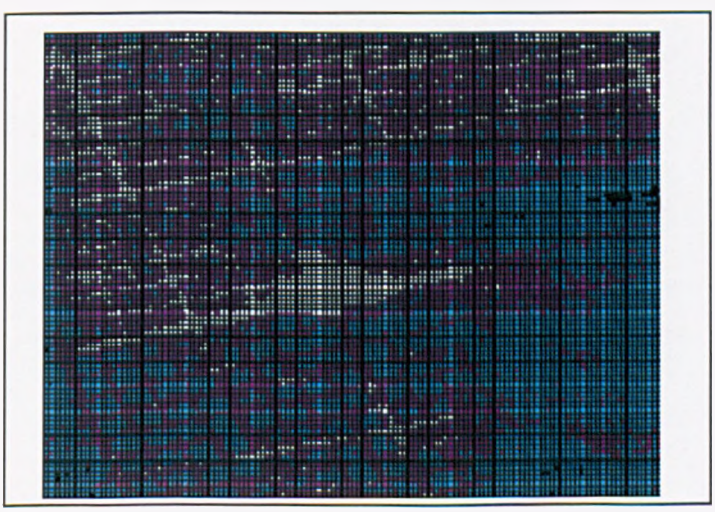

Figure 7.31 : A low resolution image for coarse sample positioning

7.2.2.5.2 Auger Imaging

Collection of an Auger image is carried out by the function '***sam***', which allows the user to setup the parameters and gather Auger images from all six detector channels simultaneously, over the area selected from the SEM display. Auger images are collected by raster scanning over the sample and collecting electron counts at three points for each pixel of the scan. Multiple sets of three points can be defined if more than one Auger peak is to be mapped at any one time.

The height of the Auger peak of interest can be estimated by selecting one point as the expected position of the peak, two at an energy slightly higher than the peak. The values taken to one side of the peak can be used to provide an extrapolated estimate of the background count under the peak, which together with the peak height can be processed in a variety of formulae, the results of which are used to modulate the intensity (or colour) of pixels on the screen and form an image.

An SEM image is collected from the secondary electron detector at the same time as the Auger image, and over the same area. The data for both types of image is saved to ASCII text files with the following formats:

- ***FILENAME*.INF** : The header file containing information about the size of the image, the number of elements in the Auger images and the exact energies at which measurements have been made.
- ***FILENAME*.SEM** : A text file of numbers corresponding to the SEM image, delimited by end-of-line (EOL) characters.
- ***FILENAME*1.SAM, *FILENAME*2.SAM .... *FILENAME*6.SAM** : Tab delimited files containing the count values at the three energies selected for each element comprising the SAM images.

The Warnier-Orr diagram shown in Figure 7.32 illustrates the SAM imaging function, and shows the construction of the ASCII data files. The mouse-driven parameter setup routines are functionally similar to those already covered in terms of the programming methods used.

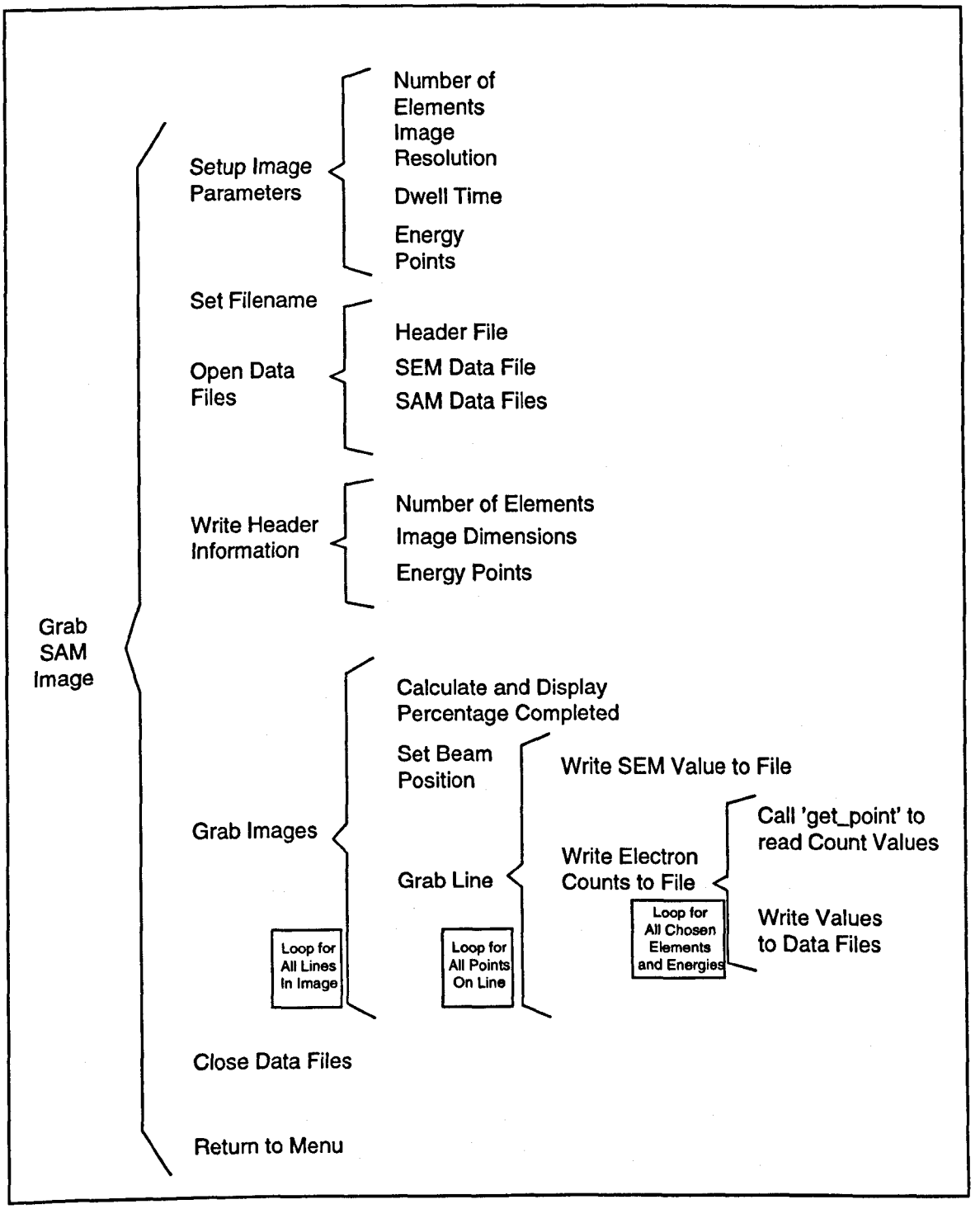

Figure 7.32 : The Auger imaging function

7.2.2.5.3 Auger Linescans

The process of collecting an Auger linescan is straightforward to understand. We simply incrementally move the electron beam along a straight line path, and collect spectra at a user-defined number of points along the line. Because this process has so much in common with the normal 'grab scan' function, the same functions are used to collect data from the analyst and to grab the spectra themselves.

Linescans are saved in the same format as normal single-point scans but with an extra number added to their filename to indicate which point in the line the spectrum set corresponds to. For example, *FILENAME*2.4 corresponds to the second point in the line and the spectrum from detector channel four. Linescan spectra can then be viewed either by loading them individually as normal spectrum sets, or with the linescan display routine which will be covered later in this chapter.

7.2.2.5.4 Fixed Point Auger

The user has the option of selecting a point on an SEM image at which to carry out subsequent spectrum collection, either through the 'grab scan' option, or with 'tune'. This allows the analyst to single out interesting features without the need to manually set the physical position of the sample to line up with the electron beam, or to adjust the beam offset to bring the required section of the sample into view on very high magnification.

To achieve this, the selected pixel on the screen must be converted to an offset voltage. The relationship between screen pixel and offset voltage is given by:

$$V_{OFFSET} = 2.5 + (m(p-50) / 5) \qquad (7.1)$$

where m is the number of pixels in the image in the chosen direction, and p is the chosen position in pixels.

7.2.2.6 Loading and Saving Files

The control software package allows spectrum sets, linescans and auger images to be saved to, and loaded from the hard drive so that a permanent record of any data can be kept. Since all data is written directly to temporary files on the disk during collection, 'loading' a file involves copying its contents from a permanent to a temporary file and vice-versa for 'saving'.

In the case of SAM images, the user is prompted for a filename before they are collected, and the images cannot be viewed without explicitly loading these data files into the 'display' routine. Single-point spectrum sets and linescans however can be loaded into and saved from the 'workspace' dynamically, then viewed or overwritten as required.

The loading operation is carried out by the functions '*load_single*' and '*load_linescan*'. When spectra are loaded, the files are checked and an error message displayed if they cannot be found. Loading a linescan set is slightly complicated by the fact that the number of spectrum sets is arbitrary, for this reason the '*load_linescan*' function attempts to load up to 99 scan sets, and stops when the next set in sequence cannot be found. The functions '*save_single*' and '*save_linescan*' allow the user to make a permanent record of their data ; prompting the user for a filename, then writing the appropriate files to disk, again it is the task of the software to ascertain how many points the linescan was collected at, since no record of this is stored in memory.

Figure 7.33 shows a Warnier-Orr representation of the '*save_linescan*' function, and 7.34 illustrates the '*load_linescan*' function. The single point spectrum routines are relatively simple and are not shown in diagrammatic form; the reader is referred to the source code listing for further information if required.

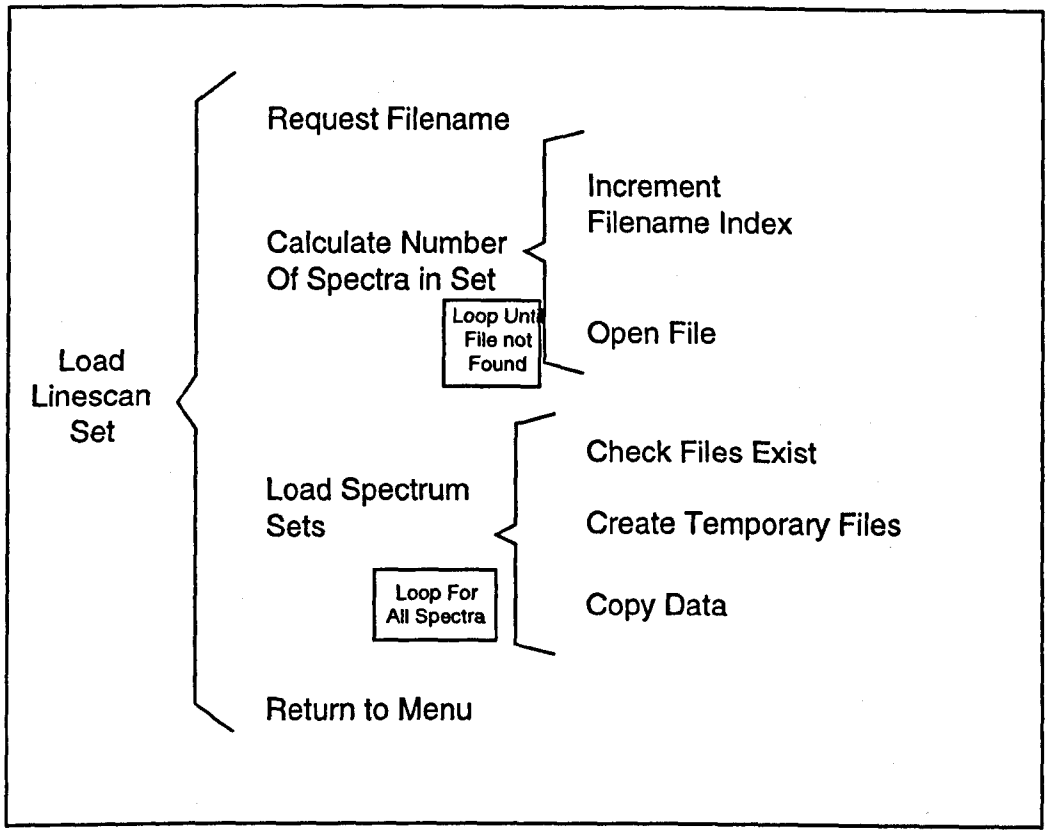

Figure 7.33 : Warnier-Orr Diagram of the linescan load function

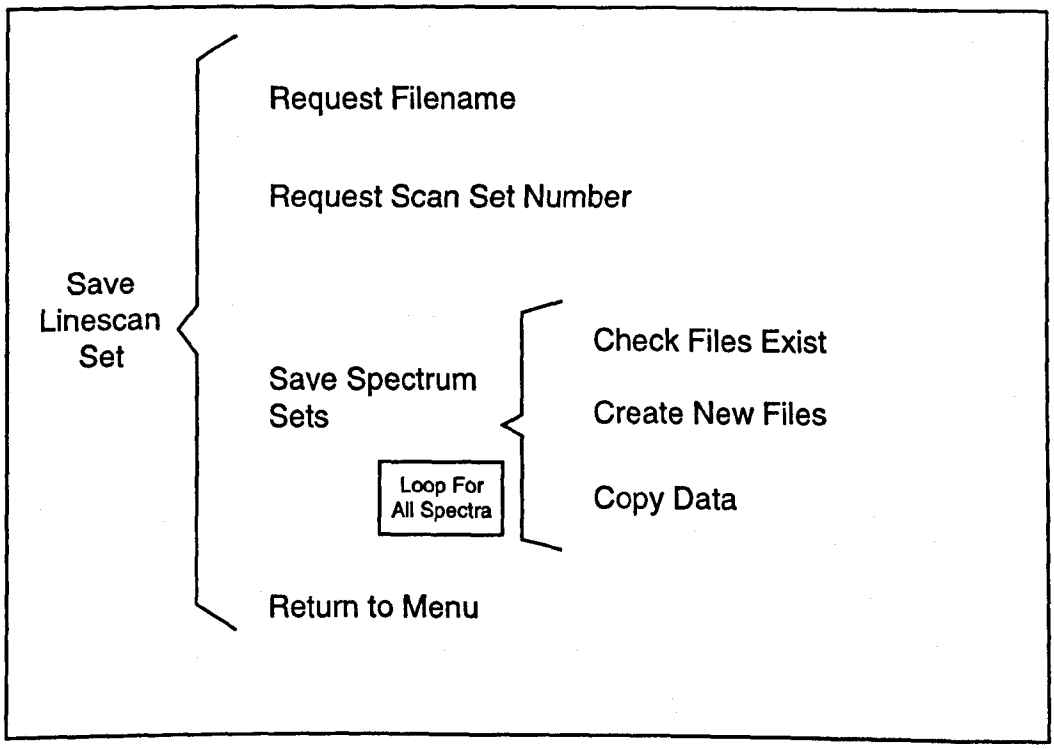

Figure 7.34 : Warnier-Orr Diagram of the linescan save function

7.2.2.7 Displaying Spectra and Images

The 'Display a Scan' menu option, presents the user with a further choice between three functions which have been written to cater for single point spectra, linescans as well as scanning Auger images.

7.2.2.7.1 Displaying Single Point Spectra

The display routine to show single point spectra is very similar to the real time display routine which runs in parallel to the spectrum collection process. Unlike the real time display, the dedicated display routine auto-scales the axes to fit the spectra and allows the results to be plotted as either points or lines.

The '***single_display***' function is first called which in turn invokes '***plot_scan***'. The processes involved in these functions are described by the Warnier-Orr diagram in Figure 7.35. Auto-scaling of the x-axis (energy) is simply a matter of reading the first and last values in the left hand column of any of the six spectrum files (all six will have the same energy scale). Setting the limit for the y-axis (counts) is slightly more complicated since it involves examining every count value in all six of the data files to determine the peak. Over estimating the maximum count due to spurious values (which may be attributed to electrical noise or mechanical interference) is avoided by only taking into account values which are less then three times as large as their neighbour.

Switching between line and point display can be carried out by simply 'clicking' a button in the corner of the screen (at which point the '*plot_scan*' function is invoked again with an appropriate flag set). In practice, although there is a certain amount of processing involved in calculating the axis limits and displaying the spectra, the display routine is close to instantaneous from the point of view of the user. Figures 7.36 and 7.37 show the results of the single-point spectrum display routine with lines and points respectively. The results shown are for a 2kV beam on the same gold grid used in the SEM imaging in previous sections of this chapter.

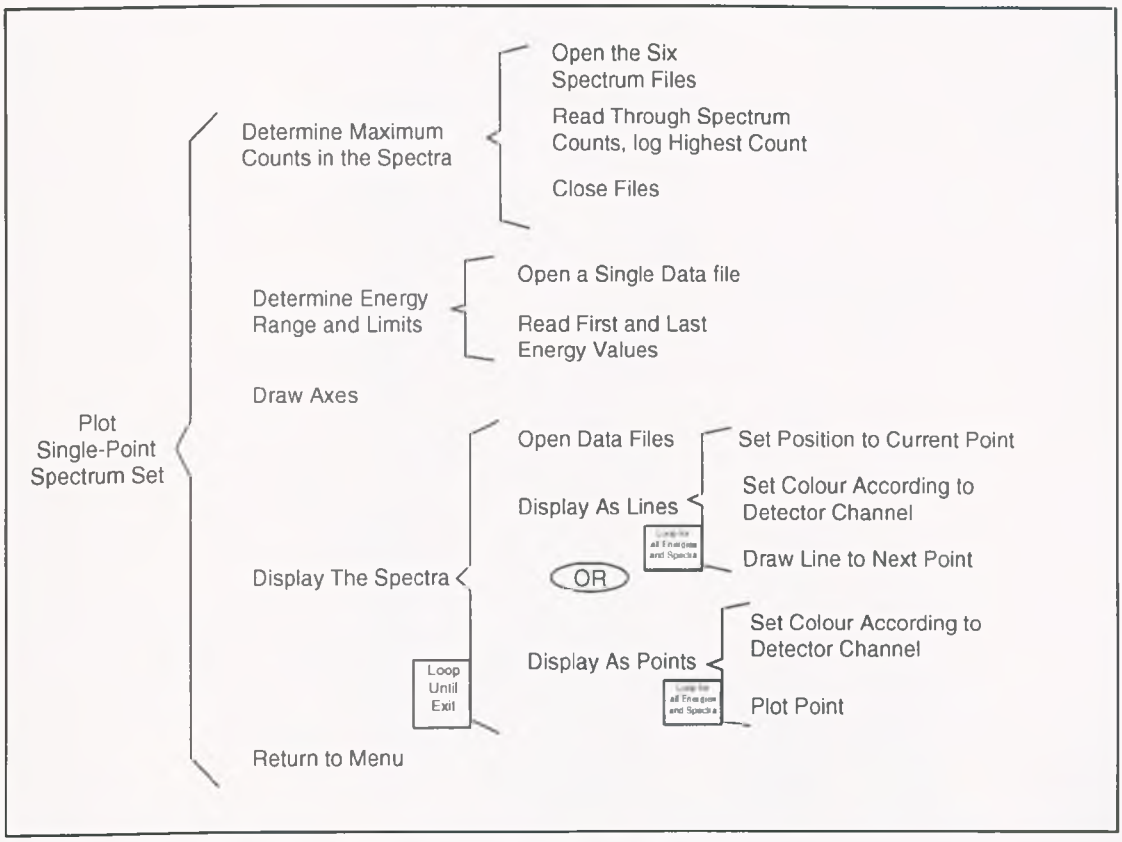

Figure 7.35 : The single-point spectrum display function

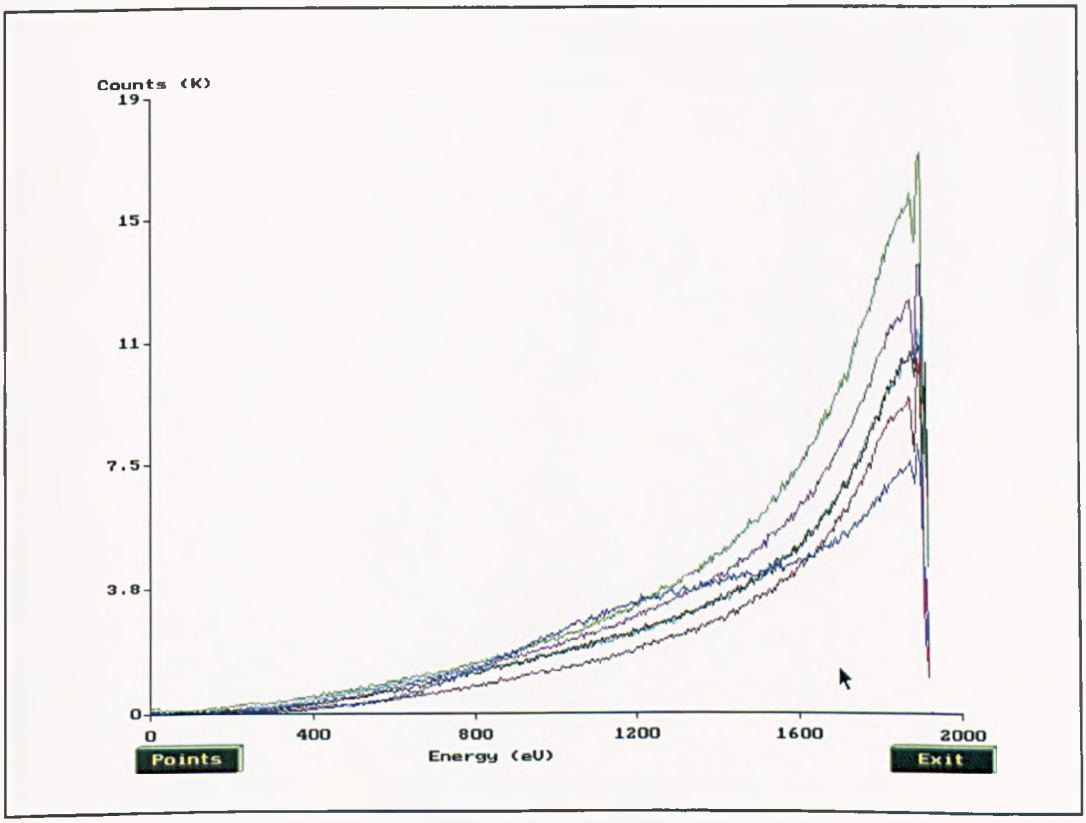

Figure 7.36 : Screen dump of the single-point spectrum display function using lines

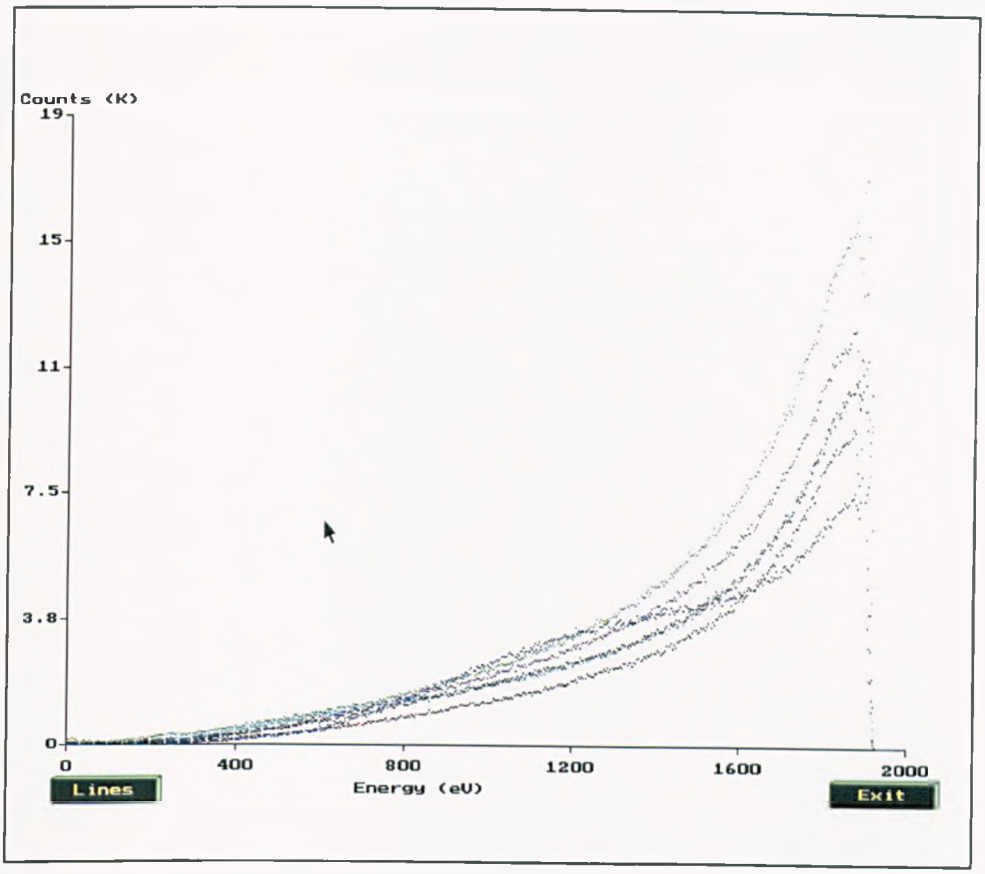

Figure 7.37 : Screen dump of the single-point spectrum display function using points

7.2.2.7.2 Displaying Linescans

Since a linescan is essentially just a set of spectra taken sequentially, they can be displayed in much the same way as standard spectra. In the current version of the software, the spectra from each point in the line are displayed simultaneously on a single set of axes and spread out by placing an incremental offset to the count values. In this scheme, the y-axis is no longer an absolute measure of counts, but instead only relates to the difference in counts between successive points in a given spectrum.

The Warnier-Orr representation of this function is illustrated in Figure 7.38. To avoid cluttering the display, the linescan display only shows spectra from one of the six detector channels at any time; the user selects which channel to show before the spectra are displayed.

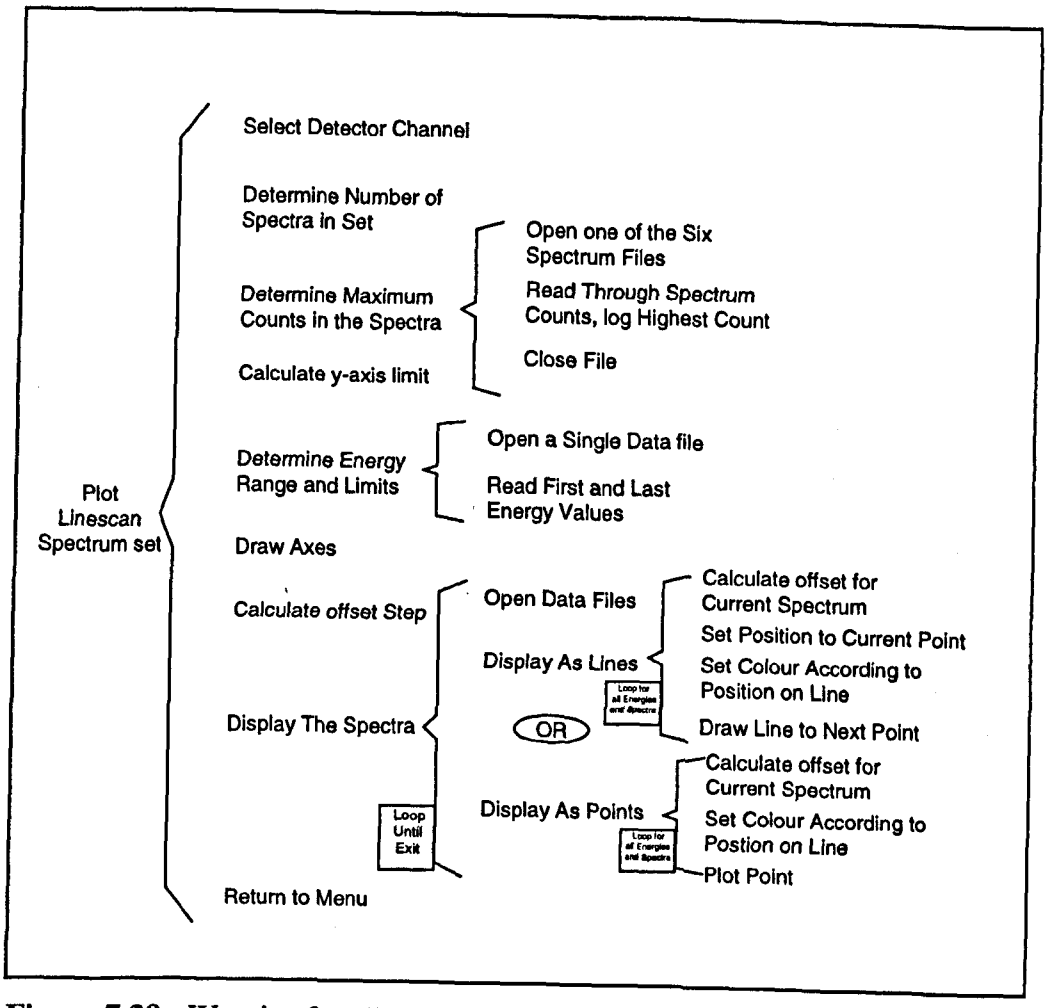

Figure 7.38 : Warnier-Orr diagram of the linescan display function

Because of time limitations, the current linescan display routine uses a rather crude method of selecting a suitable y-axis limit, and incremental offset step. Only one of the spectra in the set is examined for a peak count value, this is then doubled to give a y-axis limit with some margin. The incremental step applied to each successive spectrum in the set is calculated by the following equation:

$$step = 320 / 2n \qquad (7.2)$$

where n is the number of spectra in the set. It is strongly suggested that this simplistic technique should be replaced by more rigorous calculation to ensure that all spectra are visible and optimally placed; however in practice the current method gives some satisfactory preliminary results.

Figure 7.39 is a screen dump of the linescan display routine. The spectra displayed are all from channel five of the analyser and show the elastic peak of a 2kV electron beam as it scans over the Gold grid. Examination of the spectra reveals some with fairly well defined peaks, probably those related to the gold area of the grid, and some with less well defined features, where the electron beam struck the area between the grid lines.

The colours of the lines are cycled through the current palette as the spectra are displayed. Note that the image in Figure 7.39 is inverted and has been processed for maximum clarity in print ; the colours do not necessarily relate directly to those on the screen.

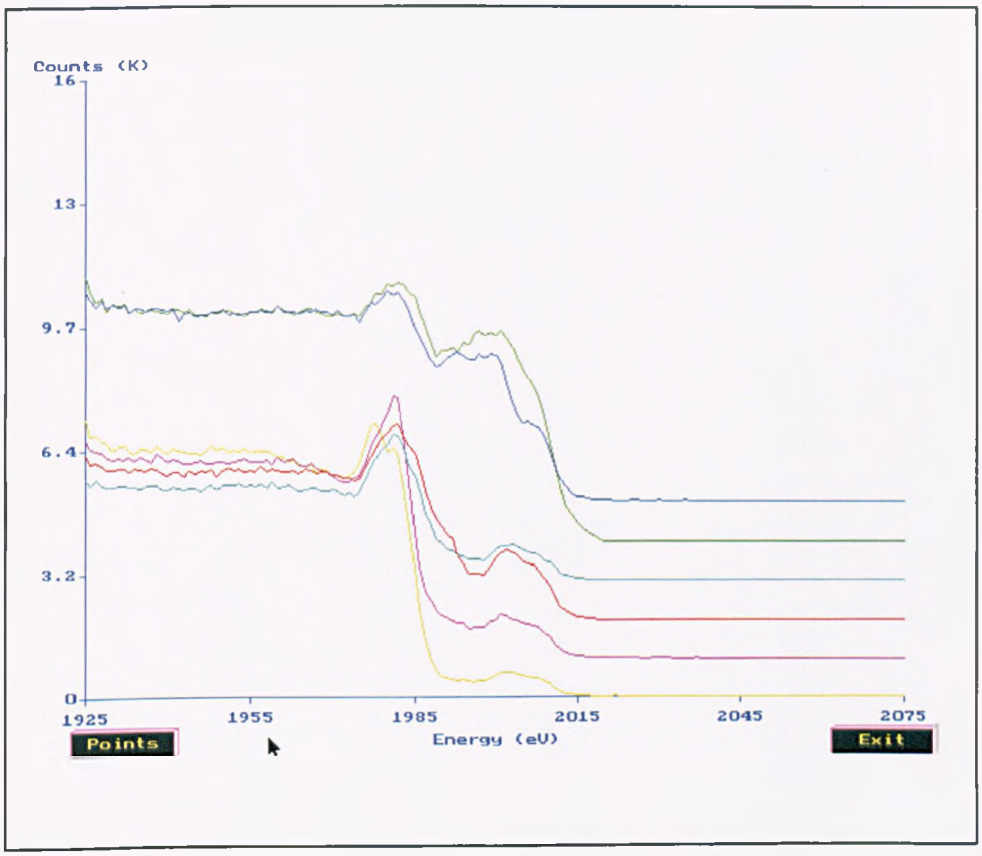

Figure 7.39 : Screen dump of the linescan display function

7.2.2.7.3 Displaying SAM Images

Unlike linescans and single-point spectra, the present software package does not allow Auger images to be displayed in real time as they are collected. This is partly due to the problems associated with setting the contrast of the image according to maximum and minimum expected counts. Undoubtedly, this problem could have been overcome at least partially by, collecting counts at representative points around the scan area, then setting the contrast according to these results. Had time allowed, the possibility of dynamically setting the contrast of the Auger images would have been investigated.

The Warnier-Orr diagram in Figure 7.40 represents the SAM image display function. The current version displays four different images from each data set; the peak count, the background count and the following two ratios:

$$R_1 = \frac{P-B}{P+B} \qquad (7.3)$$

$$R_2 = \frac{P-B'}{P+B'} \qquad (7.4)$$

Where P is the count at the energy defined as the peak, B is the count at an energy a few eV above the peak, and B' is the background extrapolated to energy of under the peak, given by

$$B' = c_3 + \frac{(e_3 - e_1)(c_2 - c_3)}{(e_3 - e_2)} \qquad (7.5)$$

where c_1, c_2 and c_3 are the three count values taken for each pixel in the image and e_1, e_2,e_3 are their corresponding energies. Automatic setting of the contrast is achieved by determining the range of values in each data set, then setting pixel colours according to the following formula:

$$C = R_1(S_P/r) \qquad (7.6)$$

where C is the colour index, S_P is the palette size, r the range of count values in the current image and R_1 the count value at any given point.

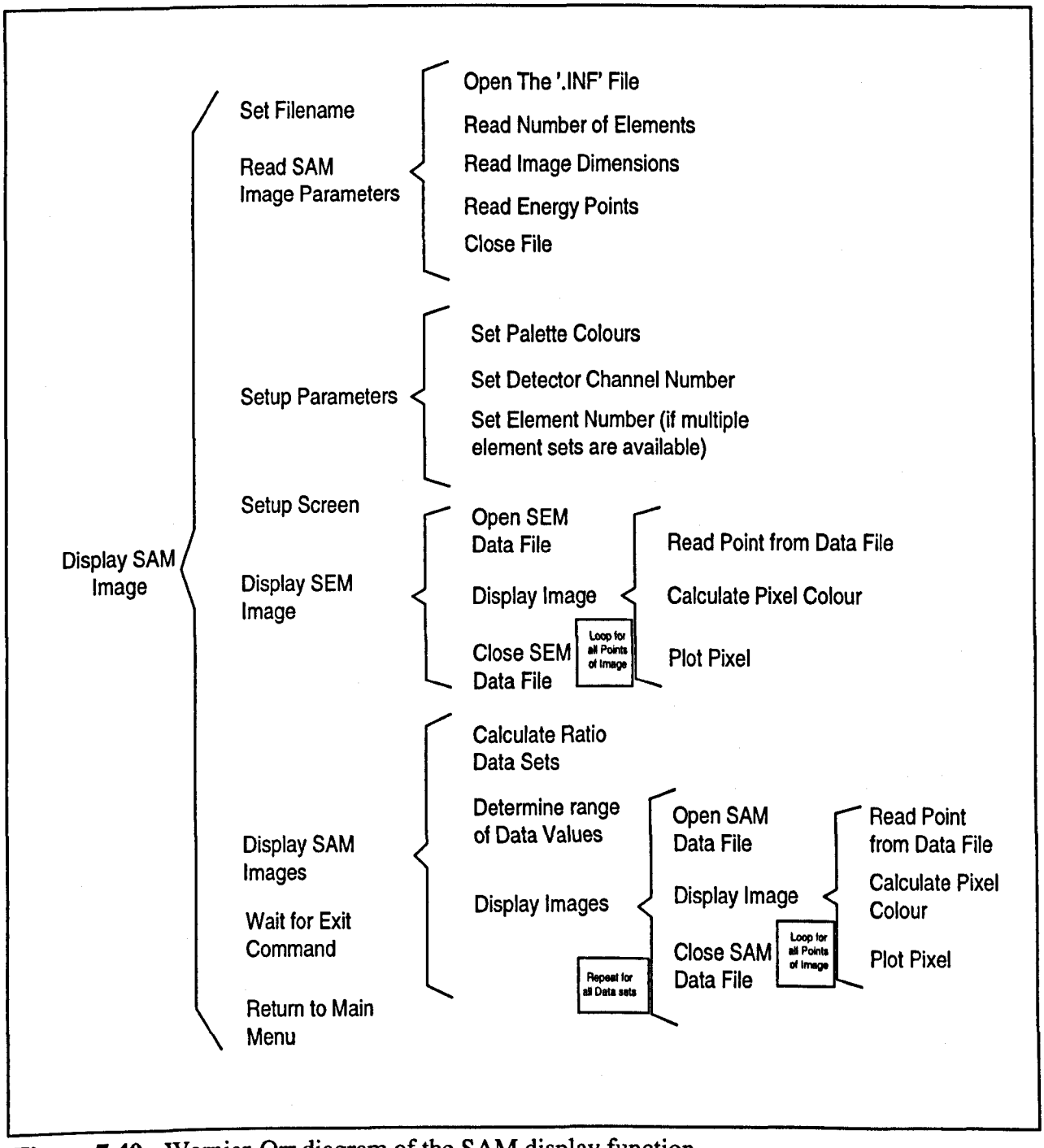

Figure 7.40 : Warnier-Orr diagram of the SAM display function

Two sets of preliminary data from the SAM display function have been chosen, the sample used in both experiments is a 30µm Gold grid, and is examined with a 2kV electron beam. The first set shows a set of 59x47 pixel images, with the peak energy set to 240eV and the two background energies at 290 and 300eV. As Figure 7.41 shows, the SEM image is quite clear, although slightly distorted and the first two Auger images (peak and background) display some recognisable features of the grid. The two ratio functions fail to yield a well defined image, and clearly more work is required to resolve this problem.

Figure 7.42 is an example of a somewhat higher resolution image of 176x120 pixels. Again the SEM image is quite clear, but this time distortion is visible on all four Auger images. This is thought to be a result of either poor sample positioning, or a convolution of the wanted image with the spatial response function of the CMA. Again, further work is required to resolve this anomaly. Both these image sets are displayed with the thermal palette, although the SAM display function also supports greyscale and colour imaging.

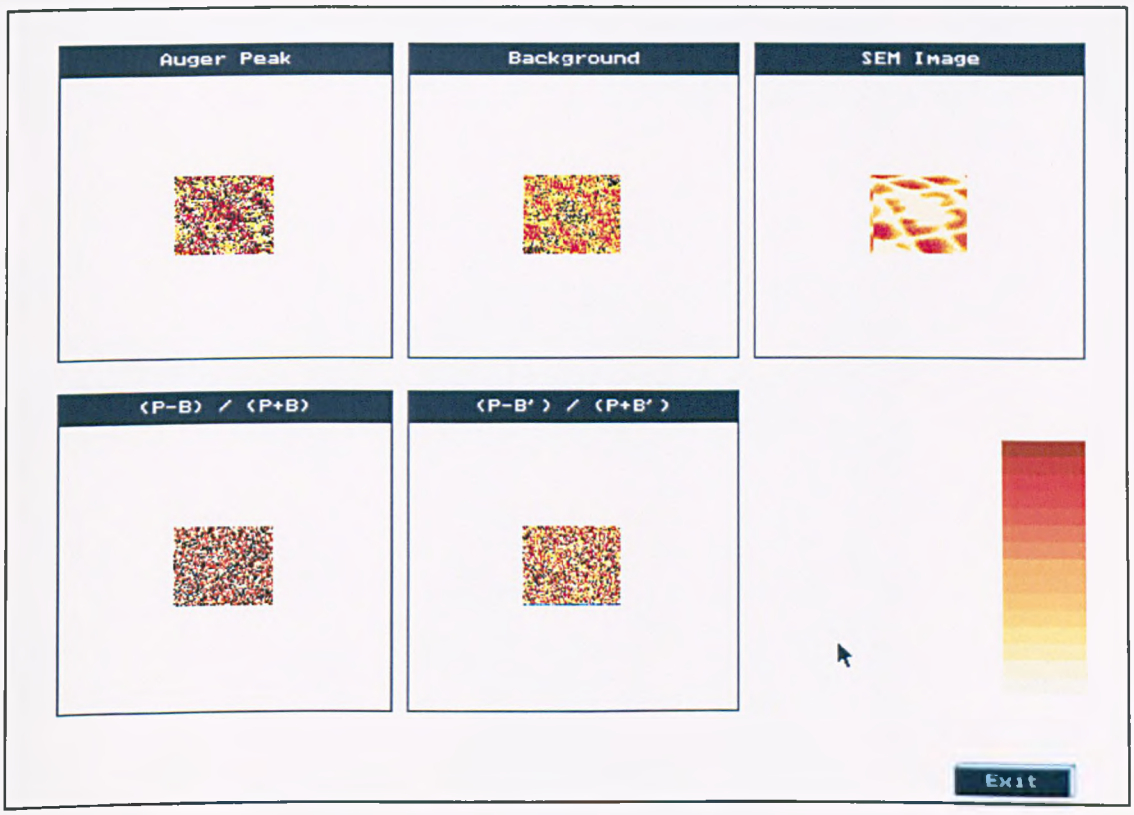

Figure 7.41 : Low resolution SAM images of the gold grid

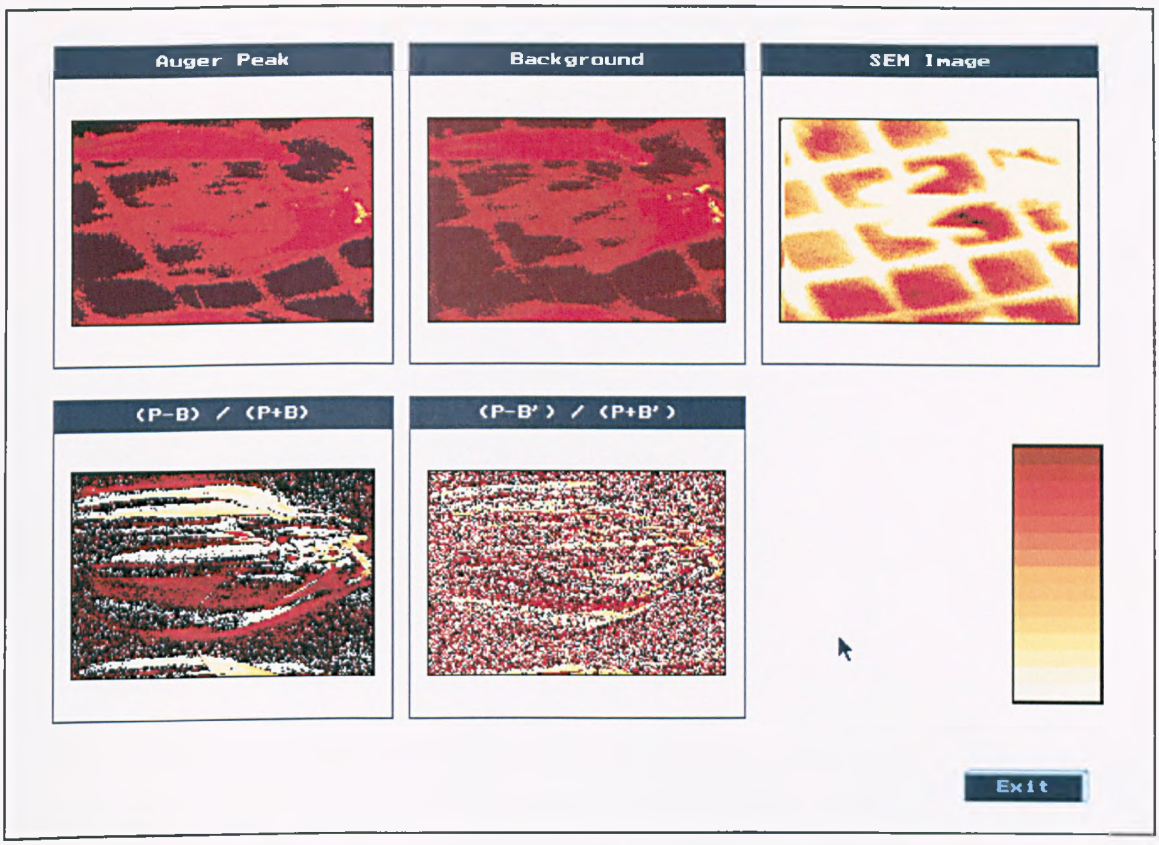

Figure 7.42 : High resolution SAM images of the gold grid, showing significant distortion

7.3 Discussion of the CMA Control System

The basis of a fully integrated acquisition and display system for the angle resolved CMA has been described. The interfacing hardware is based on the Burr-Brown PCI instrumentation system, which provides for almost unlimited future expansion due to the modular nature of its design. A set of low-level interfacing functions have been developed specifically for the purpose of communication with this hardware in order to make future software development as straightforward as possible.

A fully mouse driven user interface is described, which provides a robust and user-friendly environment for hardware control and image processing. The software has been designed and documented in a modular structure to allow the future addition of extra software components, or code re-use as required.

Spectra can be simultaneously acquired from all six channels of the detector, and a function has been provided for displaying these spectrum sets on automatically scaled axes. Because the spectra are saved in a standard format as ASCII text files, the results can easily be ported into other graph drawing packages (such as Gnuplot), or spreadsheets (including Microsoft Excel and Lotus123) for post processing.

Preliminary results have been obtained in order to determine the initial performance of the hardware and software, and on the whole these are encouraging. It is clear, however, that further work is required to improve some aspects of the control system, particularly the linescan and SAM image collection and display functions.

CHAPTER EIGHT

RESULTS AND FURTHER WORK

8.0 Introduction

Throughout the preceding chapters of this thesis, preliminary results have been presented which relate to the various developments undertaken, including the computer controlled SEM, the graphical user interface and the scanning Auger microscope. In this chapter, we examine a more comprehensive set of results from the six-channel angle resolved CMA. Time constraints meant that only a limited range of collected spectra were available for analysis. A number of spectra are examined, however, with a view to understanding some of the characteristics of the new multichannel detector system and defining future research objectives.

8.1 Measured Spectra from the angle resolved CMA

Although the ability to resolve the azimuthal angle of the detected electrons is the novel feature of the new detector head, we will firstly examine spectra obtained as the sum of signals from all the detector channels. Whilst this will, of course, yield no angular information, it will enable some of the general characteristics of the detector to be established.

8.1.1 Single Channel and Summed Spectra

Firstly, we examine the elastic peak of a 1keV beam, onto an un-prepared sample. In this experiment, the sample is a thin layer of silver on an aluminium sheet (although as we are only looking at the elastic peak, the precise constitution of the sample is of secondary importance). Figure 8.1 shows the signal from a single channel of the detector over an energy range of 200eV around the elastic peak. The measured sample current is 10nA and the spectrum is collected with a 1000ms dwell time.

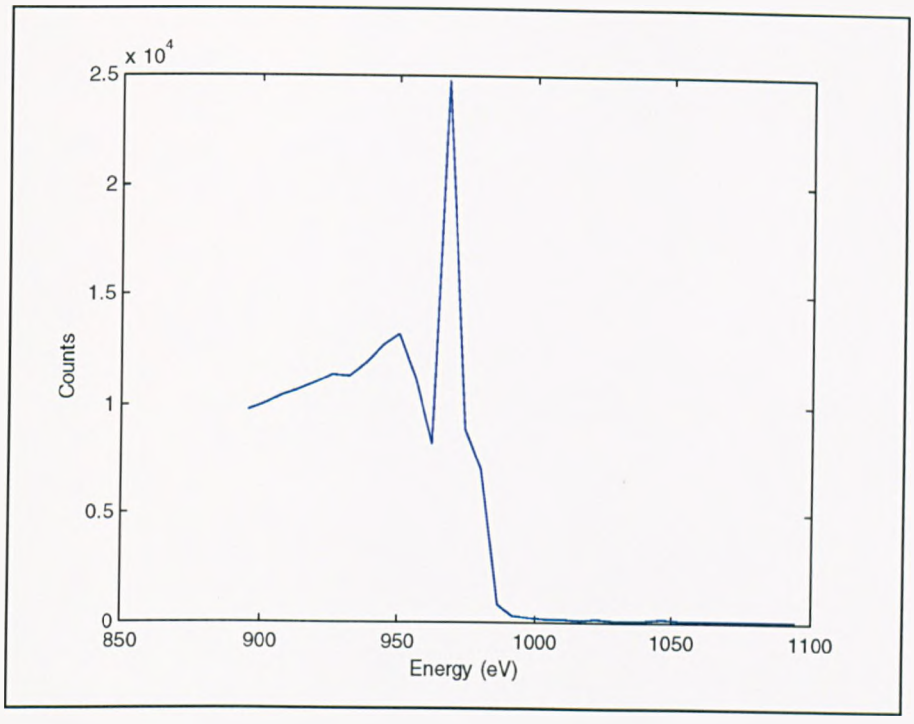

Figure 8.1 : The elastic peak for 1keV electrons gathered from an unprepared sample with a 10nA sample current

Examination of Figure 8.1 yields a full width at half maximum (FWHM) peak width of approximately 8eV. The actual position at which the elastic peak appears on the spectrum is approximately 970eV, some 3% adrift from the true energy. This suggests that further calibration of the spectrometer energy scale may be required (this should be carried out by observation of Auger peak energies from well characterised samples such as Cu). It is also interesting to note that at energies higher than the beam energy, the count from the channel plate detector is zero. This is an important indication that there is no spurious background count from the detector or head amplifier, which may have arisen either from noise sensitive amplification/discrimination or channel plate artefacts.

Figure 8.2 shows an entire measured spectrum, optimised by adjustment of the sample position for maximum amplitude of the elastic peak. This time a 2.2keV incident electron beam is used with a 30nA beam current (measured as a sample current), again the dwell time is 1000ms and 500 points are taken (giving 4.4eV per division). Although the elastic peak in Figure 8.2 is well defined, the low energy secondary electron cascade is absent. There are several possible reasons for this, three of which will be investigated.

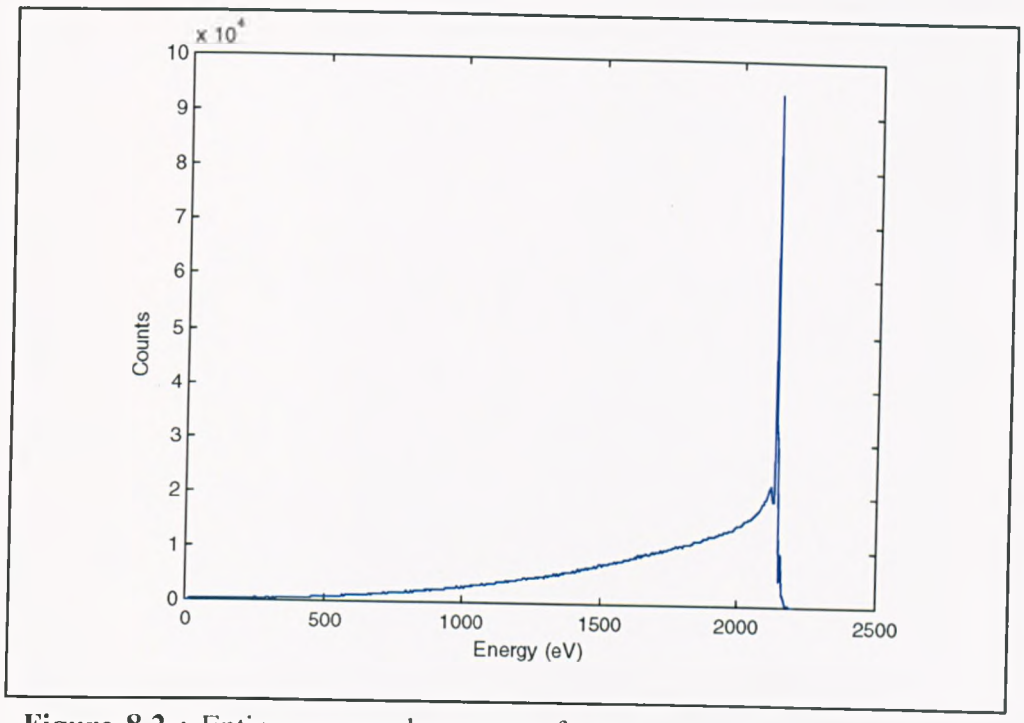

Figure 8.2 : Entire measured spectrum for an unprepared sample for 2keV electrons at 30nA sample current

8.1.1.1 The Effect of Magnetic Fields on Low Energy Spectra

It is well known that magnetic fields influence the trajectory of electrons depending on their magnitude. In the case of the CMA, these fields may act either in the region between the sample and the analyser, or in the analyser itself to change the characteristics of measured spectra. If an electron travels through a magnetic field, then the deflection, d, from its straight line path over length l, due to a magnetic flux density B is given by (Holloway, 1977)

$$d = 14800 l^2 B E^{-0.5} \qquad (8.1)$$

where E is the electron energy in eV, B is measured in Tesla and l and d in metres. As we see from equation 8.1, the lower the electron energy, the more susceptible it is to deflection by a given magnetic flux.

El-Bakush (1994) attributed a similar loss of the low energy side of the spectra measured on the Varian CMA at York, to a 30μT magnetic field in the exposed region

between the sample and the entrance to the CMA (as measured with a Hall probe). This field would have caused a deflection of 320μm for a 5eV electron travelling the 12.7mm between the sample and the CMA entrance slit. Figure 8.3 shows spectra collected by El-Bakush (1994) before and after the reduction of magnetic fields in the instrument.

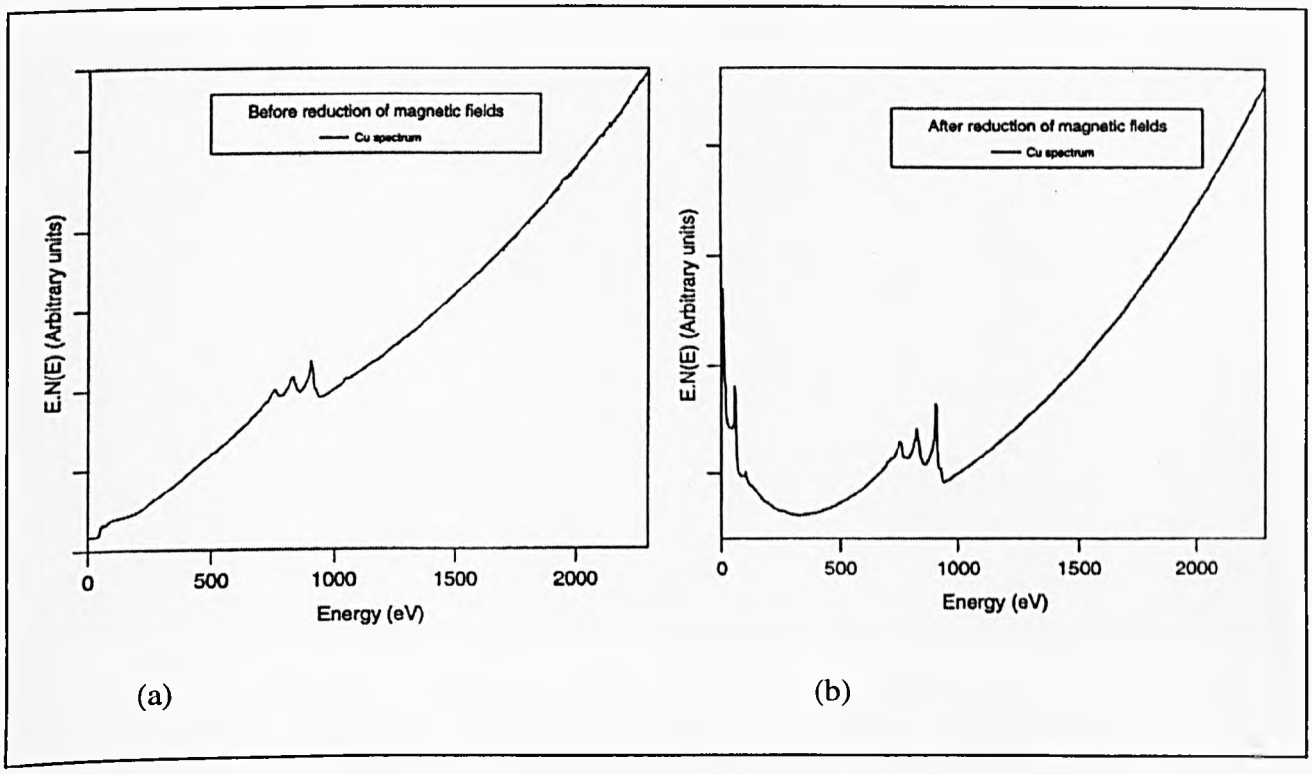

Figure 8.3 : (from El-Bakush, 1994) A measured copper spectrum for a 5.0keV beam, (a) before, and (b) after the reduction of magnetic fields

Figure 8.3 quite clearly shows that a good improvement has been made by the reduction of the magnetic field. Both the signal from the low energy true secondary electron peak and the low energy Auger peak at 58eV are dramatically improved.

8.1.1.2 The Effect of Sample Cleanliness on Low Energy Spectra

The spectrum shown in Figure 8.2 was collected from an un-prepared sample. A layer of contaminants on the surface of a sample may lead to a poor secondary electron yield from a sample which would otherwise have given a large true secondary peak (due to charging of particulate surface contaminants). To test this, a multi-layer semiconductor sample with a top layer of $Al_{0.3}Ga_{0.7}As$ was examined with a 5keV incident electron beam before and after cleaning by ion bombardment. The resulting spectra from a single channel of the multi-channel detector are shown in Figure 8.4. We can see from these results that there is a marked improvement in the low energy signal from the cleaned sample (and in fact the signal in general).

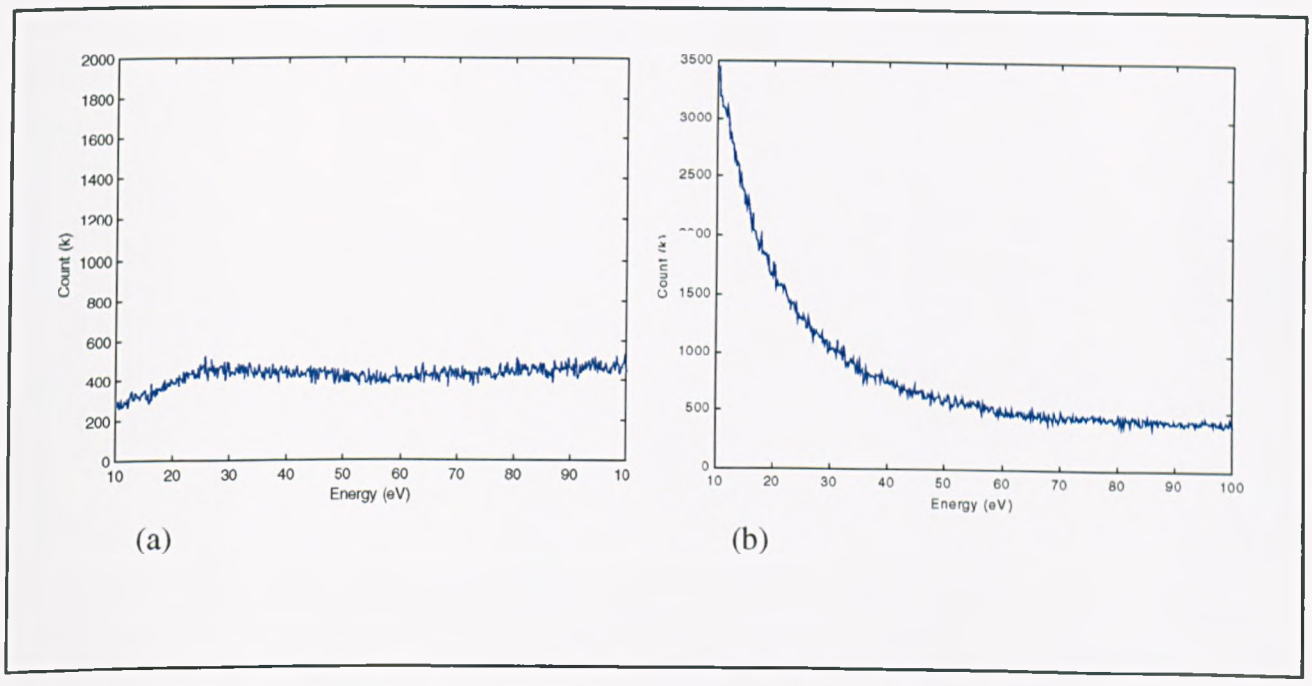

Figure 8.4 : Low energy spectra collected from (a) un-prepared and (b) ion beam sputtered samples

8.1.1.3 'Towing in' Effects on Low Energy Spectra

Figure 8.5 illustrates schematically the sectioned anode detector that forms the detector in the angle resolved CMA. As well as the 'dead' areas between the anode plates themselves, no signal will be detected from electrons which fall in the centre of the detector where the etched lines cross.

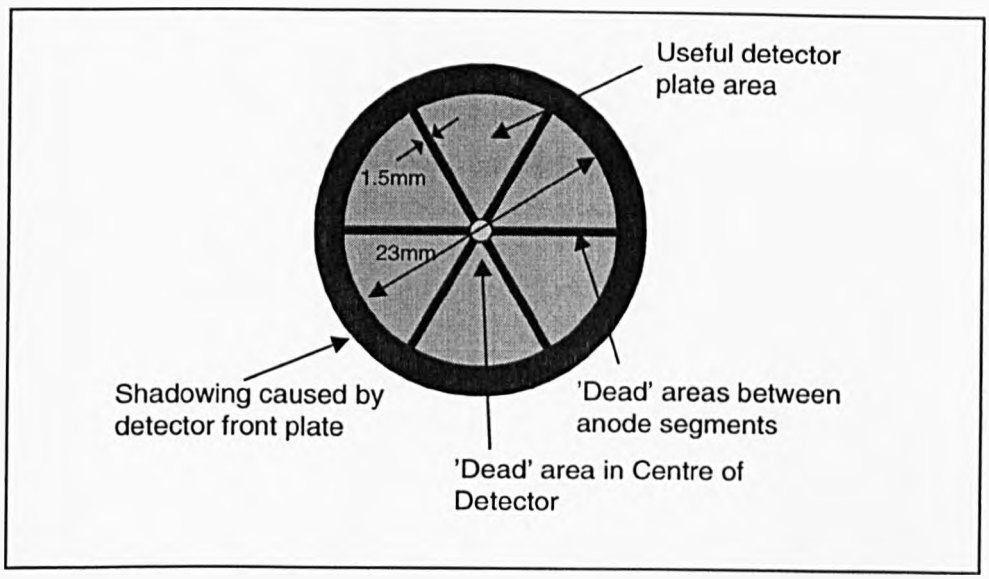

Figure 8.5 : Schematic diagram of the sectioned detector plate in the CMA

Electrons leaving the exit aperture of the CMA are accelerated towards the channel plate detector by the high positive potential on the gold anodes. This accelerating potential has the effect of 'towing in' electrons from their normal paths towards the centre of the detector (see Chapter six). Although the trajectories of high energy electrons are not greatly affected by this 'towing in' effect, it is possible that very low energy electrons may actually be drawn towards the dead area in the present design.

A SIMION simulation was designed to investigate a scenario in which low energy electrons may be drawn towards the centre of the detector and be lost. Figure 8.6 shows the path of 1eV electrons over the range 36.3^{o} - 48.3^{o} passing through the CMA and striking the detector plates.

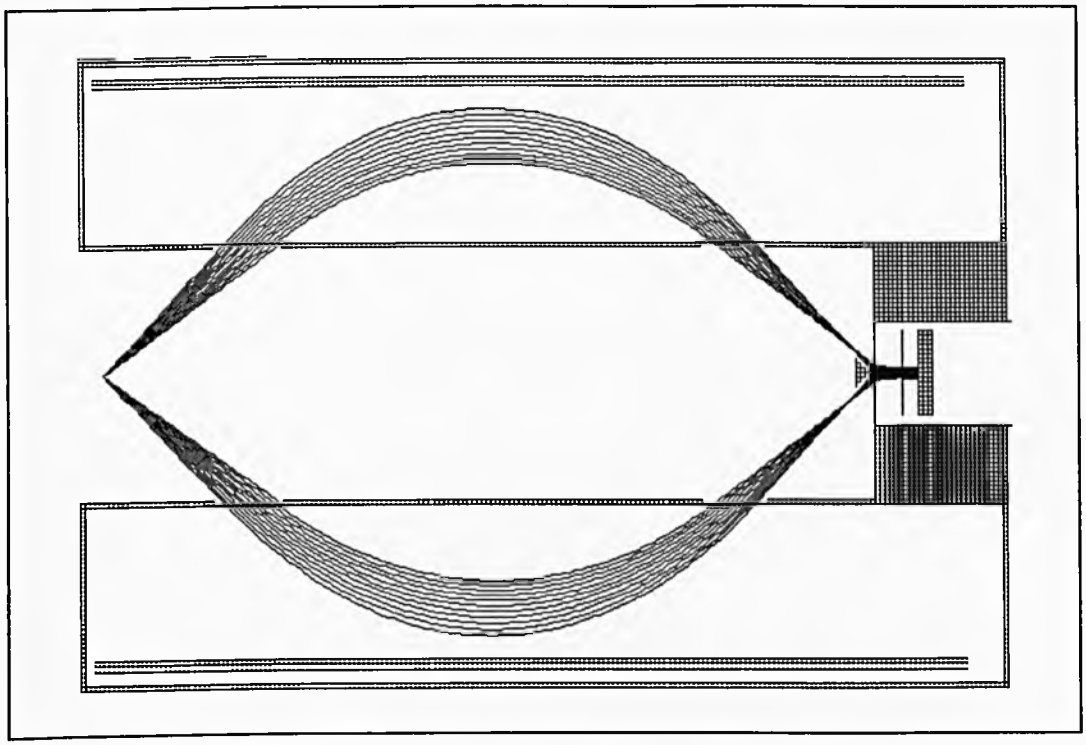

Figure 8.6 : SIMION simulation of very low energy (1eV) electrons passing through the CMA over the 12° range of collection angles

From the results shown in Figure 8.6, it is reasonable to conclude that under some experimental circumstances, low energy electrons may escape detection by this mechanism. However, even at this very low energy, the angular spread over the 12° acceptance angle of the CMA is such that not all of the electrons fall into the 'dead' area at the centre of the detector plate. For this reason, it would seem unlikely that the 'towing in' of low energy electrons into the etched area of the detector anode contributes significantly to the lack of a low energy signal in some spectra, and the mechanisms described in 8.1.1.1 and 8.1.1.2 are more likely to dominate.

8.1.2 Multichannel Detection

The method of displaying data from the six detector channels will depend on the information required. For example, in some cases noisy signals from shadowed detector channels will simply be disregarded, whereas in others we may wish to process the angle resolved data more intensively. For the purpose of this preliminary investigation, the data from the six detectors will be displayed as individual spectra, either on separate axes for clarity, or together for comparison.

Figure 8.7 shows a spectrum collected with a 2.2keV incident electron beam giving a 30nA measured sample current on an un-prepared Au/Al sample, collected with a 1000ms dwell time. Note that in this case, the y-axis limit has been set well below the elastic peak so that spectral features can be more clearly seen. Figure 8.8 shows the same results on separate axes.

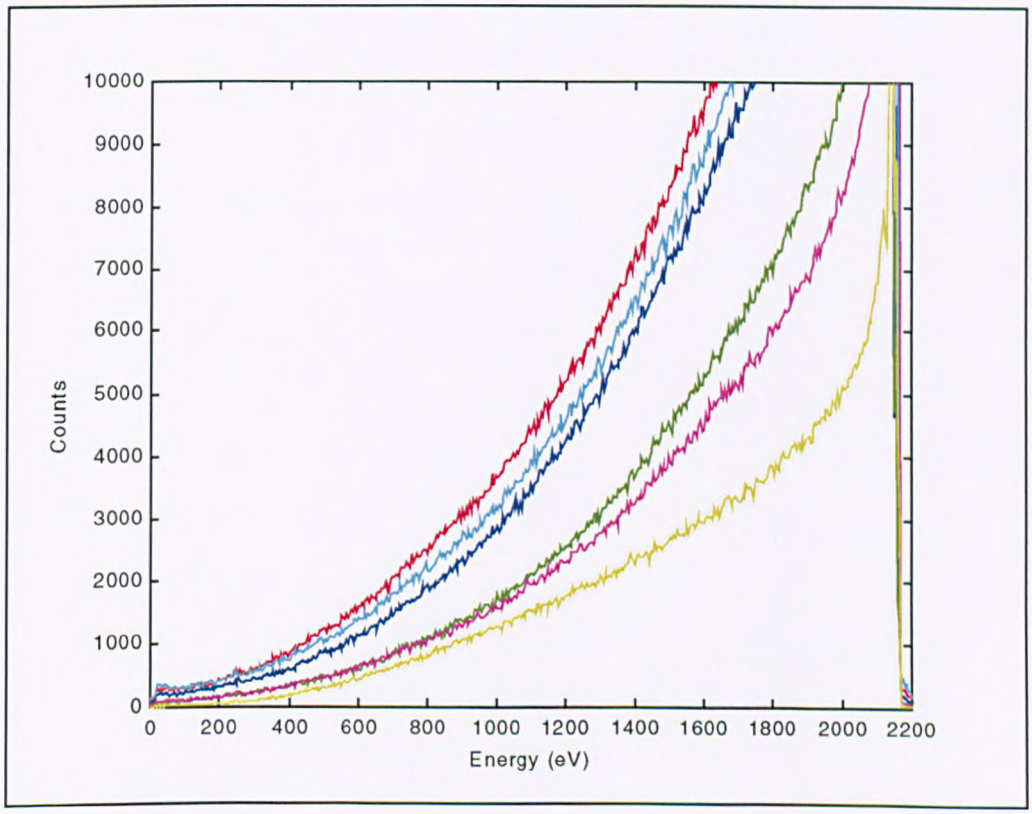

Figure 8.7 : Multichannel spectrum showing the signal from all 6 detector channels for an un-prepared sample with 2.2keV incident beam

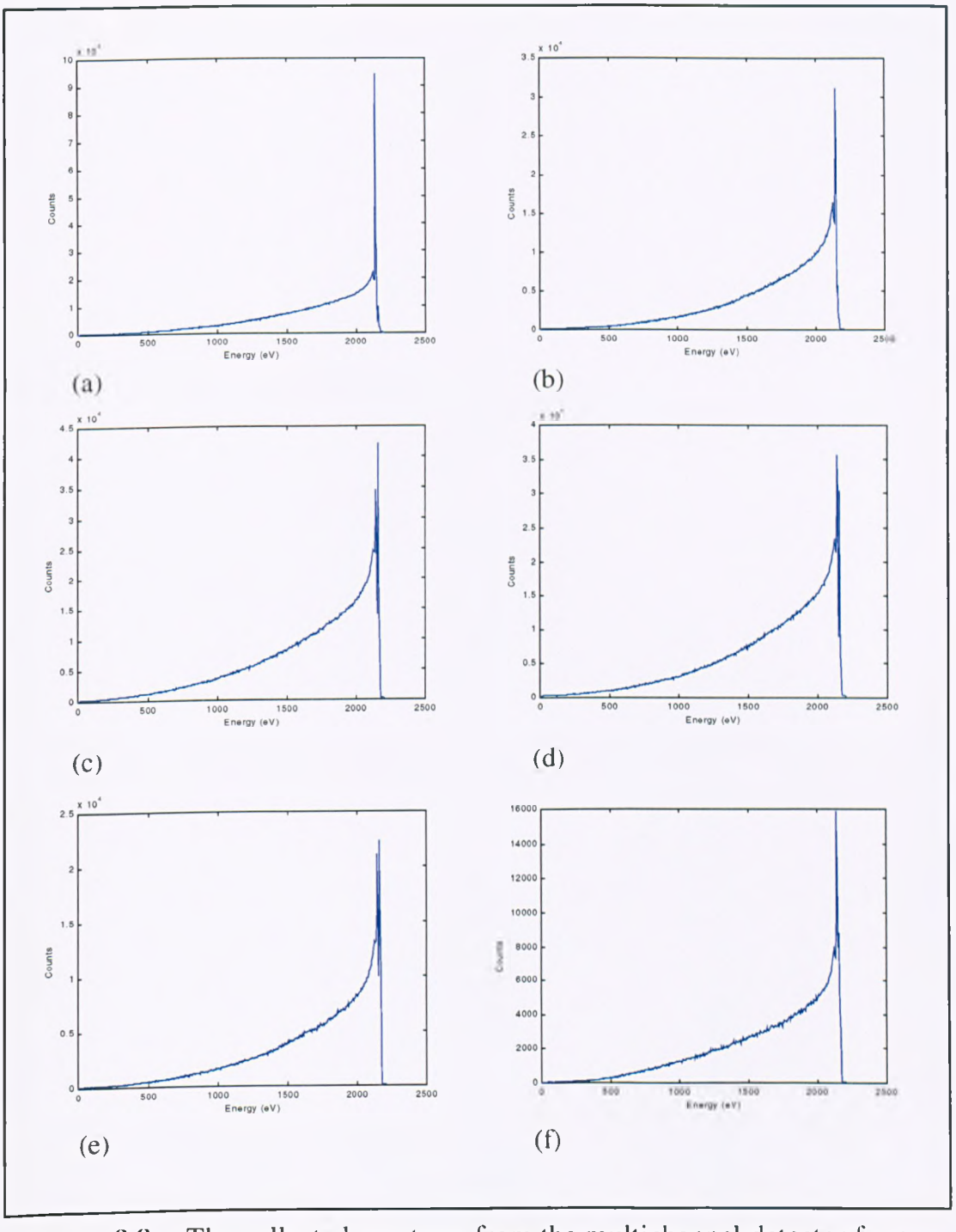

Figure 8.8 : The collected spectrum from the multichannel detector for a 2.2keV incident beam on a un-prepared sample from (a) channel 1, (b) channel 2, (c) channel 3, (d) channel 4, (e) channel 5 and (f) channel 6

It is clear, from examination of Figures 8.7 and 8.8 that there are significant differences between the spectra from the six channels both in terms of spectral features, and absolute magnitude. Since the sample under examination is positioned

normal to the incident electron beam and does not possess a complex surface morphology, we can assume isotropic emission of electrons (at least in the azimuthal angle of emission), and would therefore expect any differences in the spectra to be a result of gain variations in the electron energy analyser and detector.

8.1.2.1 Normalisation Issues with the Multichannel detector

The discrepancies between spectra from the six channels seen in Figures 8.7 and 8.8 could possibly be explained in terms of a uniform gain variation across the six channel plate detectors. An attempt is therefore made to normalise the six measured spectra by application of a single multiplying factor. Figure 8.9 shows the same spectra as Figure 8.7, normalised to 1000 counts at 450eV.

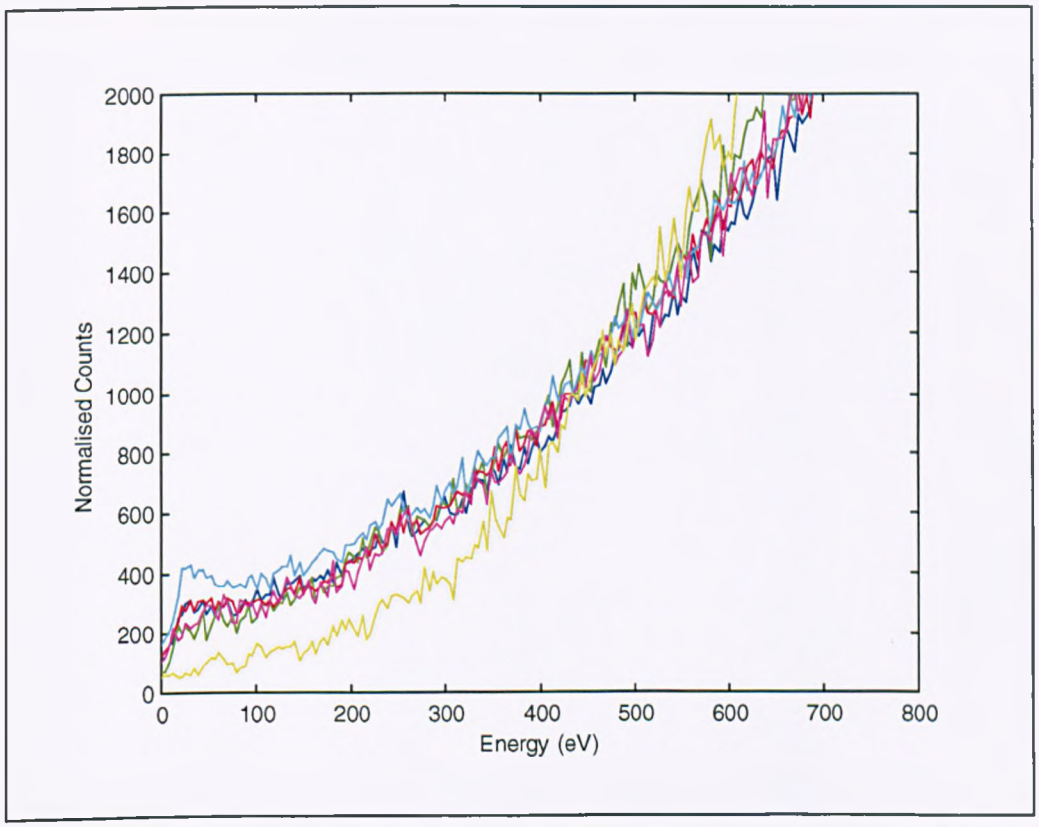

Figure 8.9 : The same results as shown in Figure 8.7, normalised to 450eV

As figure 8.9 reveals, the gain differences between the six channels are not constant with energy. Within 100eV of the normalised point, spectra from the six channels begin to diverge dramatically with no clear pattern across the six detectors. For example, the channel depicted in yellow falls below the group at energies lower than

450eV, but becomes significantly higher than the group at the high energy side of the plot. This result points towards an energy dependent gain variation in the current design. We will now explore possible causes of this gain variation with a view to its elimination.

8.1.2.1.1 Detector Alignment

The six segment detector is designed to have three axes of symmetry, to correspond with shadowed areas created by electrical feedthroughs which are inherent to the CMA design. Figure 8.10 illustrates this arrangement schematically.

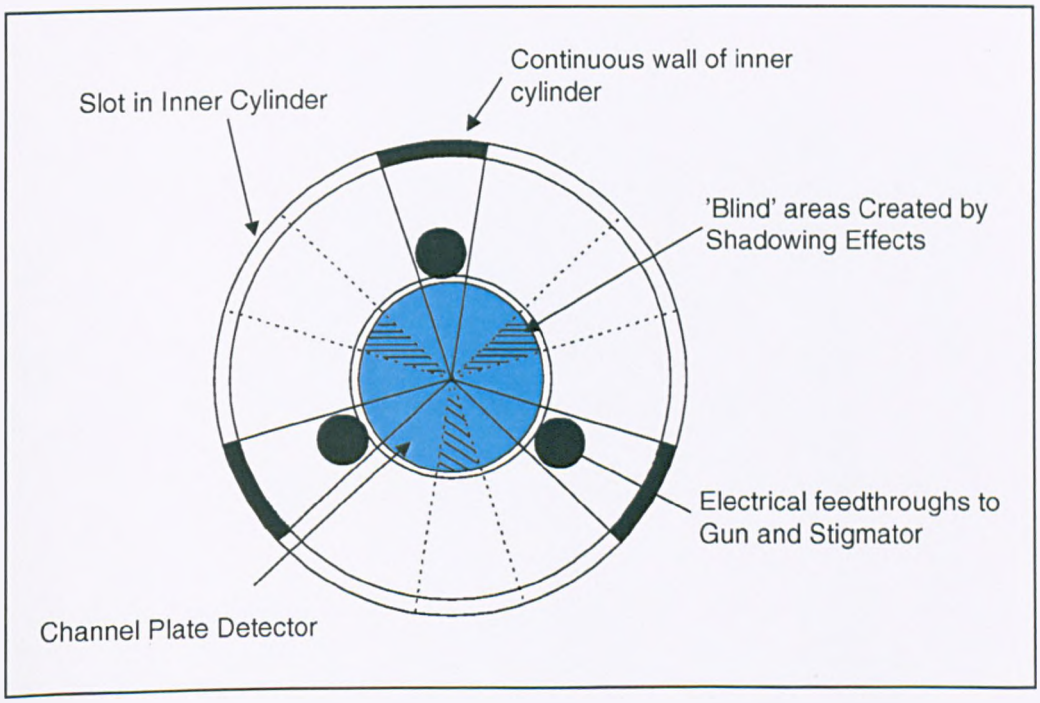

Figure 8.10 : Shadowing effects in the region of the channel plate detector

In the present design, it is intended that the lines of symmetry of the segmented detector coincide with the shadows created by the wall of the inner cylinder and the electrical feedthroughs. If this is achieved, then all six channels should be equally shadowed, and therefore give uniform signals under isotropic illumination. Figure 8.11 shows the correct alignment between the axes of symmetry of the channel plate detector and the areas of shadow described in Figure 8.10.

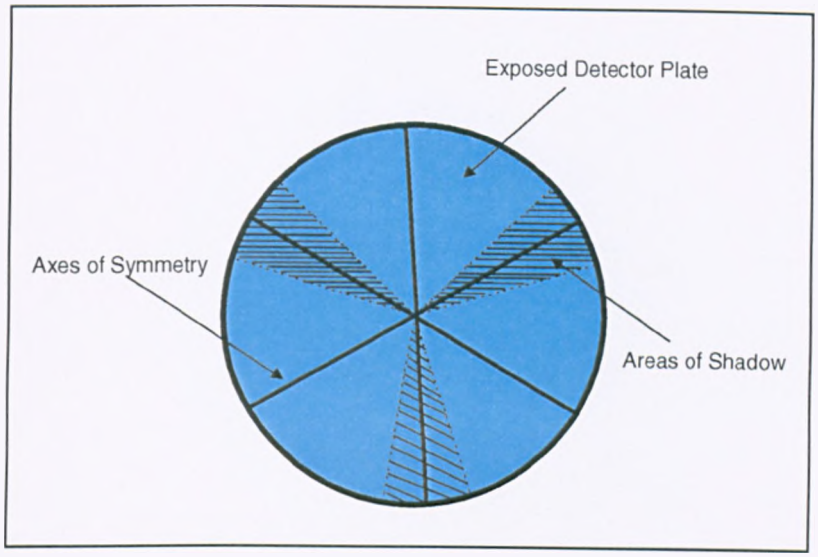

Figure 8.11 : Correct alignment of the sectioned anode detector plates with the shadowed areas of the CMA

We know that low energy electrons fall closer to the centre of the detector than those of higher energy. Although a misalignment of the detector head would result in a gain artefact due to different areas of each plate being exposed, this would not give an energy related component providing that the detector head is concentric with the optical axis of the CMA.

8.1.2.1.2 Channel Plate gain Variation

The gain of a microchannel plate varies over its surface area, depending on its age and quality. The plates used in the current detector were stored for several months under atmospheric conditions. Although the storage case was sealed, degradation of the channel plate performance is to be expected. Because the detector operates in pulse counting mode, the effect of a variation in gain will depend on the threshold of the discriminator used in the head amplifier. As well as a variation in gain of the individual channels, ageing of the plates may lead to some channels becoming completely inoperative. This effect will produce 'dead' areas of the channel plates where incident electrons are not detected. The SIMION simulation in Figure 8.12 shows the variation in area illuminated by electrons entering the CMA at 42.3°, for electron energies of 10eV and 3000eV.

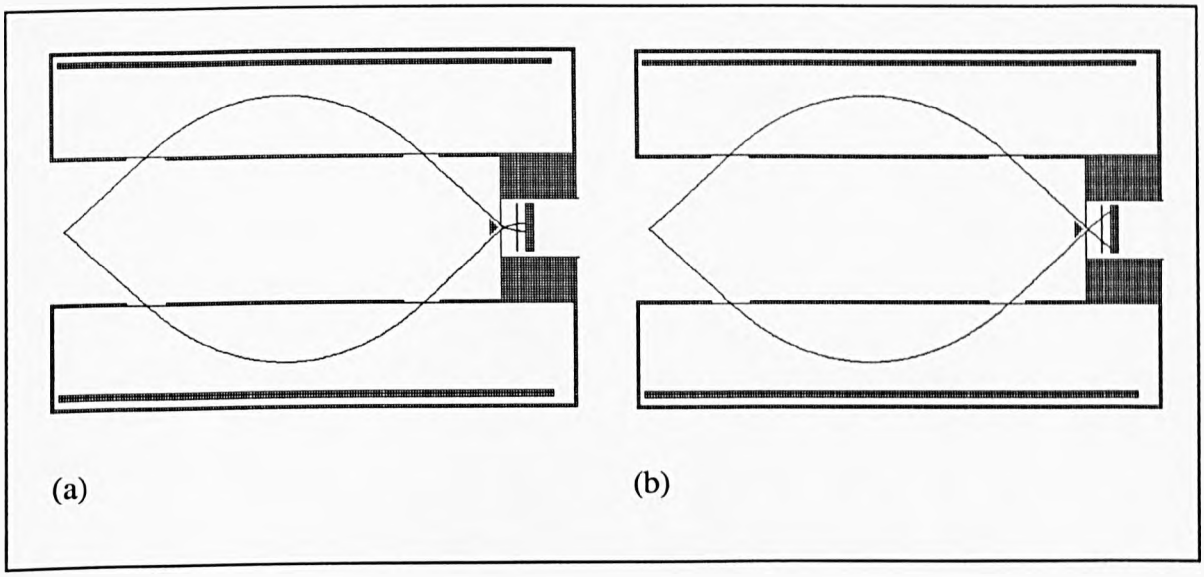

Figure 8.12 : SIMION simulation of electrons passing through the CMA analyser at 42.3° for (a) 100eV electrons and (b) 3000eV electrons

As Figure 8.12 indicates, over the energy range typically covered by the CMA the radial point at which electrons strike the channel plate detector varies between approximately 10% and 90% of their radius. If there is a substantial gain variation over the area of the plates then this simulation indicates that it would lead to an energy dependent gain variation in the measured signal (which may not be the same for all detector channels).

8.1.2.1.3 Normalisation with energy-related factor

If the gain variations observed in the measured six-channel spectra are energy dependent, it follows that there will be a unique scaling factor for each energy in a given spectrum which will correct for this effect.

To investigate this, two sets of spectra from the same sample material with the same beam energy are taken, the first at 15nA sample current and the second at 30nA. A set of scaling factors is calculated from the first spectrum set which normalises the spectra from each of the six detectors to the spectrum from channel 1. The spectra (shown in Figure 8.13) are collected with a one second dwell time, 2.2keV beam current and 500 points, giving five arrays of scaling factors, each of 500 elements.

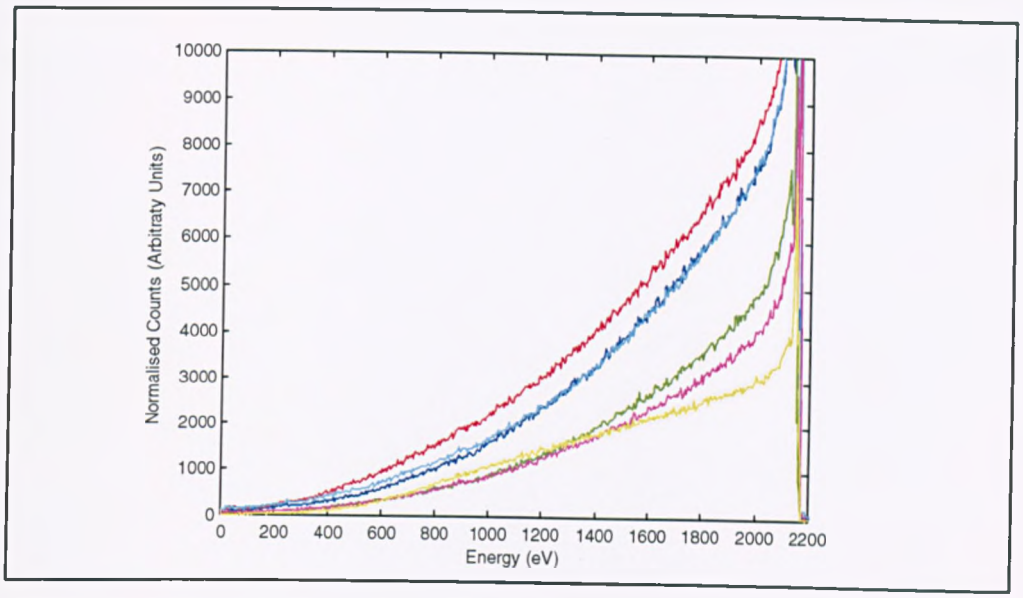

Figure 8.13 : Angle-resolved auger spectra from the six-channel detector for a 2.2keV incident beam at 15nA sample current

Figure 8.14 shows the results of applying these scaling factors to a second spectrum set obtained with a higher sample current of 30nA. If the energy-specific normalisation has been successful, we would expect the six spectra shown in Figure 8.14(b) to be co-incident.

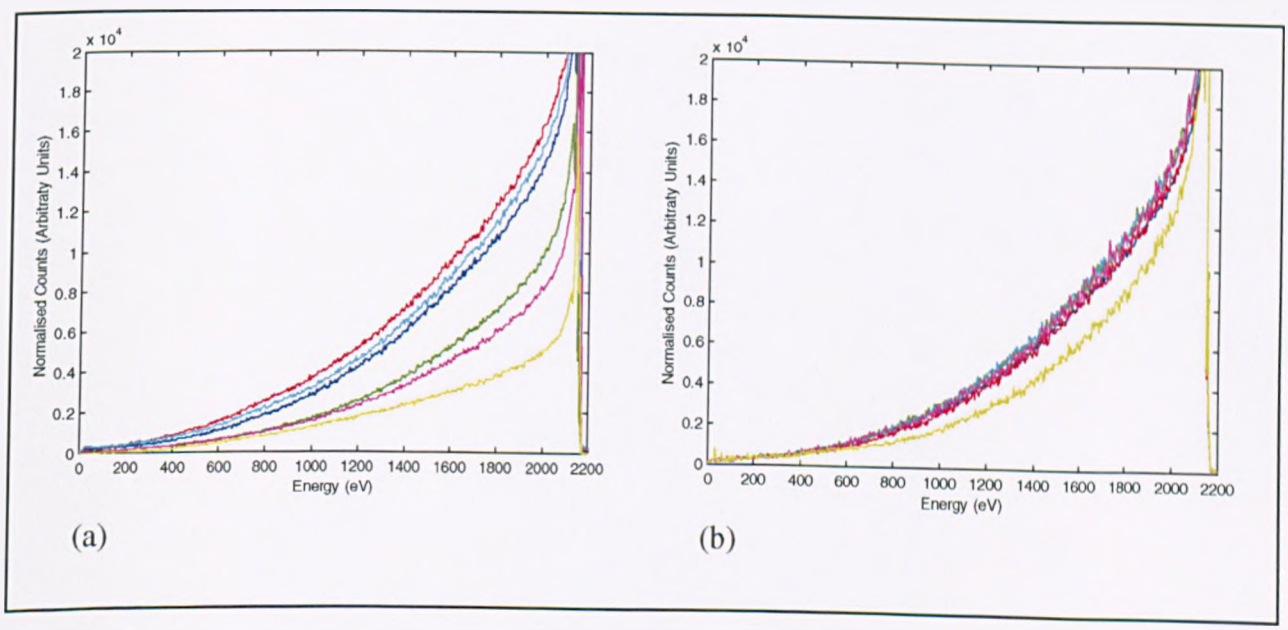

Figure 8.14 : Six channel angle resolved spectra of 2.2keV electrons at 30nA sample current on an un-prepared sample (a) before and (b) after application of energy-specific normalisation factors calculated from Figure 8.13

As figure 8.14 reveals, the application of energy-specific scaling factors has, on the whole, been successful. The notable exception is the signal drawn in yellow in Figure 8.14 which still lies apart from the group.

This study points to the influence of an energy dependent gain shift in the measured spectra from the angle-resolved analyser, but also suggests the presence of more complex underlying trends. It is possible that these effects may be focus related if, for example the detector plates are misaligned such that their face does not lie normal to the CMA axis. Further experiments will be required to fully establish the various phenomena which contribute towards gain variations in the multichannel detector and the extent to which each plays a part in the measured signal.

8.1.3 Examination of Auger Features

Due to the time intensive nature of spectrum acquisition and the limited duration of the research work presented here, time did now allow for a full Auger study to be carried out. However, in this section, we look at the spectrum from an un-prepared sample collected over the energy region of the carbon Auger peak. Figure 8.15 shows the spectrum collected from each channel of the angle resolved analyser for a 1keV incident beam, 30nA sample current over the energy range 0-550eV.

On this un-prepared sample, the carbon KLL Auger peak at 258eV is clearly visible, particularly on the spectrum collected from channel 2 of the detector. As we found in the previous studies in this chapter, the signal varies significantly according to the detector channel observed, even in this case where the spectra are scaled in magnitude to fit the axes. More work is needed to establish the validity of spectra acquired with this instrument; this will be discussed subsequently.

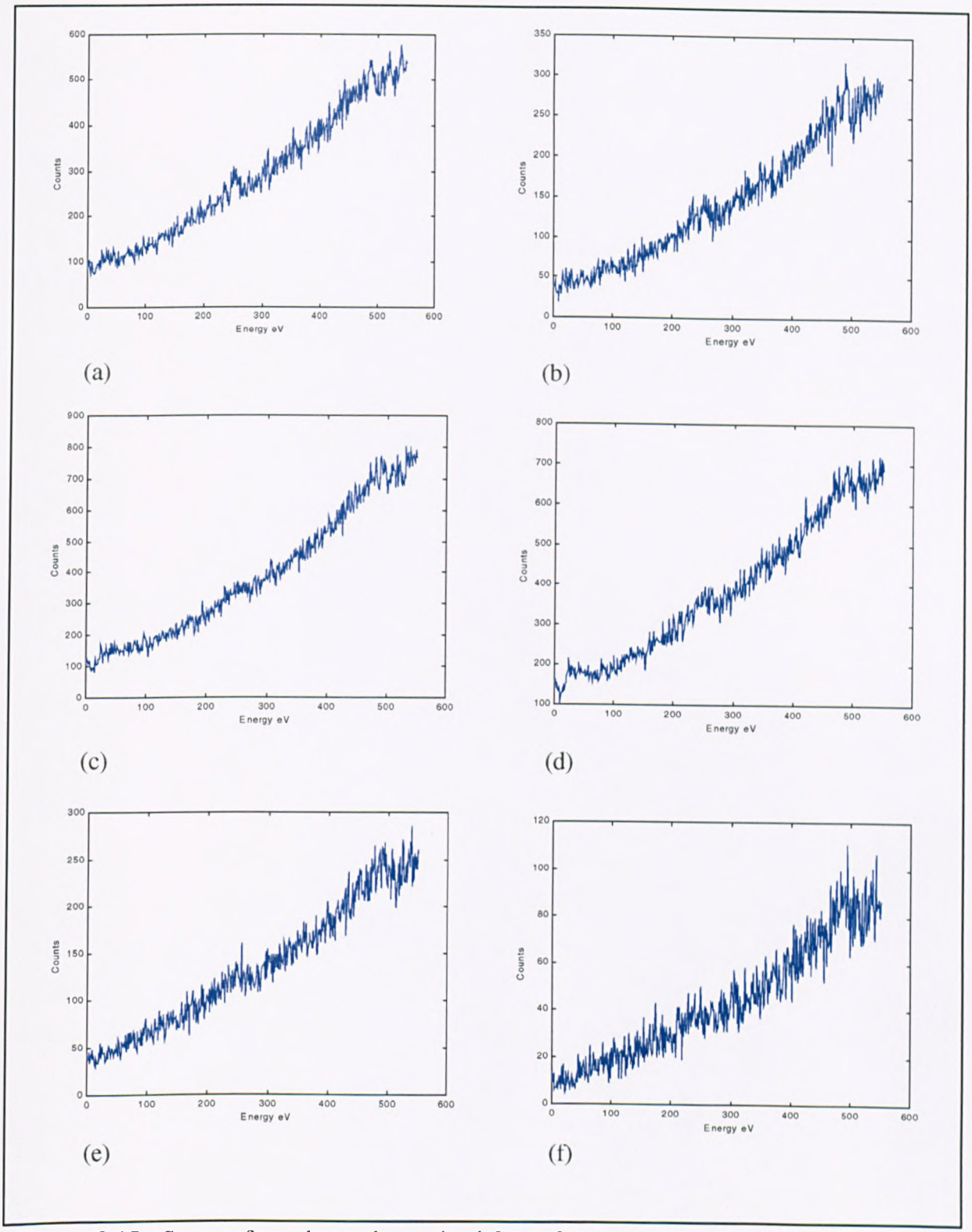

Figure 8.15 : Spectra from the angle-resolved CMA for an un-cleaned sample with 2.2keV incident electron beam from (a) channel 1, (b) channel 2, (c) channel 3, (d) channel 4, (e) channel 5 and (f) channel 6

8.2 Suggested areas for Further Development

It is apparent from the results presented in this chapter, that a substantial amount of work is still required before the angle resolved CMA will reach its full potential as an analytical instrument. The spectrometer described in this section of the thesis is intended to combine secondary electron microscopy with scanning angle-resolved auger analysis at very high resolution. Each element of the current instrument will be discussed separately with a view to identifying both short and long term areas which are in need of development.

8.2.1 The Secondary Electron Microscope

The computer controlled secondary electron microscope (SEM) has been designed, and was shown in Chapter seven to give reasonable images of a test grid. However, in direct comparison with a CRT-based analogue scanned instrument using the same detector head, the contrast produced in the computer generated images was found to be inferior (Gelsthorpe, 1998).

There are three components which play a part in the quality of computer generated SEM images (as opposed to CRT-based systems): the digital-to-analogue converter (DAC), the analogue-to-digital converter (ADC) and the method by which the image is ultimately displayed. In the present design, two 16-bit DACs are used, this gives a maximum resolution of 65536 pixels in each direction. Similarly, the ADCs are 12 bit, giving 4096 possible greyscales in the displayed image. Both of these are considered to be more than adequate to generate good quality images. The method by which SEM images are displayed is considerably more critical.

At present, the display routines offered by the Borland C++ compiler allow only 16 colours to be displayed simultaneously, it was therefore thought that ADC quantisation could not degrade performance. However, both the gain and black level on the SEM head amplifier are manually controlled. It is therefore conceivable that in some cases the signal from the head amplifier is of such an offset and magnitude that

clipping occurs at the analogue-to-digital conversion stage which leads to a reduction in contrast.

The suggested solution to this problem is to replace the variable control of the black level with a computer controlled voltage supply (a further DAC would be required for this), and to re-design the software under a compiler which allows at least 256 simultaneous shades to be displayed. This would provide a much more rigorous solution and should show a marked performance improvement over the present design.

8.2.2 The Electron Energy Analyser

The Varian CMA used in this study has already been well characterised (El-Bakush, 1994). Much work will now be required, examining the spectra obtained with the new angle-resolved analyser in order to quantify and ultimately eliminate unwanted artefacts.

General characterisation should now be carried out using cleaned samples of Cu and Au for comparison with results obtained before the introduction of the channel plate detector (El-Bakush, 1994). Perhaps the most significant issue highlighted by the results of this chapter is the calibration and normalisation of spectra from the six channel detector. A variation both in the absolute gain, and general response has been observed between the six detector channels.

One possible contributory factor to this variation is the position of the channel plate detector itself. A systematic study is required to ascertain the extent to which the alignment of the segmented detector head affects the measured spectrum. In future, a procedure is required by which the axes of symmetry of the detector can be accurately aligned with the shadowed areas of the CMA, as well as with the axis of the analyser.

Gain variations over the area of the channel plates has also been suggested as a possible cause of spectral artefacts. Two methods are suggested by which this effect may be reduced. Firstly, the detector plates themselves could be replaced by higher

quality microchannel plates with a known (and minimal) gain variation. Secondly, and perhaps more rigorously, a lens could be designed which would ensure that electrons impinging on the channel plate detectors fall over a well defined narrow annulus regardless of their energy. This second option is currently being pursued (Gelsthorpe, 1998) with the implementation of a modified einzel lens which derives its bias voltage from the outer cylinder of the CMA.

It may also be of interest to carry out a systematic study of the effect of gain and threshold variations in the A111 amplifier/discriminator used in the CMA head amplifier. Currently, these are set to the factory default and a full investigation of their effects would be useful.

8.2.3 Scanning Auger Microscopy

The control system described in Chapter seven is designed to carry out scanning Auger studies both in the form of linescans and full Auger images. Although preliminary linescans have been carried out to establish the operation of the computer interfacing hardware and software, time did not allow for a representative set of data to be fully analysed. Early attempts to gather full scanning Auger images failed to yield conclusive results.

A review is now required of the software currently used to obtain such images. The collection of test data from clean samples with known Auger features would allow a critical analysis of the data processing and display routines in the software described in Chapter seven.

8.2.4 The DDF in Multilayers, a Theory/Experiment Comparison

Overlayer experiments have been used by number of workers (Tilinin *et al.*, 1997; Cumpson, 1995) as a means of providing experimental results with which to validate theoretical procedures. Chapter four described a new Monte Carlo simulation designed to calculate the depth distribution function (DDF) in heterogeneous structures. A theory/experiment comparison of, for example, a single overlayer

system would provide an extremely useful basis for further validation of the model used in the multilayer simulation.

Figure 8.16 shows a possible experimental arrangement for such an experiment. An overlayer of known thickness, a, of material Z_A is deposited on a bulk substrate of a different material Z_B. A bevel is then cut with the computer controlled EX050S ion gun such that a staircase effect is created in the overlayer material (see Figure 8.16). Knowledge of the depth, b, of each 'step' in the bevel would be required. For this reason, accurate characterisation of the ion beam prior to this experiment would be essential.

An Auger linescan would then be performed along the line of the bevel. The Auger signal from the substrate material could then be mapped as a function of overlayer thickness. This signal, in effect, would correspond to the sum total of Auger emission from the substrate material with a given overlayer thickness. A series of Monte Carlo simulations based on a successive overlayer thickness could then be performed and subsequent comparison drawn between theoretical and experimental yields. The angle resolved CMA may be particularly useful in this experiment to enable the analyst to understand spectral characteristics introduced by the complex topology of this experiment.

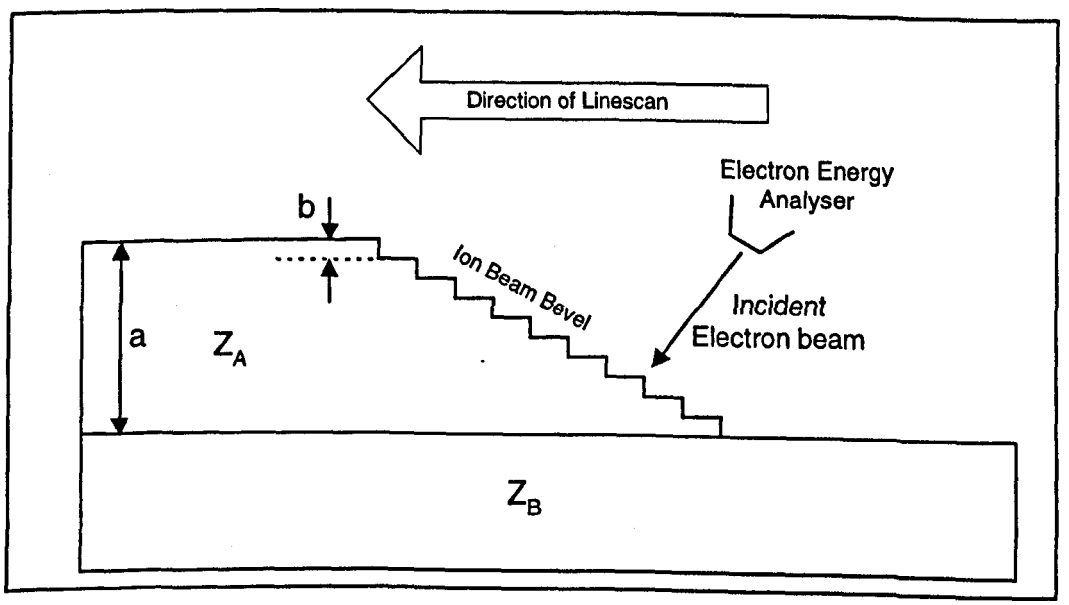

Figure 8.16 : Schematic diagram of theory/experiment comparison for validation of the multilayer Monte Carlo simulation

8.3 Conclusions

Although, due to time limitations, only a preliminary study of the multichannel detector has been made, it has been possible to establish the general operation of the angle resolved CMA and highlight some important areas for further development. Results have been presented for un-prepared samples which reveal two important features in the spectra observed: a lack of a low energy signal, and an energy dependent gain variation across the six detector channels.

The absence of a significant low energy secondary electron cascade has been investigated. Three possible sources were suggested: contamination of the analyser by a magnetic field, 'towing in' effects of electrons towards the 'dead' area at the centre of the channel plate analyser, and sample contamination. Investigation of these possible causes indicates that sample cleanliness and magnetic fields are the most plausible of these possibilities. Further work is suggested to gain a better understanding of this effect.

The gain variation across the six detector channels is investigated. Attempts are made to normalise spectra both with a single multiplying factor, and an energy dependent factor. The results of this study indicate that an energy related gain is present in the current detector arrangement, but that focusing effects and detector alignment may also be important.

Suggestions have been made for further development of the angle resolved CMA and supporting instrumentation. It is hoped that these will overcome some of the limitations of the present design uncovered in the course of this chapter.

CHAPTER NINE

CONCLUSIONS

Quantitative Auger electron spectroscopy is now a very well established technique. There are several issues currently under examination by researchers striving to improve our understanding of the physical processes leading to a measured Auger signal. The work presented in this thesis has attempted to address two of these issues. Firstly, two theoretical methods of modelling electron-solid interaction were explored; transport theory and Monte Carlo simulation. Secondly, a novel angle-resolved electron energy analyser is described which combines a high resolution field emission electron gun (FEG) and a multichannel electron detector (MCD) with a conventional Varian cylindrical mirror analyser (CMA) .

Transport theory has gained acceptance in recent years, and as described in Chapter two has been shown to give reliable results in theory-experiment comparisons. However, the model used in transport theory is only applicable at near-normal angle of incidence and when the ratio of λ_i/λ_{tr} can be considered constant, limiting its practical application to homogeneous sample structures. In Chapter three, we examined some of the fundamental parameters of transport theory with a view to establishing the extent to which generalisations can be made. These include the transport mean free path, λ_{tr} and the inelastic mean free path, λ_i. Variations in λ_{tr} with energy and atomic number are investigated in detail and some anomalous features are explained in terms of the generalised Ramsauer-Townsend effect.

Chapter four described a new Monte Carlo simulation based on the extension of the statistical weights method described by Cumpson (1993) to deal with heterogeneous sample structures. The multi-layer simulation presented here is shown to give excellent agreement with previously published results where available, and also to agree well with transport theory calculations within the limits of their application. It is quite conceivable that this multi-layer model could be extended to account for a more complex three-dimensional sample morphology, making it useful in a wide range of practical experiments.

Attention is turned to the development of a high resolution, angle resolved Auger electron microscope. This part of the thesis covers the design of a versatile spectrometer which combines a secondary electron microscope with an electron energy analyser to allow computer control over the entire instrument from a single graphical software environment. As explained in Chapter one, current device technology demands very high resolution analysis (of the order of 10-50nm) and to allow for this, a miniature field emission electron column has been designed to be accommodated in the CMA.

To make angle-resolved analysis possible with the CMA, a sectioned anode multi-channel detector head is designed, with channel electron multiplier plates to provide the required gain in the signal. The detector head electronics are based around a commercially produced charge sensitive amplifier/discriminator. In Chapter six, the SIMION electrostatic lens simulation program is used to determine optimum detector position.

A complete graphical user environment has been designed to allow integration of the secondary electron microscope and Auger electron spectrometer. This allows the user to graphically define areas on an SEM image for subsequent Auger analysis. The current software also allows Auger linescans and images to be acquired.

Chapter eight is an investigation into spectral features resulting from the use of the new detector. Preliminary results from the angle resolved spectrometer are encouraging and demonstrate basic functionality of all six detector channels. Examination of the spectra reveals some anomalies including a complex, and partially energy-dependent, gain variation, and a reduced low energy signal. Suggestions are made for further work which may lead to better understanding of the new detector and the elimination of these artefacts.

REFERENCES

Alrad Instruments Limited, *'Microchannel Plates' data sheet*, Turnpike Road Industrial Estate, Newbury RG13 2NS

AMPTEK Inc. 6 De Angelo Drive, Bedford, Mass. 01730 U.S.A.

Ashley (1988), *J. Electron Spectrosc. Relat. Phenm.*,**46**, 199

Auger P. (1925), *J. Phys. Radium*, **6**, 205

Barthés-Labrousse M.-G. (1998), *Report on the 11th IUVSTA workshop 'Auger Electron Spectroscopy : from Physics to Data', Surf. Interface Anal.*,**26**, 72

Barton J. J., Hussain Z. and Shirley D. A. (1987), *Phys. Rev.* **B35**, 933

Baschenko O. A. and Nefedov V. I. (1979), *J Electron Spectrosc. Relat Phem.*,**17**, 405

Berger M. J. and Doggett J. (1956), *Natl. Bur. Stand. J. Res,*. **56**, 89

Bishop H. E. , Coad J. P. and Rivière J. C. (1972/1973), *J. Electron Spectrsc.Relat. Phenm.*, **1**, 398

Bosch A., Fiel H. and Sawatzky G. A. (1984), *J Phys E: Sci. Instrum.*,**17**, 1187

Briggs D. and Rivière J. C. (1990) in *Practical Surface Analysis (Second Edition) Volume 1 – Auger and X-ray Photoelectron Spectroscopy*, Ed. Briggs D. and Seah M. P., Wiley

Briggs D. and Seah M. P. (1990) in *Practical Surface Analysis (Second Edition) Volume 1 – Auger and X-ray Photoelectron Spectroscopy*, Ed. Briggs D. and Seah M. P., Wiley

Casnati E., Tartari A. and Baraldi C. (1982), *J. Phys.B*, **15**, 155

Chambers A., Fitch R. K. and Halliday B. S. (1989), *Basic Vacuum Technology*, Adam Hilger, Bristol

Chambers S. A., Greenlee T. R., Smith C. P. and Weaver J. H. (1985), *Phys. Rev.***B32(6)**, 4245

Chang C. C. (1974), *Characterisation of Solid Surfaces*, Plenum Press, New York, 509

Cumpson P. J. (1993), *Surf. Interface Anal*, **20**, 727

Cumpson P..J. and Seah M. P. (1997), *Surf. Interface Anal*, **25**, 430

Cumpson P. J. (1997), *Surf. Interface Anal*, **25**, 447

Czyzewski Z., MacCallum D. O., Romig A. and Joy D. C., (1990), *J Appl. Phys.* **68**, 3066

Dahl D. A. and Delmore J. E. (1988), *The SIMION PC/PS2 users Manual Version 4.0*, Idaho National Engineering Laboratory.
Davis L. E., MacDonald N. C., Palmberg P. W., Riach G. E. and Weber R. E. (1976), *Handbook of Auger Electron Spectroscopy*, Second Edition, Physical Electronics Industries Inc., Eden Prairie, Minnesota

Dwyer V. M. and Matthew J. A. D. (1984), *Surf. Science*, **143**, 57

Dwyer V. M. and Richards J. M. (1992), *Surf. and Interface Anal.*, **18** , 555

Dwyer V. M. (1994a), *Surf. and Interface Anal.*, **21**, 637

Dwyer V. M. (1994b), *Surf Science*, **310**, L621-L624

Dwyer V. M. (1994c), *J. Vac. Sci. Technol.* **A 12(5)**, 2680

Eagen C. F. and Sichafus E. N. (1977), *Rev. Sci. Instrum.* **48** (10) 1269

Ebel H. and Jablonski A. (1984), *Surf. and Interface Anal.*, **6**, 21

Egelhoff W. F. (1993), *Phys.Rev. Lett.*, **71**, 2883

El-Bakush T. A. (1994) Dphil Thesis, University of York.

El-Gomati M. M. (1983) Dphil Thesis, University of York.

El-Gomati M. M., Prutton M., Browning R. (1985), *J.Phys.E:Sci Instrum.*,**18**, 32

El-Gomati M. M. (1988), *Vacuum* **38**, 337

El-Gomati M. M., Prutton M., Lamb B. and Tuppen C. G. (1988), *Surf. Interface Anal.*,**11**, 251

El-Gomati M. M. (1998), *Private Communication*

Fitzgerald A. G. (1998), *Mikrochim. Acta*, **15**, 351

Galileo Electro-Optics Corporation (1996), *'HOT Microchannel Plates' Data Sheet*

Gelsthorpe A. (1997), *University of York, Internal Report*

Gelsthorpe A. (1998), *University of York, Private Communication*

Gries W. H. and Werner W. S. M. (1990), *Surf. Interface Anal.*, **16**, 149

Gries W. H. (1995), *J. Vac. Sci. Technol.*, **A13**, 1304

Gries W. H. (1996), *Surf. Interface Anal.*, **24**, 38

Gryzinski M. (1965), *Phys Rev.* **138A**, 336

Hafner H., Simpson J. A., Kuyatt C. E. (1968), *Rev. Sci. Instrum.*, **39**, 33

Hall P. M. and Morabito J. M. (1979), *Surf. Sci.*,**83**, 391

Harris Semiconductor (1986) 82C54 Data Sheet No. 2970.1

Higatsberger M. (1981), *Advances in Electronics and Electron Physics*, **56**, 291

Hoflund G. B., Asbury D. A., Corallo C. F. and Corallo G. R. (1988), *J. Vac. Sci. Technol.* **A6(1)**, 70

Hofmann S. (1990) in *Practical Surface Analysis (Second Edition) Volume 1 – Auger and X-ray Photoelectron Spectroscopy*, Ed. Briggs D. and Seah M. P., Wiley

Holloway P. (1977), *Surf. Sci.*,**66**, 635

Huang M., Harland C. J. and Venables J. A. (1993), *Surf. Interface Anal.*,**20**, 666

ICI Advanced Materials, P.O. Box 90, Wilton, Middlesborough, Cleveland, TS6 8JE

Idzerda Y. U., Lind D. M. and Prinz G. A. (1988), *J. Vac. Sci. Technol.* **A7(3)**, 1341

Inokuti M. (1971), *Rev. Mod. Phys.*, **43(3)**, 297

Jablonski A. (1987), *Surf. Sci.*,**188**, 164

Jablonski A. (1989), *Surf. Interface Anal.*, **14**, 659

Jablonski A. and Tilinin I. S. (1995), *J Electron Spectrosc.***74**, 207

Jablonski A. (1996), *Surf. Sci.* **364**, 380

Jablonski A. and Tougaard S. (1998), *Surf. Interface Anal.* **26**, 17

Jackson A. R. (1994) *Third Year Project Report, University of York*

Joy D. C. (1988), *Inst. Phys.Conf. Ser.*, **93**, 23

Kirk R. (1992) 'Second Year Software Engineering' Course Notes, University of York Publication.

Lampton M. and Malina R. F. (1976), *Rev. Sci. Instrum.*,**47**, 1360

Lampton M. and Carlson C. W. (1979), *Rev. Sci. Instrum.*,**50**, 1093

Martin C., Jelinsky P., Lampton M., Malina R. F. and Anger H. O. (1981),*Rev. Sci. Instrum.*,**52(7)**, 1067

McGuire G. E. (1979), *Auger Electron Spectroscopy Reference Manual.* Plenum Press.

McKale A. G., Veal B. W., Paulikas A. P., Chan S. K. and Knapp G. S. (1988), *Phys. Rev.* **B38** 10919

Mott N. F. (1929), *Proc. R. Soc.* **A 124,** 425

Mróz A., Mróz S., Zagórski M. (1988), *Surf. Interface Anal.*,**12**, 49

Orloff J. and Swanson L. W. (1979), *J. App. Phys.*, **50**, 2494

Palmberg P. W., McDonald N. C., Riach G. E. and Weber R. E. (1972), *Handbook of Auger Electron Spectroscopy*, Physical Electronics Inc., Edina

Palmberg P. W. (1974), *J. Electron Spectrosc. Relat. Phenm,* **5**, 691

Penn D. R. (1976), *J. Electron Spectrosc. Relat. Phenm.*,**9**, 28

Powell C. J. (1974), *Surf. Sci.*,**44**, 29

Powell C. J. (1985), *J. Vac. Sci. Technol.*, **A3** 1338

Powell C. J. (1986), *J. Vac. Sci. Technol.*, **A4** 1532

Powell C. J. (1988), *J. Electron Spectrosc. Relat. Phenm.*, **47** 197

Powell C. J., Jablonski A., Tanuma S., and Penn D. R. (1994), *J Electron Specrosc. Relat. Phenm.*, **68**, 605

Purcell E. M. (1938), *Phys. Rev,* **54**, 818

Prutton M. (1994), *Introduction to Surface Physics*, Oxford University Press

QUE Programming Series (1989), *DOS Programmers reference, Second edition*

Rehr J. J., Booth C. H., Bridges F. and Zabinsky S. I. (1994), *Phys Rev* **B49** 12347

Risley J. S. (1972), *Rev Sci Instrum,* **43,** 95

Rivière J. C. (1990), in *Practical Surface Analysis (Second Edition) Volume 1 – Auger and X-ray Photoelectron Spectroscopy*, Ed. Briggs D. and Seah M. P., Wiley

Roy D. and Carette J. D. (1977), *Electron Spectroscopy for Surface Analysis*, Topics in Current Physics **4**, Ed. Ibach H, Springer Verlag, Berlin

Sar El H. Z. (1967), *Rev. Sci. Instrum*, **38**, 1210

Sar El H. Z. (1968), *Rev. Sci. Instrum*, **39**, 533

Sar El H. Z. (1970), *Rev. Sci. Instrum*, **41**, 561

Seah M. P. and Dench W. A. (1979), *Surf. Interface Anal.*,**1**, 2

Seah M. P. (1989), *Methods of Surface Analysis*, Chapter 3. Cambrige

Seah M. P., Lim C. S. and Tong K. L. (1989), *J Electron Spectrosc.Relat. Phenm.*, **48**, 209

Seah M. P. (1990) in *Practical Surface Analysis (Second Edition) Volume 1 – Auger and X-ray Photoelectron Spectroscopy*, Ed. Briggs D. and Seah M. P., Wiley

Seah M. P., Smith G. C. and Anthony M. T. (1990), *Surf. Interface Anal.***15**, 293

Seah M. P. and Gilmore I. S. (1998a), *Surf. Interface Anal.*,**26**, 815

Seah M. P. and Gilmore I. S. (1998b), *Surf. Interface Anal.*,**26**, 908

Sekine T., Nagasawa Y., Kudoh M., Sakai Y., Parkes A. S., Geller J. D., Mogami A. and Hirata K. (1982), *Handbook of Auger Electron Spectroscopy*, JEOL ltd., Tokyo

Shimizu R. and Ding Z. (1992), *Rep. Prog. Phys.*, **55**, 487

Shiokawa Y., Isida T. and Hayashi Y. (1979), *Auger Electron Spectra Catalogue : A Data Collection of Elements.* Anelva

Tanuma S., Powell C. J. and Penn D. R. (1988), *Surf. Interface Anal.* **11**, 577

Tanuma S., Powell C. J. and Penn D. R. (1990), *J. Electron. Spectrosc. Relat. Phenm.*,**52**, 285

Tanuma S., Powell C. J. and Penn D. R. (1991), *Surf. Interface Anal.* **17**, 911

Tanuma S., Powell C. J. and Penn D. R. (1991), *Surf. Interface Anal.* **17**, 927

Tanuma S., Powell C. J. and Penn D. R. (1997), *Surf. Interface Anal.* **25**, 25

Thompson M., Baker M. D., Christie A. and Tyson J. (1985), *Auger Electron Spectroscopy*, Ed. Elving and Windefordner, John Wiley & Son, Chichester

Tilinin I. S. (1988), *Sov. Phys. JETP* **67(8)**, 1570

Tilinin I. S. and Werner W. S. M. (1993), *Surf. Sci.* **290**, 119

Tilinin I. S. and Werner W. S. M. (1994), *Microchim. Acta.* **114/115**, 485

Tilinin I. S. (1992), *Soviet Physics JETP* **67**, 1570

Tilinin I. S., Zemek J. and Hucek S. (1997), *Surf. Interface Anal.* **25**, 683

Tougaard S., Sigmund P. (1982), *Phys. Rev. B*, **25**, 4452

Van Hoof H. A., Van der Wiel M. J. (1980), *J. Phys. E: Sci. Instrum.*,**13**, 409

Vasina P. and Frank L. (1979), *J. Phys. E: Sci. Instrum*, **12**, 744

Walker C., Peacock D., Prutton M. and El-Gomati M. M. (1988), *Surf. Interface Anal.*,**11**, 266

Werner W. S. M., Gries W. H. and Stori H. (1991), *Surf. Interface Anal.*, **17**, 693

Werner W. S. M. (1991), *Surf. Science*, **257**, 319

Werner W. S. M. (1992), *Surf Interface Anal.*, **18**, 217

Werner W. S. M. and Tilinin I. S. (1992), *Surf. Sci. Lett.* **268**, L319

Werner W. S. M. and Tilinin I. S. (1993), *Appl. Surf Sci.*. **70/71**, 29

Werner W. S. M. and Tilinin I. S. (1994), *Progress in Surface Science*, **46**, 241

William P. (1979), *Surf. Sci.*,**90**, 588

APPENDIX A

LOW-LEVEL CONTROL SOFTWARE

The C++ source code listings in this appendix relate to the low level control software, which forms the interface to the spectrometer. A modular design has been adopted to allow for future expansion.

FILE 'BB.H'

```
/* ------------------------ HEADER FILE BB.H ------------------------ */
/* -- Initialisation routines for the Burr-Brown interfacing system -- */
/* ------------------------------------------------------------------ */


/* Function bb_check returns 1 if it detects a BB PCI-200098C Carrier */
/* with a PCI-20006M DAC board in Slot 2 and a PCI-20007M Counter     */
/* module in slot 1. Otherwise it returns 0                          */

int bb_check(void) {

        unsigned char base = 0, mod1 = 0, mod2 = 0;
        base = peekb(base_id , 0x0000);  /* Read Board ID for Carrier */
        mod1 = peekb(base_id , 0x0100);  /* Read Board ID for Module in Slot 1 */
        mod2 = peekb(base_id , 0x0200);  /* Read Board ID for Module in Slot 2 */
        if ((base == 1) && (mod1 == 234) && (mod2 == 227))
          return(1);
        else
          return(0);
        }

/* Function bb_init initialises the counters on the BB carrier and on the */
/* PCI-20007M counter module                                               */

void bb_init(void) {
  /* Initialise DAC */
  dac_out(0.0,0);
  /*  %%%%  Set up the 20007M Counter Board as 4 16 bit Event cntrs. %%%%  */

  pokeb (base_id, 0x010B, 32+16); /* Programming Counter 0 of chip 1 to Mode 0 (count
  mode)*/
  pokeb (base_id, 0x010B, 64+32+16); /* Programming Counter 1 of chip 1 to Mode 0*/
  pokeb (base_id, 0x010B, 128+32+16); /* Programming Counter 2 of chip 1 to Mode 0 */

  pokeb (base_id, 0x0107, 32+16); /* Programming Counter 0 of chip 2 to Mode 0 (count
  mode)*/
  pokeb (base_id, 0x0107, 64+32+16); /* Programming Counter 1 of chip 2 to Mode 0*/
  pokeb (base_id, 0x0107, 128+32+16); /* Programming Counter 2 of chip 2 to Mode 0 */

  pokeb (base_id, 0x0108, 0); /* Reset Low Byte of Counter 0 Chip 1 */
  pokeb (base_id, 0x0108, 0); /* Reset High Byte of Counter 0 Chip 1 */

  pokeb (base_id, 0x0109, 0); /* Reset Low Byte of Counter 1 Chip 1 */
  pokeb (base_id, 0x0109, 0); /* Reset High Byte of Counter 1 Chip 1 */

  pokeb (base_id, 0x010A, 0); /* Reset Low Byte of Counter 2 Chip 1 */
  pokeb (base_id, 0x010A, 0); /* Reset High Byte of Counter 2 Chip 1 */

  pokeb (base_id, 0x0106, 0); /* Reset Low Byte of Counter 2 Chip 2 */
  pokeb (base_id, 0x0106, 0); /* Reset High Byte of Counter 2 Chip 2 */

  /* %%%%%%%%%%%%%% Set Up Counters on the BB Carrier Board %%%%%%%%%%%%%% */
  pokeb (base_id, 0x0024,3); /* Sets up Counter1 as 16 bit Event counter */
  pokeb (base_id, 0x0034,3); /* Sets up Counter2 as 16 bit Event counter */
```

```
    pokeb (base_id, 0x0025,0); /* Sets non-inverted Gate Signal, Gate high for Count */
    pokeb (base_id, 0x0035,0);

    pokeb (base_id, 0x0026,0); /* Sends Start Measurement Command to Cntr' 1 */
    pokeb (base_id, 0x0036,0); /* Sends Start Measurement Command to Cntr' 2 */

    pokeb (base_id, 0x002E,0); /* Resets Counter1 to zero */
    pokeb (base_id, 0x003E,0); /* Resets Counter2 to zero */
    } /* End of bb_init */
```

FILE 'BG.H'

```
/* ------------------------ HEADER FILE BG.H ------------------------ */
/* ---- Privides the Control Functionality of the Burst Generator ---- */
/* ------------------------------------------------------------------- */


/* ------ BG_START sets the burst generator to ------- */
/* ---- a square wave output of a given frequency ---- */

void bg_start(int freq)
{
        unsigned char high, low ;
        unsigned int value;
        int incr;

        int flags;
        value = 8e6 / freq;
        high = (floor(value/256));
        low = (char)(value) - (high * 256);
        pokeb (base_id, 0x0060, low);  /* Division Ratio from 8MHz clock */
        pokeb (base_id, 0x0061, high); /* High Byte of Division Ratio */
        pokeb (base_id, 0x0062, 0);
        pokeb (base_id, 0x0063, 0);

       pokeb (base_id, 0x0064, 1);    /* Gap between Pulses in the same burst (in
units of 125ns) */
       pokeb (base_id, 0x0065, 0);   /* High Byte of Pulse Gap */

       pokeb (base_id, 0x0066, 1);   /* Pulses per Burst */
       pokeb (base_id, 0x0067, 1);     /* Mode of Burst Generator (synchronous
interrupt driven or whatever) */

       }

/* ------ BG_STOP Switches off the output of the Burst Generator ------ */

void bg_stop(void) {
  pokeb (base_id, 0x0067, 0);   /* Writing 0 to Mode Reg. Stops BG */
  }
```

FILE 'HANDIO.H'

```
/* ---------------------- HEADER FILE HANDIO.H ---------------------- */
/* - Provides Functionality of the Handshaking lines on the BB Board - */
/* ------------------------------------------------------------------ */


/* Returns the status of the STBA Handshaking line                    */
/* DIO_WRITE must have been used to initialise the IO and Handshaking  */
/* Before this will Work */

int stba(void) {
  int flags;
  flags = peekb (base_id, 0x000B);
  if (flags & 16)
    return(1);
  else
    return(0);
}
```

FILE 'DIO.H'

```
/* ------------------------ HEADER FILE DIO.H ------------------------ */
/* --- Provides the required functionality of the digital I/O port --- */
/* ------------------------------------------------------------------- */


/* --- The Function dio_write writes --- */
/* --- out a number to the two 8 bit --- */
/* ---    Digital I/O lines on the   --- */
/* ---         carrier board.        ---*/

void dio_write(int value) {
  unsigned high, low ;

  high = (floor(value/256));
  low = (char)(value) - (high * 256);

  pokeb (base_id, 0x000B,128+32+8); /* Set up for A as OP, B as OP */
                                    /* .... With Handshaking */
  pokeb (base_id, 0x0009, high);    /* Write high 8 bits */
  pokeb (base_id, 0x0008, low);     /* Write low 8 bits */
  }
```

FILE 'DAC.H'

```
/* ------------------------ HEADER FILE DAC.H ------------------------ */
/* - Provides software control of the Digital-to-Analogue converters - */
/* ------------------------------------------------------------------- */

/* Function dac_out converts a floating point number to the high and  */
/* parts of a 16 bit digital number and sends them to the DAC board   */

void dac_out(double voltage, int channel)
{
       unsigned char high, low ;
       unsigned int value;

       value = value / 2.0;
       value = floor((voltage - 5.0) * 6553.4); /* Gives the required number to load
the DAC */

       high = (floor(value/256));     /* Strips the top 8 bits of the 16 bit word */
       low = (char)(value) - (high * 256);  /* Strips the low 8 bits of 16 bit word */

       /* See p6 of DAC manual for these addresses */

       if (channel == 3) {
         pokeb (base_id2,0x0315,low);     /*  Write low bit */
         pokeb (base_id2, 0x0316, high); /* Write High bit */
         pokeb (base_id2, 0x031B, 12);   /* Strobe DAC */
         }
       if (channel == 2) {
         pokeb (base_id2,0x030D,low);     /*  Write low bit */
         pokeb (base_id2, 0x030E, high); /* Write High bit */
         pokeb (base_id2, 0x031B, 12);   /* Strobe DAC */
         }
       if (channel == 1) {
         pokeb (base_id,0x0215,low);      /* Write low bit to DAC */
         pokeb (base_id, 0x0216, high);   /* Write high bit to DAC */
         pokeb (base_id, 0x021B, 12);     /* Strobe the DAC */
         }
       if (channel == 0) {
         pokeb (base_id,0x020D,low);     /*  Write low bit */
         pokeb (base_id, 0x020E, high); /* Write High bit */
         pokeb (base_id, 0x021B, 12);   /* Strobe DAC */
         }

       }
```

APPENDIX B

GRAPHICAL USER INTERFACE SOFTWARE

The following source code listings form the basis of the graphical user interface of the new six-channel CMA control system.

FILE 'GUIHEAD.H'

This is the header file, declaring user interface functions and global structures.

```
/* ---------------------- HEADER FILE GUIHEAD.H ---------------------- */
/* -- Declares global constants and functions for the User Interface -- */
/* ------------------------------------------------------------------- */

/* -- Define Structures for Mouse Functions -- */

#define TRUE   -1
#define FALSE !TRUE

#define MOUSEFUNC int86(0x33,&regs,&regs);
#define LEFT      0x01
#define RIGHT     0x02
#define CENTRE    0x04

/* -- Function Declarations for Mouse Driver -- */

void restore_mouse(void);
void mouse_off(void);
void mouse_on(void);
void set_mouse_pos(int x,int y);
void set_mouse_x_limit(int minimum,int maximum);
void set_mouse_y_limit(int minimum,int maximum);
void mouse_cursor(char unsigned ANDcolour,
                  char unsigned XORcolour,
                  char unsigned ANDchar,
                  char unsigned XORchar   );

int mouse_x(void);
int mouse_y(void);
char unsigned get_mouse_button(void);

char signed mouse_setup(void);
char signed mouse_in_box(int x1,
                         int y1,
                         int x2,
                         int y2);


#define MOUSE 0x33
int mouse = 0;

/* -- Funcion Decalrations for Other GUI elements -- */

extern void initialise_graphics_environment(void);
void button(int status, int x1, int y1, int x2, int y2, char text[]);
```

FILE 'GUI.CPP'

```
/*-------------------------------------------------------------------------*/
/*------------ GUI.CPP, Graphical User Interface Functions ----------------*/
/*-------------------------------------------------------------------------*/


/*-----------------------Graphics Initialisation---------------------------*/

void initialise_graphics_environment(void) /* required to also set mouse up */
{
  int gdriver = DETECT, gmode, errorcode;

  initgraph(&gdriver, &gmode, "EGAVGA.BGI");
  cleardevice();
  errorcode = graphresult();
  if (errorcode != 0 )
  {
    printf("Graphics error: %s \n",grapherrormsg(errorcode));
    printf("Press any key to halt");
    getch();
    exit(1);
  }
}

/*--------------------------Function BUTTON--------------------------------*/
/*------- Draws a text button at the required co-ordindates--------*/
/*-------------------------------------------------------------------------*/

void button(int status, int x1,int y1,int x2,int y2,char text[]) {

  int text_x, text_y;
  text_x = (x1+x2)/2;
  text_y = (y1+y2)/2;
  mouse_off();
  if (status != 0) {
    settextjustify(CENTER_TEXT,CENTER_TEXT);
    if (status == 1) setfillstyle(1,WHITE);
    if (status == 2) setfillstyle(1,YELLOW);
    if (status == 3) setfillstyle(1,CYAN);
    setcolor(GREEN);
    bar3d(x1,y1,x2,y2,3,3);
    setcolor(BLUE);
    outtextxy(text_x, text_y,text);
    } else {
    setcolor(WHITE);
    outtextxy(text_x, text_y,text);
    }

    mouse_on();

  }

/*--- ---------------------- MOUSE FUNCTIONS ----------------------*/

/*-------------------------------------------------------------------------*/

/* -- Setting up the Mouse Interrupts -- */

char signed mouse_setup(void)
{
 union REGS regs;
 char signed present;

 regs.x.ax=0x0000;
 MOUSEFUNC
 present=regs.x.ax;

 regs.h.ah=0x0f;
 int86(0x10,&regs,&regs);

 regs.h.bl=0x00;
 regs.x.ax=0x001d;
 MOUSEFUNC
```

```
 return(present ? TRUE:FALSE);
}

/*-------------------------------------------------------------------------*/

/* -- Function to return the mouse 'x' co-ordinate -- */

int mouse_x(void)
{
 union REGS regs;

 regs.x.ax=0x0003;
 MOUSEFUNC

 return(regs.x.cx);
}

/*-------------------------------------------------------------------------*/

/* -- Function to return the mouse 'y' co-ordinate -- */

int mouse_y(void)
{
 union REGS regs;

 regs.x.ax=0x0003;
 MOUSEFUNC

 return(regs.x.dx);
}

/*-------------------------------------------------------------------------*/

/* -- Replaces mouse pointer on Screen -- */

void restore_mouse(void)
{
 union REGS regs;

 regs.x.ax=0x0003;
 MOUSEFUNC
 regs.x.ax=0x0004;
 MOUSEFUNC
}

/*-------------------------------------------------------------------------*/

/* -- Switches off the mouse Pointer -- */

void mouse_off(void)
{
 union REGS regs;

 regs.x.ax=0x0002;
 MOUSEFUNC
}

/*-------------------------------------------------------------------------*/

/* -- Switches on the mouse Pointer -- */

void mouse_on(void)
{
 union REGS regs;

 regs.x.ax=0x0001;
 MOUSEFUNC
}

/*-------------------------------------------------------------------------*/

/* -- Forces the mouse pointer to appear at a given position -- */

void set_mouse_pos(int x,int y)
{
```

```
  union REGS regs;

  regs.x.ax=0x0004;
  regs.x.cx=x;
  regs.x.dx=y;
  MOUSEFUNC
 }

 /*-------------------------------------------------------------------------*/

 /* -- Limits movement of the mouse in the 'x' direction -- */

 void set_mouse_x_limit(int minimum,int maximum)
 {
  union REGS regs;

  regs.x.ax=0x0007;
  regs.x.cx=minimum;
  regs.x.dx=maximum;
  MOUSEFUNC
 }

 /*-------------------------------------------------------------------------*/

 /* -- Limits movement of the mouse in the 'y' direction -- */

 void set_mouse_y_limit(int minimum,int maximum)
 {
  union REGS regs;

  regs.x.ax=0x0008;
  regs.x.cx=minimum;
  regs.x.dx=maximum;
  MOUSEFUNC
 }

 /*-------------------------------------------------------------------------*/

 /* -- Sets attribtes of the mouse pointer -- */

 void mouse_cursor(char unsigned ANDcolour,
                   char unsigned XORcolour,
                   char unsigned ANDchar,
                   char unsigned XORchar   )
 {
  union REGS regs;

  regs.x.ax=0x000a;
  regs.x.bx=0x0000;
  regs.x.cx=ANDchar+(ANDcolour<<8);
  regs.x.dx=XORchar+(XORcolour<<8);
  MOUSEFUNC
 }

 /*-------------------------------------------------------------------------*/

 /* -- Allows detection of mouse position within a given box area -- */

 char signed mouse_in_box(int x1,
                          int y1,
                          int x2,
                          int y2)
 {
  int x=mouse_x(),y=mouse_y();

  return((x<=x2 && x>=x1 && y<=y2 && y>=y1) ?TRUE:FALSE);
 }

 /*-------------------------------------------------------------------------*/

 /* -- Determination of Mouse button status -- */

 char unsigned get_mouse_button(void)
 {
  union REGS regs;
```

```
 regs.x.ax=0x0003;
 MOUSEFUNC

 return(regs.x.bx&7);
}

/*--------------------------------------------------------------------------*/

void chk_mouse()

{
 union REGS regs;

 struct SREGS sregs;
 regs.x.ax = 0x3533;
 intdosx(&regs,&regs,&sregs);
 if ((regs.x.bx | sregs.es) == 0) {
      closegraph();
      }
 regs.x.ax = 0;
 int86(MOUSE,&regs,&regs);
 if (regs.x.ax != 0) {
     mouse = regs.x.bx;
     regs.x.ax = 0x01;
     int86(MOUSE,&regs,&regs);
     }
}
}
```

FILE 'IMAGE.CPP'

```
/* ---------------------------- FILE IMAGE.CPP ----------------------------*/
/* ------------ Contains Functions related to all aspects of imaging ----*/
/* ------------ with the 6 channel CMA and SEM detector ---------------- */
/* ------------------------------------------------------------------------ */

/* ---------------------------- Function image() ----------------------------*/
/* ------------ SEM and 6 channel Auger Imaging Software ---------------- */
void image(void) {
  sem();
  } /* end of function image() */


/* ---------------------- Function get_adc_value() ---------------------- */
/* - Returns the current voltage on the analogue-to-digital converter -- */
/* ------------------------------------------------------------------------ */
int get_adc_value(void) {
  unsigned char low, high;
  int adc_value;
  pokeb (base_id, 0x0014,0); /* Clears overrun Flag */
  pokeb (base_id, 0x0005,0); /* Clears eoc Flag */
  pokeb (base_id, 0x0011,0); /* Starts Conversion */
  low = peekb (base_id, 0x0012); /* Get low part of 12 bit word */
  high = peekb (base_id, 0x0013);/* Get high part of 12 bit word */
  adc_value = low + (256*high); /* Calculate true ADC result */
  return(adc_value); /* pass back result */
  } /* end function get_adc_value */

/* ------------------------------------------------------------------------ */
/* ---------------------- Function SEM ------------------------------------ */
/* ---- Provides the functionality of the Computer controlled SEM ----- */
/* ------------------------------------------------------------------------ */
int sem(void) {

  /* -- Variable declarations -- */
  unsigned char status, eocflag;
  int incr, adc_value;
  int color, current_col, cross_col;
  int s,i,i2, exitflag, exitflag2;
  double m, zfactor = 1, offset = 0;
  int yincr, xincr, strobe, samflag = 0, bigflag = 0;
  int ymax, xmax, xpos, ypos;
```

```
  double yfactor, xfactor;
  double yout, xout;
  int step = 1;
  int line1[13], line2[13], lincr;

  /* -- Define structures -- */
  LINE *line_data;
  BOX *box_data;

  /* -- Set up the Screen -- */
  mouse_off();
  setfillstyle(11,MAGENTA);
  setcolor(CYAN);
  bar3d(180,90,570,330,3,3);
  setcolor(GREEN);
  line(200,200,550,200);
  line(200,201,550,201);
  line(200,270,550,270);
  line(200,271,550,271);
  button(1,315,100,465,120,"Large / Fast Scan");
  button(1,315,125,465,145,"Small Scan");
  button(1,315,150,465,170,"Large / Slow Scan");
  button(2,440,220,550,240,"Palette");
  setcolor(CYAN);
  line(344,230,440,230);
  line(344,231,440,231);
  line(344,229,440,229);
  line(435,225,440,230);
  line(435,235,440,230);

  /* -- Provide Cycling Options for Display Mode -- */
  if (colourflag == 0) {
    button(1,190,220,340,240,"Switch to Colour");
    button(1,440,242,550,262,"Grey Scale");
    }
  if (colourflag == 1) {
    button(1,190,220,340,240,"Switch to Thermal");
    button(1,440,242,550,262,"Colour Scale");
    }
  if (colourflag == 2) {
    button(1,190,220,340,240,"Switch to Grey");
    button(1,440,242,550,262,"Thermal Scale");
    }
 button(1,315,290,465,310,"Cancel");
 mouse_on();

 /* -- Poll User input to select Current Display Mode -- */
 exitflag = 0;
 while (exitflag == 0) {
   if ( (mouse_in_box(315,290,465,310) == TRUE) && (get_mouse_button() == LEFT) ) {
     return(0);
     }
   if ( (mouse_in_box(190,220,340,240) == TRUE) && (get_mouse_button() == LEFT) ) {
     if (colourflag == 0) { /* Start on Mono - go to Colour */
       colourflag = 1;
       button(0,190,220,340,240,"Switch to Colour");
       button(1,190,220,340,240,"Switch to Thermal");
       button(0,440,242,550,262,"Grey Scale");
       button(1,440,242,550,262,"Colour Scale");
       while(get_mouse_button() == LEFT);
       }
     else if (colourflag == 1) { /* Start on Colour, go to Thermal */
       colourflag = 2;
       button(0,190,220,340,240,"Switch to Thermal");
       button(1,190,220,340,240,"Switch to Grey");
       button(0,440,242,550,262,"Colour Scale");
       button(1,440,242,550,262,"Thermal Scale");
       while(get_mouse_button() == LEFT);
       }
     else if (colourflag == 2) { /* Start on Thermal, go to Mono */
       colourflag = 0;
       button(0,190,220,340,240,"Switch to Grey");
       button(1,190,220,340,240,"Switch to Colour");
       button(0,440,242,550,262,"Thermal Scale");
       button(1,440,242,550,262,"Grey Scale");
       while(get_mouse_button() == LEFT);
```

```
          }
        }
       if ( (mouse_in_box(315,100,465,120) == TRUE) && (get_mouse_button() == LEFT) ) {
         xmax = 300;
         ymax = 220;
         step = 2;
         exitflag = 1;
         }
       if ( (mouse_in_box(315,125,465,145) == TRUE) && (get_mouse_button() == LEFT) ) {
         xmax = 120;
         ymax = 100;
         exitflag = 1;
         }
       if ( (mouse_in_box(315,150,465,170) == TRUE) && (get_mouse_button() == LEFT) ) {
         xmax = 500;
         ymax = 320;
         bigflag = 1;
         exitflag = 1;
         }
       if ( (mouse_in_box(315,150,465,170) == TRUE) && (get_mouse_button() == LEFT) ) {
         xmax = 500;
         ymax = 320;
         bigflag = 1;
         exitflag = 1;
         }
       } /* end of While loop */

    mouse_off();
    cleardevice();
    mouse_on();


    palette_setup(1);
    /* -- If a Large Scan is displayed, Show Options for Auger studies -- */
    if (bigflag == 1) {
      button(1,50,380,180,400,"Auger Line Scan");
      button(1,50,405,180,425,"Auger Imaging");
      button(1,50,430,180,450,"Fixed Auger");
      }
    button(1,570,430,630,450,"Abort");
    pokeb (base_id, 0x0071,0); /* Sets software jumpers to none DMA ADC software start
 */
    pokeb (base_id, 0x0080,8); /* Sets up analogue input channel 0 */
    status = peekb (base_id, 0x0010); /* Read Status register */
    status = (status && (64 + 32)) + 8;
    pokeb (base_id, 0x0010, status); /* sets ADC mode to normal, single ended 0-10V */

    /* -- Poll User input for Selection of Auger Imaging or Point analysis -- */
    mouse_on();
    set_mouse_pos(620,460);
    exitflag = 0;
    set_mouse_y_limit(370,480);
    while (exitflag == 0) {
      for (yincr = 0; yincr < ymax; yincr = yincr + step) {
        if (  (mouse_in_box(570,430,630,450) == TRUE) && (get_mouse_button() == LEFT) )
{
         exitflag = 1;
         yincr = ymax;
         }
       if (  (mouse_in_box(50,430,180,450) == TRUE) && (get_mouse_button() == LEFT) &&
(bigflag == 1) ) {
         exitflag = 1;
         samflag = 1;
         button(1,200,405,400,425,"Please Wait a Moment...");
         }
       if (  (mouse_in_box(50,380,180,400) == TRUE) && (get_mouse_button() == LEFT) &&
(bigflag == 1) ) {
         exitflag = 1;
         samflag = 2;
         button(1,200,405,400,425,"Please Wait a Moment...");
         }
       if (  (mouse_in_box(50,405,180,425) == TRUE) && (get_mouse_button() == LEFT) &&
(bigflag == 1) ) {
         exitflag = 1;
         samflag = 3;
         button(1,200,405,400,425,"Please Wait a Moment...");
```

```
        }
      yfactor =  5.0 / (double)(ymax);
      yout  = 2.5 + ( (double)(yincr) * yfactor );
      dac_out(yout,3);
      xfactor = 5.0 / (double)(xmax);
      for (xincr = 0; xincr < xmax; xincr = xincr + step) {
        xout = 2.5 + ( (double)(xincr) * xfactor );
        dac_out(xout,2);
        adc_value = get_adc_value();
        color =  (int)(floor( (double)((zfactor*adc_value) + offset) * (16.0/4096.0) )
);
        putpixel(xincr+ 50, yincr + 50, color);

              } /* end of X loop */
      } /* end of Y loop */
    } /* end of While loop */

      set_mouse_y_limit(0,480);

/* ------------------------------------------------------------------ */
/* ------------ This section deals with Fixed Point Auger ------------*/
/* ------------------------------------------------------------------ */

    if (samflag == 1) {
        exitflag = 0;
        button(0,200,405,400,425,"Please Wait a Moment...");
        button(1,200,405,400,425,"Select point with Mouse");
        while (exitflag == 0) {
          if (get_mouse_button() == LEFT) {
            xpos = mouse_x();
            ypos = mouse_y();
               /*-- Check mouse is in Picture Range before allowing Selection -- */
            if ( (xpos < 550) && (xpos > 50) && (ypos < 370) && (ypos > 50) ) {
              button(0,200,405,400,425,"Select point with Mouse");
              button(1,200,405,400,425,"Is this Ok ? |Yes| |No|");
              cross(xpos, ypos, line1, line2, 1);
              exitflag2 = 0;
              while (exitflag2 == 0) {
                /* -- Exit loop -- */
                if ( (mouse_in_box(300,405,350,425) == TRUE) && (get_mouse_button() ==
LEFT) ) {
                  exitflag = 1;
                  exitflag2 = 1;
                  }
                /* -- Place Cross on Screen, verify for correct selection -- */
                if ( (mouse_in_box(350,405,400,425) == TRUE) && (get_mouse_button() ==
LEFT) ) {
                  cross(xpos, ypos, line1, line2, 0);
                  button(0,200,405,400,425,"Is this Ok ? |Yes| |No|");
                  exitflag2 = 1;
                  button(1,200,405,400,425,"Select point with Mouse");
                  }
                } /* end While exitflag2 == 0 */
              } /* end if Mouse is in picture range */
            } /* end of Mouse button = LEFT */
          } /* end of exitflag bit */

          mouse_off();
          button(0,200,405,400,425,"Is this Ok ? |Yes| |No|");
          button(1,200,405,400,425,"Ok, Click here to finish");
          mouse_on();
          /* -- Checking Selected point ok -- */
          while(get_mouse_button() == LEFT);
          exitflag2 = 0;
          while (exitflag2 == 0) {
           if ( (mouse_in_box(200,405,400,425) == TRUE) && (get_mouse_button() == LEFT)
) exitflag2 = 1;
            }
       /* -- Work out x and y Voltage for shift During AES -- */
          yfactor =  5.0 / (double)(ymax);
          y_voltage  = 2.5 + ( (double)(ypos - 50) * yfactor );

          xfactor = 5.0 / (double)(xmax);
          x_voltage = 2.5 + ( (double)(xpos - 50) * xfactor );
        }  /* end of samflag == 1 bit */
/* --------------  End of Single point Auger Routine --------------*/
```

```
/* --------------   Next Part Deals with Linescans  ---------------*/

    if (samflag == 2) {
         get_line(line_data, line1, line2, 0);
         get_line(line_data, line1, line2, 1);
         initialise_graphics_environment();
         mouse_on();
         linescan(line_data);
         } /* end of Linescan */

/* ----------- This section selects a box for SAM Imaging --------- */

    if (samflag == 3) {
         get_box(box_data); /* get the box parameters required */
         sam(box_data); /* go for the sam Image ! */
         }

/* -- Reset the palette before returning to main menu -- */
    initialise_graphics_environment();
    mouse_on();
    return(1);
  } /* end of Function sem() */


/* ----------------------- Function palette_setup ----------------------- */
/* ------------------ Selects Palette for SEM Display -------------------- */
/* ---------------------------------------------------------------------- */

void palette_setup(int flag) {
  int i,m,r,g,b;
  int current_col;
  struct palettetype pal;
  int RED1 [16]   = { 0, 0, 0, 0, 0, 0, 5,10,20,30,40,50,60,60,60,60};
  int GREEN1 [16] = { 0, 0, 5, 5,20,25,30,35,30,25,15,10,30,45,50,60};
  int BLUE1 [16]  = {10,20,30,40,35,30,20,10, 0, 0, 0, 0,10,20,30,55};
  int RED_T [16]   = {20,35,45,55,60,60,60,60,60,60,60,60,60,60,60,60};
  int GREEN_T [16] = {10,10,12,14,18,23,30,36,42,45,49,52,55,57,59,60};
  int BLUE_T [16]  = { 0, 0, 0, 0, 1, 5,10,13,17,20,23,27,34,42,50,55};

  /* grab a copy of the palette */

  getpalette(&pal);
  if (colourflag == 0) {
    /* create gray scale */
    m=3.5;
    for (i=0; i<15; i++) {
       r=( (int)(floor( (double)(i)*(double)(m) ) ) +7);
       g=( (int)(floor( (double)(i)*(double)(m) ) ) +7);
       b=( (int)(floor( (double)(i)*(double)(m) ) ) +7);
       setrgbpalette(pal.colors[i], r,g,b);
       }
    } /* end if colour flag == 0 */
  if (colourflag == 1){
       /* Set colours for Colour Scale */
       for(i=0;i<16;i++)
       setrgbpalette(pal.colors[i], RED1[i], GREEN1[i], BLUE1[i]);

       } /* end if colourflag == 1 */

  if (colourflag == 2){
       /* Set colours for Thermal Scale */
       for(i=1;i<16;i++)
       setrgbpalette(pal.colors[i], RED_T[i], GREEN_T[i], BLUE_T[i]);
       } /* end if colourflag == 2 */
  if (flag == 1) {
  current_col = 0;
    for (i = 50; i < 307; i = i + 16) {
      setfillstyle(SOLID_FILL, current_col);
      bar(570,i,620,i+16);
      current_col++;
      }
    }
  if (flag == 0) {
    current_col = 0;
    for (i = 260; i < 421; i = i + 10) {
```

```
      setfillstyle(SOLID_FILL, current_col);
      bar(580,i,630,i+10);
      current_col++;
      }
    }
  } /* end of palette_setup */


/* ------------------------ Function cross ----------------------------- */
/* ------- Draws a Cross on the Screen to Mark selected Point ---------- */
/* --------------------------------------------------------------------- */
void cross(int xpos, int ypos, int *line1, int *line2, int flag) {
  int cross_col; /* -- Colour of the Cross -- */
  int lincr;     /* -- Incremental variable for line -- */

  mouse_off();
  if (flag == 1) {
    if (colourflag == 2) {
      cross_col = 0;
      } else cross_col = 15;
      for (lincr = 0; lincr < 13; ++lincr) {
      line1[lincr] = getpixel(xpos-6+lincr, ypos-6+lincr);
      line2[lincr] = getpixel(xpos-6+lincr, ypos+6-lincr);
      putpixel(xpos-6+lincr, ypos-6+lincr,cross_col);
      putpixel(xpos-6+lincr, ypos+6-lincr,cross_col);
      }
    } /* end if flag == 1 */
  if (flag == 0) {
     for (lincr = 0; lincr < 13; ++lincr) {
     putpixel(xpos-6+lincr, ypos-6+lincr,line1[lincr]);
     putpixel(xpos-6+lincr, ypos+6-lincr,line2[lincr]);
     }
   } /* end if flag == 0 */
  mouse_on();
  } /* end function cross */


/* ------------------------- Function get_line ------------------------- */
/* - Allows the analyst to define a line along which to perform --------- */
/* ------------------  sucessive Auger experiments --------------------- */
void get_line(LINE *line_data,int *line1,int *line2, int flag) {
  int exitflag, exitflag2;
  int xpos, ypos;
  int line3[600];
  int no_pixels, col;
  int incr;
  int xdiff, ydiff;
  double xstep, ystep;
  double x,y;

  if (colourflag == 2) col = 0; else col = 15;
  button(0,200,405,400,425,"Please Wait a Moment..."); /*-- Remove this box -- */

  if (flag == 0) button(1,200,405,400,425,"Point to Start Position"); else
    button(1,200,405,400,425,"Point to End Position"); /* -- Prompt User -- */
  exitflag = 0;
  while (exitflag == 0) {
    if (get_mouse_button() == LEFT) {
      xpos = mouse_x();
      ypos = mouse_y();
      if ( (xpos < 550) && (xpos > 50) && (ypos < 370) && (ypos > 50) ) {
        if (flag == 0) button(0,200,405,400,425,"Point to Start Position"); else
          button(0,200,405,400,425,"Point to End Position");
          button(1,200,405,400,425,"Is this Ok ? |Yes| |No|");
          cross(xpos, ypos, line1, line2, 1);
        if (flag == 1) {
          xdiff = xpos - line_data->x1;
          ydiff = ypos - line_data->y1;
          if (abs(xdiff) > abs(ydiff)) no_pixels = abs(xdiff); else no_pixels =
abs(ydiff);
          xstep = (double)(xdiff) / (double)(no_pixels);
          ystep = (double)(ydiff) / (double)(no_pixels);
          mouse_off();
          for (incr = 0; incr < no_pixels; incr++) {
            x = line_data->x1 + (int)floor((double)(incr)*xstep);
```

```
            y = line_data->y1 + (int)floor((double)(incr)*ystep);
            line3[incr] = getpixel(x,y);
            putpixel(x,y,col);
            }
          mouse_on();
          } /* end if flag == 1 */
        exitflag2 = 0;
        while (exitflag2 == 0) {
        if ( (mouse_in_box(300,405,350,425) == TRUE) && (get_mouse_button() == LEFT) )
{
          exitflag = 1;
          exitflag2 = 1;
          }
        if ( (mouse_in_box(350,405,400,425) == TRUE) && (get_mouse_button() == LEFT) )
{
           if (flag == 1) {
            mouse_off();
              for (incr = 0; incr < no_pixels; incr++) {
              x = line_data->x1 + (int)floor((double)(incr)*xstep);
              y = line_data->y1 + (int)floor((double)(incr)*ystep);
              putpixel(x,y,line3[incr]);
              }
              mouse_on();
            } /* end if flag == 1 */

          cross(xpos, ypos, line1, line2, 0);
          button(0,200,405,400,425,"Is this Ok ? |Yes| |No|");
          exitflag2 = 1;
          if (flag == 0) button(1,200,405,400fn25,"Point to Start Position"); else
            button(1,200,405,400,425,"Point to End Position");
          mouse_on();
          }
        } /* end While exitflag2 == 0 */
      } /* end if Mouse is in picture range */
   } /* end of Mouse button = LEFT */
  } /* end of exitflag bit */


  button(0,200,405,400,425,"Is this Ok ? |Yes| |No|");
  /* -- Write Data for Start or End of line as appropriate -- */
  if (flag == 0) {
    line_data->x1 = xpos;
    line_data->y1 = ypos;
    } else {
     line_data->x2 = xpos;
     line_data->y2 = ypos;
     }

  } /* end of get_line() */


/* ------------------------ Function get_box() ------------------------- */
/* -------- Defines Area for SAM Imaging by 'Dragging' a box on Screen -- */
/* ----------------------------------------------------------------------- */

void get_box(BOX *box_data) {
  int col;
  int exitflag, exitflag2;
  int mx, my, mox,moy,x,y, lmx , lmy;
  int startx, starty;
  int incr;
  int top_line[500], bottom_line[500], left_line[320], right_line[320];


  if (colourflag == 2) col = 0; else col = 15; /* - Set colour according to background
- */

  button(0,200,405,400,425,"Please Wait a Moment...");
  /* -- Prompt User for Selection -- */
  button(1,200,405,400,425,"Set Top Left Position");


  /* -- 'Drag' out the box -- */
  exitflag = 0;
  while (exitflag == 0) {
    if ( (mouse_in_box(50,50,550,370) == TRUE) && (get_mouse_button() == LEFT) ) {
```

```
          button(0,200,405,400,425,"Set Top Left Position");
          button(1,200,405,400,425,"Hold button & Drag BOX");
       /* -- Define start Position --*/
          mx = mouse_x();
          my = mouse_y();
          set_mouse_x_limit(mx+1,550);
          set_mouse_y_limit(my+1,370);
          startx = mx;
          starty = my;
          box_data->tlx = mx;
          box_data->tly = my;
          mox = mx;
          moy = my;
          lmx = 0;
          lmy = 0;
          /* - Drag out the box -- */
          while (get_mouse_button() == LEFT) {
            mx = mouse_x();
            my = mouse_y();
            if ((mx > startx) && (my > starty) && ((mox != mx) || (moy != my)) )  { /*
if we have moved in the positive X direction */
              if ( (lmx != 0) || (lmy != 0) ) {
                for (incr = 0; incr < abs(lmx - startx); incr++) {
                     x = startx + incr;
                     mouse_off();
                     putpixel(x,starty,top_line[incr]);
                     putpixel(x,lmy,bottom_line[incr]);
                     mouse_on();
                     }
                }
              if ( (lmx != 0) || (lmy != 0) ) {
                for (incr = 0; incr < abs(lmy - starty); incr++) {
                     y = starty + incr;
                     mouse_off();
                     putpixel(startx,y,left_line[incr]);
                     putpixel(lmx,y,right_line[incr]);
                     mouse_on();
                     }
                }
              for (incr = 0; incr < abs(my - starty); incr++) {
                lmx = mx;
                lmy = my;
                y = starty + incr;
                left_line[incr] = getpixel(startx,y);
                mouse_off();
                putpixel(startx,y,col);
                right_line[incr] = getpixel(mx,y);
                putpixel(mx,y,col);
                mouse_on();
                }
              for (incr = 0; incr < abs(mx - startx); incr++) {
                lmx = mx;
                lmy = my;
                x = startx + incr;
                top_line[incr] = getpixel(x,starty);
                mouse_off();
                putpixel(x,starty,col);
                bottom_line[incr] = getpixel(x,my);
                putpixel(x,my,col);
                mouse_on();
                }
              } /* end we have moved in +ve X direction */
              mox = mx;
              moy = my;
            } /* end while left button */
            button(0,200,405,400,425,"Hold button & Drag BOX");
            button(1,200,405,400,425,"Is this Ok ? |Yes| |No|");
            set_mouse_x_limit(0,640);
            set_mouse_y_limit(0,480);
            exitflag2 = 0;
         while (exitflag2 == 0) {
              if ( (mouse_in_box(300,405,350,425) == TRUE) && (get_mouse_button() ==
LEFT) ) {
                exitflag = 1;
                exitflag2 = 1;
                }
```

```
                if ( (mouse_in_box(350,405,400,425) == TRUE) && (get_mouse_button() ==
  LEFT) ) {
                  exitflag2 = 1;
                  for (incr = 0; incr < abs(lmx - startx); incr++) {
                  x = startx + incr;
                  mouse_off();
                  putpixel(x,starty,top_line[incr]);
                  putpixel(x,lmy,bottom_line[incr]);
                  mouse_on();
                  }
                 for (incr = 0; incr < abs(lmy - starty); incr++) {
                   y = starty + incr;
                   mouse_off();
                   putpixel(startx,y,left_line[incr]);
                   putpixel(lmx,y,right_line[incr]);
                   mouse_on();
                   }
                 button(0,200,405,400,425,"Is this Ok ? |Yes| |No|");
                 button(1,200,405,400,425,"Set Top Left Position");
                 }
               }

           box_data->brx = mx;
           box_data->bry = my;
           } /* end mouse in box and button left */
      } /* end exitflag */

   } /* end get_box() */


/* ------------------------- Function linescan() ---------------------- */
/* ---------------- Harness to Deal with Auger Linescans ---------------- */
/* ------------------------------------------------------------------------ */
void linescan(LINE *line_data) {
  int exitflag;
  int statusflag;

  cleardevice();
  fs_linescan(line_data);

  } /* end function linescan */


/* ----------------------- Function fs_linescan() --------------------- */
/* ---------------- Carries out a Full Spectrum Linescan ------------- */
/* ------------------------------------------------------------------- */

void fs_linescan(LINE *line_data) {
  int o_emin, o_emax, o_points, o_dwell;
  long int o_max_count;
  int incr;
  int scan_no;
  int status;
  double yfactor, xfactor;
  double y_shift, x_shift;
  double xstep, ystep;
  double xpos, ypos;

 o_emin = emin;
 o_emax = emax;
 o_points = points;
 o_dwell = dwell;
 o_max_count = max_count;
 emin = l_emin;
 emax = l_emax;
 points = l_points;
 dwell = l_dwell;
 max_count = l_max_count;

 /* -- Set current line number (max is 10, so wraps round at this point !) -- */
 current_line++;
 if (current_line == 10) current_line = 1;
 /* -- Call Functions to Set up Scan parameters -- */
 display_params(2);
 set_params(2);
```

```
    /* -- Work out the length of each step along the scan -- */
    xstep = (double)(line_data->x2 - line_data->x1) / (double)(linepoints);
    ystep = (double)(line_data->y2 - line_data->y1) / (double)(linepoints);

    /* -- Get starting position -- */
    xpos = (double)(line_data->x1);
    ypos = (double)(line_data->y1);

    /* -- Carry out linescan -- */
    for (incr = 0; incr < linepoints; incr++ ){
      scan_no = (current_line * 100) + incr;
      yfactor =  5.0 / 320.0;
      y_shift  = 2.5 + ((ypos - 50.0) * yfactor );
      xfactor = 5.0 / 500.0;
      x_shift = 2.5 + ((xpos - 50.0) * xfactor );
      dac_out(x_shift,2);
      dac_out(y_shift,3);
      status = grab_scan(scan_no,(incr+1),linepoints);
      if (status == 0) {
        current_line--;
        del_files( (current_line * 100), scan_no);
        incr = linepoints;
        }
      xpos = xpos + xstep;
      ypos = ypos + ystep;
      }

    l_emin = emin;
    l_emax = emax;
    l_points = points;
    l_dwell = dwell;
    l_max_count = max_count;
    emin = o_emin;
    emax = o_emax;
    points = o_points;
    dwell = o_dwell;
    max_count = o_max_count;
    dac_out(0.0,2);
    dac_out(0.0,3);
    if (current_line > max_line) max_line = current_line;
    } /* end of fs_linescan */



/* ----------------- Function line_display ---------------------- */
/* ------ Displays Auger Linescans as a set of offset spectra --- */
/* ------------------------------------------------------------- */

void line_display(void) {
  int scan_no;
  char text[20];
  int exitflag;

  scan_no = current_line;

  /* -- Set up Screen Environment -- */
  mouse_off();
  cleardevice();
  setfillstyle(11,MAGENTA);
  setcolor(CYAN);
  bar3d(180,90,450,230,3,3);
  setcolor(BLUE);
  button(1,240,240,390,260,"Continue");
  setcolor(YELLOW);
  settextjustify(CENTER_TEXT, CENTER_TEXT);
  outtextxy(315,110,"Linescan Display");
  outtextxy(315,120,"Parameters");
  setfillstyle(1,BLUE);
  bar(340,140,410,160);
  bar(340,175,410,195);
  setcolor(WHITE);
  outtextxy(330,150,"<<");
  outtextxy(420,150,">>");
  outtextxy(330,185,"<<");
  outtextxy(420,185,">>");
  setcolor(YELLOW);
```

```
    outtextxy(280,150,"Scan No.");
    outtextxy(257,185,"Detector Ch. No.");
    setcolor(YELLOW);
    sprintf(text,"%i",scan_no);
    outtextxy(375,150,text);
    sprintf(text,"%i",det_channel);
    outtextxy(375,185,text);
    mouse_on();

    exitflag = 0;
    while (exitflag == 0) {
      if ( (mouse_in_box(240,240,390,260) == TRUE) && (get_mouse_button() == LEFT) ) {
        exitflag = 1;
        }
      if ( (mouse_in_box(320,145,340,155) == TRUE) && (get_mouse_button() == LEFT) ) {
        if (scan_no > 1) {
         mouse_off();
         setcolor(BLUE);
         sprintf(text,"%i",scan_no);
         outtextxy(375,150,text);
         scan_no--;
         setcolor(YELLOW);
         sprintf(text,"%i",scan_no);
         outtextxy(375,150,text);
         mouse_on();
         }
       while(get_mouse_button() == LEFT);
       }

     if ( (mouse_in_box(410,145,430,155) == TRUE) && (get_mouse_button() == LEFT) ) {
       if (scan_no < max_line) {
        mouse_off();
        setcolor(BLUE);
        sprintf(text,"%i",scan_no);
        outtextxy(375,150,text);
        scan_no++;
        setcolor(YELLOW);
        sprintf(text,"%i",scan_no);
        outtextxy(375,150,text);
        mouse_on();
        }
      while(get_mouse_button() == LEFT);
      }

    if ( (mouse_in_box(320,180,340,190) == TRUE) && (get_mouse_button() == LEFT) ) {
      if (det_channel > 1) {
           mouse_off();
           setcolor(BLUE);
           sprintf(text,"%i",det_channel);
           outtextxy(375,185,text);
           det_channel--;
           setcolor(YELLOW);
           sprintf(text,"%i",det_channel);
           outtextxy(375,185,text);
           mouse_on();
           }
     while(get_mouse_button() == LEFT);
     }

    if ( (mouse_in_box(410,180,430,190) == TRUE) && (get_mouse_button() == LEFT) ) {
      if (det_channel < 6) {
           mouse_off();
           setcolor(BLUE);
           sprintf(text,"%i",det_channel);
           outtextxy(375,185,text);
           det_channel++;
           setcolor(YELLOW);
           sprintf(text,"%i",det_channel);
           outtextxy(375,185,text);
           mouse_on();
           }
     while(get_mouse_button() == LEFT);
     }
   } /* end exitflag == 0 */
exitflag = 0;
plot_line_scan(scan_no);
```

```
   while (exitflag == 0) {
     if ( (mouse_in_box(550,463,620,480) == TRUE) && (get_mouse_button() == LEFT) ) {
       exitflag = 1;
       }
     if ( (mouse_in_box(30,463,100,480) == TRUE) && (get_mouse_button() == LEFT) ) {
      while(get_mouse_button() == LEFT);
      if (plotflag == 0) {
        plotflag = 1;
        plot_line_scan(scan_no);
        } else {
         plotflag = 0;
         plot_line_scan(scan_no);
         }
     }
    }
   } /* --- End of Function line_display --- */



/* -------------------- Function plot_line_scan() ------------------------ */
/* ---------- Part of the linescan Display Function -------------------- */
/* -- Actually draws the scans on the screens, working out scaling etc. -- */
/* ------------------------------------------------------------------------ */

void plot_line_scan(int line_no) {
  FILE *ifp1, *ifp2, *ifp3, *ifp4, *ifp5, *ifp6;
  char fname[20];
  char text[20];
  int n, incr, i;
  long double a, lasta1, lasta2, lasta3, lasta4, lasta5, lasta6;
  long double amax, offset;
  long double number1, number2, number3, number4, number5, number6, divisor;
  long double buffer[3];
  long int count;
  int low_e, high_e;
  int no_points;
  int locator;
  int c_scan;
  int no_spectra;
  double ystep;
  int yoffset;
  int colorincr;
  int sem_result;


 /* -- Load up Files and Work out number of Scans -- */
 for (incr = 0; incr < 99; ++incr) {
   c_scan = ((100*line_no)+incr);
   sprintf(fname,"scan%i.%i",det_channel,c_scan);
   if ( (ifp1 = fopen(fname, "rt")) != NULL )
     no_spectra = incr + 1; else
     incr = 100;
   fclose(ifp1);
   }

 /* --  Work out the Number of points in each spectrum -- */
 amax = 0.0;
 for (i = 0; i < no_spectra; ++i) {
   c_scan = ((100*line_no)+i);
   sprintf(fname,"scan%i.%i",det_channel,c_scan);
   ifp1 = fopen(fname, "rt");
   incr = 0;
   for (n=0; fscanf(ifp1,"%Lf",&a) == 1; ++n) {
     if (incr == 1) {
       if (a > amax) amax = a;
       }
     incr++;
     if(incr == 2) incr = 0;
     }
   fclose(ifp1);
   }

/* - Determine the minumum and maximum energies in the spectra - */
sprintf(fname,"scan1.%i",(100*line_no));
ifp1 = fopen(fname, "rt");
```

```
  incr = 0;
  for (n=0; fscanf(ifp1,"%Lf",&a) == 1; ++n) {
    if (n==0) low_e = floor(a);
    buffer[incr] = a;
    incr++;
    if(incr == 3) incr = 0;
    }
  if (incr == 2) locator = 0;
  if (incr == 1) locator = 2;
  if (incr == 0) locator = 1;
  high_e = floor(buffer[locator]);
  fclose(ifp1);


  /* -- Set up the Screen -- */
  count = (long int)(floor1(amax));
  if (count == 0) {
    count = 1;
    } else {
    count = (count + (floor((double)(count) * 0.1)) );
    }
  count = count*2; /* Double the Y axis to make room for spectra */
  no_points = (n/2);

  amax = 0.0;

  mouse_off();
  draw_axes(count,high_e,low_e);
  button(1,550,463,620,480,"Exit");
  if (plotflag == 0) {
    button(1,30,463,100,480,"Points");
    } else {
    button(1,30,463,100,480,"Lines");
    }
 ystep = (320.0 / (double)(no_spectra))/2.0;


 /* -- Load in the 6 data files and Display the spectra -- */
 if (plotflag == 0) {
 /* -- Plot as Lines -- */
 colorincr = 9;
 divisor = (double)(count)/420.0;
 for (i = 0; i < no_spectra; ++i) {
   setcolor(colorincr);
   colorincr++;
   if(colorincr == 16) colorincr = 9;
   c_scan = ((100*line_no) + i);
   sprintf(fname,"scan%i.%i",det_channel,c_scan);

   ifp1 = fopen(fname, "rt");

   for (incr = 0; incr <= no_points; ++incr) {
     offset = ( (560.0 / (double)(no_points)) * (double)(incr) );
     fscanf(ifp1,"%Lf",&number1);
     fscanf(ifp1,"%Lf",&number1);
     yoffset = ((double)(i)*(ystep));
     if (incr == 0) {
       moveto( (40+floor(offset)) , ((440-floor(number1/divisor)) - yoffset) );
       }
     lineto( (40+floor(offset)) , ((440-floor(number1/divisor)) - yoffset) );
     }

   fclose(ifp1);
   } /* end of 'i' loop for number of spectra */
 } else {
 /* -- Plot as points -- */
 colorincr = 8;
 divisor = (double)(count)/420.0;
 for (i = 0; i < no_spectra; ++i) {
   colorincr++;
   if(colorincr == 16) colorincr = 9;
   c_scan = ((100*line_no) + i);
   sprintf(fname,"scan%i.%i",det_channel,c_scan);

   ifp1 = fopen(fname, "rt");
```

```
      for (incr = 0; incr <= no_points; ++incr) {
        offset = ( (560.0 / (double)(no_points)) * (double)(incr) );
        fscanf(ifp1,"%Lf",&number1);
        fscanf(ifp1,"%Lf",&number1);
        yoffset = ((double)(i)*(ystep));
        if ((int)(floor1(number1)) < count) putpixel( (40+floor(offset)),(440-
 floor(number1/divisor) - yoffset), colorincr);
        }
      fclose(ifp1);
      } /* end of 'i' loop for number of spectra */
    }
    mouse_on();
  } /* end of plot_scan */


  /* ------------------------ Function SAM ---------------------------- */
  /* -- Performs scanning Auger Microscopy over a selected area -------- */
  /* -------------------------------------------------------------------- */

  void sam(BOX *box_data) {
    POINT *point_data;
    FILE *ofp1, *ofp2, *ofp3, *ofp4, *ofp5, *ofp6, *ofp7, *ofp8;
    int no_els = 1;
    int exitflag = 0;
    int eincr;
    double e1,e2,e3;
    int exitflag2;
    int strobe;
    int samx, samy;
    int ysize, xsize;
    char text[20], filename[10], fname[20];
    int temdwell;
    int xincr, yincr, n, i;
    int sem_result;
    double xstep, ystep;
    double xpos, ypos;
    double yfactor, xfactor;
    double y_shift, x_shift;
    double energies[12] = {-1.0,-1.0,-1.0,-1.0,-1.0,-1.0,-1.0,-1.0,-1.0,-1.0,-1.0,-1.0};
    long double buf;
    int percent;
    int old_percent = 0;

    temdwell = dwell;
    dwell = 200;
    samx = (box_data->brx) - (box_data->tlx);
    samy = (box_data->bry) - (box_data->tly);
    ysize = samy;
    xsize = samx;

    /* -- Set up Screen -- */
    closegraph(); /* -- Re-initialise the screen to reset palette -- */

    initialise_graphics_environment();
    setfillstyle(11,MAGENTA);
    settextjustify(CENTER_TEXT,CENTER_TEXT);
    setcolor(CYAN);
    bar3d(150,200,430,400,3,3);
    button(1,240,410,390,430,"Continue");
    setfillstyle(1,BLUE);
    bar(310,260,390,280);
    bar(310,290,390,310);
    bar(310,320,390,340);
    bar(310,350,390,370);
    setcolor(YELLOW);
    outtextxy(315,210,"Please Set Parameters");
    outtextxy(315,220,"For the SAM Image ?");
    sprintf(text,"%i",no_els);
    outtextxy(350,270,text);
    sprintf(text,"%i",samx);
    outtextxy(350,300,text);
    sprintf(text,"%i",samy);
    outtextxy(350,330,text);
    sprintf(text,"%i",dwell);
    outtextxy(350,360,text);
    settextjustify(RIGHT_TEXT,CENTER_TEXT);
```

```
  outtextxy(280,270,"No. Elements");
  outtextxy(280,300,"X - Pixels");
  outtextxy(280,330,"Y - Pixels");
  outtextxy(280,360,"Dwell Time");
  settextjustify(LEFT_TEXT,CENTER_TEXT);
  setcolor(WHITE);
  outtextxy(284,270,"<<");
  outtextxy(401,270,">>");
  outtextxy(284,300,"<<");
  outtextxy(401,300,">>");
  outtextxy(284,330,"<<");
  outtextxy(401,330,">>");
  outtextxy(284,360,"<<");
  outtextxy(401,360,">>");
  mouse_on();
  settextjustify(CENTER_TEXT,CENTER_TEXT);


  /* ----- Wait for User input ----- */
  /* -- This section uses offboard timing to allow smooth data input with the mouse --
*/

  exitflag = 0;
  while (exitflag == 0) {
    if ( (mouse_in_box(240,410,390,430) == TRUE) && (get_mouse_button() == LEFT) ) {
      exitflag = 1;
      }
    if ( (mouse_in_box(280,260,300,280) == TRUE) && (get_mouse_button() == LEFT) ) {
      setcolor(BLUE);
      sprintf(text,"%i",no_els);
      outtextxy(350,270,text);
      no_els--;
      if (no_els == 0) no_els = 1;
      setcolor(YELLOW);
      sprintf(text,"%i",no_els);
      outtextxy(350,270,text);
      while(get_mouse_button() == LEFT);
      }
    if ( (mouse_in_box(400,260,420,280) == TRUE) && (get_mouse_button() == LEFT) ) {
      setcolor(BLUE);
      sprintf(text,"%i",no_els);
      outtextxy(350,270,text);
      no_els++;
      if (no_els == 5) no_els = 4;
      setcolor(YELLOW);
      sprintf(text,"%i",no_els);
      outtextxy(350,270,text);
      while(get_mouse_button() == LEFT);
      }
    if ( (mouse_in_box(283,295,310,305) == TRUE) && (get_mouse_button() == LEFT)) {
      setcolor(BLUE);
      sprintf(text,"%i",samx);
      outtextxy(350,300,text);
      samx--;
      if (samx < 0) samx = 0;
      setcolor(YELLOW);
      sprintf(text,"%i",samx);
      outtextxy(350,300,text);
      if(get_mouse_button() == LEFT) {
        exitflag2 = 0;
        dio_write(7000);
        bg_start(10000);
        strobe = stba();
        while (strobe == 0) strobe = stba();
        while ( (exitflag2 == 0) && (strobe == 1) ) {
          strobe = stba();
          if (get_mouse_button() != LEFT) exitflag2 = 1;
          }
        bg_stop();
        while(get_mouse_button() == LEFT) {
          delay(D);
          setcolor(BLUE);
          sprintf(text,"%i",samx);
          outtextxy(350,300,text);
          samx--;
          if (samx < 0) samx = 0;
```

```
        setcolor(YELLOW);
        sprintf(text,"%i",samx);
        outtextxy(350,300,text);
        }
      bg_stop();
      }
    }
  if ( (mouse_in_box(283,325,310,335) == TRUE) && (get_mouse_button() == LEFT)) {
    setcolor(BLUE);
    sprintf(text,"%i",samy);
    outtextxy(350,330,text);
    samy--;
    if (samy < 0) samy = 0;
    setcolor(YELLOW);
    sprintf(text,"%i",samy);
    outtextxy(350,330,text);
    if(get_mouse_button() == LEFT) {
     exitflag2 = 0;
     dio_write(7000);
     bg_start(10000);
     strobe = stba();
     while (strobe == 0) strobe = stba();
     while ( (exitflag2 == 0) && (strobe == 1) ) {
       strobe = stba();
       if (get_mouse_button() != LEFT) exitflag2 = 1;
       }
     bg_stop();
     while(get_mouse_button() == LEFT) {
       delay(D);
       setcolor(BLUE);
       sprintf(text,"%i",samy);
       outtextxy(350,330,text);
       samy--;
       if (samy < 0) samy = 0;
       setcolor(YELLOW);
       sprintf(text,"%i",samy);
       outtextxy(350,330,text);
       }
     bg_stop();
     }
   }

 if ( (mouse_in_box(283,355,310,365) == TRUE) && (get_mouse_button() == LEFT)) {
   setcolor(BLUE);
   sprintf(text,"%i",dwell);
   outtextxy(350,360,text);
   dwell--;
   if (dwell < 0) dwell = 0;
   setcolor(YELLOW);
   sprintf(text,"%i",dwell);
   outtextxy(350,360,text);
   if(get_mouse_button() == LEFT) {
    exitflag2 = 0;
    dio_write(7000);
    bg_start(10000);
    strobe = stba();
    while (strobe == 0) strobe = stba();
    while ( (exitflag2 == 0) && (strobe == 1) ) {
      strobe = stba();
      if (get_mouse_button() != LEFT) exitflag2 = 1;
      }
    bg_stop();
    while(get_mouse_button() == LEFT) {
      delay(D);
      setcolor(BLUE);
      sprintf(text,"%i",dwell);
      outtextxy(350,360,text);
      dwell--;
      if (dwell < 0) dwell = 0;
      setcolor(YELLOW);
      sprintf(text,"%i",dwell);
      outtextxy(350,360,text);
      }
    bg_stop();
    }
  }
```

```
 if ( (mouse_in_box(400,295,410,305) == TRUE) && (get_mouse_button() == LEFT)) {
   setcolor(BLUE);
   sprintf(text,"%i",samx);
   outtextxy(350,300,text);
   samx++;
   if (samx > 500) samx = 500;
   setcolor(YELLOW);
   sprintf(text,"%i",samx);
   outtextxy(350,300,text);
   if(get_mouse_button() == LEFT) {
    exitflag2 = 0;
    dio_write(7000);
    bg_start(10000);
    strobe = stba();
    while (strobe == 0) strobe = stba();
    while ( (exitflag2 == 0) && (strobe == 1) ) {
      strobe = stba();
      if (get_mouse_button() != LEFT) exitflag2 = 1;
      }
    bg_stop();
    while(get_mouse_button() == LEFT) {
      delay(D);
      setcolor(BLUE);
      sprintf(text,"%i",samx);
      outtextxy(350,300,text);
      samx++;
      if (samx > 500) samx = 500;
      setcolor(YELLOW);
      sprintf(text,"%i",samx);
      outtextxy(350,300,text);
      }
   bg_stop();
   }
 }

if ( (mouse_in_box(400,325,410,335) == TRUE) && (get_mouse_button() == LEFT)) {
  setcolor(BLUE);
  sprintf(text,"%i",samy);
  outtextxy(350,330,text);
  samy++;
  if (samy > 500) samy = 500;
  setcolor(YELLOW);
  sprintf(text,"%i",samy);
  outtextxy(350,330,text);
  if(get_mouse_button() == LEFT) {
   exitflag2 = 0;
   dio_write(7000);
   bg_start(10000);
   strobe = stba();
   while (strobe == 0) strobe = stba();
   while ( (exitflag2 == 0) && (strobe == 1) ) {
     strobe = stba();
     if (get_mouse_button() != LEFT) exitflag2 = 1;
     }
   bg_stop();
   while(get_mouse_button() == LEFT) {
     delay(D);
     setcolor(BLUE);
     sprintf(text,"%i",samy);
     outtextxy(350,330,text);
     samy++;
     if (samy > 500) samy = 500;
     setcolor(YELLOW);
     sprintf(text,"%i",samy);
     outtextxy(350,330,text);
     }
   bg_stop();
   }
 }

if ( (mouse_in_box(400,355,410,365) == TRUE) && (get_mouse_button() == LEFT)) {
  setcolor(BLUE);
  sprintf(text,"%i",dwell);
  outtextxy(350,360,text);
  dwell++;
  if (dwell > 3000) dwell = 3000;
```

```
        setcolor(YELLOW);
        sprintf(text,"%i",dwell);
        outtextxy(350,360,text);
        if(get_mouse_button() == LEFT) {
         exitflag2 = 0;
         dio_write(7000);
         bg_start(10000);
         strobe = stba();
         while (strobe == 0) strobe = stba();
         while ( (exitflag2 == 0) && (strobe == 1) ) {
           strobe = stba();
           if (get_mouse_button() != LEFT) exitflag2 = 1;
           }
         bg_stop();
         while(get_mouse_button() == LEFT) {
           delay(D);
           setcolor(BLUE);
           sprintf(text,"%i",dwell);
           outtextxy(350,360,text);
           dwell++;
           if (dwell > 3000) dwell = 3000;
           setcolor(YELLOW);
           sprintf(text,"%i",dwell);
           outtextxy(350,360,text);
           }
        bg_stop();
          }
      }
     } /* end while exitflag == 0 */

   while(get_mouse_button() == LEFT);

   /* Get the Parameters for the SAM imaging --- up to Auger peaks --------- */

   for (eincr = 0; eincr < no_els; ++eincr) {

   mouse_off();
   cleardevice();
   setfillstyle(11,MAGENTA);
   settextjustify(CENTER_TEXT,CENTER_TEXT);
   setcolor(CYAN);
   bar3d(150,200,430,400,3,3);
   button(1,240,410,390,430,"Continue");
   setfillstyle(1,BLUE);
   bar(310,260,390,280);
   bar(310,290,390,310);
   bar(310,320,390,340);
   setcolor(YELLOW);
   outtextxy(290,210,"Please Set the Parameters");
   sprintf(text,"For element Number %i",eincr+1);
   outtextxy(290,220,text);
   e1 = 300.0;
   e2 = 300.0;
   e3 = 300.0;
   sprintf(text,"%g eV",e1);
   outtextxy(350,270,text);
   sprintf(text,"%g eV",e2);
   outtextxy(350,300,text);
   sprintf(text,"%g eV",e3);
   outtextxy(350,330,text);
   setcolor(WHITE);
   settextjustify(LEFT_TEXT,CENTER_TEXT);
   outtextxy(284,270,"<<");
   outtextxy(401,270,">>");
   outtextxy(284,300,"<<");
   outtextxy(401,300,">>");
   outtextxy(284,330,"<<");
   outtextxy(401,330,">>");

   settextjustify(RIGHT_TEXT,CENTER_TEXT);
   outtextxy(280,270,"Peak Energy");
   outtextxy(280,300,"Background (e1)");
   outtextxy(280,330,"Background (e2)");

   settextjustify(CENTER_TEXT,CENTER_TEXT);
   mouse_on();
```

```
/* ------------------------------------ */
/* --- User input of Auger Parameters --- */
/* ------------------------------------ */

exitflag = 0;
while(exitflag == 0) {
  if ( (mouse_in_box(240,410,390,430) == TRUE) && (get_mouse_button() == LEFT) ) {
    exitflag = 1;
    }

  if ( (mouse_in_box(283,265,310,275) == TRUE) && (get_mouse_button() == LEFT)) {
    setcolor(BLUE);
    sprintf(text,"%g eV",e1);
    outtextxy(350,270,text);
    e1--;
    if (e1 < 0) e1 = 0;
    setcolor(YELLOW);
    sprintf(text,"%g eV",e1);
    outtextxy(350,270,text);
    if(get_mouse_button() == LEFT) {
     exitflag2 = 0;
     dio_write(7000);
     bg_start(10000);
     strobe = stba();
     while (strobe == 0) strobe = stba();
     while ( (exitflag2 == 0) && (strobe == 1) ) {
       strobe = stba();
       if (get_mouse_button() != LEFT) exitflag2 = 1;
       }
     bg_stop();
     while(get_mouse_button() == LEFT) {
       delay(D);
       setcolor(BLUE);
       sprintf(text,"%g eV",e1);
       outtextxy(350,270,text);
       e1--;
       if (e1 < 0) e1 = 0;
       setcolor(YELLOW);
       sprintf(text,"%g eV",e1);
       outtextxy(350,270,text);
       }
     bg_stop();
     }
   }
 if ( (mouse_in_box(283,295,310,305) == TRUE) && (get_mouse_button() == LEFT)) {
   setcolor(BLUE);
   sprintf(text,"%g eV",e2);
   outtextxy(350,300,text);
   e2--;
   if (e2 < 0) e2 = 0;
   setcolor(YELLOW);
   sprintf(text,"%g eV",e2);
   outtextxy(350,300,text);
   if(get_mouse_button() == LEFT) {
    exitflag2 = 0;
    dio_write(7000);
    bg_start(10000);
    strobe = stba();
    while (strobe == 0) strobe = stba();
    while ( (exitflag2 == 0) && (strobe == 1) ) {
      strobe = stba();
      if (get_mouse_button() != LEFT) exitflag2 = 1;
      }
    bg_stop();
    while(get_mouse_button() == LEFT) {
      delay(D);
      setcolor(BLUE);
      sprintf(text,"%g eV",e2);
      outtextxy(350,300,text);
      e2--;
      if (e2 < 0) e2 = 0;
      setcolor(YELLOW);
      sprintf(text,"%g eV",e2);
      outtextxy(350,300,text);
      }
```

```
      bg_stop();
      }
    }
  if ( (mouse_in_box(283,325,310,335) == TRUE) && (get_mouse_button() == LEFT)) {
    setcolor(BLUE);
    sprintf(text,"%g eV",e3);
    outtextxy(350,330,text);
    e3--;
    if (e3 < 0) e3 = 0;
    setcolor(YELLOW);
    sprintf(text,"%g eV",e3);
    outtextxy(350,330,text);
    if(get_mouse_button() == LEFT) {
     exitflag2 = 0;
     dio_write(7000);
     bg_start(10000);
     strobe = stba();
     while (strobe == 0) strobe = stba();
     while ( (exitflag2 == 0) && (strobe == 1) ) {
       strobe = stba();
       if (get_mouse_button() != LEFT) exitflag2 = 1;
       }
     bg_stop();
     while(get_mouse_button() == LEFT) {
       delay(D);
       setcolor(BLUE);
       sprintf(text,"%g eV",e3);
       outtextxy(350,330,text);
       e3--;
       if (e3 < 0) e3 = 0;
       setcolor(YELLOW);
       sprintf(text,"%g eV",e3);
       outtextxy(350,330,text);
       }
     bg_stop();
     }
    }
 if ( (mouse_in_box(400,265,410,275) == TRUE) && (get_mouse_button() == LEFT)) {
   setcolor(BLUE);
   sprintf(text,"%g eV",e1);
   outtextxy(350,270,text);
   e1++;
   if (e1 > 3000.0) e1 = 3000.0;
   setcolor(YELLOW);
   sprintf(text,"%g eV",e1);
   outtextxy(350,270,text);
   if(get_mouse_button() == LEFT) {
    exitflag2 = 0;
    dio_write(7000);
    bg_start(10000);
    strobe = stba();
    while (strobe == 0) strobe = stba();
    while ( (exitflag2 == 0) && (strobe == 1) ) {
      strobe = stba();
      if (get_mouse_button() != LEFT) exitflag2 = 1;
      }
    bg_stop();
    while(get_mouse_button() == LEFT) {
      delay(D);
      setcolor(BLUE);
      sprintf(text,"%g eV",e1);
      outtextxy(350,270,text);
      e1++;
      if (e1 > 3000.0) e1 = 3000.0;
      setcolor(YELLOW);
      sprintf(text,"%g eV",e1);
      outtextxy(350,270,text);
      }
    bg_stop();
    }
  }
if ( (mouse_in_box(400,295,410,305) == TRUE) && (get_mouse_button() == LEFT)) {
  setcolor(BLUE);
  sprintf(text,"%g eV",e2);
  outtextxy(350,300,text);
  e2++;
```

```
      if (e2 > 3000.0) e2 = 3000.0;
      setcolor(YELLOW);
      sprintf(text,"%g eV",e2);
      outtextxy(350,300,text);
      if(get_mouse_button() == LEFT) {
       exitflag2 = 0;
       dio_write(7000);
       bg_start(10000);
       strobe = stba();
       while (strobe == 0) strobe = stba();
       while ( (exitflag2 == 0) && (strobe == 1) ) {
         strobe = stba();
         if (get_mouse_button() != LEFT) exitflag2 = 1;
         }
       bg_stop();
       while(get_mouse_button() == LEFT) {
         delay(D);
         setcolor(BLUE);
         sprintf(text,"%g eV",e2);
         outtextxy(350,300,text);
         e2++;
         if (e2 > 3000.0) e2 = 3000.0;
         setcolor(YELLOW);
         sprintf(text,"%g eV",e2);
         outtextxy(350,300,text);
         }
       bg_stop();
       }
     }

   if ( (mouse_in_box(400,325,410,335) == TRUE) && (get_mouse_button() == LEFT)) {
     setcolor(BLUE);
     sprintf(text,"%g eV",e3);
     outtextxy(350,330,text);
     e3++;
     if (e3 > 3000.0) e3 = 3000.0;
     setcolor(YELLOW);
     sprintf(text,"%g eV",e3);
     outtextxy(350,330,text);
     if(get_mouse_button() == LEFT) {
      exitflag2 = 0;
      dio_write(7000);
      bg_start(10000);
      strobe = stba();
      while (strobe == 0) strobe = stba();
      while ( (exitflag2 == 0) && (strobe == 1) ) {
        strobe = stba();
        if (get_mouse_button() != LEFT) exitflag2 = 1;
        }
      bg_stop();
      while(get_mouse_button() == LEFT) {
        delay(D);
        setcolor(BLUE);
        sprintf(text,"%g eV",e3);
        outtextxy(350,330,text);
        e3++;
        if (e3 > 3000.0) e3 = 3000.0;
        setcolor(YELLOW);
        sprintf(text,"%g eV",e3);
        outtextxy(350,330,text);
        }
      bg_stop();
      }
    }
   } /* end of exitflag == 1 loop */
  while(get_mouse_button() == LEFT);
  energies[(eincr*3)] = (double)(e1);
  energies[(eincr*3) + 1] = (double)(e2);
  energies[(eincr*3) + 2] = (double)(e3);

  } /* end of loop for Energy incrementor */

mouse_off();
cleardevice();
get_filename(filename,7);
cleardevice();
```

```
setfillstyle(11,MAGENTA);
settextjustify(CENTER_TEXT,CENTER_TEXT);
setcolor(CYAN);
bar3d(210,205,430,275,3,3);
setcolor(YELLOW);
outtextxy(320,215,"Grabbing SAM Image");
outtextxy(320,225,"Please Wait");

mouse_on();

/* -- Open output Files for SAM Data -- */
sprintf(fname,"%s1.SAM",filename);
ofp1 = fopen(fname, "wt");
sprintf(fname,"%s2.SAM",filename);
ofp2 = fopen(fname, "wt");
sprintf(fname,"%s3.SAM",filename);
ofp3 = fopen(fname, "wt");
sprintf(fname,"%s4.SAM",filename);
ofp4 = fopen(fname, "wt");
sprintf(fname,"%s5.SAM",filename);
ofp5 = fopen(fname, "wt");
sprintf(fname,"%s6.SAM",filename);
ofp6 = fopen(fname, "wt");

sprintf(fname,"%s.SEM",filename);
ofp7 = fopen(fname, "wt");

sprintf(fname,"%s.INF",filename);
ofp8 = fopen(fname, "wt");

/* ------------------------------------------------------ */
/* Write the information about the scans to the data file */
/* ------------------------------------------------------ */

fprintf(ofp8,"%i\n%i\n%i\n%i\n%i\n",no_els,samx,samy,xsize,ysize);
for (i = 0; i<12; ++i) {
  fprintf(ofp8,"%g\n",energies[i]);
  }

xstep = (double)(box_data->brx - box_data->tlx) / (double)(samx);
ystep = (double)(box_data->bry - box_data->tly) / (double)(samy);

xpos = (double)(box_data->tlx);
ypos = (double)(box_data->tly);

for (yincr = 0; yincr < samy; ++yincr) {
  yfactor =  5.0 / 320.0;
  y_shift  = 2.5 + ((ypos - 50.0) * yfactor );
  dac_out(y_shift,3);

  percent = yincr*100/samy;
  sprintf(text,"%i Percent Completed",old_percent);
  button(0,220,280,420,300,text);
  sprintf(text,"%i Percent Completed",percent);
  button(1,220,280,420,300,text);
  setfillstyle(1,WHITE);
  setcolor(YELLOW);
  bar3d(220,245,(220+(percent*2)),265,3,3);
  old_percent = percent;

  xpos = (double)(box_data->tlx);
  for (xincr = 0; xincr < samx; ++xincr) {
    xfactor = 5.0 / 500.0;
    x_shift = 2.5 + ((xpos - 50.0) * xfactor );
    dac_out(x_shift,2);
    sem_result = get_adc_value();
    fprintf(ofp7,"%i\n",sem_result);
    for (n = 0; n < no_els; n++) {
      for (i = 0; i<3; ++i) {
        get_point(energies[(n*3)+i],point_data);
        fprintf(ofp1,"%Lf\t",point_data->c1);
        fprintf(ofp2,"%Lf\t",point_data->c2);
        fprintf(ofp3,"%Lf\t",point_data->c3);
        fprintf(ofp4,"%Lf\t",point_data->c4);
        fprintf(ofp5,"%Lf\t",point_data->c5);
        fprintf(ofp6,"%Lf\t",point_data->c6);
```

```
          } /* end of i loop */
        } /* end of n loop */
       /* -- Write SAM Data to the Files -- */
       fprintf(ofp1,"\n");
       fprintf(ofp2,"\n");
       fprintf(ofp3,"\n");
       fprintf(ofp4,"\n");
       fprintf(ofp5,"\n");
       fprintf(ofp6,"\n");
       xpos = xpos + xstep;
       } /* end of xincr loop */
     ypos = ypos + ystep;
     } /* end of yincr loop */

   /* Reset dwell time to original value */
   dwell = temdwell;
   fclose(ofp1);
   fclose(ofp2);
   fclose(ofp3);
   fclose(ofp4);
   fclose(ofp5);
   fclose(ofp6);
   fclose(ofp7);
   fclose(ofp8);
   } /* end function sam() */



/* ---------------------- Function display_sam_images ------------- */
/* -- This allows the on-screen Display of Data Collected in the -- */
/* ---------------------- previous Function ----------------------- */
/* ---------------------------------------------------------------- */

void display_sam_image(void) {
  /* Declare Variables */
  FILE *ifp, *ofp;
  char filename[20], fname[20], text[20];
  int exitflag;
  int no_els, samx, samy, xsize, ysize;
  int i,n,yincr,xincr;
  double xstep, ystep;
  double xpos, ypos;
  int el_no = 1;
  double e1,e2,e3;
  double c1,c2,c3,result;
  double maxc1 = 0.0,minc1,maxc2 = 0.0, minc2, maxc3 = 0.0, minc3;
  double rat1, rat2, rat1_range, rat2_range;
  double rat1_max = 0.0, rat2_max = 0.0, rat1_min, rat2_min;
  double e1_range, e2_range, e3_range;
  double p_to_b_range;
  double energies[12];
  double b_dash;
  int e_input;
  int x,y,startx,starty;
  int color;
  int topleft[10] = {10,20,220,20,430,20,10,240,220,240};
  int sem_value;

  /* - Prompt for Root Filename for Stored SAM Images - */
  mouse_off();
  get_filename(fname,7);
  sprintf(filename,"%s.INF",fname);
  ifp = fopen(filename, "rt");

  /* -- Read in Parameters for this Set of Scans -- */
  fscanf(ifp,"%i",&no_els);
  fscanf(ifp,"%i",&samx);
  fscanf(ifp,"%i",&samy);
  fscanf(ifp,"%i",&xsize);
  fscanf(ifp,"%i",&ysize);
  for (i = 0; i<(no_els*3); i++) {
    fscanf(ifp,"%i",&e_input);
    energies[i] = (double)(e_input);
    }
  fclose(ifp);
```

```
  /* -- Set up Graphical Environment -- */
  setfillstyle(1,BLACK);
  setcolor(BLACK);
  bar(200,0,640,480);
  setfillstyle(11,MAGENTA);
  settextjustify(CENTER_TEXT,CENTER_TEXT);
  setcolor(CYAN);
  bar3d(240,90,510,250,3,3);
  button(1,300,255,450,275,"Continue");
  setfillstyle(1,BLUE);
  bar(370,195,450,215);
  bar(370,225,450,245);
  setcolor(YELLOW);
  outtextxy(375,100,"Please Set Parameters");
  outtextxy(375,110,"For the SAM Image ?");

  /* -- Set Up Cycling Prompt for Display Colour Map -- */
  if (colourflag == 0) {
    button(1,260,130,410,150,"Grey Scale");
    }
  if (colourflag == 1) {
    button(1,260,130,410,150,"Colour Scale");
    }
  if (colourflag == 2) {
    button(1,260,130,410,150,"Thermal Scale");
    }
  button(2,300,155,450,175,"Change Palette");
  setcolor(YELLOW);

  sprintf(text,"%i",det_channel);
  outtextxy(410,205,text);
  sprintf(text,"%i",el_no);
  outtextxy(410,235,text);

  /* -- Set up Prompt for User input -- */
  settextjustify(RIGHT_TEXT,CENTER_TEXT);
  outtextxy(350,205,"Channel No");
  outtextxy(350,235,"Element No");
  setcolor(WHITE);
  settextjustify(LEFT_TEXT,CENTER_TEXT);
  outtextxy(451,205,">>");
  outtextxy(451,235,">>");
  settextjustify(RIGHT_TEXT,CENTER_TEXT);
  outtextxy(369,205,"<<");
  outtextxy(369,235,"<<");

mouse_on();
settextjustify(CENTER_TEXT,CENTER_TEXT);
exitflag = 0;


/* -- Poll Mouse Interrupt for User Input --*/
while (exitflag == 0) {
  if ( (mouse_in_box(300,255,450,275) == TRUE) && (get_mouse_button() == LEFT) ) {
    exitflag = 1;
    }
  if ( (mouse_in_box(300,155,450,175) == TRUE) && (get_mouse_button() == LEFT) ) {
    if (colourflag == 0) { /* Start on Mono - go to Colour */
      colourflag = 1;
      button(0,260,130,410,150,"Grey Scale");
      button(1,260,130,410,150,"Colour Scale");
      while(get_mouse_button() == LEFT);
      }
    else if (colourflag == 1) { /* Start on Colour, go to Thermal */
      colourflag = 2;
      button(0,260,130,410,150,"Colour Scale");
      button(1,260,130,410,150,"Thermal Scale");
      while(get_mouse_button() == LEFT);
      }
    else if (colourflag == 2) { /* Start on Thermal, go to Mono */
      colourflag = 0;
      button(0,260,130,410,150,"Thermal Scale");
      button(1,260,130,410,150,"Grey Scale");
      while(get_mouse_button() == LEFT);
      }
```

```
    }
  if ( (mouse_in_box(350,225,370,245) == TRUE) && (get_mouse_button() == LEFT) ) {
    setcolor(BLUE);
    sprintf(text,"%i",el_no);
    outtextxy(410,235,text);
    el_no--;
    if (el_no == 0) el_no = 1;
    setcolor(YELLOW);
    sprintf(text,"%i",el_no);
    outtextxy(410,235,text);
    while(get_mouse_button() == LEFT);
    }

  if ( (mouse_in_box(450,225,470,245) == TRUE) && (get_mouse_button() == LEFT) ) {
    setcolor(BLUE);
    sprintf(text,"%i",el_no);
    outtextxy(410,235,text);
    el_no++;
    if (el_no > no_els) el_no = no_els;
    setcolor(YELLOW);
    sprintf(text,"%i",el_no);
    outtextxy(410,235,text);
    while(get_mouse_button() == LEFT);
    }

  if ( (mouse_in_box(350,195,370,215) == TRUE) && (get_mouse_button() == LEFT) ) {
    setcolor(BLUE);
    sprintf(text,"%i",det_channel);
    outtextxy(410,205,text);
    det_channel--;
    if (det_channel == 0) det_channel = 1;
    setcolor(YELLOW);
    sprintf(text,"%i",det_channel);
    outtextxy(410,205,text);
    while(get_mouse_button() == LEFT);
    }

  if ( (mouse_in_box(450,195,470,215) == TRUE) && (get_mouse_button() == LEFT) ) {
    setcolor(BLUE);
    sprintf(text,"%i",det_channel);
    outtextxy(410,205,text);
    det_channel++;
    if (det_channel == 7) det_channel = 6;
    setcolor(YELLOW);
    sprintf(text,"%i",det_channel);
    outtextxy(410,205,text);
    while(get_mouse_button() == LEFT);
    }


  } /* end of exitflag bit */

/* -- Initialise Screen for Image Display -- */
mouse_off();
cleardevice();
palette_setup(0);
setfillstyle(1,15);
bar(10,30,210,220);
bar(220,30,420,220);
bar(430,30,630,220);
bar(10,240,210,430);
bar(220,240,420,430);
setfillstyle(1,0);
bar(11,50,209,219);
bar(221,50,419,219);
bar(431,50,629,219);
bar(11,260,209,429);
bar(221,260,419,429);
setcolor(0);
settextjustify(CENTER_TEXT,CENTER_TEXT);
outtextxy(110,40,"Auger Peak");
outtextxy(320,40,"Background");
outtextxy(530,40,"SEM Image");
outtextxy(110,250,"(P-B) / (P+B)");
outtextxy(320,250,"(P-B') / (P+B')");
```

```
 /* -- Limit Image Size -- */
 if (xsize > 200) xsize = 200;
 if (ysize > 200) ysize = 200;

 /* -- Work out Incremental variable for Image display -- */
 xstep = (double)(xsize)/(double)(samx);
 ystep = (double)(ysize)/(double)(samy);
 xpos = 0.0;
 ypos = 0.0;

 /* -- Extract Current Energy points from Array of Energies for this scan -- */
 e1 = energies[(el_no-1)*3];
 e2 = energies[(el_no-1)*3 + 1];
 e3 = energies[(el_no-1)*3 + 2];

 /* -- Calculate Co-ordinates of Image Corner -- */
 startx = 430+((200-xsize)/2);
 starty = 50 +((170-ysize)/2);

 /* -- Load SEM data from file and Display -- */
 sprintf(filename,"%s.SEM",fname);
 ifp = fopen(filename, "rt");
 for (yincr = 0; yincr < samy; ++yincr) {
   y = starty + (int)(floor(ypos));
   xpos = 0.0;
   for (xincr = 0; xincr < samx; ++xincr) {
     x = startx + (int)(floor(xpos));
     fscanf(ifp,"%i",&sem_value);
     color =  (int)(floor( (double)(sem_value) * (16.0/4096.0) ) );
     putpixel(x, y, color);
     xpos = xpos + xstep;
     } /* end x loop */
   ypos = ypos + ystep;
   }/* end y loop */
/* -- Close SEM Image file */
fclose(ifp);


/* -- Load SAM Image data for this Detector Channel --*/

sprintf(filename,"%s%i.SAM",fname,det_channel);
ifp = fopen(filename, "rt");
for (i = 0; i < (samx*samy); ++i) {

  for (n = 0; n < (el_no-1); ++n) {
    fscanf(ifp,"%lf",&c1);
    fscanf(ifp,"%lf",&c1);
    fscanf(ifp,"%lf",&c1);
    }
  fscanf(ifp,"%lf",&c1);
  fscanf(ifp,"%lf",&c2);
  fscanf(ifp,"%lf",&c3);

  /* - Calculate (P-B)/(P+B) - */
  rat1 = (c1-c2)/(c1+c2);

  b_dash = fabsl(c3 + ( ((e3-e1)*(c2-c3)) / (e3-e2) ));

  /* - Calculate (P-B')/(P+B'), where B' is Extrapolated background - */

  rat2 = (c1-b_dash)/(c1+b_dash);

  if (i == 0) {
   rat1_min = rat1;
   rat2_min = rat2;
   minc1 = c1;
   minc2 = c2;
   minc3 = c3;
   }

  /*- Calculate limits of SAM results -*/

  if (c1 > maxc1) maxc1 = c1;
  if (c2 > maxc2) maxc2 = c2;
  if (c3 > maxc3) maxc3 = c3;
  if (rat1 > rat1_max) rat1_max = rat1;
```

```
    if (rat2 > rat2_max) rat2_max = rat2;
    if (c1 < minc1) minc1 = c1;
    if (c2 < minc2) minc1 = c2;
    if (c3 < minc3) minc1 = c3;
    if (rat1 < rat1_min) rat1_min = rat1;
    if (rat2 < rat2_min) rat2_min = rat2;

    for (n = 0; n < (no_els-el_no); ++n) {
      fscanf(ifp,"%lf",&c1);
      fscanf(ifp,"%lf",&c1);
      fscanf(ifp,"%lf",&c1);
      }
    } /* end of for i loop */
  fclose(ifp);

  /* - Calculate Range of SAM results for Contrast Adjustment -*/
  e1_range = maxc1-minc1;
  e2_range = maxc2-minc2;
  rat1_range = rat1_max - rat1_min;
  rat2_range = rat2_max - rat2_min;


  /* Define Start Co-ordinates */
  startx = 10+((200-xsize)/2);
  starty = 50 +((170-ysize)/2);

  /* - Load SAM Image Data - */
  sprintf(filename,"%s%i.SAM",fname,det_channel);
  ifp = fopen(filename, "rt");
  ypos = 0.0;
  for (yincr = 0; yincr < samy; ++yincr) {
    y = starty + (int)(floor(ypos));
    xpos = 0.0;
    for (xincr = 0; xincr < samx; ++xincr) {
      x = startx + (int)(floor(xpos));

    for (n = 0; n < (el_no-1); ++n) {
      fscanf(ifp,"%lf",&c1);
      fscanf(ifp,"%lf",&c1);
      fscanf(ifp,"%lf",&c1);
      }
    fscanf(ifp,"%lf",&c1);
    fscanf(ifp,"%lf",&c2);
    fscanf(ifp,"%lf",&c3);

    /* -- Plot Data Points (PEAK)-- */
    color =  (int)(floor1(c1 * (16.0/e1_range)) );
    fprintf(ofp,"\n%i",color);
    putpixel(x, y, color);
    xpos = xpos + xstep;

    for (n = 0; n < (no_els-el_no); ++n) {
      fscanf(ifp,"%lf",&c1);
      fscanf(ifp,"%lf",&c1);
      fscanf(ifp,"%lf",&c1);
      }

    } /* end x loop */
  ypos = ypos + ystep;
  }/* end y loop */

/* -- Load Image Data -- */
startx = 220 +((200-xsize)/2);
starty = 50 +((170-ysize)/2);
sprintf(filename,"%s%i.SAM",fname,det_channel);
ifp = fopen(filename, "rt");
ypos = 0.0;
for (yincr = 0; yincr < samy; ++yincr) {
  y = starty + (int)(floor(ypos));
  xpos = 0.0;
  for (xincr = 0; xincr < samx; ++xincr) {
    x = startx + (int)(floor(xpos));

  for (n = 0; n < (el_no-1); ++n) {
    fscanf(ifp,"%lf",&c1);
    fscanf(ifp,"%lf",&c1);
```

```
      fscanf(ifp,"%lf",&c1);
      }
    fscanf(ifp,"%lf",&c1);
    fscanf(ifp,"%lf",&c2);
    fscanf(ifp,"%Lf",&c3);

    /* -- Plot Image (BACKGROUND) -- */
    color =  (int)(floor1(c2 * (16.0/e2_range)) );
    fprintf(ofp,"\n%i",color);
    putpixel(x, y, color);
    xpos = xpos + xstep;

    for (n = 0; n < (no_els-el_no); ++n) {
      fscanf(ifp,"%lf",&c1);
      fscanf(ifp,"%lf",&c1);
      fscanf(ifp,"%lf",&c1);
      }
    } /* end x loop */
    ypos = ypos + ystep;
    }/* end y loop */

  /* -- Load Image Data -- */
  startx = 10+((200-xsize)/2);
  starty = 260 +((170-ysize)/2);
  sprintf(filename,"%s%i.SAM",fname,det_channel);
  ifp = fopen(filename, "rt");
  ypos = 0.0;
  for (yincr = 0; yincr < samy; ++yincr) {
    y = starty + (int)(floor(ypos));
    xpos = 0.0;
    for (xincr = 0; xincr < samx; ++xincr) {
      x = startx + (int)(floor(xpos));
    for (n = 0; n < (el_no-1); ++n) {
      fscanf(ifp,"%lf",&c1);
      fscanf(ifp,"%lf",&c1);
      fscanf(ifp,"%lf",&c1);
      }
    fscanf(ifp,"%lf",&c1);
    fscanf(ifp,"%lf",&c2);
    fscanf(ifp,"%lf",&c3);
    rat1 = (c1-c2)/(c1+c2);

    /* -- Plot Image Data (P-B)/(P+B) -- */
    color =  (int)(floor1(rat1 * (16.0/rat1_range)) );
    fprintf(ofp,"\n%i",color);
    putpixel(x, y, color);
    xpos = xpos + xstep;
    for (n = 0; n < (no_els-el_no); ++n) {
      fscanf(ifp,"%lf",&c1);
      fscanf(ifp,"%lf",&c1);
      fscanf(ifp,"%lf",&c1);
      }
    } /* end x loop */
   ypos = ypos + ystep;
   }/* end y loop */

/* -- Load Image Data -- */
startx = 220+((200-xsize)/2);
starty = 260 +((170-ysize)/2);
sprintf(filename,"%s%i.SAM",fname,det_channel);
ifp = fopen(filename, "rt");
ypos = 0.0;
for (yincr = 0; yincr < samy; ++yincr) {
  y = starty + (int)(floor(ypos));
  xpos = 0.0;
  for (xincr = 0; xincr < samx; ++xincr) {
    x = startx + (int)(floor(xpos));

  for (n = 0; n < (el_no-1); ++n) {
    fscanf(ifp,"%lf",&c1);
    fscanf(ifp,"%lf",&c1);
    fscanf(ifp,"%lf",&c1);
    }
  fscanf(ifp,"%lf",&c1);
  fscanf(ifp,"%lf",&c2);
  fscanf(ifp,"%lf",&c3);
```

```
    b_dash = fabsl(c3 + ( (e3-e1)*(c2-c3) / (e1+(e3-e1)-e2) ));
    if(c1 == 0.0) c1 = 0.1;
    rat2 = (c1-b_dash)/(c1+b_dash);
    /* -- Plot Image Data (P-B')/(P+B') -- */
    color =  (int)(floorl(rat2 * (16.0/rat2_range)) );
    fprintf(ofp,"\n%i",color);
    putpixel(x, y, color);
    xpos = xpos + xstep;

    for (n = 0; n < (no_els-el_no); ++n) {
      fscanf(ifp,"%lf",&c1);
      fscanf(ifp,"%lf",&c1);
      fscanf(ifp,"%lf",&c1);
      }


     } /* end x loop */
   ypos = ypos + ystep;
   }/* end y loop */
 fclose(ifp);
 fclose(ofp);

 mouse_on();
 button(1,550,463,620,480,"Exit");
 exitflag = 0;

 /* -- Wait for User to Select Exit, then Return -- */
 while (exitflag == 0) {
   if ( (mouse_in_box(550,463,620,480) == TRUE) && (get_mouse_button() == LEFT) ) {
     exitflag = 1;
     }
   }
 closegraph();
 initialise_graphics_environment();
 mouse_on();
 } /* end of function display_sam_image */
```

APPENDIX C

MAIN CMA CONTROL PROGRAM

FILE 'HEADER.H'

```
/* ========================== FUNCTION DECLARATIONS ======================*/
void display_sam_image(void);
int get_adc_value(void);
void sam(BOX *box_data);
void get_point(double energy, POINT *point_data);
int load_single(int current_scan);
int load_options(int current_scan);
void load_linescan(void);
void save_single(int current_scan);
void save_linescan(void);
void save_options(int current_scan);
void display_options(int current_scan);
void single_display(int current_scan);
void plot_line_scan(int line_no);
void line_display(void);
void del_files(int low, int high);
void fs_linescan(LINE *line_data);
void el_linescan(LINE *line_data);
void linescan(LINE *line_data);
void get_box (BOX *box_data);
void get_line(LINE *line_data, int *line1,int *line2, int flag);
void cross(int x, int y, int *line1, int *line2, int flag);
void palette_setup(int flag);
void image(void);
int tune_ch(void);
int tune(void);
void plot_scan(int c_scan);
int write_scan(char *filein, char *fileout);
void get_filename(char filename[], int length);
void set_params(int flag);
void display_params(int flag);
int scan_params(int current_scan);
```

```
void close_down(void);

void menu_input(void);

void main_screen_setup(void);

void draw_menu(void);

int bb_check(void);

void bb_init(void);

void dac_out(double voltage, int channel);

void dio_write(int value);

void bg_start(int freq);

void bg_stop(void);

int stba(void);

int grab_scan(int scan_no, int current, int total);

void draw_axes(long int count, int max, int min);

int sem(void);

int sam(void);

void image_menu(void);

/* ============================  GLOBAL #defines  ============================*/

#define base_id 0xCE00  /* Base Address of First BB Carrier */
#define base_id2 0xCE40  /* Base Address of Second BB Carrier */
```

FILE 'CMA6.CPP'

```
/* ----------------------  CMA6.CPP  --------------------------- */
/* - The Control program fot the Six-Channel Angle resolved CMA  - */
/* ------------------------------------------------------------- */

#include <stdio.h>
#include <stdlib.h>
#include <process.h>
#include <conio.h>
#include <dos.h>
#include <time.h>
#include <assert.h>
#include <string.h>
#include <math.h>
#include <graphics.h>

/* -------- Define Structrures to Hold Information about ------- */
/* --- Scanning Auger parameters, Line and box co-ordinates ---- */

typedef struct line_params {
 int x1, y1, x2, y2;
 } LINE;
typedef struct box_params {
 int tlx, tly, brx, bry;
 } BOX;
typedef struct point_results {
 long double c1,c2,c3,c4,c5,c6;
 } POINT;


/* - Include Supporting Functions for Interfacing and GUI - */
#include "header.h"
```

```
#include "guihead.h"
#include "dio.h"
#include "dac.h"
#include "handio.h"
#include "bg.h"
#include "bb.h"
#include "gui.cpp"
#include "image.cpp"


/* - Define Global Constants - */
#define D 5
#define base_id 0xCE00
#define bgcol BLACK
#define fgcol WHITE
#define maxy 1000

/* - Define Global variables for grab_scan - */
int emin = 0, emax = 2000, points = 200, dwell = 50;
long int max_count = 2000;

/* - Define Global Position variables for electron scan - */
double y_voltage = 5.0, x_voltage = 5.0;

/* - Define Global Variables for Tune parameters - */
int t_emin = 1000, t_emax = 2000, t_points = 60, t_dwell = 10;
long int t_max_count = 500;

/* - Define Global variables for Display Routines - */
int plotflag = 1;
int colourflag = 2;
int det_channel = 1;

/* Global Variables for linescans */
int l_emin = 0, l_emax = 2000, l_points = 200, l_dwell = 50;
long int l_max_count = 2000;
int current_line = 0;
int max_line = 0;
int linepoints = 3;


void main(void)  {
  /* Set up the counters on the Burr Brown I/O Card */
  bb_init();

  /* Set Up the Mouse and Graphics Environment */
  main_screen_setup();

  /* Draw the main Menu for the start of the Program */
  draw_menu();

  /* Launch the Menu - Other functions are then called from Here */
  menu_input();

  /* Back from menu and main program - Close Down */
  close_down();
  }

/* ------------------------------------------------------------------ */
/* -- Function GRAB_SCAN, gets a spectrum from the 6 Channel CMA -- */
/* ------------------------------------------------------------------ */

int grab_scan(int scan_no, int current, int total) {
  FILE *ofp1, *ofp2, *ofp3, *ofp4, *ofp5, *ofp6;
  char fname[20];
  char text[50];
  int strobe, incr;
  int dio_val;
  double dac_voltage;
  double offset;
  double energy;
  double divisor;
  POINT *point_data;

  /* - Draw the Axes for Real Time Spectrum Plot -*/
  draw_axes(max_count,emax,emin);
```

```
    /* - Display Current Grab Scan Status (if in linescan Mode) -*/
    /* - This could also be useful if Depth profiling is added  -*/
    if (current != 0) {
      sprintf(text,"Grabbing scan %i of %i",current,total);
      button(1,200,2,440,19,text);
      }

    /* - Display Abort Button - */
    button(1,550,463,620,480,"Abort");

    /* - Open Data Files for Spectral Data - */
    sprintf(fname,"scan1.%i",scan_no);
    ofp1 = fopen(fname, "w");
    sprintf(fname,"scan2.%i",scan_no);
    ofp2 = fopen(fname, "w");
    sprintf(fname,"scan3.%i",scan_no);
    ofp3 = fopen(fname, "w");
    sprintf(fname,"scan4.%i",scan_no);
    ofp4 = fopen(fname, "w");
    sprintf(fname,"scan5.%i",scan_no);
    ofp5 = fopen(fname, "w");
    sprintf(fname,"scan6.%i",scan_no);
    ofp6 = fopen(fname, "w");

    /* - Iterate through scan for every Point - */
    for (incr = 0; incr < points; ++incr ) {
      /* - If scan aborted then exit Function returning Zero - */
      if( (mouse_in_box(550,463,620,480) == TRUE) && (get_mouse_button() == LEFT) ) {
        return(0);
        }
      /* Calculate Energy */
      energy = (((double)(incr)*((double)(emax-emin)/(double)(points-1))) +
(double)(emin));
      /* Get Signal for Curent energy and dwell time */
      get_point(energy,point_data);

      /* Calculate X co-ordinate on Screen */
      offset = ( (560.0 / (double)(points-1)) * (double)(incr) );

      /* If on screen then Display Current Point */
      divisor = (double)(max_count)/420.0;
      if ((int)(floorl(point_data->c1)) < max_count) putpixel( (40+floor(offset)),(440-
floor(point_data->c1/divisor)), LIGHTMAGENTA);
      if ((int)(floorl(point_data->c2)) < max_count) putpixel( (40+floor(offset)),(440-
floor(point_data->c2/divisor)), LIGHTRED);
      if ((int)(floorl(point_data->c3)) < max_count) putpixel( (40+floor(offset)),(440-
floor(point_data->c3/divisor)), CYAN);
      if ((int)(floorl(point_data->c4)) < max_count) putpixel( (40+floor(offset)),(440-
floor(point_data->c4/divisor)), LIGHTGREEN);
      if ((int)(floorl(point_data->c5)) < max_count) putpixel( (40+floor(offset)),(440-
floor(point_data->c5/divisor)), WHITE);
      if ((int)(floorl(point_data->c6)) < max_count) putpixel( (40+floor(offset)),(440-
floor(point_data->c6/divisor)), YELLOW);

      /* Save Data To File */
      fprintf(ofp1,"%lf\t%Lf\n",energy,point_data->c1);
      fprintf(ofp2,"%lf\t%Lf\n",energy,point_data->c2);
      fprintf(ofp3,"%lf\t%Lf\n",energy,point_data->c3);
      fprintf(ofp4,"%lf\t%Lf\n",energy,point_data->c4);
      fprintf(ofp5,"%lf\t%Lf\n",energy,point_data->c5);
      fprintf(ofp6,"%lf\t%Lf\n",energy,point_data->c6);

      /* Ensure the Burst Generator is reset */
      bg_stop();
      }
      /* Close Data Files */
      fclose(ofp1);
      fclose(ofp2);
      fclose(ofp3);
      fclose(ofp4);
      fclose(ofp5);
      fclose(ofp6);
      return(1);
}
```

```
/* ---------------------- End of Function GRAB_SCAN ---------------------- */


/* ------------------------ Function draw_axes ---------------------- */
/* ------------------ Draws Graph axes for the Specra --------------- */
/* ------------------------------------------------------------------ */

void draw_axes(long int count, int max, int min) {
  int yincr, xincr, xincr2;
  double yincr2;
  long double factor;
  char text[20];

  /* Set up Screen Environment, Draw Axes */
  mouse_off();
  cleardevice();
  setcolor(YELLOW);
  line(40,440,600,440);
  line(40,440,40,20);
  settextjustify(RIGHT_TEXT,CENTER_TEXT);

  /* Label the Y Axis Tick Marks*/
  yincr2 = 0.0;
  for(yincr = 440; yincr >=20; yincr = yincr - 84) {
    line(40,yincr, 36, yincr);
    factor = (long double)(yincr2) * (long double)(count/420.0);
    sprintf(text,"%3.2Lg",factor/1000.0);
    outtextxy(35,yincr,text);
    yincr2 = yincr2 + 84.0;
    }

  /* Label the X Axis Tick Marks*/
  settextjustify(CENTER_TEXT,CENTER_TEXT);
  for(xincr = 40; xincr <= 600; xincr = xincr + 112) {
    line(xincr, 440, xincr,444);
    sprintf(text,"%g", ( (double)(min) + ( (double)(max-min) / 560.0
)*((double)(xincr) -40) ) );
    outtextxy(xincr,455,text);
    }

  /* Label Axes */
  outtextxy(320,470,"Energy (eV)");
  settextjustify(LEFT_TEXT,CENTER_TEXT);
  outtextxy(5,10,"Counts (K)");

  mouse_on();
}  /* ---- End of Function draw_axes ---- */

/* ------------------------------------------------------------------ */
/* ------------------ Function main_screen_setup -------------------- */
/* --------  -------Sets up basic global conditions ---------------- */
/* ------------------------------------------------------------------ */

void main_screen_setup(void) {

  initialise_graphics_environment();
  setbkcolor(BLACK);
  settextjustify(CENTER_TEXT,CENTER_TEXT);
  setcolor(WHITE);
  cleardevice();
  mouse_setup();
  mouse_on();
  } /* End of main_screen_setup */

/* ------------------------------------------------------------------ */
/* ----------------- Function draw_menu ---------------------------- */
/* ------------- Displays the Main Menu Options --------------------- */
/* ------------------------------------------------------------------ */
void draw_menu(void) {

  button(1, 20,45, 170,65, "Grab Scan");
  button(1, 20,70, 170,90, "Tune CMA");
  button(1, 20,95, 170,115, "Display a Scan");
  button(1, 20,120, 170,140, "Save Scan");
  button(1, 20,145, 170,165, "SAM/SEM Imaging");
  button(1, 20,170,170,190,"Load Scan");
```

```
   button(1, 20,450, 170,470, "Exit Program");

   } /* End of draw_menu */

 /* ------------------------------------------------------------------- */
 /* ----------------------  Function menu_input ---------------------- */
 /* ---- This is the main Harness from which everything is launched ---- */
 /* ------------------------------------------------------------------- */
 void menu_input(void) {
   int exitflag = 0;
   int status = 2, full = 0;
   int load_result = 0;
   int current_scan = 0, top_scan = 0;
   int incr, yincr, shift;
   int done_draw = 0;
   char *filename;
   char text[20];

   while(exitflag == 0) {
     if(current_scan == 29) {
       current_scan = 1;
       full = 1;
       }
     if (done_draw == 0) {
       if(current_scan != 0) {
       /* -- Display Currently Available Scans (Max 21) --*/
        button(1,25,200,165,220,"Current Scan");
        button(2,10,200,24,220,"<<");
        button(2,166,200,180,220,">>");
        }
       yincr = 222;
       shift = 0;
       if (full == 0) {
        for(incr = 1; incr <= top_scan; ++incr) {
          sprintf(text,"%i",incr);
          if (incr != current_scan)
          button(1,(25 + (((incr -1)*20) - shift)), yincr, ((45 + ((incr-1)*20)) -
shift), yincr + 20, text);
          if (incr == current_scan)
          button(2,(25 + (((incr -1)*20) - shift)), yincr, ((45 + ((incr-1)*20)) -
shift), yincr + 20, text);

          if((incr == 7) || (incr == 14) || (incr == 21)) {
             yincr = yincr + 22;
             shift = shift + 140;
             }
           }
        }
      if (full == 1) {
       for(incr = 1; incr <= 28; ++incr) {
         sprintf(text,"%i",incr);
         if (incr != current_scan)
         button(1,(25 + (((incr -1)*20) - shift)), yincr, ((45 + ((incr-1)*20)) -
shift), yincr + 20, text);
         if (incr == current_scan)
         button(2,(25 + (((incr -1)*20) - shift)), yincr, ((45 + ((incr-1)*20)) -
shift), yincr + 20, text);

         if((incr == 7) || (incr == 14) || (incr == 21)) {
           yincr = yincr + 22;
           shift = shift + 140;
           }
         }
       }
      while(get_mouse_button() == LEFT);
      done_draw = 1;
      }

    if ( (mouse_in_box(166,200,180,220) == TRUE) && (get_mouse_button() == LEFT) ) {
      if (full == 0) {
       if (current_scan < top_scan) current_scan++;
       }
      if (full == 1) {
       current_scan++;
       if (current_scan == 29) current_scan = 1;
       }
```

```
    mouse_off();
    cleardevice();
    draw_menu();
    mouse_on();
    done_draw = 0;
    }

  if ( (mouse_in_box(10,200,24,220) == TRUE) && (get_mouse_button() == LEFT) ) {
    if (full == 0) {
      if (current_scan > 1) current_scan--;
      }
    if (full == 1) {
      current_scan--;
      if (current_scan == 0) current_scan = 28;
      }
    mouse_off();
    cleardevice();
    draw_menu();
    mouse_on();
    done_draw = 0;
    }

  /* -- Grab An Auger Scan -- */
  if ( (mouse_in_box(20,45,170,65) == TRUE) && (get_mouse_button() == LEFT) ) {
    current_scan++;
    mouse_off();
    cleardevice();
    mouse_on();
    status = scan_params(current_scan);
    mouse_off();
    cleardevice();
    mouse_on();
    draw_menu();
    if (status == 0) current_scan--;
    if(current_scan > top_scan) top_scan = current_scan;
    done_draw = 0;
    }

  /* -- Save Scan Data -- */
  if ( (mouse_in_box(20,120,170,140) == TRUE) && (get_mouse_button() == LEFT) ) {
    save_options(current_scan);
    mouse_off();
    cleardevice();
    draw_menu();
    mouse_on();
    done_draw = 0;
    }

  /* -- Load a Scan -- */
  if ( (mouse_in_box(20,170,170,190) == TRUE) && (get_mouse_button() == LEFT) ) {
    load_result = load_options(current_scan);
    if (load_result == 1) current_scan ++;
    if(current_scan > top_scan) top_scan = current_scan;
    mouse_off();
    cleardevice();
    draw_menu();
    done_draw = 0;
    mouse_on();
    }

  /* -- Display Scan Data -- */
  if ( (mouse_in_box(20,95,170,115) == TRUE) && (get_mouse_button() == LEFT) ) {
    display_options(current_scan);
    mouse_off();
    cleardevice();
    draw_menu();
    done_draw = 0;
    mouse_on();
    }

  /* -- Tune the Spectrometer -- */
  if ( (mouse_in_box(20,70,170,90) == TRUE) && (get_mouse_button() == LEFT) ) {
    status = tune();
    mouse_off();
    cleardevice();
    draw_menu();
```

```
        done_draw = 0;
        mouse_on();
        }

      /* -- SEM Imaging, Followed by Auger if required -- */
      if ( (mouse_in_box(20,145,170,165) == TRUE) && (get_mouse_button() == LEFT) ) {
        image();
        mouse_off();
        cleardevice();
        draw_menu();
        done_draw = 0;
        mouse_on();
        }

      /* -- Exit the Entire Program -- */
      if ( (mouse_in_box(20,450,170,470) == TRUE) && (get_mouse_button() == LEFT) ) {
        exitflag = 1;
        }
      }
  } /* End of menu_input */


  /* ------------------------------------------------------------------- */
  /* --------------------- Function close_down ------------------------- */
  /* - Deletes all Temporary Files, resets I/O outputs, closes graphics -- */
  /* ------------------------------------------------------------------- */
  void close_down(void) {

    int incr, result;
    int low, high;
    char fname[20];
    del_files(1,27);
    mouse_off();
    cleardevice();
    setfillstyle(11,MAGENTA);
    setcolor(CYAN);
    bar3d(240,90,510,165,3,3);
    button(2,300,110,450,130,"Shutting Down");
    button(2,300,135,450,155,"Please Wait...");
    for (incr = 1; incr <= max_line; incr++) {
      low = (incr*100);
      high = (incr*100) + 99;
      del_files(low,high);
      }
    dac_out(0.0,0);
    dio_write(0);
    closegraph();
    }

  /* ------------------------------------------------------------------- */
  /* --------------------- Function scan_params ------------------------ */
  /* ------------------- Gets Parameters for the Scan ------------------- */
  /* ------------------------------------------------------------------- */

  int scan_params(int current_scan) {
    int status;
    int exitflag;

    display_params(0);
    set_params(0);

    dac_out(x_voltage,2);
    dac_out(y_voltage,3);


    status = grab_scan(current_scan,0,0);
    if (status == 1) {
      button(1,200,2,440,19,"Scan Completed - Click here");
      exitflag = 0;
      while (exitflag == 0) {
        if ((mouse_in_box(200,2,440,19) == TRUE) && (get_mouse_button() == LEFT) )
  exitflag = 1;
        }
      }
    return(status);
```

```
  } /* --- End of Function scan_params --- */

/* ------------------------------------------------------------------- */
/* --------------------- Function display_params ---------------------- */
/* ------------Displays the Screen for Parameter Selection------------- */
/* ------------------------------------------------------------------- */

void display_params(int flag) {
  char text[20];

  mouse_off();

  /* Set up the Screen Environment */
  setfillstyle(11,MAGENTA);
  settextjustify(CENTER_TEXT,CENTER_TEXT);
  setcolor(CYAN);
  bar3d(190,80,440,300,3,3);
  setfillstyle(9,BLUE);
  bar3d(200,320,430,340,3,3);
  setcolor(YELLOW);
  if (flag == 0) {
    outtextxy(315,330,"Grab Scan");
    }
  if (flag == 1) {
    outtextxy(315,330,"Tune");
    }
  setfillstyle(1,BLUE);
  bar(310,120,390,140);
  bar(310,150,390,170);
  bar(310,190,390,210);
  bar(310,220,390,240);
  bar(310,250,390,270);
  setcolor(YELLOW);
  outtextxy(270,100,"Energy Range :-");
  outtextxy(250,130,"From :");
  outtextxy(250,160," To :");
  outtextxy(240,200,"Dwell Time");
  outtextxy(240,230,"No. Points");
  outtextxy(240,260,"Max Count");
  sprintf(text,"%i eV",emin);
  outtextxy(350,130,text);
  sprintf(text,"%i eV",emax);
  outtextxy(350,160,text);
  sprintf(text,"%i ms",dwell);
  outtextxy(350,200,text);
  sprintf(text,"%i",points);
  outtextxy(350,230,text);
  sprintf(text,"%li",max_count);
  outtextxy(350,260,text);
  settextjustify(LEFT_TEXT,CENTER_TEXT);

  setcolor(WHITE);

  outtextxy(284,130,"<<");
  outtextxy(284,160,"<<");
  outtextxy(284,200,"<<");
  outtextxy(284,230,"<<");
  outtextxy(284,260,"<<");

  outtextxy(401,130,">>");
  outtextxy(401,160,">>");
  outtextxy(401,200,">>");
  outtextxy(401,230,">>");
  outtextxy(401,260,">>");

  if (flag == 2) {
    setfillstyle(11,MAGENTA);
    settextjustify(CENTER_TEXT,CENTER_TEXT);
    setcolor(CYAN);
    bar3d(190,360,440,410,3,3);
    setcolor(YELLOW);
    outtextxy(315,330,"Start Linescan");
    setfillstyle(1,BLUE);
    bar(310,380,390,400);
    outtextxy(240,390,"No. Points");
    outtextxy(315,370,"Line Scan Parameters");
```

```
    sprintf(text,"%i",linepoints);
    outtextxy(350,390,text);

    settextjustify(LEFT_TEXT,CENTER_TEXT);
    setcolor(WHITE);
    outtextxy(284,390,"<<");
    outtextxy(401,390,">>");
    }


  mouse_on();
} /* End of display_params */

/* ------------------------------------------------------------------ */
/* ---------------------  Set Params  ------------------------------- */
/* ---------------- Allows the User to set the Scan Parameters ------- */
/* ------------------------------------------------------------------ */

void set_params(int flag) {
  int exitflag = 0;
  char text[20];
  int exitflag2 = 0;
  int strobe;

  settextjustify(CENTER_TEXT,CENTER_TEXT);

  /* --- Polling Loop Waiting for Mouse Events to indicate User Input -- */
  while (exitflag == 0) {

    /* Exit Option */
    if ( (mouse_in_box(200,320,430,340) == TRUE) && (get_mouse_button() == LEFT)) {
      exitflag = 1;
      }

    /* Reduce Minimum Energy */
    if ( (mouse_in_box(283,125,310,135) == TRUE) && (get_mouse_button() == LEFT)) {
      setcolor(BLUE);
      sprintf(text,"%i eV",emin);
      outtextxy(350,130,text);
      emin--;
      if (emin < 0¼žemin = 0;
      setcolor(YELLOW);
      sprintf(text,"%i eV",emin);
      outtextxy(350,130,text);
      if(get_mouse_button() == LEFT) {
       exitflag2 = 0;
      /* This section waits 700ms Before fast scrolling of the variable */
      /* Outboard counters are used so that user can interrupt and continue */
      /* at any time without having to wait the full 700ms to continue */
       dio_write(7000);
       bg_start(10000);
       strobe = stba();
       while (strobe == 0) strobe = stba();
       while ( (exitflag2 == 0) && (strobe == 1) ) {
         strobe = stba();
         if (get_mouse_button() != LEFT) exitflag2 = 1;
         }
       bg_stop();
      /* This section performs the fast Scrolling */
       while(get_mouse_button() == LEFT) {
         delay(D);
         setcolor(BLUE);
         sprintf(text,"%i eV",emin);
         outtextxy(350,130,text);
         emin--;
         if (emin < 0) emin = 0;
         setcolor(YELLOW);
         sprintf(text,"%i eV",emin);
         outtextxy(350,130,text);
         }
       bg_stop();
       }
     }

   /* Increase Minimum Energy */
```

```
  if ( (mouse_in_box(385,125,410,135) == TRUE) && (get_mouse_button() == LEFT)) {
    setcolor(BLUE);
    sprintf(text,"%i eV",emin);
    outtextxy(350,130,text);
    emin++;
    if (emin > emax) emin = emax;
    setcolor(YELLOW);
    sprintf(text,"%i eV",emin);
    outtextxy(350,130,text);
    if(get_mouse_button() == LEFT) {
     exitflag2 = 0;
     dio_write(7000);
     bg_start(10000);
     strobe = stba();
     while (strobe == 0) strobe = stba();
     while ( (exitflag2 == 0) && (strobe == 1) ) {
       strobe = stba();
       if (get_mouse_button() != LEFT) exitflag2 = 1;
       }
     bg_stop();
     while(get_mouse_button() == LEFT) {
       delay(D);
       setcolor(BLUE);
       sprintf(text,"%i eV",emin);
       outtextxy(350,130,text);
       strobe = stba();
       emin++;
       if (emin > emax) emin = emax;
       setcolor(YELLOW);
       sprintf(text,"%i eV",emin);
       outtextxy(350,130,text);
       }
       bg_stop();
     }
   }

 /* Reduce Upper Energy Limit */
 if ( (mouse_in_box(285,155,310,165) == TRUE) && (get_mouse_button() == LEFT)) {
   setcolor(BLUE);
   sprintf(text,"%i eV",emax);
   outtextxy(350,160,text);
   emax--;
   if(emax < emin) emax = emin;
   setcolor(YELLOW);
   sprintf(text,"%i eV",emax);
   outtextxy(350,160,text);
   if(get_mouse_button() == LEFT) {
    exitflag2 = 0;
    dio_write(7000);
    bg_start(10000);
    strobe = stba();
    while (strobe == 0) strobe = stba();
    while ( (exitflag2 == 0) && (strobe == 1) ) {
      strobe = stba();
      if (get_mouse_button() != LEFT) exitflag2 = 1;
      }
    bg_stop();
    while(get_mouse_button() == LEFT) {
      delay(D);
      setcolor(BLUE);
      sprintf(text,"%i eV",emax);
      outtextxy(350,160,text);
      emax--;
      if (emax < emin) emax = emin;
      setcolor(YELLOW);
      sprintf(text,"%i eV",emax);
      outtextxy(350,160,text);
      }
    }
  }

/* Increase Upper Energy Limit */
if ( (mouse_in_box(385,155,410,165) == TRUE) && (get_mouse_button() == LEFT)) {
  setcolor(BLUE);
  sprintf(text,"%i eV",emax);
  outtextxy(350,160,text);
```

```
        emax++;
        if (emax > 3000) emax = 3000;
        setcolor(YELLOW);
        sprintf(text,"%i eV",emax);
        outtextxy(350,160,text);
        if(get_mouse_button() == LEFT) {
         exitflag2 = 0;
         dio_write(7000);
         bg_start(10000);
         strobe = stba();
         while (strobe == 0) strobe = stba();
         while ( (exitflag2 == 0) && (strobe == 1) ) {
           strobe = stba();
           if (get_mouse_button() != LEFT) exitflag2 = 1;
           }
         bg_stop();
         while(get_mouse_button() == LEFT) {
           delay(D);
           setcolor(BLUE);
           sprintf(text,"%i eV",emax);
           outtextxy(350,160,text);
           emax++;
           if (emax > 3000) emax = 3000;
           setcolor(YELLOW);
           sprintf(text,"%i eV",emax);
           outtextxy(350,160,text);
           }
         }
       }


     /* Decrease Number of Points in linescan */
     if ( (flag == 2) && (mouse_in_box(285,383,310,397) == TRUE) && (get_mouse_button()
== LEFT)) {
       setcolor(BLUE);
       sprintf(text,"%i",linepoints);
       outtextxy(350,390,text);
       linepoints--;
       if (linepoints < 2) linepoints = 2;
       setcolor(YELLOW);
       sprintf(text,"%i",linepoints);
       outtextxy(350,390,text);
       if(get_mouse_button() == LEFT) {
        exitflag2 = 0;
        dio_write(7000);
        bg_start(10000);
        strobe = stba();
        while (strobe == 0) strobe = stba();
        while ( (exitflag2 == 0) && (strobe == 1) ) {
          strobe = stba();
          if (get_mouse_button() != LEFT) exitflag2 = 1;
          }
        bg_stop();
        while(get_mouse_button() == LEFT) {
          delay(D);
          setcolor(BLUE);
          sprintf(text,"%i",linepoints);
          outtextxy(350,390,text);
          linepoints--;
          if (linepoints < 2) linepoints = 2;
          setcolor(YELLOW);
          sprintf(text,"%i",linepoints);
          outtextxy(350,390,text);
          }
        }
       }


     /* Increase Number of Points in Linescan */
     if ( (flag == 2) && (mouse_in_box(385,383,410,397) == TRUE) && (get_mouse_button()
== LEFT)) {
       setcolor(BLUE);
       sprintf(text,"%i",linepoints);
       outtextxy(350,390,text);
       linepoints++;
```

```
    if (linepoints > 99) linepoints = 99;
    setcolor(YELLOW);
    sprintf(text,"%i",linepoints);
    outtextxy(350,390,text);
    if(get_mouse_button() == LEFT) {
     exitflag2 = 0;
     dio_write(7000);
     bg_start(10000);
     strobe = stba();
     while (strobe == 0) strobe = stba();
     while ( (exitflag2 == 0) && (strobe == 1) ) {
       strobe = stba();
       if (get_mouse_button() != LEFT) exitflag2 = 1;
       }
     bg_stop();
     while(get_mouse_button() == LEFT) {
       delay(D);
       setcolor(BLUE);
       sprintf(text,"%i",linepoints);
       outtextxy(350,390,text);
       linepoints++;
       if (linepoints > 99) linepoints = 99;
       setcolor(YELLOW);
       sprintf(text,"%i",linepoints);
       outtextxy(350,390,text);
       }
    }
   }


 /* Increase Dwell Time */
 if ( (mouse_in_box(385,190,410,210) == TRUE) && (get_mouse_button() == LEFT)) {
   setcolor(BLUE);
   sprintf(text,"%i ms",dwell);
   outtextxy(350,200,text);
   dwell++;
   if (dwell > 10000) dwell = 10000;
   setcolor(YELLOW);
   sprintf(text,"%i ms",dwell);
   outtextxy(350,200,text);
   if(get_mouse_button() == LEFT) {
    exitflag2 = 0;
    dio_write(7000);
    bg_start(10000);
    strobe = stba();
    while (strobe == 0) strobe = stba();
    while ( (exitflag2 == 0) && (strobe == 1) ) {
      strobe = stba();
      if (get_mouse_button() != LEFT) exitflag2 = 1;
      }
    bg_stop();
    while(get_mouse_button() == LEFT) {
      delay(D);
      setcolor(BLUE);
      sprintf(text,"%i ms",dwell);
      outtextxy(350,200,text);
      dwell++;
      if (dwell > 10000) dwell = 10000;
      setcolor(YELLOW);
      sprintf(text,"%i ms",dwell);
      outtextxy(350,200,text);
      }
    }
  }

/* Decrease Dwell Time */
if ( (mouse_in_box(285,190,310,210) == TRUE) && (get_mouse_button() == LEFT)) {
  setcolor(BLUE);
  sprintf(text,"%i ms",dwell);
  outtextxy(350,200,text);
  dwell--;
  if (dwell < 1) dwell = 1;
  setcolor(YELLOW);
  sprintf(text,"%i ms",dwell);
  outtextxy(350,200,text);
  if(get_mouse_button() == LEFT) {
```

```
      exitflag2 = 0;
      dio_write(7000);
      bg_start(10000);
      strobe = stba();
      while (strobe == 0) strobe = stba();
      while ( (exitflag2 == 0) && (strobe == 1) ) {
        strobe = stba();
        if (get_mouse_button() != LEFT) exitflag2 = 1;
        }
      bg_stop();
      while(get_mouse_button() == LEFT) {
        delay(D);
        setcolor(BLUE);
        sprintf(text,"%i ms",dwell);
        outtextxy(350,200,text);
        dwell--;
        if (dwell < 1) dwell = 1;
        setcolor(YELLOW);
        sprintf(text,"%i ms",dwell);
        outtextxy(350,200,text);
        }
      }
    }

  /* Increase Number of Points */
  if ( (mouse_in_box(385,223,410,237) == TRUE) && (get_mouse_button() == LEFT)) {
    setcolor(BLUE);
    sprintf(text,"%i",points);
    outtextxy(350,230,text);
    points++;
    if (points > 5000) points = 5000;
    setcolor(YELLOW);
    sprintf(text,"%i",points);
    outtextxy(350,230,text);
    if(get_mouse_button() == LEFT) {
      exitflag2 = 0;
      dio_write(7000);
      bg_start(10000);
      strobe = stba();
      while (strobe == 0) strobe = stba();
      while ( (exitflag2 == 0) && (strobe == 1) ) {
        strobe = stba();
        if (get_mouse_button() != LEFT) exitflag2 = 1;
        }
      bg_stop();
      while(get_mouse_button() == LEFT) {
        delay(D);
        setcolor(BLUE);
        sprintf(text,"%i",points);
        outtextxy(350,230,text);
        points++;
        if (points > 5000) points = 5000;
        setcolor(YELLOW);
        sprintf(text,"%i",points);
        outtextxy(350,230,text);
        }
      }
    }

/* Decrease Number of Points */
if ( (mouse_in_box(285,223,310,237) == TRUE) && (get_mouse_button() == LEFT)) {
  setcolor(BLUE);
  sprintf(text,"%i",points);
  outtextxy(350,230,text);
  points--;
  if (points < 2) points = 2;
  setcolor(YELLOW);
  sprintf(text,"%i",points);
  outtextxy(350,230,text);
  if(get_mouse_button() == LEFT) {
    exitflag2 = 0;
    dio_write(7000);
    bg_start(10000);
    strobe = stba();
    while (strobe == 0) strobe = stba();
    while ( (exitflag2 == 0) && (strobe == 1) ) {
```

```
      strobe = stba();
      if (get_mouse_button() != LEFT) exitflag2 = 1;
      }
    bg_stop();
    while(get_mouse_button() == LEFT) {
      delay(D);
      setcolor(BLUE);
      sprintf(text,"%i",points);
      outtextxy(350,230,text);
      points--;
      if (points < 2) points = 2;
      setcolor(YELLOW);
      sprintf(text,"%i",points);
      outtextxy(350,230,text);
      }
    }
  }

/* Decrease Maximum Expected Count */
if ( (mouse_in_box(285,255,310,265) == TRUE) && (get_mouse_button() == LEFT)) {
  setcolor(BLUE);
  sprintf(text,"%li",max_count);
  outtextxy(350,260,text);
  max_count = (max_count - 50);
  if (max_count < 50) max_count = 50;
  setcolor(YELLOW);
  sprintf(text,"%li",max_count);
  outtextxy(350,260,text);
  if(get_mouse_button() == LEFT) {
    exitflag2 = 0;
    dio_write(7000);
    bg_start(10000);
    strobe = stba();
    while (strobe == 0) strobe = stba();
    while ( (exitflag2 == 0) && (strobe == 1) ) {
      strobe = stba();
      if (get_mouse_button() != LEFT) exitflag2 = 1;
      }
    bg_stop();
    while(get_mouse_button() == LEFT) {
      delay(D);
      setcolor(BLUE);
      sprintf(text,"%li",max_count);
      outtextxy(350,260,text);
      max_count = max_count - 50;
      if (max_count < 50) max_count = 50;
      setcolor(YELLOW);
      sprintf(text,"%li",max_count);
      outtextxy(350,260,text);
      }
    }
  }

/* Increase Maximum Expected Count */
if ( (mouse_in_box(385,255,410,265) == TRUE) && (get_mouse_button() == LEFT)) {
  setcolor(BLUE);
  sprintf(text,"%li",max_count);
  outtextxy(350,260,text);
  max_count = max_count + 50;
  if (max_count > 100000) max_count = 100000;
  setcolor(YELLOW);
  sprintf(text,"%li",max_count);
  outtextxy(350,260,text);
  if(get_mouse_button() == LEFT) {
    exitflag2 = 0;
    dio_write(7000);
    bg_start(10000);
    strobe = stba();
    while (strobe == 0) strobe = stba();
    while ( (exitflag2 == 0) && (strobe == 1) ) {
      strobe = stba();
      if (get_mouse_button() != LEFT) exitflag2 = 1;
      }
    bg_stop();
    while(get_mouse_button() == LEFT) {
```

```
            delay(D);
            setcolor(BLUE);
            sprintf(text,"%li",max_count);
            outtextxy(350,260,text);
            max_count = max_count + 50;
            if (max_count > 100000) max_count = 100000;
            setcolor(YELLOW);
            sprintf(text,"%li",max_count);
            outtextxy(350,260,text);
            }
          }
        }
    } /* End of main WHILE loop */

} /* End of set_params */

/* ------------------------------------------------------------------ */
/* ---------------------- Function get_filename --------------------- */
/* ----------- General Routine used for selection of filenames ------- */
/* ------------------------------------------------------------------ */
void get_filename(char filename[],int length) {
  char input;
  int exitflag = 0;
  char fname[20];
  char text[100];
  int incr;

  mouse_off();
  setfillstyle(11,MAGENTA);
  setcolor(CYAN);
  bar3d(300,300,500,400,3,3);
  setcolor(YELLOW);
  settextjustify(CENTER_TEXT,CENTER_TEXT);
  outtextxy(400,320,"Please Enter Filename");
  outtextxy(400,340,"(not including extention");
  outtextxy(400,360,"Type Enter when Done");
  button(1,350,403,450,423,NULL);

  /* Only allows input of Valid characters */
  /* Text string is gradually built up     *
  fname[0] = NULL;
  filename[0] = NULL;
  incr = 0;
  while(exitflag == 0) {
    input = getch();
    if(input == 13) exitflag = 1;
    if(input != 13) {
      if(input == 8) {
        if (incr > 0) {
          filename[incr - 1] = NULL;
          incr --;
          } else {
          filename[incr] = NULL;
          }
        } else {
        if ( (incr < length) && ( ((input > 64) && (input < 123)) || ((input > 47) &&
(input < 58)) ) ){
          filename[incr] = input;
          filename[incr+1] = NULL;
          incr ++;
          button(0,350,403,450,423,fname);
          sprintf(fname,"%s",filename);
          button(1,350,403,450,423,fname);
          }
        }
      if (input == 8) {
        button(0,350,403,450,423,fname);
        sprintf(fname,"%s",filename);
        button(1,350,403,450,423,fname);
        }
      }

    }
    mouse_on();
} /* - End of get_filename - */
```

```
/* ------------------------------------------------------------------ */
/* --------------------  Function load_options() -------------------- */
/* ---- Allows the Loading of previously Saved Spectrum Data --------- */
/* ------------------------------------------------------------------ */

int load_options(int current_scan) {

  int exitflag;
  int status = 3;

  /* Set up Screen Environment */
  mouse_off();
  setfillstyle(11,MAGENTA);
  setcolor(CYAN);
  bar3d(240,90,510,225,3,3);
  /* Display the available Options */
  button(2,302,110,448,130,"Load Options");
  button(1,290,140,460,160,"Linescan");
  button(1,290,165,460,185,"Spectrum Set");
  button(1,290,195,460,215,"Cancel");
  mouse_on();

  /* Poll for user Input and Launch Appropriate Function */
  exitflag = 0;
  while (exitflag == 0) {
    if ( (mouse_in_box(290,140,460,160) == TRUE) && (get_mouse_button() == LEFT) ) {
      load_linescan();
      exitflag = 1;
      }
    if ( (mouse_in_box(290,165,460,185) == TRUE) && (get_mouse_button() == LEFT) ) {
      status = load_single(current_scan);
      exitflag = 1;
      }
    if ( (mouse_in_box(290,195,460,215) == TRUE) && (get_mouse_button() == LEFT) ) {
      exitflag = 1;
      }

    } /* end while exitflag == 0 */
  return(status);
  } /* end of function load_options */

/* ------------------------------------------------------------------ */
/* ------------------------  Function load_linescan ----------------- */
/* -------  Loads a previously collected set of linescan spectra ------ */
/* ------------------------------------------------------------------ */
void load_linescan(void) {
  char filename[20];
  char fileout[20];
  char filein[20];
  char fname[20];
  int status = 0;
  int exitflag;
  int incr, incr2;
  int no_spectra;
  int c_scan;
  int scan_no;
  FILE *ifp;


  scan_no = current_line+1;
  /* Select Filename for Root File */
  get_filename(filename,7);
  no_spectra = 0;
  /* Attempt to load sets 0 to 99 until there are no more ! */
  for (incr = 0; incr < 99; ++incr) {
    c_scan = incr + 1;
    sprintf(fname,"%s1.%i",filename,c_scan);
    if ( (ifp = fopen(fname, "rt")) != NULL )
      no_spectra = incr + 1; else
      incr = 100;
    fclose(ifp);
    }
  /* Load Spectrum data as long as there are valid files */
  if (no_spectra != 0) {
    for (incr = 1; incr <= 6; ++incr) {
      for (incr2 = 0; incr2 < no_spectra; ++incr2) {
```

```
          c_scan = (incr2 + 1);
          sprintf(filein,"%s%i.%i",filename,incr,c_scan);
          c_scan = ((scan_no)*100) + incr2;
          sprintf(fileout,"scan%i.%i",incr,c_scan);
          status = write_scan(filein,fileout);
          }
        }
      } /* end if no_spectra != 0 */

    /* Increment Current line number is there is another one available */
    if (status == 1) {
        current_line++;
        if (max_line < current_line) max_line = current_line;
        }

    /* Warn if Valid File is not Found */
    if (status == 0) {
      mouse_off();
      cleardevice();
      button(1,230,100,410,120,"File not Found !");
      button(1,230,122,410,142,"Click here to Continue");
      mouse_on();
      exitflag = 0;
      while (exitflag == 0) {
        if ( (mouse_in_box(235,100,405,145) == TRUE) && (get_mouse_button() == LEFT) )
exitflag = 1;
        }
      }
    } /* end of Function load_linescan */

/* ------------------------------------------------------------------ */
/* -------------------  Function load_single ---------------------- */
/* --------- Loads single (sets of six) spectra into workspace -------- */
/* ------------------------------------------------------------------ */

int load_single(int current_scan) {
  char filename[20];
  char fileout[20];
  char filein[20];
  int status;
  int exitflag;
  int incr;

  /* Prompt for Filename */
  get_filename(filename,8);

  /* Sequentially load the six files */
  for (incr = 1; incr <= 6; ++incr) {
    sprintf(filein,"%s.%i",filename,incr);
    sprintf(fileout,"scan%i.%i",incr,(current_scan+1));
    status = write_scan(filein,fileout);
    }
  /* Warn if it was not possible to write the scan into the workspace */
  if (status == 0) {
    mouse_off();
    cleardevice();
    button(1,230,100,410,120,"File not Found !");
    button(1,230,122,410,142,"Click here to Continue");
    mouse_on();
    exitflag = 0;
    while (exitflag == 0) {
      if ( (mouse_in_box(235,100,405,145) == TRUE) && (get_mouse_button() == LEFT) )
exitflag = 1;
      }
    }
  return(status);
  } /* end of Function load_single */


/* ------------------------------------------------------------------ */
/* -------------------  Function save_options() ------------------- */
/* ----------------- Launches the funtions to Save Spectra ---------- */
/* ------------------------------------------------------------------ */

void save_options(int current_scan) {
  int exitflag;
```

```
    mouse_off();
    setfillstyle(11,MAGENTA);
    setcolor(CYAN);
    bar3d(240,90,510,225,3,3);

    /* Display the Options */
    button(2,302,110,448,130,"Save Options");
    button(1,290,140,460,160,"Linescan");
    button(1,290,165,460,185,"Current Spectrum");
    button(1,290,195,460,215,"Cancel");

    mouse_on();

    /* Poll for User Input and Launch Appropriate Functions */
    exitflag = 0;
    while (exitflag == 0) {
      if ((max_line > 0) && (mouse_in_box(290,140,460,160) == TRUE) &&
  (get_mouse_button() == LEFT) ) {
        save_linescan();
        exitflag = 1;
        }
      if ( (mouse_in_box(290,165,460,185) == TRUE) && (get_mouse_button() == LEFT) &&
  (current_scan != 0) ) {
        save_single(current_scan);
        exitflag = 1;
        }
      if ( (mouse_in_box(290,195,460,215) == TRUE) && (get_mouse_button() == LEFT) ) {
        exitflag = 1;
        }

      } /* end while exitflag == 0 */
    } /* end of function save_scan */

  /* ------------------------------------------------------------------ */
  /* ------------------------- Function save_single ------------------- */
  /* --- Saves a single Spectrum set (Six spectra, one per channel)----- */
  /* ------------------------------------------------------------------ */
  void save_single(int current_scan) {
    char filename[20];
    char fileout[20];
    char filein[20];

    int incr;

    /* Prompt For Filename */
    get_filename(filename,8);

    /* Save Currently Selected Spectrum set */
    for (incr = 1; incr <= 6; ++incr) {
      sprintf(filein,"scan%i.%i",incr,current_scan);
      sprintf(fileout,"%s.%i",filename,incr);
      write_scan(filein,fileout);
      }
    } /* end of Function save_single */


/* ------------------------------------------------------------------ */
/* --------------------- Function save_linescan ---------------------- */
/* ------------------ Saves a set of Linescan Data ------------------- */
/* ------------------------------------------------------------------ */
void save_linescan(void) {
  char filename[20];
  char fileout[20];
  char filein[20];
  char fname[20];
  int incr, incr2;
  FILE *ifp;
  int scan_no;
  char text[20];
  int exitflag;
  int no_spectra;
  int c_scan;

  scan_no = current_line;
  /* Set up Screen */
  mouse_off();
```

```
  cleardevice();
  setfillstyle(11,MAGENTA);
  setcolor(CYAN);
  bar3d(180,90,450,230,3,3);
  setcolor(BLUE);

  /* Prompt User to Select a Scan set */

  button(1,240,240,390,260,"Continue");
  setcolor(YELLOW);
  settextjustify(CENTER_TEXT, CENTER_TEXT);
  outtextxy(315,110,"Select Scan Set");
  outtextxy(315,120,"To Save");
  setfillstyle(1,BLUE);
  bar(340,140,410,160);
  setcolor(WHITE);
  outtextxy(330,150,"<<");
  outtextxy(420,150,">>");
  setcolor(YELLOW);
  outtextxy(280,150,"Scan No.");
  setcolor(YELLOW);
  sprintf(text,"%i",scan_no);
  outtextxy(375,150,text);
  mouse_on();

  /* Poll Mouse Interrupt for User Request */
  exitflag = 0;
  while (exitflag == 0) {
    if ( (mouse_in_box(240,240,390,260) == TRUE) && (get_mouse_button() == LEFT) ) {
      exitflag = 1;
      }
    if ( (mouse_in_box(320,145,340,155) == TRUE) && (get_mouse_button() == LEFT) ) {
      if (scan_no > 1) {
        mouse_off();
        setcolor(BLUE);
        sprintf(text,"%i",scan_no);
        outtextxy(375,150,text);
        scan_no--;
        setcolor(YELLOW);
        sprintf(text,"%i",scan_no);
        outtextxy(375,150,text);
        mouse_on();
        }
      while(get_mouse_button() == LEFT);
      }

    if ( (mouse_in_box(410,145,430,155) == TRUE) && (get_mouse_button() == LEFT) ) {
      if (scan_no < max_line) {
        mouse_off();
        setcolor(BLUE);
        sprintf(text,"%i",scan_no);
        outtextxy(375,150,text);
        scan_no++;
        setcolor(YELLOW);
        sprintf(text,"%i",scan_no);
        outtextxy(375,150,text);
        mouse_on();
        }
      while(get_mouse_button() == LEFT);
      }

    } /* end exitflag == 0 */

/* - Prompt For filename -*/
get_filename(filename,7);

/* - Attempt to load all sets to 99 to determine number of spectrum sets */
for (incr = 0; incr < 99; ++incr) {
  c_scan = ((100*scan_no)+incr);
  sprintf(fname,"scan1.%i",c_scan);
  if ( (ifp = fopen(fname, "rt")) != NULL )
    no_spectra = incr + 1; else
    incr = 100;
  fclose(ifp);
  }
```

```
   /* Save Files under New name */
   for (incr = 1; incr <= 6; ++incr) {
     for (incr2 = 0; incr2 < no_spectra; ++incr2) {
       c_scan = (scan_no*100) + incr2;
       sprintf(filein,"scan%i.%i",incr,c_scan);
       c_scan = (incr2 + 1);
       sprintf(fileout,"%s%i.%i",filename,incr,c_scan);
       write_scan(filein,fileout);
       }
     }
   } /* end of function save_linescan */

/* ------------------------------------------------------------------ */
/* ---------------------- Function write_scan ----------------------- */
/* --------- Universal Funciton to Move one scan to a new file ------- */
/* ------------------------------------------------------------------ */
int write_scan(char *filein, char *fileout) {
  char fname[20];
  FILE *ifp;
  FILE *ofp;
  long double buffer;
  int status;

  /* Open input file - return with error if can't do this */
  if ( (ifp = fopen(filein, "rt")) == NULL) return(0);
  /* Open output file */
  ofp = fopen(fileout, "wt");
  /* Move data from one to the other until the end of the File */
  while (!feof(ifp))
  fputc(fgetc(ifp), ofp);
  fclose(ifp);
  fclose(ofp);
  return(1);
  } /* end of write_scan */

/* ------------------------------------------------------------------ */
/* --------------------- function single_display -------------------- */
/* --------- Harness funciton to display a single spectrum set ------- */
/* ------------------------------------------------------------------ */
void single_display(int current_scan) {
  int exitflag2;

  exitflag2 = 0;

  /* Call main Funciton to display the Scan */
  plot_scan(current_scan);

  /* Poll for User request */
  /* This harness enables switching between point and line display by */
  /* Recursive calling of the 'plot_scan' function */

  while (exitflag2 == 0) {
    if ( (mouse_in_box(550,463,620,480) == TRUE) && (get_mouse_button() == LEFT) ) {
      exitflag2 = 1;
      }
    if ( (mouse_in_box(30,463,100,480) == TRUE) && (get_mouse_button() == LEFT) ) {
      while(get_mouse_button() == LEFT);
      if (plotflag == 0) {
       plotflag = 1;
       plot_scan(current_scan);
       } else {
         plotflag = 0;
         plot_scan(current_scan);
         }
       }
      }
   }

/* ------------------------------------------------------------------ */
/* --------------------- Function plot_scan() ----------------------- */
/* ---------- Plots a single spectrum set on the screen ------------- */
/* ------------------------------------------------------------------ */
void plot_scan(int c_scan) {

  FILE *ifp1, *ifp2, *ifp3, *ifp4, *ifp5, *ifp6;
  char fname[20];
```

```
  int n, incr;
  long double a, lasta1, lasta2, lasta3, lasta4, lasta5, lasta6;
  long double amax, offset;
  long double number1, number2, number3, number4, number5, number6, divisor;
  long double buffer[3];
  long int count;
  int low_e, high_e;
  int no_points;
  int locator;

/* -- This part Works out the maximum count for the spectrum, ignoring -- */
/* -- peaks that are more than 3 times the previous count -- */
  sprintf(fname,"scan1.%i",c_scan);
  ifp1 = fopen(fname, "rt");
  sprintf(fname,"scan2.%i",c_scan);
  ifp2 = fopen(fname, "rt");
  sprintf(fname,"scan3.%i",c_scan);
  ifp3 = fopen(fname, "rt");
  sprintf(fname,"scan4.%i",c_scan);
  ifp4 = fopen(fname, "rt");
  sprintf(fname,"scan5.%i",c_scan);
  ifp5 = fopen(fname, "rt");
  sprintf(fname,"scan6.%i",c_scan);
  ifp6 = fopen(fname, "rt");
  amax = 0.0;
  incr = 0;
  for (n=0; fscanf(ifp1,"%Lf",&a) == 1; ++n) {
  if (n==0) lasta1 = 0.0;
  if (incr == 1) {
    if ( (a > amax) && (a < (3.0*lasta1) ) ) amax = a;
    lasta1 = a;
    }
  incr++;
  if(incr == 2) incr = 0;
  }
  incr = 0;
  for (n=0; fscanf(ifp2,"%Lf",&a) == 1; ++n) {
  if (n==0) lasta2 = 0.0;
  if (incr == 1) {
    if ( (a > amax) && (a < (3.0*lasta2) ) ) amax = a;
    lasta2 = a;
    }
  incr++;
  if(incr == 2) incr = 0;
  }
  incr = 0;
  for (n=0; fscanf(ifp3,"%Lf",&a) == 1; ++n) {
  if (n==0) lasta3 = 0.0;
  if (incr == 1) {
    if ( (a > amax) && (a < (3.0*lasta3) ) ) amax = a;
    lasta3 = a;
    }
  incr++;
  if(incr == 2) incr = 0;
  }
  incr = 0;
  for (n=0; fscanf(ifp4,"%Lf",&a) == 1; ++n) {
  if (n==0) lasta4 = 0.0;
  if (incr == 1) {
    if ( (a > amax) && (a < (3.0*lasta4) ) ) amax = a;
    lasta4 = a;
    }
  incr++;
  if(incr == 2) incr = 0;
  }

  incr = 0;
  for (n=0; fscanf(ifp5,"%Lf",&a) == 1; ++n) {
  if (n==0) lasta5 = 0.0;
  if (incr == 1) {
    if ( (a > amax) && (a < (3.0*lasta5) ) ) amax = a;
    lasta5 = a;
    }
  incr++;
  if(incr == 2) incr = 0;
  }
```

```
  incr = 0;
  for (n=0; fscanf(ifp6,"%Lf",&a) == 1; ++n) {
  if (n==0) lasta6 = 0.0;
  if (incr == 1) {
    if ( (a > amax) && (a < (3.0*lasta6) ) ) amax = a;
    lasta6 = a;
    }
  incr++;
  if(incr == 2) incr = 0;
  }
  /* Close File Pointers */
  fclose(ifp1);
  fclose(ifp2);
  fclose(ifp3);
  fclose(ifp4);
  fclose(ifp5);
  fclose(ifp6);

  /* Determine the minumum and maximum energies in the spectra */

  /* Open first Spectrum */
  sprintf(fname,"scan1.%i",c_scan);
  ifp1 = fopen(fname, "rt");

  /* Run through to find maximum, ignoring peaks */
  incr = 0;
  for (n=0; fscanf(ifp1,"%Lf",&a) == 1; ++n) {
  if (n==0) low_e = floor(a);
  buffer[incr] = a;
  incr++;
  if(incr == 3) incr = 0;
  }

  if (incr == 2) locator = 0;
  if (incr == 1) locator = 2;
  if (incr == 0) locator = 1;
  high_e = floor(buffer[locator]);

  fclose(ifp1);

/* -- Load in the 6 data files and Display the spectra -- */

if (plotflag == 0) {
  sprintf(fname,"scan1.%i",c_scan);
  ifp1 = fopen(fname, "rt");
  sprintf(fname,"scan2.%i",c_scan);
  ifp2 = fopen(fname, "rt");
  sprintf(fname,"scan3.%i",c_scan);
  ifp3 = fopen(fname, "rt");
  sprintf(fname,"scan4.%i",c_scan);
  ifp4 = fopen(fname, "rt");
  sprintf(fname,"scan5.%i",c_scan);
  ifp5 = fopen(fname, "rt");
  sprintf(fname,"scan6.%i",c_scan);
  ifp6 = fopen(fname, "rt");

  /* Work out normalised Count */
  count = (long int)(floor1(amax));
  if (count == 0) {
    count = 1;
    } else {
    count = (count + (floor((double)(count) * 0.1)) );
    }
  no_points = (n/2);
  mouse_off();
  /* Display the required Axes */
  draw_axes(count,high_e,low_e);
  /* Display the 'Exit' Button */
  button(1,550,463,620,480,"Exit");
  /* Display option of points of lines display */
  if (plotflag == 0) {
    button(1,30,463,100,480,"Points");
    } else {
    button(1,30,463,100,480,"Lines");
    }
```

```
/* Display spectra as Lines or points as required */
for (incr = 0; incr <= no_points; ++incr) {
offset = ( (560.0 / (double)(no_points)) * (double)(incr) );
divisor = (double)(count)/420.0;
fscanf(ifp1,"%Lf",&number1);
fscanf(ifp1,"%Lf",&number1);
setcolor(LIGHTMAGENTA);
if (incr == 0) {
  lasta1 = 0.0;
  moveto( (40+floor(offset)) , (440-floor(number1/divisor)) );
  }
if (number1 < (lasta1 * 10.0) ) {
  lineto( (40+floor(offset)) , (440-floor(number1/divisor)) );
  }
lasta1 = number1;
}

for (incr = 0; incr <= no_points; ++incr) {
offset = ( (560.0 / (double)(no_points)) * (double)(incr) );
divisor = (double)(count)/420.0;
fscanf(ifp2,"%Lf",&number2);
fscanf(ifp2,"%Lf",&number2);
setcolor(LIGHTRED);
if (incr == 0) {
  lasta2 = 0.0;
  moveto( (40+floor(offset)) , (440-floor(number2/divisor)) );
  }
if (number2 < (lasta2 * 10.0) ){
  lineto( (40+floor(offset)) , (440-floor(number2/divisor)) );
  }
lasta2 = number2;
}

for (incr = 0; incr <= no_points; ++incr) {
offset = ( (560.0 / (double)(no_points)) * (double)(incr) );
divisor = (double)(count)/420.0;
fscanf(ifp3,"%Lf",&number3);
fscanf(ifp3,"%Lf",&number3);
setcolor(CYAN);
if (incr == 0) {
  lasta3 = 0.0;
  moveto( (40+floor(offset)) , (440-floor(number3/divisor)) );
  }
if (number3 < (lasta3 * 10.0) ) {
  lineto( (40+floor(offset)) , (440-floor(number3/divisor)) );
  }
lasta3 = number3;
}

for (incr = 0; incr <= no_points; ++incr) {
offset = ( (560.0 / (double)(no_points)) * (double)(incr) );
divisor = (double)(count)/420.0;
fscanf(ifp4,"%Lf",&number4);
fscanf(ifp4,"%Lf",&number4);
setcolor(LIGHTGREEN);
if (incr == 0) {
  lasta4 = 0.0;
  moveto( (40+floor(offset)) , (440-floor(number4/divisor)) );
  }
if (number4 < (lasta4 * 10.0) ) {
  lineto( (40+floor(offset)) , (440-floor(number4/divisor)) );
  }
lasta4 = number4;
}

for (incr = 0; incr <= no_points; ++incr) {
offset = ( (560.0 / (double)(no_points)) * (double)(incr) );
divisor = (double)(count)/420.0;
fscanf(ifp5,"%Lf",&number5);
fscanf(ifp5,"%Lf",&number5);
setcolor(WHITE);
if (incr == 0) {
  lasta5 = 0.0;
  moveto( (40+floor(offset)) , (440-floor(number5/divisor)) );
  }
```

```
if (number5 < (lasta5 * 10.0) ) {
  lineto( (40+floor(offset)) , (440-floor(number5/divisor)) );
  }
lasta5 = number5;
}

for (incr = 0; incr <= no_points; ++incr) {
offset = ( (560.0 / (double)(no_points)) * (double)(incr) );
divisor = (double)(count)/420.0;
fscanf(ifp6,"%Lf",&number6);
fscanf(ifp6,"%Lf",&number6);
setcolor(YELLOW);
if (incr == 0) {
  lasta6 = 0.0;
  moveto( (40+floor(offset)) , (440-floor(number6/divisor)) );
  }
if (number6 < (lasta6 * 10.0) ) {
  lineto( (40+floor(offset)) , (440-floor(number6/divisor)) );
  }
lasta6 = number6;
}
/* -- Close the Data Files -- */
fclose(ifp1);
fclose(ifp2);
fclose(ifp3);
fclose(ifp4);
fclose(ifp5);
fclose(ifp6);

mouse_on();
} else {
/* -------------- Display AS POINTS ------------ */
  /* -- Open Files -- */
  sprintf(fname,"scan1.%i",c_scan);
  ifp1 = fopen(fname, "rt");
  sprintf(fname,"scan2.%i",c_scan);
  ifp2 = fopen(fname, "rt");
  sprintf(fname,"scan3.%i",c_scan);
  ifp3 = fopen(fname, "rt");
  sprintf(fname,"scan4.%i",c_scan);
  ifp4 = fopen(fname, "rt");
  sprintf(fname,"scan5.%i",c_scan);
  ifp5 = fopen(fname, "rt");
  sprintf(fname,"scan6.%i",c_scan);
  ifp6 = fopen(fname, "rt");

  /*- Calculate normalised Count -*/
  count = (long int)(floor1(amax));
  if (count == 0) {
    count = 1;
    } else {
    count = (count + (floor((double)(count) * 0.1)) );
    }
  no_points = (n/2);

  mouse_off();
  /* Display Axes */
  draw_axes(count,high_e,low_e);
  if (plotflag == 0) {
    button(1,30,463,100,480,"Points");
    } else {
    button(1,30,463,100,480,"Lines");
    }
  button(1,550,463,620,480,"Exit");

  for (incr = 0; incr <= no_points; ++incr) {
  /* Read in new Spectrum Points for six channels */
  fscanf(ifp1,"%Lf",&number1);
  fscanf(ifp1,"%Lf",&number1);
  fscanf(ifp2,"%Lf",&number2);
  fscanf(ifp2,"%Lf",&number2);
  fscanf(ifp3,"%Lf",&number3);
  fscanf(ifp3,"%Lf",&number3);
  fscanf(ifp4,"%Lf",&number4);
  fscanf(ifp4,"%Lf",&number4);
  fscanf(ifp5,"%Lf",&number5);
```

```
    fscanf(ifp5,"%Lf",&number5);
    fscanf(ifp6,"%Lf",&number6);
    fscanf(ifp6,"%Lf",&number6);

    offset = ( (560.0 / (double)(no_points)) * (double)(incr) );

    divisor = (double)(count)/420.0;
    /* Display Points */
    if ((int)(floorl(number1)) < count) putpixel( (40+floor(offset)),(440-
floor(number1/divisor)), LIGHTMAGENTA);
    if ((int)(floorl(number2)) < count) putpixel( (40+floor(offset)),(440-
floor(number2/divisor)), LIGHTRED);
    if ((int)(floorl(number3)) < count) putpixel( (40+floor(offset)),(440-
floor(number3/divisor)), CYAN);
    if ((int)(floorl(number4)) < count) putpixel( (40+floor(offset)),(440-
floor(number4/divisor)), LIGHTGREEN);
    if ((int)(floorl(number5)) < count) putpixel( (40+floor(offset)),(440-
floor(number5/divisor)), WHITE);
    if ((int)(floorl(number6)) < count) putpixel( (40+floor(offset)),(440-
floor(number6/divisor)), YELLOW);
    }
    /* Close the Data Files */
    fclose(ifp1);
    fclose(ifp2);
    fclose(ifp3);
    fclose(ifp4);
    fclose(ifp5);
    fclose(ifp6);
    mouse_on();
    }
} /* end of plot_scan */



/* ------------------------------------------------------------------ */
/* -------------------------- Function tune ------------------------- */
/* --------- Self contained function to Tune the CMA over a given ---- */
/* ----------              Energy Range                        ------- */
/* ------------------------------------------------------------------ */

int tune(void) {
  char fname[20];
  int strobe, incr;
  int dio_val;
  int channel;

  long double number1;
  long double number2;
  long double number3;
  long double number4;
  long double number5;
  long double number6;
  long double number;
  long double total_count;
  long double last_total_count;
  long double max_total_count;
  int linepoint;
  double dac_voltage;
  double offset;
  double energy;
  double divisor;
  int o_emin, o_emax, o_points, o_dwell;
  long int o_max_count;
  POINT *point_data;

  /* Write Existing Scan Parameters to Temporary Variables */
  o_emin = emin;
  o_emax = emax;
  o_points = points;
  o_dwell = dwell;
  o_max_count = max_count;

  /* Set up parameters to pre-defined tune params -- */
  emin = t_emin;
  emax = t_emax;
  points = t_points;
```

```
  dwell = t_dwell;
  max_count = t_max_count;
  channel = tune_ch();

  /* Position electron beam as set in the SEM stage -- */
  dac_out(x_voltage,2);
  dac_out(y_voltage,3);

  /* Set up Screen */
  mouse_off();
  cleardevice();
  display_params(1);
  mouse_on();
  set_params(1);
  mouse_off();
  cleardevice();
  draw_axes(max_count,emax,emin);
  button(1,550,463,620,480,"Exit");
  button(1,20,463,130,480,"Reset Peak");
  mouse_on();

  total_count = 0.0;
  max_total_count = 0.0;
  settextjustify(LEFT_TEXT,CENTER_TEXT);

/* Loop forever, end function with 'return' command */
while(1) {
  last_total_count = total_count;
  total_count = 0.0;

  for (incr = 1; incr <= points; ++incr ) {

    /* Exit has been selected */
    if( (mouse_in_box(550,458,620,478) == TRUE) && (get_mouse_button() == LEFT) ) {
      /* Save current parameters to Tune Parameters */
      t_emin = emin;
      t_emax = emax;
      t_points = points;
      t_dwell = dwell;
      t_max_count = max_count;

      /* Replace original Scan Parameters */
      emin = o_emin;
      emax = o_emax;
      points = o_points;
      dwell = o_dwell;
      max_count = o_max_count;

      /* Disable Burst Generator and Exit back to menu */
      bg_stop();
      return(0);
      }

    /* Reset Max count */
    if( (mouse_in_box(20,463,130,480) == TRUE) && (get_mouse_button() == LEFT) ) {
      max_total_count = 0.0;
      }

    /* -------- This section grabs the signal at this energy ------- */
    energy = ((incr*((emax-emin)/points)) + emin);
    get_point(energy,point_data);

    /* -- get the signal for this point */
    number1 = point_data->c1;
    number2 = point_data->c2;
    number3 = point_data->c3;
    number4 = point_data->c4;
    number5 = point_data->c5;
    number6 = point_data->c6;


    /* Set total count on centre energy (could be improved on in future */
    if (incr == (int)(floor(points/2)) )
      total_count = floor1(((number1 + number2 + number3 + number4 + number5 +
number6) / 6.0));
```

```
    offset = ( (560.0 / (double)(points)) * (double)(incr) );

    /* Select signal according to which channel is selected (or sum if required) */
    if (channel == 1) number = number1;
    if (channel == 2) number = number2;
    if (channel == 3) number = number3;
    if (channel == 4) number = number4;
    if (channel == 5) number = number5;
    if (channel == 6) number = number6;
    if (channel == 7) number = ((number1 + number2 + number3 + number4 + number5 +
number6) / 6.0);

    if (number > (long double)(max_count) ) number = (long double)(max_count);

    divisor = (double)(max_count)/420.0;
    mouse_off();
    setcolor(CYAN);
    /* Erase the old line at this energy */
    line((40+floor(offset)), 439,(40+floor(offset)),(440-floor(number/divisor)) );
    setcolor(BLACK);
    /* Draw the new line */
    line( (40+floor(offset)), (440-floor(number/divisor)), (40+floor(offset)), 20);
    mouse_on();

    } /* (end of loop for each scan of the tune process) */

    /* Update the total Count Bars at the top of the screen */
    if(total_count < (last_total_count * 1.2)) {
      mouse_off();
      setfillstyle(SOLID_FILL,LIGHTGREEN);
      bar(108,1,115,7);
      setfillstyle(SOLID_FILL,LIGHTRED);
      bar(288,1,295,7);

      setcolor(BLACK);
      setfillstyle(SOLID_FILL,BLACK);
      bar(120,0,287,7);
      sprintf(fname,"Total = %8Lg",last_total_count);
      outtextxy(120,4,fname);
      setcolor(WHITE);
      sprintf(fname,"Total = %8Lg",total_count);
      outtextxy(120,4,fname);

      linepoint = 100 + (floor)(500.0 * (total_count / max_count) );
      setfillstyle(SOLID_FILL,LIGHTGREEN);
      bar(100,16,linepoint,19);
      setfillstyle(SOLID_FILL,BLACK);
      bar(linepoint,16,600,19);

      /* Display Peak Total Count */
      setcolor(BLACK);
      setfillstyle(SOLID_FILL,BLACK);
      bar(300,0,500,7);
      sprintf(fname,"Peak = %8Lg",max_total_count);
      outtextxy(300,4,fname);
      if (total_count > max_total_count) max_total_count = total_count;
      setcolor(WHITE);
      sprintf(fname,"Peak = %8Lg",max_total_count);
      outtextxy(300,4,fname);
      mouse_on();

      linepoint = 100 + (floor) (500.0 * (max_total_count / max_count) );
      setfillstyle(SOLID_FILL,LIGHTRED);
      bar(100,12,linepoint,14);
      setfillstyle(SOLID_FILL,BLACK);
      bar(linepoint,12,600,14);
      }

    } /*end of While(1) loop*/

} /* -- End of Function TUNE -- */

/* ------------------------------------------------------------------ */
/* ------------------------ Function tune_ch() ---------------------- */
/* ------------ Gets the number of the channel to tune from ---------- */
/* ------------------------------------------------------------------ */
```

```
int tune_ch(void) {
  char text[20];
  mouse_off();
  cleardevice();
  sprintf(text,"Select Channel");
  button(2, 240,100,365,120, text);

  /* Display the Options */
  sprintf(text,"1");
  button(1, 250, 122, 283, 142,text);
  sprintf(text,"2");
  button(1, 285, 122, 318, 142,text);
  sprintf(text,"3");
  button(1, 320, 122, 353, 142,text);
  sprintf(text,"4");
  button(1, 250, 144, 283, 164,text);
  sprintf(text,"5");
  button(1, 285, 144, 318, 164,text);
  sprintf(text,"6");
  button(1, 320, 144, 353, 164,text);
  sprintf(text,"Average");
  button(1, 250, 166, 353, 186,text);
  mouse_on();

  /* Loop forever, Poll Mouse for Selection then let 'return' end the loop */
  while(1) {
    if ( (mouse_in_box(250,122,283,142) == TRUE) && (get_mouse_button() == LEFT) ) {
      return(1);
      }
    if ( (mouse_in_box(285,122,318,142) == TRUE) && (get_mouse_button() == LEFT) ) {
      return(2);
      }
    if ( (mouse_in_box(320,122,353,142) == TRUE) && (get_mouse_button() == LEFT) ) {
      return(3);
      }
    if ( (mouse_in_box(250,144,283,164) == TRUE) && (get_mouse_button() == LEFT) ) {
      return(4);
      }
    if ( (mouse_in_box(285,144,318,164) == TRUE) && (get_mouse_button() == LEFT) ) {
      return(5);
      }
    if ( (mouse_in_box(320,144,353,164) == TRUE) && (get_mouse_button() == LEFT) ) {
      return(6);
      }
    if ( (mouse_in_box(250,166,353,186) == TRUE) && (get_mouse_button() == LEFT) ) {
      return(7);
      }
    }

  }



/* ------------------------------------------------------------------ */
/* ------------------- Function del_files ---------------------------- */
/* ---------- Deletes the Temporary Files as required --------------- */
/* ------------------------------------------------------------------ */

void del_files(int low, int high) {

  int incr, result;
  char fname[20];

  /* Deletes files over requested range (corresponds to all files in the workspace */
  for(incr = low;incr <= high; ++incr) {
    sprintf(fname,"scan1.%i",incr);
    result = remove(fname);
    }
  for(incr = low;incr <= high; ++incr) {
    sprintf(fname,"scan2.%i",incr);
    result = remove(fname);
    }
  for(incr = low;incr <= high; ++incr) {
    sprintf(fname,"scan3.%i",incr);
    result = remove(fname);
    }
```

```
  for(incr = low;incr <= high; ++incr) {
    sprintf(fname,"scan4.%i",incr);
    result = remove(fname);
    }
  for(incr = low;incr <= high; ++incr) {
    sprintf(fname,"scan5.%i",incr);
    result = remove(fname);
    }
  for(incr = low;incr <= high; ++incr) {
    sprintf(fname,"scan6.%i",incr);
    result = remove(fname);
    }
  }


/* ------------------------------------------------------------------ */
/* ------------------------- Function Display_options ----------------*/
/* ---------- Presents Options to Display Spectra and Images --------- */
/* ------------------------------------------------------------------ */
void display_options(int current_scan) {
  int exitflag;

  /* Set up Screen and Display the Options */
  mouse_off();
  setfillstyle(11,MAGENTA);
  setcolor(CYAN);
  bar3d(240,90,510,250,3,3);
  button(2,302,110,448,130,"Display Options");
  button(1,290,140,460,160,"Linescan");
  button(1,290,165,460,185,"SAM Image");
  button(1,290,190,460,210,"Single Spectrum");
  button(1,290,220,460,240,"Cancel");
  mouse_on();

  /* Poll for User request, call appropriate Funciton */
  exitflag = 0;
  while (exitflag == 0) {
    /* Linescan Display */
    if ((max_line > 0) && (mouse_in_box(290,140,460,160) == TRUE) &&
(get_mouse_button() == LEFT) ) {
      line_display();
      exitflag = 1;
      }
    /* Single Spectrum Display */
    if ( (mouse_in_box(290,190,460,210) == TRUE) && (get_mouse_button() == LEFT) &&
(current_scan != 0) ) {
      single_display(current_scan);
      exitflag = 1;
      }
    /* SAM Image Display */
    if ( (mouse_in_box(290,165,460,185) == TRUE) && (get_mouse_button() == LEFT) ) {
      display_sam_image();
      exitflag = 1;
      }
    /* ABORT */
    if ( (mouse_in_box(290,220,460,240) == TRUE) && (get_mouse_button() == LEFT) ) {
      exitflag = 1;
      }
    } /* end while exitflag == 0 */
  } /* end function display_options() */

/* ------------------------------------------------------------------ */
/* ---------------------- function get_point ------------------------ */
/* -- General Function to acquire the CMA signal at a given energy --- */
/* -- and dwell time. Directly controls low-level counter functions -- */
/* ------------------------------------------------------------------ */
void get_point(double energy, POINT *point_data) {

  unsigned char low1, high1, old1;
  unsigned char low2, high2, old2;
  unsigned char low3, high3, old3;
  unsigned char low4, high4, old4;
  unsigned char low5, high5, old5;
  unsigned char low6, high6, old6;
```

```
int strobe, incr;
int dio_val;
int roll1, roll2, roll3, roll4, roll5, roll6;

long double number1;
long double number2;
long double number3;
long double number4;
long double number5;
long double number6;

long double start3;
long double start4;
long double start5;
long double start6;

double dac_voltage;
double offset;
double divisor;


roll1 = 0;  /* Reset Counter Roll Increment */
old1 = 0;   /* Reset Old Values for High Byte */
roll2 = 0;
old2 = 0;

roll3 = 0; /* Reset all roll Increment */
roll4 = 0;
roll5 = 0;
roll6 = 0;

dac_voltage = energy / 500.0;  /* Set DAC Value for this */
dac_out(dac_voltage,0);                /* Energy */


pokeb (base_id, 0x002E,0); /* Resets Counter1 to zero */
pokeb (base_id, 0x003E,0); /* Resets Counter2 to zero */

low3 = peekb (base_id, 0x0108);  /* Two consecutive reads gets you the low */
high3 = peekb (base_id, 0x0108); /* Byte then the high byte of the count */
start3 = (long double)((long double)(low3) + (256.0*(long double)(high3)));
old3 = high3; /* Saves the old High Byte to detect counter rollover */

low4 = peekb (base_id, 0x0109);  /* Two consecutive reads gets you the low */
high4 = peekb (base_id, 0x0109); /* Byte then the high byte of the count */
start4 = (long double)((long double)(low4) + (256.0*(long double)(high4)));
old4 = high4; /* Saves the old High Byte to detect counter rollover */


low5 = peekb (base_id, 0x010A);  /* Two consecutive reads gets you the low */
high5 = peekb (base_id, 0x010A); /* Byte then the high byte of the count */
start5 = (long double)((long double)(low5) + (256.0*(long double)(high5)));
old5 = high5; /* Saves the old High Byte to detect counter rollover */


low6 = peekb (base_id, 0x0106);  /* Two consecutive reads gets you the low */
high6 = peekb (base_id, 0x0106); /* Byte then the high byte of the count */
start6 = (long double)((long double)(low6) + (256.0*(long double)(high6)));
old6 = high6; /* Saves the old High Byte to detect counter rollover */

dio_val = (dwell * 10); /* Calculate value for Dwell time counter */
dio_write(dio_val);    /* Write the Dwell Time to the 16 bit Counter */
bg_start(10000);    /* Clock the timer and Gate Counters */

strobe = stba();         /* ---------------------------------  */
while (strobe == 0) {    /* Wait until Counters have Been Loaded */
  strobe = stba();       /* ---------------------------------  */
  }

while (strobe == 1) {  /* Loop Until End of Dwell time */
  strobe = stba();     /* Check for End of Dwell time */

  pokeb (base_id, 0x002F,0); /* Loads count1 value to offsets 0x0028 (LSB) */
                             /* and 0x0029 (MSB) */
  low1 = peekb (base_id, 0x0020);   /* Read Low Byte of Counter 1 */
  high1 = peekb (base_id, 0x0021);  /* Read High Byte of Counter 1 */
```

```
    if (high1 < old1) roll1++; /* Detect Rollover */
    old1 = high1; /* Saves the old High Byte to detect counter rollover */


    pokeb (base_id, 0x003F,0); /* Loads count2 value to offsets 0x0030 (LSB) */
                               /* and 0x0031 (MSB) */
    low2 = peekb (base_id, 0x0030);   /* Read Low Byte of Counter 2 */
    high2 = peekb (base_id, 0x0031);  /* Read High Byte of Counter 2 */
    if (high2 < old2) roll2++; /* Check for rollover */
    old2 = high2; /* Saves the old High Byte to detect counter rollover */


    pokeb (base_id, 0x010B, 0); /* Latch counter 3 */
    low3 = peekb (base_id, 0x0108);  /* Two consecutive reads gets you the low */
    high3 = peekb (base_id, 0x0108); /* Byte then the high byte of the count */
    if (high3 > old3) roll3++; /* Check for Rollover */
    old3 = high3; /* Saves the old High Byte to detect counter rollover */


    pokeb (base_id, 0x010B, 64); /* Latch counter 4 */
    low4 = peekb (base_id, 0x0109);  /* Two consecutive reads gets you the low */
    high4 = peekb (base_id, 0x0109); /* Byte then the high byte of the count */
    if (high4 > old4) roll4++; /* Check for Rollover */
    old4 = high4; /* Saves the old High Byte to detect counter rollover */


    pokeb (base_id, 0x010B, 128); /* Latch counter 5 */
    low5 = peekb (base_id, 0x010A);  /* Two consecutive reads gets you the low */
    high5 = peekb (base_id, 0x010A); /* Byte then the high byte of the count */
    if (high5 > old5) roll5++; /* Check for Rollover */
    old5 = high5; /* Saves the old High Byte to detect counter rollover */


    pokeb (base_id, 0x0107, 128); /* Latch counter 6 */
    low6 = peekb (base_id, 0x0106);  /* Two consecutive reads gets you the low */
    high6 = peekb (base_id, 0x0106); /* Byte then the high byte of the count */
    if (high6 > old6) roll6++; /* Check for Rollover */
    old6 = high6; /* Saves the old High Byte to detect counter rollover */

    }

    /* Write the Results to the 'point_data' structure */
    point_data->c1 = (long double)((long double)(low1) + (256.0*(long double)(high1)))
+ (65536.0*(long double)(roll1));
    point_data->c2 = (long double)((long double)(low2) + (256.0*(long double)(high2)))
+ (65536.0*(long double)(roll2));
    point_data->c3 = start3 - (long double)((long double)(low3) + (256.0*(long
double)(high3))) + (65536.0*(long double)(roll3));
    point_data->c4 = start4 - (long double)((long double)(low4) + (256.0*(long
double)(high4))) + (65536.0*(long double)(roll4));
    point_data->c5 = start5 - (long double)((long double)(low5) + (256.0*(long
double)(high5))) + (65536.0*(long double)(roll5));
    point_data->c6 = start6 - (long double)((long double)(low6) + (256.0*(long
double)(high6))) + (65536.0*(long double)(roll6));


  } /* end Function get_point() */
```