To the Graduate Council:

I am submitting herewith a dissertation written by Andrew Curtis Stephan entitled "Monte Carlo Simulation of Neutron Detectors." I have examined the final electronic copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Nuclear Engineering.

Laurence F. Miller

_____

Major Professor

We have read this dissertation
and recommend its acceptance:

Ronald Pevey

_____

Lawrence Townsend

_____

Thomas Thundat

_____

Accepted for the Council:

Anne Mayhew

_____

Vice Provost and Dean of
Graduate Studies

(Original signatures are on file with official student records.)

MONTE CARLO SIMULATION OF NEUTRON DETECTORS

A Dissertation Presented for the
Doctor of Philosophy Degree

The University of Tennessee

Andrew Curtis Stephan
December 2003

DEDICATION

This dissertation has a two-fold dedication. First, to my parents, Gerald and Nancy Stephan, who always sought the best for me. Second, to the much-doubted proposition that Truth exists and that it may be known.

ACKNOWLEDGEMENTS

Many persons have contributed in one way or another to the successful completion of this dissertation and I would be remiss if I did not mention them here. I would like to thank my committee, Drs. R. Pevey, L. Townsend, T. Thundat, and especially my major professor, L. F. Miller, for the time and effort they have invested in guiding me through the Ph.D. process. My thanks also to our Departmental Chairman, Dr. H. L. Dodds, Jr., support staff Gary Graves and Richard Bailey, and to our secretaries, Ellen Fisher and Kristen England. Ellen and Kristen always went out of their way to help me when difficulties arose, such as the main printer taking three hours to print a third of my dissertation before going nuts. Dr. J. F. Ziegler of the US Naval Academy did me a tremendous service in modifying the TRIM code to produce additional information that is critical to some of my simulations. Dr. R. G. Cooper of the Spallation Neutron Source at ORNL provided much assistance in the early stages of this project (prior to it becoming my dissertation project), including financial support. My parents urged me repeatedly to keep my eyes on the ball – finishing the dissertation – and that was a help to me. The congregation of Bible Presbyterian Church in Knoxville, TN and numerous other people such as the Resnicks, Unruhs, and Gardners were diligent in praying for me as I finished up this project. I would like to thank the many persons who have contributed to my education, both teachers in the formal educational system and friends, acquaintances, authors, and others who taught me important lessons through many other means. Lastly, I should mention that a great many people besides those mentioned above helped me through this process in one way or another. My thanks to you all.

ABSTRACT

Neutron detectors are simulated using Monte Carlo methods in order to gain insight into how they work and optimize their performance. Simulated results for a Micromegas neutron beam monitor using a custom computer code are compared with published experimental data to verify the accuracy of the simulation. Different designs (e.g. neutron converter material, gas chamber width, gas pressure) are tested to assess their impact on detector performance. It is determined that a ^{10}B converter foil and 1 mm drift gap width work best for a neutron beam monitor. The Micromegas neutron beam monitor neutronics are evaluated using the computer code MCNP. An optimized set of design criteria are determined that minimize neutron scattering probability in the device. In a best-case scenario, the thermal neutron scattering probability in the detector is $1.1*10^{-3}$. Lastly, composite neutron scintillators consisting of fluorescent dopant particles in a lithiated matrix material are simulated using a custom Monte Carlo code. The effects of design parameters such as dopant particle size, dopant volumetric concentration, and dopant and matrix material densities on scintillator characteristics are quantified. For ZnS:Ag particles in a lithiated glass matrix, it is found that dopant particle radii of 1 micron or less result in approximately Gaussian-shaped pulse height spectra and dopant particle radii of 5 microns or less result in practically all neutron absorption events producing scintillation light emission. Self-absorption of scintillation light is not treated in the simulation. Both the Micromegas and composite neutron scintillator simulations use the TRIM code as a heavy-charged particle transport engine.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION TO DISSERTATION

(a) Overview

This dissertation concerns the simulation of neutron detectors using Monte Carlo methods. The two types of neutron detectors simulated are the Micromegas, a gas-based parallel plate radiation detector with particular application to neutron beam monitoring, and composite neutron scintillators consisting of fluorescent dopant particles embedded in a matrix containing neutron target material [1-2]. It was decided to model these two particular neutron detectors for several reasons. First, the author has personal involvement in developing both of these devices. Second, many hitherto unresolved questions concerning the optimization of these detectors are answerable using these simulations. Third, these simulations are very novel and do not substantial duplicate prior published work.

Neutrons and the ability to detect them are important in materials structure research, homeland security, power plant operations, nuclear medicine, particle accelerators, and so forth. In many ways, neutron detection is more complex than gamma ray, electron, or heavy charged particle detection. Gammas behave in approximately the same way in different materials with attenuation nearly proportional to the density of the material through which they pass, and electrons and charged particles show the same tendency in that regard. Neutrons, by contrast, are highly sensitive to the isotopic makeup of the materials through which they pass, have interaction probabilities (cross sections) that can change very dramatically with only small changes in neutron energy, and frequently scatter many times with very large changes in direction prior to being absorbed. As these behavioral characteristics can make it very difficult to accurately calculate neutron transport and interactions (especially for non-homogeneous materials and

geometries) using deterministic methods, neutron transport is a very natural application of Monte Carlo techniques. Further, many of the events occurring inside a neutron detector after the interaction of a neutron inside the detector (e.g. transport of heavy charged particles from the neutron interaction, electron showers in gas-based detectors) are also very well suited to modeling using Monte Carlo techniques.

A brief overview of the Micromegas neutron beam monitor and composite neutron scintillators is as follows. The Micromegas is a parallel plate neutron detector with a solid neutron-to-charged particle conversion layer on one side of the gas chamber. A Frisch grid separates the gas chamber into a drift gap (several mm wide) where most of the electron-ion pairs are created and an amplification gap (~100 microns) where electrons liberated in the drift gap pass through a very strong electric field and create an electron shower. Anode readout strips collect the electrons in the electron shower and the spatial distribution of electrons across the anode readout strips indicates where the neutron absorption event occurred. Composite neutron scintillators consist of fluorescent particles embedded in a matrix containing neutron target material. Composite scintillators are more common than Micromegas devices; for example, ZnS:Ag particles suspended in lithiated epoxies are commonly used in thermal neutron detection. Composite scintillators allow fluorescent materials (e.g. ZnS:Ag) that do not contain neutron target materials to be used for neutron detection, a useful characteristic in a world in which only a small range of neutron scintillators exist. There are two major downsides to composite scintillators. First, their optical transparency is frequently limited due to poorly matched indices of refraction between the matrix and fluorescent particle materials. Second, if the size and concentration of the fluorescent dopant particles (based in part on their density) are not chosen judiciously, scintillation pulse amplitudes may vary widely and make reliable identification of events difficult. The first factor is reasonably well understood and may be characterized experimentally without excessive difficulty. The second factor is

very difficult to investigate experimentally but is readily investigated using simulations.

Both the Micromegas neutron detector and the composite neutron scintillator have a number of physical processes occuring within them that are random with a known probability density function and thus are eminently suited to modeling using Monte Carlo techniques. (Some examples are neutron absorption in the target material, heavy charged particle transport, particle distribution in the scintillator, and electron transport and showers in the Micromegas.) The value of modeling these two types of neutron detectors lies in the ability to predict changes in behavior as a result of changes in design parameters and to characterize the response of the detector to very specific neutron fields and neutron absorption events. Predicting detector behavior based on design parameters allows one to get a good idea of the performance characteristics of very large numbers of different possible device designs without actually building and testing very large numbers of prototypes, an approach that is usually not feasible. Comparison of the simulated behavior of different designs allows one to quickly determine which design paths should be pursued and which rejected. Response characterization is very useful in cases in which it is difficult to know precisely where and how a radiation particle interacts in a detector, this usually being true of neutrons. The simulation of both the Micromegas and the composite scintillator neutron detectors answers questions that cannot be easily or readily gleaned from experimentation alone. Thus, the simulation results described in this work provide useful information for persons seeking to design detectors based on these technologies.

(b) Organization

There are three major areas of work within this dissertation. The first involves simulation of a Micromegas neutron beam monitor using an original simulation code written expressly for this purpose. This simulation is concerned with what happens inside the neutron beam monitor from the point

that a neutron enters the device to the point that an output signal is generated. The theory behind this simulation is discussed in Chapter 3 and results obtained from this simulation are detailed and analyzed in Chapter 4. The second area involves the aspects of the Micromegas neutronics that do not contribute to production of a signal from the detector. This is effectively a study of the neutron scattering in Micromegas neutron beam monitors using the MCNP code and of how to minimize such scattering [3]. This is important because a well-designed neutron beam monitor must have the smallest possible effect on the beam being characterized by the monitor. Micromegas neutronics studies are described in Chapter 5. The third area of work is a study of the characteristics of composite neutron scintillators using a second custom simulation code. The theory behind this simulation is discussed in Chapter 6 and the results obtained from this simulation are detailed and analyzed in Chapter 7. Both the Micromegas and composite neutron scintillator simulations utilize the TRIM code as a heavy ion transport engine [4]. As a result, Chapter 2 is devoted to a brief overview of TRIM and the means by which it was used.

The order of work in this dissertation is slightly unusual in that the grouping of chapters is by topic rather than in the order of methods of investigation followed by results and discussion. (The conventional order of methods of investigation followed by results and discussion is followed within each of the three topical areas discussed in the dissertation.) This order is used to enhance reader comprehension of the material. The methods of investigation used in the three topical areas considered are quite different from one another, although all are based on Monte Carlo simulations. The properties being studied in each area are also very different from each other. Grouping material relating to each of the three topical areas together should make the dissertation much clearer to the reader and more readily comprehended.

(c) Novelty and Relevancy

The investigations described in this dissertation are all significantly novel; that is, no equivalent work has been published. The work in all three areas of investigation (Micromegas operation, Micromegas neutronics, and composite scintillator behavior) is also sufficiently interesting and useful to warrant publication in each case. An analysis of the utility and novelty of the three topical areas of investigation is as follows.

Limited simulations of the Micromegas have been done in support of some experimental results [1-2, 5-6]. Based on what is reported in the literature, the simulation described in this dissertation is significantly more comprehensive than previous work and can produce results that are significantly more detailed than those previously reported. It describes the physics of the Micromegas neutron detector from the neutron interaction in the detector to signal collection on the anode strips and may be used to predict the effects of changing many parameters that affect detector response. Such parameters include incident neutron energy and direction; neutron to charged particle conversion material type and thickness; detector gas composition, pressure, ionization energy, gain, and electron diffusion coefficient; gas chamber thickness; anode readout strip pitch; and detector dimensions. Responses affected include detector pulse amplitude, pulse topology across the anode readout strips, and position resolution.

The MCNP simulations of the Micromegas neutron beam monitor determine what materials are suitable for device construction from a neutronics standpoint. In designing a neutron beam monitor there is an inherent tension between minimizing neutron scattering in the detector, accomplished in part by minimizing the amount of material in the beam monitor, and producing a rugged device for operational use. To achieve the optimum Micromegas neutron beam monitor design, it is necessary that the neutronics consequences of various designs be known and that a design be selected exhibiting both low

disruption of the neutron beam and reasonable strength. While the simulations in this area do use MCNP as opposed to a custom Monte Carlo code, custom codes are used to automate the running of large numbers of MCNP cases and the analysis of the MCNP output files. This allows very large numbers of possible designs to be evaluated without excessive need for user intervention. There are no publications in the literature dealing with the effects of Micromegas design on transient neutron beams.

Finally, the composite scintillator simulation is significantly novel; there is no comparable work reported in the literature. Composite scintillators (mostly ZnS:Ag/^{6}Li) are widely used for neutron detection [7]. The investigations are significant because of the lack of information (until now) concerning the effect of critical design parameters (e.g. fluorescent dopant particle size and concentration) on the performance of such scintillators.

CHAPTER 2
ABOUT TRIM

(a) Reasons for Using TRIM

In developing the Micromegas and composite neutron scintillator simulations, it seemed most valuable to focus on creating new tools rather than functional equivalents of existing tools. Both types of neutron detectors contain target atoms (e.g. ^{6}Li, ^{10}B) for converting neutrons into energetic heavy charged particle reaction products, and in each case it is necessary to track these products as they move along and deposit energy. Since an accurate simulation of heavy charged particle transport is very complex to develop entirely from scratch and since simulation codes already exist, the TRIM code was used for heavy charged particle transport calculations [4].

(b) Introduction to TRIM

TRIM (Transport of Ions in Matter) is part of the SRIM (Stopping and Range of Ions in Matter) code package developed by Dr. J. F. Ziegler of the US Naval Academy located in Annapolis, Maryland [4]. The SRIM code package is based on work by J. F. Ziegler on stopping theory and by J. P. Biersack on range algorithms [4, 8]. Descriptions of the calculational methods used in SRIM are found in the book "The Stopping and Range of Ions in Solids" [4].

The SRIM manual includes the following description of SRIM:

> SRIM is a group of programs which calculate the stopping and range of ions (10 eV - 2 GeV/amu) into matter using a full quantum mechanical treatment of ion-atom collisions… This calculation is made very efficient by the use of statistical algorithms which allow the ion to make jumps between calculated collisions and then averaging the

collision results over the intervening gap. During the collisions, the ion and atom have a screened Coulomb collision, including exchange and correlation interactions between the overlapping electron shells. The ion has long range interactions creating electron excitations and plasmons within the target. These are described by including a description of the target's collective electronic structure and interatomic bond structure when the calculation is setup (tables of nominal values are supplied). The charge state of the ion within the target is described using the concept of effective charge, which includes a velocity dependent charge state and long range screening due to the collective electron sea of the target [9].

(c) Methods for Using TRIM

TRIM is capable of calculating many different parameters associated with ion transport through matter. For example, it can calculate:

(1) the stopping points of ions directed into a target material (and produce a graphical plot illustrating the path of the ions),

(2) sputtering from a surface,

(3) recoil atoms from ion-atom collisions,

(4) locations, directions, and energies of transmitted and backscattered ions from an ion beam directed into a layer of material of finite thickness.

At my request, Dr. Ziegler kindly provided a modified version of TRIM that produces an output file that lists the position, energy, and direction of simulated ions at periodic intervals of energy loss along their path [10]. Results that include a series of position-energy pairs at regular intervals along the path of an ion permits one to obtain a very good approximation of the energy transfer history of the ion. In the case of the studies described in this

dissertation, ions typically begin life with energies in the MeV range, and position-energy pairs are calculated at energy loss intervals of around 50 keV.

Several issues relating to how the TRIM calculations are handled should be noted as follows. In the case of the Micromegas neutron beam monitor simulation, the materials used in the TRIM calculations (e.g. ^{10}B, Ar/isobutane gas) are the same as those of which the Micromegas consisted (specified by the user). By contrast, the composite scintillator simulation uses a different method for ion transport. Travel distances of reaction product ions in 1 g/cm^3 silicon are divided by the density of the scintillator material (i.e. the matrix and fluorescent dopant materials) through which they travel. This method of scaling assumes that the ranges of the reaction product ions are inversely proportional to the material through which they travel. Since ion ranges usually scale as Z^2/A, this is a reasonable approximation for the scintillators considered as they largely consist of materials (e.g. O, S, Si, C) with Z numbers the same as or similar to Si.

Thus, ion-electron creation and electron migration are treated in the Micromegas simulation. As TRIM describes the trajectory of ions using a set of position-energy pairs, the creation of simulated ion-electron pairs in the detector gas is handled by randomly distributing the appropriate number of ion-electron pairs along each ion path segment between a set of position-energy pairs. By contrast, the scintillator simulation calculates the total energy deposited in the fluorescent (scintillator) dopant particles by each neutron capture event and uses the results from a large number of events to generate pulse height spectra and other data. The number of photons emitted by a scintillator is predicted from the energy deposited in the scintillating material, the average energy needed to produce a photon (this varies with the linear energy transfer, or LET, of the radiation particle depositing energy), and the Fano factor of the scintillator, which describes the statistical variance. Thus, the scintillator simulation does not include specifics on the number of ionizations and excitons formed in the scintillator nor on their behavior as this

is unnecessary for obtaining the desired outputs. Energy deposition in a fluorescent dopant particle is calculated by multiplying the fraction of an individual path segment (i.e. between two position-energy pairs) that falls within fluorescent dopant particles (adjusted for matrix and dopant material densities) by the energy lost within that path segment.

Ions traveling through matter sometimes produce recoil atoms, and the ions produced by neutron interactions in the Micromegas neutron beam monitor and in the composite neutron scintillator are no exception. The significance of recoil atoms in the simulations is investigated by using TRIM to find the rate of generation of recoil atoms in typical detector materials by common ions from neutron capture (e.g. alpha particles) that have energies consistent with production from ^{10}B and ^{6}Li (n, alpha) reactions. It is found that less than one in a hundred produce a recoil atom and that the energy of most recoil atoms is small (several tens of keV or less). It is determined that following recoil atoms is very time consuming in terms of the effort involved in adding them to the simulation and that their effect on calculated results is minimal. It was therefore decided to ignore recoil ions in the simulation.

When creating TRIM input decks in the Micromegas simulation for a new target material, ions that drop below 1 keV are discarded. This is done since calculating tracks for ions with very low starting energies results in ions that wander around in the simulated volume and never stop, presumably because their energy does not exceed the minimum for which TRIM can accurately simulate them. Since TRIM is only accurate for ion energies above 10 eV/amu and since in all of the cases considered (or likely to ever be considered) the final 1 keV of energy makes no appreciable difference to the final answer, this threshold is used to cease tracking ions. When an ion has an initial energy above the minimum that TRIM can accurately simulate and it remains in a single material, TRIM correctly finds its stopping location; thus particles are only dropped when a material boundary is crossed and a new TRIM run begun.

(d) Mechanics of Using TRIM

The details of how TRIM is used in the Micromegas simulation are quite complex. (The scintillator simulation is simpler for several reasons; for example, one standard target material is used.) Multiple TRIM input files are constructed based on the information provided by the user, after which TRIM runs are initiated. After TRIM has finished, the simulation code reads data from multiple input files to construct the path each particle followed and to determine what happened to it along the way. Since the Micromegas simulation includes the different regions in the detector and since ion transport in each layer of material is run as a separate TRIM calculation, TRIM is usually run numerous times in any individual Micromegas simulation run. This entails repeatedly setting up input decks based on the output of the previous TRIM run. TRIM numbers ions in its output files based on the order in which they appear in the input file, starting at one and increasing by one each time. This cannot be changed. Thus it is necessary to include a scheme for keeping track of each ion as its number changes to link it back to the correct initial neutron capture (ion-producing) event so that parts of different pulses are not inadvertently mixed together. To explain what is meant here, all of the ions of a particular type created by neutron capture events in a specific layer of material are run in a single batch TRIM run. As the ions travel through the detector, they end up in different regions composed of different types of materials. This results in TRIM input decks that may have, for example, ions number 2, 5, and 7 in them, but not 1, 3, 4, and 6. The potential for mistakes in ion identity can be appreciated when the information for ions 2, 5, and 7 is labeled as numbers 1, 2, and 3 in the TRIM output files and this process is repeated numerous times in any given simulation run. Since only one type of ion in one specific material is included in any single TRIM run, many particles must be banked while awaiting their turn in TRIM.

A detailed explanation of how to set up and use TRIM, including creation of the input files and interpretation of the output files, is found in the TRIM manual and is not described here [9]. Examples of important TRIM input and output files may be found in Appendix A. A very brief overview of the input and output files is as follows. (See Figure 2.1 for a diagram that illustrates this.)

The input file trim_win.in contains specifications concerning the characteristics of the target material, the characteristics of the ions impinging on the target, and the type of output desired. Examples of target material characteristics are the types of atoms present and their concentrations. Descriptive information for impinging ions includes the type of ion and its energy, initial location, and initial direction. Types of output can include recoil atoms, position-energy pairs for the ions, and other information as listed in Section (c).

If the file trim.dat is present, TRIM will use the information in it concerning the initial location, direction, and energy of the ions instead of using the information in trim_win.in. Trim_win.in allows only one specification to be given concerning the initial location, direction, and energy of the ions (necessitating that all ions start with these exact parameters). By contrast, trim.dat allows the initial parameters for each ion to be specified individually. Thus trim.dat is employed by the simulations.

When TRIM is run, it produces a number of output files concerned with the path and interactions of the ions, such as backscat.txt and transmit.txt, which record, respectively, the ions backscattering out of and transmitting through the layer of material in the simulation. Of special interest is exyz.txt which records the position-energy pairs for ions traveling through the material, this being the file from which the substantial majority of the ion path data is collected.

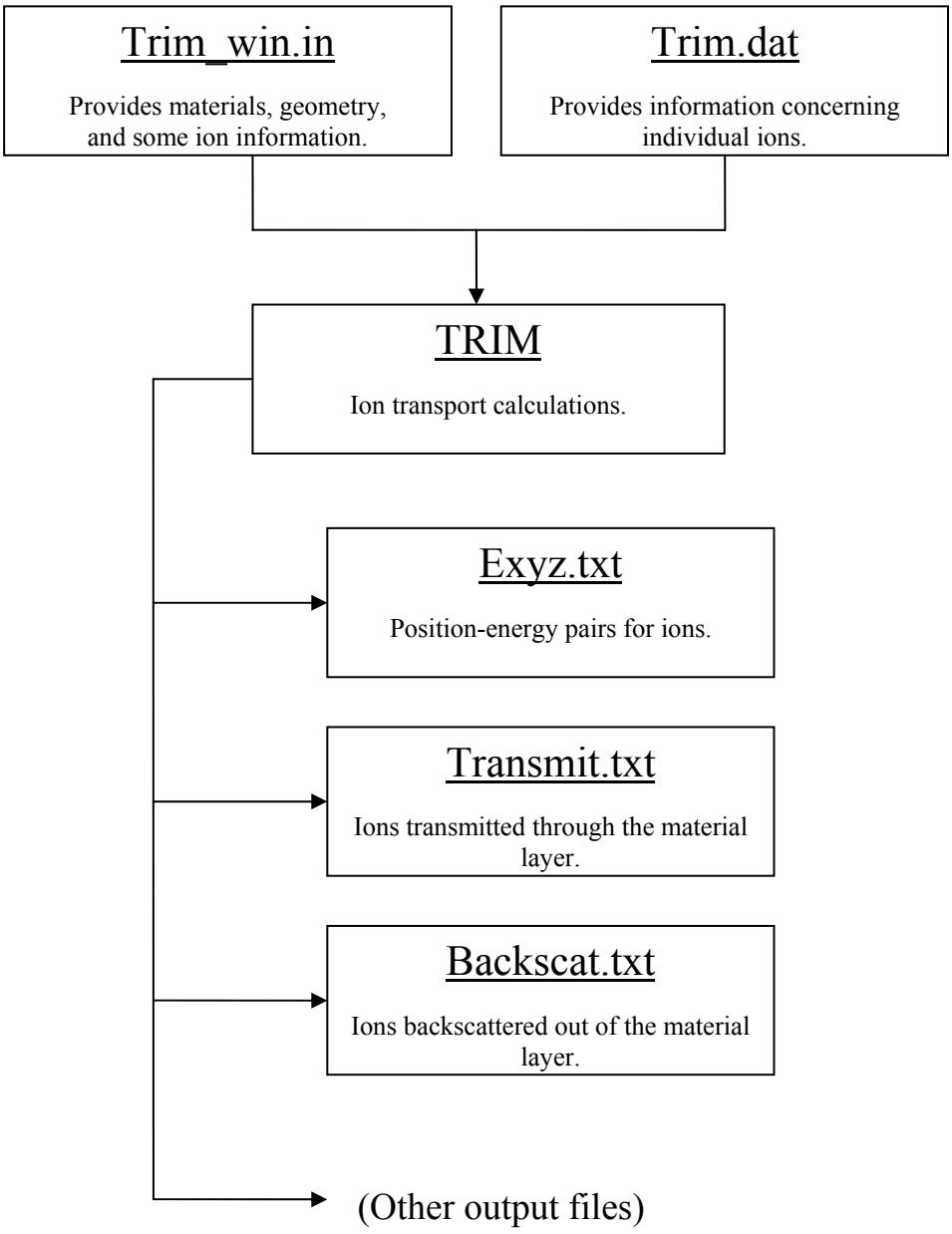


Figure 2.1. TRIM input and output files. Block diagram of the major TRIM input and output files used in the simulations and their relationship to TRIM.

CHAPTER 3
MICROMEGAS SIMULATION: MONTE CARLO METHODS

This chapter is heavily based on a paper entitled "Monte Carlo studies of a Micromegas neutron beam monitor" by A. C. Stephan (myself), R. G. Cooper, and L. F. Miller and accepted for publication in *Nuclear Instruments and Methods in Physics Research A*. Some portions of this chapter are taken from that paper verbatim (with some minor editing), while other portions are written from scratch. Drs. Cooper and Miller provided guidance to me during this project; otherwise, all work is my own.

(a) Introduction to the Micromegas Neutron Beam Monitor

History

The Micromegas (MICRO MEsh GAseous Structure) detector is a gas-based ionizing radiation detector first introduced in 1995 for use in high-rate, high-energy physics experiments [1]. After initial application to charged particle and x-ray detection, the Micromegas has been used for neutral particle detection, most recently for time-of-flight measurements for neutrons ranging from 1 eV to 250 MeV at CERN [1-2, 5-6, 11]. (CERN, the European Organisation for Nuclear Research, is located near Geneva, Switzerland, and is the world's largest particle physics research center.) Micromegas neutron beam monitors are currently slated for use at the Spallation Neutron Source (SNS) under construction at Oak Ridge National Laboratory (ORNL) in Oak Ridge, Tennessee.

Design and Design Considerations

The Micromegas design consists of two parallel plate electrodes separated by a gas chamber. (A diagram of the Micromegas layout may be found in Figure 3.1.) The gas chamber is in turn separated by a Frisch grid into two regions: a drift gap and an amplification gap. The Frisch grid is a

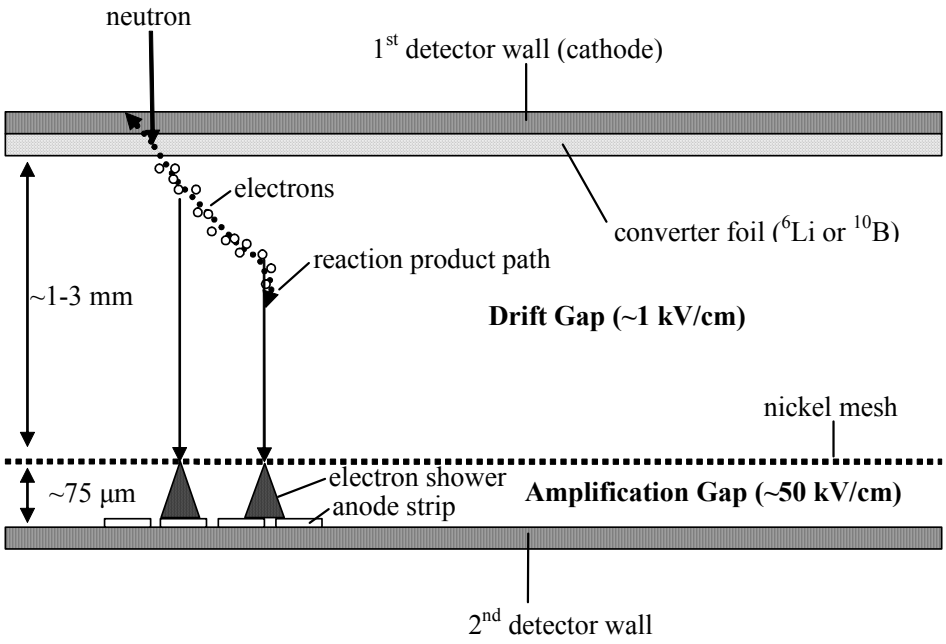


Figure 3.1. Micromegas neutron beam monitor layout, cross-sectional view.

wire mesh that acts as a third electrode between the anode and cathode and allows the electric field strength in the drift and amplification gaps to be set independently of each other. This design is very similar to a parallel-plate proportional counter, the primary difference being the use of the Frisch grid. The amplification gap, where gain occurs, is typically very small (~75 μm width) whereas the drift gap is several mm wide (where no gain occurs). The large region in which no gain occurs results in ionization pulses that are proportional to the energy deposited by an incident radiation particle in the drift gap. The separation of the drift and amplification gaps by the drift electrode (the Frisch grid) allows pulses to be generated solely from the motion of electrons and ions in the amplification gap and not in the drift gap, thus very fast pulses are produced that enable the detector to operate at a very high count rate due to the thinness of the amplification gap.

The Micromegas concept is quite flexible and can be used to detect many different types of ionizing radiation with the following caveats:

(1) The detector walls should be thin relative to the particle mean free path (neutrons, x- and gamma rays) or the particle range (charged particles).

(2) If spectroscopic information is desired, the width of the drift gap must be on the order of the range of the charged particles depositing the energy. (Neutral particles relinquish their energy via a small number of primary charged particles that produce many ionizations.)

(3) For particles traversing the detector from one side (e.g. charged particles), the drift gap width should ideally be equal to the track length of the particles. (This assumes the intent is to stop the particles as opposed to transmitting them.)

(4) For particles originating at random locations within the detector (e.g. x-rays), the drift gap width should ideally be large relative to the track length of the particles to maximize detection efficiency.

(5) As drift gap widths are decreased, particle energy resolution will fall and eventually be lost.

Neutron sensitivity in a Micromegas detector is achieved by placing a converter foil, a thin layer of neutron target material (e.g. ^{10}B), over the cathode that faces the gas chamber in the detector. When a neutron is absorbed in the converter foil, the target atom emits one or more reaction products, which are energetic ions that then travel a short distance as they relinquish their energy. Neutron-induced pulses are produced in the detector when a reaction product travels into the gas chamber in the detector and produces electron-ion pairs that travel across the gas chamber and are collected on anode readout strips.

Neutron Beam Monitoring

The purpose of a neutron beam monitor is to gather important information about the characteristics of the beam with minimal beam disruption. Typically beam intensity and distribution in space and time are used for normalizing data from experiments performed with the beam. At a pulsed neutron source facility, the time of arrival of the neutrons at the monitor may be used to determine their energy. The beam parameters that are considered important vary with the type of facility and experiment; for example, at non-pulsed neutron sources, total current fluence is more significant that neutron time distribution, while in other cases neutron spatial distribution information may not be needed. At the SNS, which produces a pulsed source, Micromegas neutron beam monitors are intended to be used where beam intensity and distribution in time and space are all important. Ideally, neutron beam monitors should capture only as many neutrons as are needed to obtain reliable statistics concerning the beam parameters and should avoid any effect on transient neutrons not captured for this purpose. In reality, scattering some of the transient neutrons and capturing others by means that do not produce a count in the monitor is unavoidable. Another unavoidable compromise concerns the neutron detection efficiency. In general, a detection efficiency that is independent of neutron energy is preferred. However, the 1/v dependence of neutron capture cross sections for many neutron target materials

(e.g. ^{10}B, ^{6}Li) means that if the incident neutrons are not monoenergetic, a compromise in detection efficiency must be made. The form of the compromise will most likely be the acceptance of higher neutron detection efficiencies than desired over certain energy ranges (and thus greater neutron beam attenuation). Another unpleasant reality that must be faced is that any in-beam neutron beam monitor must inevitably scatter some neutrons; thus, the question in design is how to minimize the probability of a neutron being scattered and what value is acceptable. Neutron absorption in monitor materials other than the neutron target material should be considered, although there are a number of materials available for making monitor walls and other components that have absorption cross sections that are practically zero.

The neutron beamlines at the SNS will have a very wide intensity range and accordingly the desired neutron detection efficiencies for the different beamlines vary from approximately $0.5x10^{-3}$ to 10^{-5}. The energies of the neutrons in the beams will vary from less than 10^{-4} to 10 eV, although each individual beamline will have a significantly narrower range than that. SNS staff would prefer to keep the neutron scattering probability in the beam monitors at or below the neutron detection probability, although this will often be unachievable as a realistic best-case is 10^{-3} scattering probability at thermal [12]. (See Chapter 5 for a detailed analysis.)

Utility of Simulation

Micromegas performance characteristics are strongly dependent on design parameters such as converter foil type and thickness, anode strip pitch, gas type, and drift gap width. While the effects of some parameter changes may be easily measured, others necessitate expensive detector modifications or very time-consuming data analysis. An accurate, comprehensive Monte Carlo Micromegas detector simulation is of considerable value as it allows some detector design modification effects to be investigated much more easily than through hardware changes. While simulation can never completely replace experimentation, it can provide valuable direction when seeking an optimized

detector design by indicating which combinations of design parameters are likely to perform particularly well and which are best avoided.

(b) Description of Micromegas Design

General Layout

Figure 3.1 shows a cross sectional diagram of the basic Micromegas neutron beam monitor design. The Micromegas design is much like that of a parallel plate proportional counter since it consists of a gas chamber located between two parallel plate electrodes with an electric field across the gas being responsible for the charge collection on the electrodes by which a signal is produced. It deviates from a parallel plate proportional counter in that it contains a Frisch grid (usually nickel mesh) that divides the gas chamber into two regions, a drift gap and an amplification gap. The drift gap between the cathode and the nickel mesh is typically several mm wide. The amplification gap between the mesh and the anode is 75-100 microns wide. The voltages on the three electrodes are set such that the field strength is around 1 kV/cm in the drift gap and 50 kV/cm or higher in the amplification gap. The significance of this is that gain occurs mostly in the amplification gap and pulses are very nearly proportional to the energy deposited in the drift gap (enabling the collection of spectroscopic information) due to the small size of the amplification gap compared with that of the drift gap.

Neutron Sensitivity

When employed as a neutron beam monitor, a Micromegas has a thin layer (a converter foil) of neutron target material (e.g. ^{10}B, ^{6}Li) deposited on the cathode. When an atom in the converter foil absorbs a neutron, it emits one or more energetic heavy charged particle reaction products (ions). The two most common reactions used (also the two that are of greatest interest to us) are as follows.

$$^{10}B + n \rightarrow {}^{7}Li + \alpha,\ Q = 2.310/2.792\ \text{MeV}\ 94\%/6\%\ \text{at thermal} \qquad (3.1)$$

$$^{6}Li + n \rightarrow {}^{3}H + \alpha,\ Q = 4.78\ \text{MeV} \qquad (3.2)$$

In the case of the ^{10}B reaction, the lesser Q value of 2.31 MeV that accompanies the majority of neutron absorptions at or near thermal is made up for by the ^{7}Li reaction product being produced in an excited state. The ^{7}Li quickly decays to a ground state by emitting a 0.482 MeV gamma ray.

Since the two reaction product ions are usually emitted in very nearly opposite directions, one ion will usually exit the converter foil and enter the drift gap. Reaction product ions usually travel in an approximately straight path with only small changes in direction as they interact with electron clouds around atoms, although they occasionally strike an atomic nucleus and undergo a large deflection. If the reaction product is sufficiently energetic to have a range in the gas that extends beyond the other side of the detector, it will exit the other side of the gas chamber, and if not, it will stop before reaching the other side. The range of the reaction product in the gas is a function of its atomic and Z numbers, initial energy, and direction of travel, along with some statistical variation due to underlying randomness in its interactions with the material through which it passes.

Signal Generation

As a reaction product travels through the drift gap it loses energy through the creation of electron-ion pairs in the gas. Under the influence of the electric field, the ions move towards the cathode and the electrons toward the anode. As the electrons travel through the drift gap, they undergo lateral diffusion. Additional lateral translation occurs when the electrons travel through the nickel mesh Frisch grid as the electrons follow the electric field lines that curve into and through the holes in the nickel mesh [6]. As the pitch of the nickel mesh is usually small compared to both the average lateral diffusion of the electrons and the pitch of the anode readout strips in a Micromegas neutron detector, the effect of the nickel mesh on position resolution is quite small.

After passing through the nickel mesh, the electrons enter the amplification gap. The high electric field in this region accelerates the

electrons, causing them to start electron avalanches that grow exponentially as the avalanches approach the anode readout strips. The electrons collect on the anode readout strips and produce a pulse that can be read on each anode strip individually. In most cases only a small fraction of the anode readout strips in a Micromegas detector collect electrons as a result of a single neutron capture event; that is, the area over which electrons are liberated by a reaction product and through which they travel on their way to the anode is small relative to the size of the detector. Position resolution is achieved by estimating the location of the neutron capture event using the topology of charge collection on the anode readout strips [1-2]. To date, Micromegas neutron beam monitor work has used anode readout strips, as opposed to other anode shapes. Anode readout strips give only 1D position resolution, although stacking two beam monitors against each other with their anode readout strips aligned at right angles will give some 2D resolution and this is being considered for some SNS beamlines [12]. Anode readout pixelation will give 2D; the main reason this has not yet been done being the complexity that the large increase in the number of anode elements from a 1D case would add to the detector design.

Neutron-Gamma Discrimination

Neutron-gamma discrimination in a Micromegas beam monitor is readily accomplished using a lower-level discriminator (LLD) set to reject detector pulses that do not exceed some minimum amplitude [2]. The LLD level is set such that gamma pulses produced by the detector fall below the LLD and most pulses produced by neutrons exceed this level. The amount of energy deposited in the detector drift gap as a result of a neutron capture event varies with the neutron target material and detector design and will be discussed in Chapter 4. For a 10 cm square Micromegas, an electron traveling in a straight line deposits at most around 250 keV of energy in the drift gap under the most favorable circumstances (electron path length in the gas is roughly 60 cm/MeV). The LLD used in experimental work is 235 keV [2]. Thus, neutron-gamma discrimination in a Micromegas device is much better

than from a neutron scintillator and is probably comparable to an He-3 proportional counter. Since an event may produce pulses on many different anode readout strips, the signal from the nickel mesh (essentially a sum of the signals on the anode readout strips) is normally used [2]. The ability to differentiate pulse origin (neutron or gamma) by pulse amplitude is based on two factors. Neutron capture reactions release much greater amounts of energy than gamma ray interactions. The higher LET of the neutron reaction products compared to electrons from gammas results in greater energy deposition in a thin target (i.e. the gas chamber) by the reaction product than by the electron. Pulse topology is strongly dependent on particle LET and the analysis of pulse topology independently of total pulse amplitude will also prove very effective in separating gamma-induced events from neutron-induced events. Further, combining these methods will yield very satisfactory results.

Pulse Duration

An important feature to note about the Micromegas design compared with other gas-based radiation detectors is that the division of the gas chamber into drift and amplification gaps dramatically speeds up the detector pulses and increases the maximum count rate. The reasons for this are as follows. Following the creation of electron-ion pairs in the gas by a reaction product, the electrons move much more rapidly than the ions. In a conventional gas detector (e.g. proportional counter), the detector pulse lasts as long as it takes for all of the ions created in the gas to reach the cathode; usually several microseconds at a minimum. In a Micromegas device, pulses can be generated solely from the motion of electrons and ions in the amplification gap and not the drift gap. (Differences in electron transit times in the drift gap have only a minor effect on pulse duration.) Since the amplification gap is quite small compared to the overall width of the gas chamber, Micromegas detector pulses are very fast compared with those of other gas-based detectors. Experimental work with Micromegas radiation detectors (not specific to neutrons) has shown that the component of pulse associated with electron drift in the amplification

gap is around one nanosecond and the component associated with ion drift around one hundred nanoseconds [5].

(c) Qualitative Effect of Design Parameters on Detector Response

The Micromegas neutron beam monitor Monte Carlo simulation is intended to provide quantitative data concerning the effect of detector design parameters on detector performance. General qualitative effects of design parameters on detector performance are predictable by persons familiar with the detector design and the underlying physics but are not sufficient when seeking to optimize a detector design. However, a qualitative understanding is important – if not essential – to properly comprehend quantitative data. This section discusses the qualitative effects of Micromegas design parameters on detector behavior in order to prepare the reader for the quantitative data that is presented in Chapter 4.

The type and thickness of converter foil used for neutron detection has a very significant impact on detector behavior for several reasons. The neutron absorption cross section of the material used in the converter foil (dependent on the type of material used), and its thickness, together determine the probability that an incident neutron will be absorbed in the Micromegas neutron beam monitor. Thicker converter foils will capture more neutrons, as will materials with higher neutron absorption cross sections. If converter foils become too thick, they will start to significantly self-shield the reaction products they emit and neutron capture events may fail to produce useful pulses in the detector. The type of converter foil material used is significant for another reason also. Upon absorption of a neutron, different neutron-sensitive materials emit different types of reaction products with different energies. Since reaction product ranges and LET values can vary dramatically with particle type, different converter foil materials can produce very different pulse characteristics in the detector.

The characteristics of the gas chamber in the Micromegas device are also very important. If the chamber is thin relative to the reaction product ranges in the fill gas, the amplitudes of many detector pulses will be constrained by the amount of energy a reaction product can deposit in the chamber before exiting the other side. (This is one means by which the type of reaction product can have a significant impact on the detector pulse characteristics.) By virtue of limiting reaction product track lengths in the gas chamber due to the products exiting the other side, a thin chamber also limits the average lateral motion of reaction products and by extension the spatial extent of the electron showers created, thus improving position resolution. As gas chamber width is increased, neutron pulses first increase in size and then level off as the chamber width becomes greater than the reaction product path lengths. Neutron position uncertainty follows a very similar pattern, although instead of leveling off it rises at a much slower rate due to lateral electron diffusion while crossing the drift gap. Since the electrons from gamma interactions have a much longer range than most neutron reaction products, gamma pulses continue to increase in amplitude well past the point at which neutron pulse amplitudes level off. Given all of the prior considerations, it is clear that some optimum gas chamber width exists at which the best combination of neutron and gamma pulse amplitudes and position resolution is achieved. It is also clear that this optimum gas chamber width will vary with the type of neutron reaction product (and therefore the type of converter foil used), gamma background (energy intensity), specific detector application, and other factors.

Other gas chamber characteristics affect detector behavior as well. For example, pressurized gases have higher densities than unpressurized gases of the same type. Thus increasing pressure decreases the range of reaction products in the gas, improving position resolution and, if the gas chamber is narrow, increases neutron pulse amplitudes. The main drawback to pressurization is the additional detector wall thickness needed and this

necessarily increases the probability that neutrons will scatter or be absorbed in the detector walls, which is an undesirable outcome. (Increases in neutron scattering probability with detector wall thickness may be calculated from data presented in Chapter 5.)

(d) Monte Carlo Methods Used in Simulation

Introduction

The Micromegas neutron beam monitor simulation uses Monte Carlo methods in a series of calculations that track an event from the point at which a neutron enters the detector to the point at which signals are produced on the anode readout strips. Many different physical processes occur along the way, all of which can be modeled with known physical laws with reasonable assumptions. Specific processes include:

(1) neutron interaction in the converter foil,
(2) reaction product creation,
(3) reaction product transport,
(4) creation of electron-ion pairs in the gas chamber,
(5) transport of electrons across the gas chamber to the anode readout strips under influence of the electric field in the detector,
(6) electron shower creation in the amplification gap,
(7) collection of electrons on the anode readout strips.

The Micromegas simulation includes neutron absorption in the converter foil but does not include neutron scattering. There are several reasons for choosing this approach. Calculations of neutron scattering can already be easily and very accurately accomplished with MCNP, as can neutron absorption in other detector regions (e.g. detector walls). The probabilities of neutron scattering and absorption in other detector regions are quite low when appropriate detector materials are chosen (around 10^{-3} at thermal, see Chapter 5) and thus the probability of affecting any individual neutron that goes on to be absorbed in the converter foil is quite low. Inclusion

of neutron absorption in the simulation – as opposed to coupling it to MCNP – enables the simulation to operate as a stand-alone product. It also significantly reduces the complexity of setting up and running the Micromegas simulation.

Neutron Absorption

The simulation uses neutron absorption cross sections for ^{6}Li and ^{10}B over an energy range from very cold to 20 MeV and takes advantage of the normal methods for calculating if and where a neutron absorption event takes place. For neutron transport (absorption only; scattering is neglected) through the neutron target material layer, the probability of neutron absorption is given by

$$p = 1 - e^{-l}, \tag{3.3}$$

where

$$l = \Sigma_a t / n_x, \tag{3.4}$$

in which Σ_a is the absorption cross section, t is the thickness of the converter foil, and n_x is the direction vector x component. (The variable l represents the distance through the converter foil in neutron absorption mean free path lengths.) Given a random number ξ (0,1), if $\xi < p$ then neutron absorption occurs. To obtain a correct distribution of the depth of penetration (x direction) in the target material by the neutron, the equation used to find the distance of penetration in the x direction is

$$x = \frac{-t \ln(1-\xi)}{l}. \tag{3.5}$$

Once the distance of x travel is known, calculating travel distances in the y and z directions is straightforward.

Since the probability that a neutron will interact in the Micromegas neutron detector can be very low (e.g. when being used as a neutron beam monitor), an option is included in the simulation to force neutron absorption in the target material (i.e. disallowing transmission). In this case the resultant particle weight becomes p from Equation (3.1) and the equation for travel in the x direction becomes

$$x = \frac{-t \ln(1 - p\xi)}{l}. \qquad (3.6)$$

The Micromegas neutron beam monitor simulation includes the ability to handle two layers of converter foil (and could readily be extended to n layers, if desired). When two layers are present and forced neutron absorption is not used, transport calculations are run for the first layer and then, if the neutron passes through the first layer without being absorbed, for the second layer. When forced neutron absorption is used, the total neutron absorption probability in both layers is given by Equation (3.3) but using instead for l,

$$l = (\Sigma_{a1} t_1 + \Sigma_{a2} t_2) / n_x, \qquad (3.7)$$

where the definitions are the same as for Equation (3.4) with subscripts 1 and 2 denoting the first and second converter foil layers. The probability of neutron absorption in the first converter foil layer is calculated using Equations (3.3) and (3.4). The probability of the neutron being absorbed in the first layer given forced absorption is given by

$$p_{1\,forced} = p_1 / p_{total}, \qquad (3.8)$$

where p_1 is the unforced probability of neutron absorption in the first layer and p_{total} is the unforced probability of neutron absorption in both layers. Given a random number ξ (0,1), if $\xi < p_{1,forced}$ then neutron absorption occurs in the first layer, otherwise in the second. A second random number ξ (0,1) is generated and the location of neutron absorption within the appropriate layer is determined after the manner of Equation (3.5) (with l and t for the chosen layer).

When a neutron is absorbed in the converter foil of the Micromegas neutron beam monitor, the atom that absorbs the neutron undergoes a nuclear reaction in which energetic reaction products are released. The two neutron target materials (i.e. converter foil constituents) of interest to us are ^{10}B and ^{6}Li. Their reactions are listed in Equations (3.1) and (3.2), respectively. When a low energy (e.g. thermal) neutron is absorbed, its contribution of energy and

momentum to the reaction is minimal and the reaction products are emitted in almost exactly opposite directions (which is necessary for conservation of momentum). When an energetic neutron is absorbed, its energy contributes to the energy of the reaction products (along with the Q value of the reaction) and its momentum alters the emission direction of the reaction product. To make the simulation accurate for neutrons above thermal energy, all of these effects are included in the simulation.

Results of the Neutron Absorption Reaction

The nuclear reaction follows a standard $a + b \rightarrow c + d + Q$ reaction, where a is the incident neutron, b is the target atom, c and d are the reaction products, and Q is the energy released in the reaction. One way to calculate the effect of the additional energy and momentum from the incident neutron on the reaction product directions and energies is as follows. The target atom and the neutron are transformed from the laboratory (LAB) coordinate system to the center-of-mass (COM) coordinate system and the total kinetic energy is determined in the COM system. The Q value of the reaction is then added and the two reaction products are given random but opposite directions in the COM system, their energies being determined by the law of conservation of energy. Following that, the reaction products are transformed back into the LAB system and their energies and directions in that system determined. This will correctly determine their energies and directions using the laws of the conservation of energy and of momentum while including the random nature of the direction of reaction product emission.

The speed $v_{a,lab}$ of the incident neutron and its individual directional velocity components $v_{ax,lab}$, $v_{ay,lab}$, $v_{a,labz}$ are calculated. The initial speed $v_{b,lab}$ of the target atom and its directional velocity components are all zero.

The speed of the center of mass (COM) of the neutron and the target is given by

$$v_{com} = (m_a v_{a,lab})/(m_a + m_b), \qquad (3.9)$$

where m_a is the neutron mass and m_b is the target atom mass. The x direction component of the COM velocity is given by

$$v_{com,x} = v_{com} n_{a,x}, \tag{3.10}$$

where $n_{a,x}$ is the x component of the unit direction vector for the neutron. The y and z components are calculated in the same fashion using the obvious substitution.

The velocities of the neutron and target atom are then transformed from the LAB system to the COM system using the velocity components of the COM system relative to the LAB system that were just found. Thus

$$v_{ax,com} = v_{ax,lab} - v_{com,x} \tag{3.11}$$

and

$$v_{bx,com} = v_{bx,lab} - v_{com,x}. \tag{3.12}$$

Equivalent transformations are made for the y and z speeds. Finally, the total speeds of the particles are calculated by

$$v_{a,com} = (v_{ax,com}^2 + v_{ay,com}^2 + v_{az,com}^2)^{0.5}. \tag{3.13}$$

At this point it is necessary to know the type of reaction that takes place (see Equations (3.1) and (3.2)), the mass of the products (symbolized by m_c and m_d), and the Q value of the reaction. For the ^{10}B reaction a constant 94%/6% branching probability between the 2.310 and 2.792 MeV Q values is assumed. This is a simplification as in actuality the branching probability varies with the neutron energy.

Next, the total energy that the reaction products will have in the COM system after the reaction is calculated by summing the kinetic energies of the neutron and target atom and adding the Q value, thus

$$E_{com} = 0.5 m_a v_{a,com}^2 + 0.5 m_b v_{b.com}^2 + Q. \tag{3.14}$$

The speeds of the two reaction products produced by the nuclear reaction are governed by the laws of the conservation of energy and of momentum and are given by

$$v_d = ((2E_{com}m_c)/(m_d^2 + m_c m_d))^{0.5} \quad (3.15)$$

and

$$v_c = (m_d / m_c)v_d \,. \quad (3.16)$$

It is then straightforward to calculate the energies E_c and E_d. If the energy of the incident neutron is minimal (e.g. in the thermal range), E_{com} will be minimal and the laws of the conservation of energy and of momentum require that energy be distributed as follows: for ^{10}B with 2.310 MeV Q, 1.47 MeV is acquired by the α (^{4}He) particle and 0.84 MeV is acquired by the ^{7}Li; for ^{10}B with 2.792 MeV Q, 1.777 MeV goes to the α and 1.015 MeV goes to the ^{7}Li; and for ^{6}Li, 2.73 MeV to the ^{3}H and 2.05 MeV to the α.

The next step in the process is to determine a random direction vector for the reaction products. Random x, y, and z vector components are picked after the manner

$$n_{cx} = 2\xi - 1 \,, \quad (3.17)$$

where ξ is a random number (0,1). If the sum of the squares of the vector components is less than or equal to one, the vector components are normalized to a unit vector (i.e. the sum of their vector components is one), otherwise new sets of vector components are generated and tested in the same manner until one is found satisfying that condition. This method picks a direction vector by finding a random location within a sphere and finding the direction vector pointing from the center of the sphere to the random location, thus giving an isotropic distribution for random direction vectors. Once a unit random direction vector is found, it is assigned to one of the reaction products and its exact opposite to the other reaction product, giving $<n_{cx}, n_{cy}, n_{cz}>$ and $<n_{dx}, n_{dy}, n_{dz}>$, where $n_{dx} = -n_{cx}$ and so forth.

Knowing the speeds of the two reaction products, the directional components of their velocities are calculated after the manner

$$v_{cx,com} = v_{c,com} n_{cx} \,. \quad (3.18)$$

Knowing the directional components of the reaction products in the COM system enables them to be transformed into the LAB system after the manner

$$v_{cx,lab} = v_{cx,com} + v_{com,x}, \tag{3.19}$$

from which we obtain the speeds in the LAB system after the manner

$$v_{c,lab} = (v_{cx,lab}^2 + v_{cy,lab}^2 + v_{cz,lab}^2)^{0.5}. \tag{3.20}$$

It is then straightforward to calculate the unit direction vectors and energies of the reaction products in the LAB system.

Energy Deposition by the Reaction Products

Following creation of the reaction products, an input file is created for each type of reaction product and, if two converter foil layers are present, a set of two files is created for each layer. TRIM calculations are then run to determine where the reaction products travel and how the energy they relinquish is distributed. This process is described in greater detail in Chapter 2. Information regarding energy deposition in the gas chamber is extracted from TRIM output files and passed back to the Micromegas simulation in the form of individual segments of reaction product tracks. These segments consist of two points (the track is assumed to be a straight line between the points) and an amount of energy that the reaction product lost while traveling between the two points, usually around 50 keV.

Free Electron Creation and Migration

To simulate the creation of electron-ion pairs, the number of ionization events is calculated using the ionization energy of the gas and statistical variations added. Thus the number of ionizations is

$$n_{ionizations} = \Delta E / e_{ionization} + \xi(\Delta E / e_{ionization})^{0.5} \tag{3.21}$$

rounded to the nearest integer, where ξ is a random number of standard deviations picked according to a two-tailed normal distribution. This is a standard method for estimating statistical variations in the number of information carriers in radiation detectors [7]. In cases where $e_{ionization}$ is a

significant fraction of ΔE, the calculated $n_{ionization}$ may be less than zero, in which case it is taken as zero.

Once the number of ionizations has been determined, locations are selected for each ionization event. Ionizations are assumed to occur randomly along each track segment. (Changes in reaction product LET over energy drops of 50 keV are fairly small so an equal probability of ionization was used along the entire track segment.) Coordinates in x, y, and z for each individual ionization event location are found after the manner

$$x_{ionization} = x_s + \xi \Delta x, \tag{3.22}$$

where x_s is the start point, Δx is the track length in the x direction, and ξ is a random number in the interval (0, 1). The obvious substitutions are made for y and z and the same ξ is used in each case.

As the electrons cross the gas chamber, they drift laterally. Given the electron diffusion coefficient in units of microns/√cm (obtained from the literature) or an equivalent, the lateral diffusion for an individual electron is given by

$$\Delta l = \xi d_l \sqrt{\Delta x}, \tag{3.23}$$

where ξ is a random number of standard deviations picked according to a two-tailed normal distribution, d_l is the diffusion coefficient, and Δx is the distance traveled by the electron from the ionization location to the nickel mesh [6]. (The x axis runs directly through the detector perpendicular to the planar electrodes in our coordinate system.) Final electron locations are given by

$$y = y_s + \Delta l \sin(\xi 2\pi) \tag{3.24}$$

and

$$z = z_s + \Delta l \cos(\xi 2\pi), \tag{3.25}$$

where y_s and z_s are the y and z start locations of the electron and ξ is a random number (0, 1), with the same ξ being used for both equations.

Once electrons reach the nickel mesh side of the amplification gap, they are sorted according to the anode readout strip towards which they are

headed and the total number of electrons thus associated with each strip prior to gas multiplication is determined. As discussed earlier in this chapter, some lateral translation of electrons also occurs when the electrons pass through the nickel mesh [6]. This lateral translation is quite small compared to the anode readout strip pitch used or anticipated to be used in Micromegas neutron beam monitors and thus was judged to make little difference in the electron distribution among the strips.

Gas Amplification

Gas gains of around 100 are most appropriate for Micromegas neutron detectors given the number of primary ionizations produced by the neutron reaction products. No publications were located that define the appropriate statistical fluctuation that should be added to the number of electrons generated in the electron shower striking each targeted anode strip; thus, simple simulations were performed to estimate an appropriate statistical fluctuation as follows. The distance between the nickel mesh and the anode readout strips (normally 75-100 microns) was divided up into individual layers of one micron, and the average number of electrons that leave the layer per electron that enters the layer was calculated based on the gas gain (supplied by the user) and on the number of such layers. (The one micron layers in the amplification gap did not correspond to any physical structure but were selected as evenly spaced points at which secondary electron liberations could be calculated.) Thus, this multiplication constant for electrons in each layer is given by

$$c = g^{1/n}, \tag{3.26}$$

where g is the gas gain and n is the number of one micron layers in the amplification gap (an integer number). For the gains and amplification gap thicknesses considered, c was quite close to one, thereby ensuring that the omission of cases where more than one secondary electron was created would have a minimal effect on the estimates of statistical fluctuation. Another advantage of having c close to one was that the probability that an electron

would create a secondary electron while traversing a one micron layer could be readily calculated using the equation

$$p = c - 1. \tag{3.27}$$

Given this probability, a group of electrons can be followed as they pass through the amplification gap and each electron can be tested by using p and a random number ξ (0, 1) to see if it created a secondary electron while traversing each one micron layer. (Secondary electrons created in each layer are added to the primary electrons before being tested to determine their effect in the subsequent layers.) Repeated runs are made for different cases with starting numbers of electrons between 100 and 10000 and gains of 100. It is found that the following expression provides a good empirical predictor of $n_{e,a}$, the number of electrons reaching the an individual anode strip:

$$n_{e,a} = (1+g)n_{e,n} + 8\xi\sqrt{gn_{e,n}}\,, \tag{3.28}$$

where $n_{e,n}$ is the number of electrons entering the amplification gap at the nickel mesh aimed at that anode strip, and ξ is a random number of standard deviations picked according to a two-tailed normal distribution. (All other terms have been previously defined.)

Simplifications

Several simplifications are incoporated into the simulation. First, it is assumed that no crosstalk occurs between the electron showers aimed at each anode. In reality some electrons do end up being collected on an anode strip adjacent to the anode strip at which they were initially aimed. However, the probability of any individual electron doing this is low for several reasons. First, the distance between the mesh and the anode readout strip (75-100 microns) indicates a typical lateral diffusion of no more than a few tens of microns if the diffusion constants in the amplification gap are the same as in the drift gap. Second, investigations of the electric field shape between the mesh and the anode strips and its effect on transient electrons indicate that very little electron crosstalk between strips will take place [13]. Third, most

collisions by electrons while crossing the amplification gap are electron-electron collisions; that is, transient electrons collide with electrons associated with gas atoms in the amplification gap. Electron-electron collisions are highly forward biased due to the equal mass of the two particles, thus typically limiting the extent to which an electron acquires a lateral motion component as a result of such a collision. Fourth, were an electron to acquire a significant lateral motion component, its lateral travel distance would not be all that great. The potential difference between the mesh and the anode readout strips is typically around 500 V and most electrons will undergo several or more collisions while crossing the amplification gap, thus usually preventing their energies from building up to more than several hundred eV. This energy corresponds to a range in the gas on the order of a couple of hundred microns, which is significantly less than the average width of an anode readout strip. However, another type of indirect crosstalk does occur between the anode readout strips. Previous experimental work has demonstrated that the capacitance between adjacent anode readout strips is sufficient for signals to be induced on anode strips that collect no electrons from the event itself by nearby strips that do collect electrons, which increases the measured spatial extent of the electron shower [2]. No attempt is made to include this in the simulation. Finally, some noise is present in the signals on the anode strips, including strips that are not affected by the neutron capture event. This is not included in the simulation.

Use of Simulation

Following the simulation methods outlined in this section enables one to generate information describing the distribution of electrons collected on a series of anode readout strips (an anode strip topology) as a result of a neutron capture event in a Micromegas neutron beam monitor, given certain information about the design of the device. To understand how changes in design affect the detector response, many simulation runs of different designs must be run and the predicted behaviors compared. This is done in Chapter 4

where such parameters as position uncertainty and probability of the detector registering a count given a neutron absorption event are studied as a function of design. The simulation enables many different aspects of Micromegas neutron beam monitor design to be studied besides those covered in this work and many opportunities exist for follow-up work. For example, although anode strip topologies are already used for neutron capture location, more sophisticated analysis methods will enable determination of radiation particle type, improve position resolution, and so forth. Large numbers of events and the anode strip topologies they produce can be evaluated to identify correlations for use in estimating event parameters (e.g. neutron absorption location) from anode strip topologies.

(e) Simulation Code Organization

The Micromegas simulation is broken into five parts. The form MicromegasSim.frm contains the code associated with interacting with the user to gather all necessary information before running the simulation, calculating neutron interactions and reaction products created thereby, and setting up some of the initial input files for TRIM. The TrimAutomation.bas module contains all of the code associated with calling TRIM to process the input decks created by the EntryForm form and then analyzing the TRIM output files, including creating new TRIM input files from the data in the TRIM output files as needed until all the reaction products have either exited the Micromegas or stopped. TrimAutomation creates its own output files containing lists of reaction product segments (including energy loss along each segment) and which neutron capture event each segment was associated with. The module AnalyzeTrimData.bas combines all of the output files from TrimAutomation into a single file and then processes each event sequentially, processing including ionization events in the gas and electron transport to the nickel mesh. The module PulsesUserEditable.bas is, as its name implies, intended to be easily changed by the user if desired. For example, after being

passed information on the arrival location of each electron at the nickel from AnalyzeTrimData, PulsesUserEditable sorts electrons into their corresponding anode readout strip and then creates an electron shower at each strip according to the gas gain specified by the user. If a Micromegas with 2D position sensitivity (i.e. anodes divided into pixels rather than strips) is simulated, PulsesUserEditable could be changed to accommodate this. PulsesUserEditable would also be the appropriate code to add lateral electron translation due to the curvature of the electric field around the nickel mesh and through the holes in the mesh, crosstalk between anode readout strips, and background noise not related to the neutron capture event itself. Finally, the module MicromegasSimModule.bas contains declarations of global variables and constants. (These types of declarations are required to be in modules. Also, separating the global variables and constants in such a manner makes it easier to keep track of them.)

(f) Code Verification by Testing Calculational Methods

There are two aspects to code verification and validation. Verification involves testing of routines in the code to verify that they are indeed doing what they are intended to do. Validation involves comparing simulated results with real, measured data, or, in the absence of any such data, performing simulations for which expected results may be readily calculated by hand. Since there are many random processes that take place between the point at which a neutron enters a Micromegas neutron beam monitor and the point at which a signal is read out of the device, hand calculation of expected results for a simple case is not feasible. Experimental results are, however, available in the literature and validation through comparison with simulated results is dealt with in Chapter 4.

Significant effort is expended in verifying the code by testing individual routines in the code dealing with critical aspects of the simulation calculations. For example, routines for determining lateral drift of electrons

were repeatedly tested to ensure electron transport routines are functioning correctly. In some cases, checks are built into the code such that if unanticipated events or inconsistent variable values are encountered, an error indicating the nature of the problem is returned to the user and code execution ceases. The simulation code was tested during its development to quickly expose errors when they occurred and to allow problems to be corrected quickly.

CHAPTER 4
MICROMEGAS SIMULATION: RESULTS AND DISCUSSION

This chapter is heavily based on a paper entitled "Monte Carlo studies of a Micromegas neutron beam monitor" by A. C. Stephan (myself), R. G. Cooper, and L. F. Miller and accepted for publication in *Nuclear Instruments and Methods in Physics Research A*. Some portions of this chapter are taken from that paper verbatim (with some minor editing), while other portions are written from scratch. Drs. Cooper and Miller provided guidance to me during this project; otherwise, all work is my own.

(a) Introduction

The usefulness of a Micromegas neutron beam monitor simulation is a direct function of its ability to accurately predict how the device will perform, given a set of design criteria. Thus, a significant portion of this chapter is given over to a comparison between experimental results from a Micromegas neutron beam monitor (acquired by a research group developing instrumentation for CERN) and results from the Micromegas simulation that were generated using the same design criteria. (The team that built and tested the Micromegas neutron beam monitor prototype is affiliated with a number of different research institutions in three different countries. Since the device is intended for use at CERN, I hereafter refer to them as the CERN group.) Since the number of different possible designs (e.g. converter foil thickness, gas chamber width, anode strip pitch) is practically limitless, the design used by the CERN group is used as a starting point for calculations of the effects of changing various design parameters [2]. Since these calculations are intended to be helpful to the Micromegas neutron beam monitor development project underway at ORNL and since the energy of the SNS neutron beams at which the Micromegas neutron beam monitors will be used over the cold to 10 eV

range, calculations are restricted to neutron energies falling within that interval.

(b) Validation by Comparison with Experimental Data

Design Parameters for Simulation

Simulation results for the neutron beam profiler (monitor) design tested at CENBG (Centre d'Etudes Nucleaires de Bordeaux-Gradignan), intended for use at the n_TOF (neutron time-of-flight) facility at CERN, are compared with published experimental data as described below. Design parameters of the experimental device included an active area of 12 cm x 14 cm, a 500 nm ^{6}Li converter foil, a 3 mm drift gap filled with 90/10 Ar/isobutane gas, and an anode readout strip pitch of 317.5 microns. Several different gas gains are tested using different voltage settings on the detector electrodes. A variety of neutron energy ranges in the hundreds of keV are used for testing [2]. To generate simulated results that are comparable to those used by the CERN group and are useful for the SNS neutron beam monitor design work, the following design parameters are used in the simulation. Detector walls consisting of 50 microns of Al are placed on either side of the detector, allowing backscatter of ^{6}Li reaction products into the detector. A detector size of 10 cm x 10 cm is used since many SNS neutron beamlines are to be that size. The 3 mm drift gap is filled with Ar/isobutane at 1 atm pressure, the anode strip pitch is 317.5 microns, and the gas gain is 100. The gas is taken as having a 22.4 eV ionization constant and a 370 microns/√cm diffusion constant based on published data [6, 14]. All incident neutrons are assumed to be thermal and normally incident. These parameters are almost the same as the parameters used in the published experiments. The only differences likely to affect results are the detector size and neutron energies. In the former case, size only affects the magnitude of the edge effects in the detector and the difference in edge effects between detector sizes of 12 cm x 14 cm and 10 cm

x 10 cm will be quite small. In the latter case, the energy of the neutrons is, in any case, only a fraction of the ^{6}Li Q value, which will produce fractional increases in pulse size and in the probability that both ^{6}Li reaction products enter the gas chamber.

Anode Strip Topologies

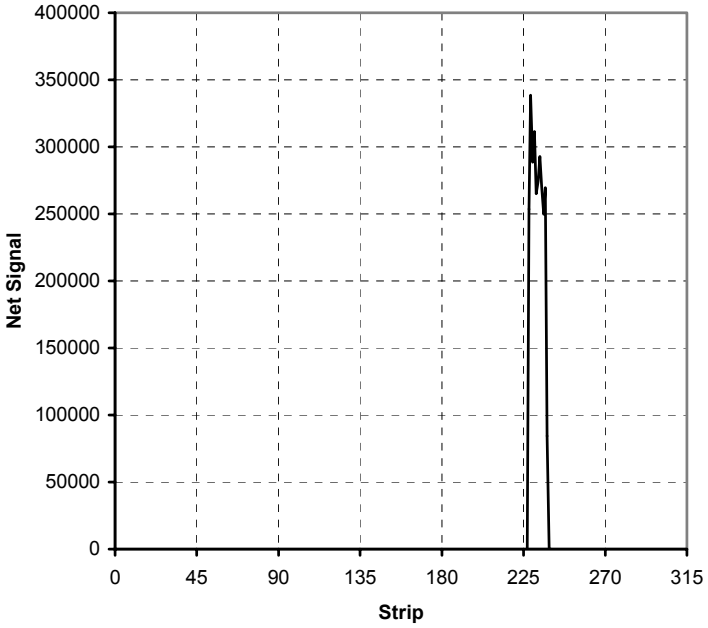
Neutron capture events in which one or both ^{6}Li reaction product particles enter the drift chamber result in charge collection on the anode readout strips. The collection of electrons on the anode strips (the number collected and their distribution across the anode strips) is the defining characteristic of an individual event from the standpoint of the signal output by the detector. Estimates of neutron capture event location and time and discrimination between genuine neutron capture events and false events (e.g. gamma interactions) are made on the basis of the pattern of the electron deposition on the anode strips (the anode strip topology) or an aggregate of the electron deposition on the strips, the total pulse amplitude. Since, in essence, all other measurements flow out of the anode strip topologies, the ability of the simulation to generate anode strip topologies that accurately depict what happens is essential to the validity of the simulation.

The number of electrons collected, and their distribution on the anode strips, varies greatly from event to event and is most heavily influenced by the type of particle entering the drift gap and its direction. Particles entering the drift gap at a shallow angle and perpendicularly to the anode strips maximize both the number of anode strips producing a signal (i.e. that collect electrons) and the total charge collected while minimizing the signal on each strip. Particles traveling straight across the drift gap minimize both charge collection and the number of anode strips producing a signal. Particles traveling at a shallow angle parallel to the anode strips maximize charge collection while producing signals on only a few anode strips. The type of particle that enters the drift gap has a strong influence on the signal topology. Tritons have a

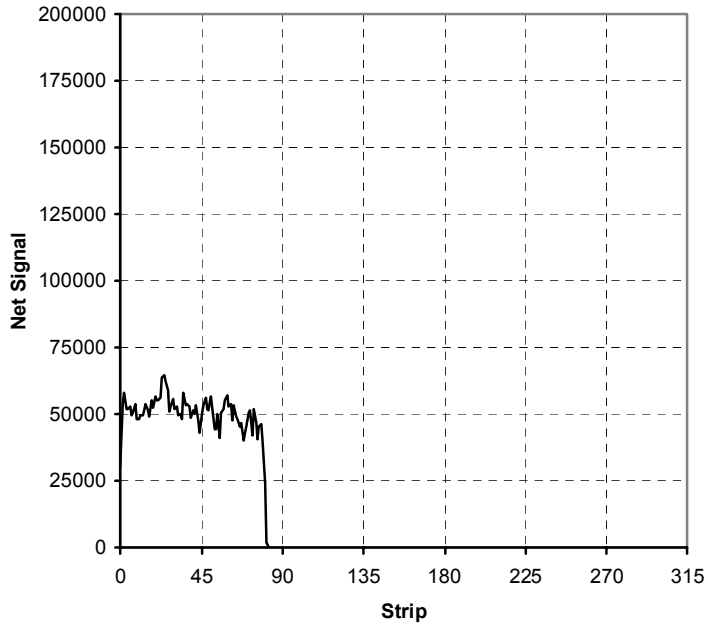
much greater range than alpha particles and are correspondingly much less densely ionizing, thus tending to produce signals on a very large number of anode strips if entering the drift gap at a shallow angle and to produce small pulses if traveling directly through the drift gap. Figure 4.1 shows the anode strip charge collection distribution for four typical event types. Note the Bragg peak in (c) produced by a reaction product passing through its energy range of maximum stopping power. (Stopping power is the closest descriptive term but is not completely accurate as secondaries are not followed.) These four charge collection distributions are very similar to the experimental results in the paper by Andriamonje et al which are reproduced in Figure 4.2 [2].

Q vs. FCS Plots

Using the anode strip signals (see Figure 4.1 for examples), single neutron capture events can be quantified by the total charge (cluster charge) collected on the anode strips, Q, and by the number of anode strips registering a signal, the Fitted Cluster Size (FCS). Since the FCS is determined by applying a fitting function to the topologic anode signal data, it varies for the same topologic data with the type of fitting function used. In the paper by Andriamonje et al, a convolution of a Gaussian and a step function is fitted to the anode strip signals to produce an FCS value [2]. In this work, a fraction method is used whereby a strip was counted if it collected more than a given fraction of the maximum number of electrons collected on any strip. As Micromegas neutron detectors will likely have gains of 10^2 or more and large numbers of electrons are collected on the strip with the largest signal, a fraction of one one-thousandth is used. Figure 4.3 shows the resulting plot of total charge collected (Q) vs. FCS. The triton and alpha particle signals are well separated with only a few triton events appearing in the predominantly alpha region and no alpha events appear in the triton region. Tritons on average deposit much less energy in the drift gap than alphas due to the much longer triton range, although the largest triton pulses are bigger than any alpha pulses due to the higher initial energy of the triton. Approximately 0.6% of the

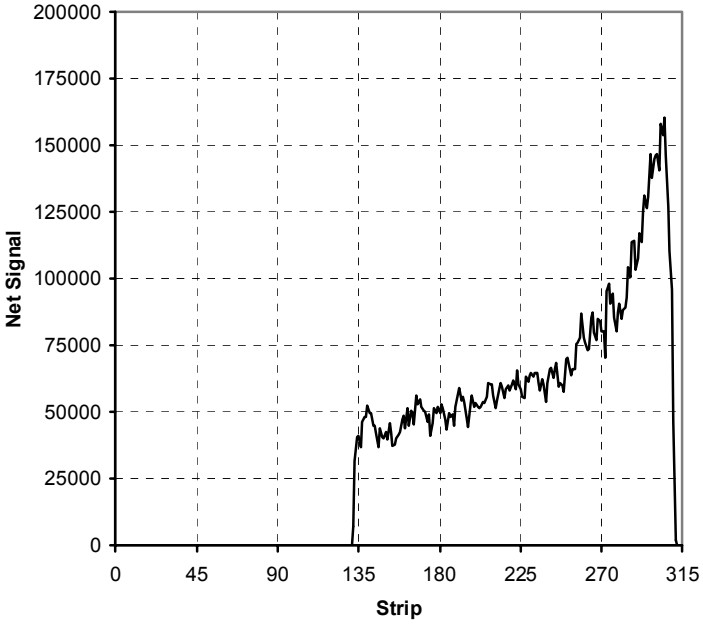


(a)

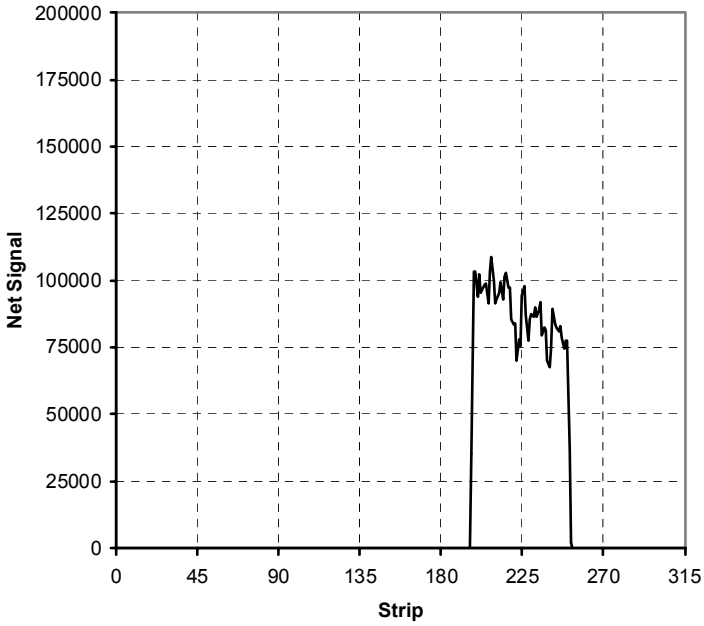


(b)

Figure 4.1. Simulated typical event types; net signal (electrons collected, relative scale) vs. anode strip number. Micromegas with ^{6}Li converter foil. Compare with corresponding measured typical event types in Figure 4.2.

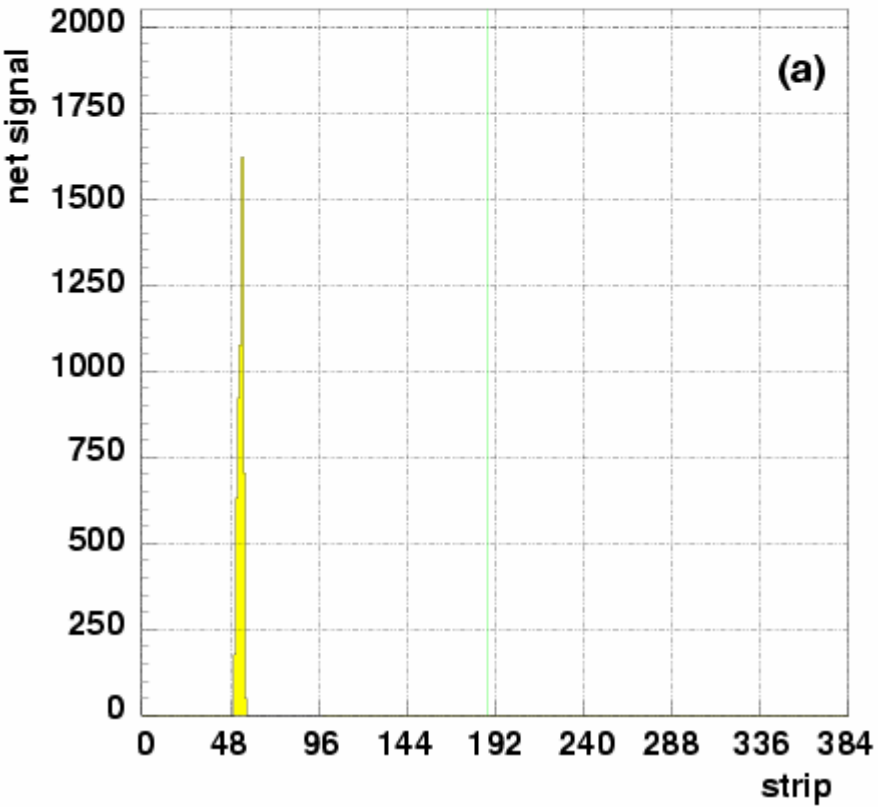


(c)



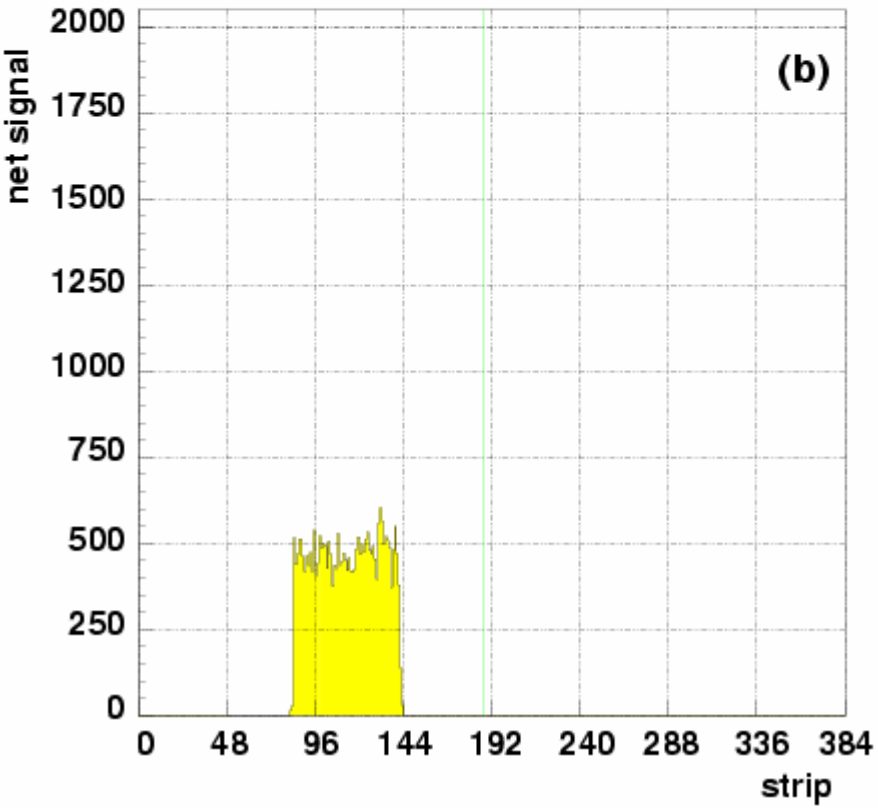
(d)

Figure 4.1 Continued.



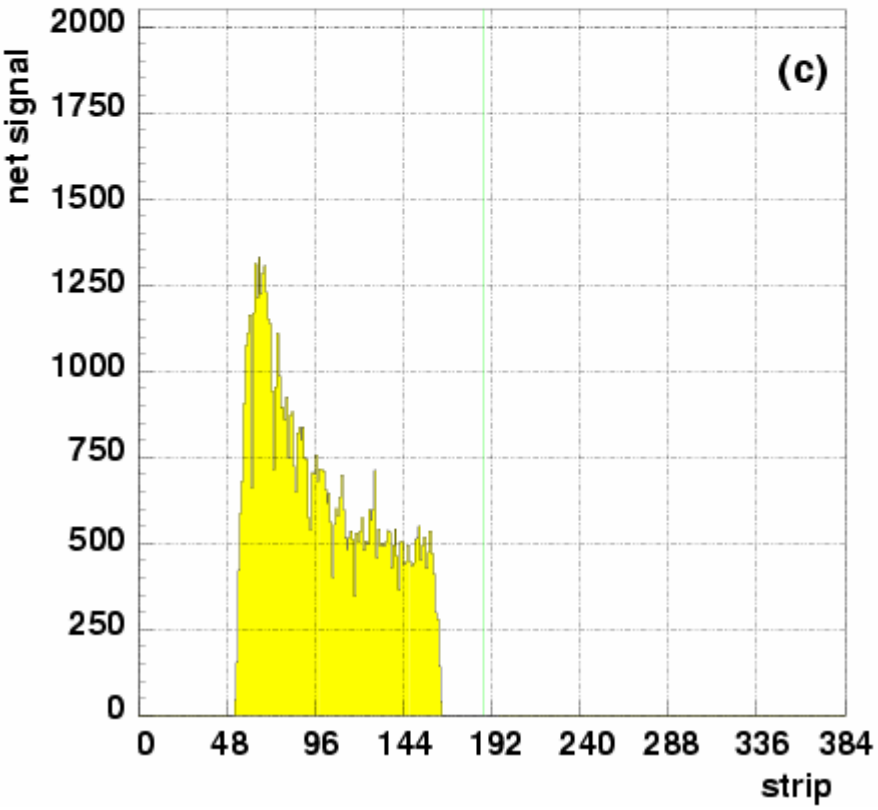
(a)

Figure 4.2. Measured typical event types; net signal (electron collected, relative scale) vs. anode strip number. Micromegas neutron detector with ^{6}Li converter foil. Courtesy Dr. Ioannis Papadopoulos, CERN [5].



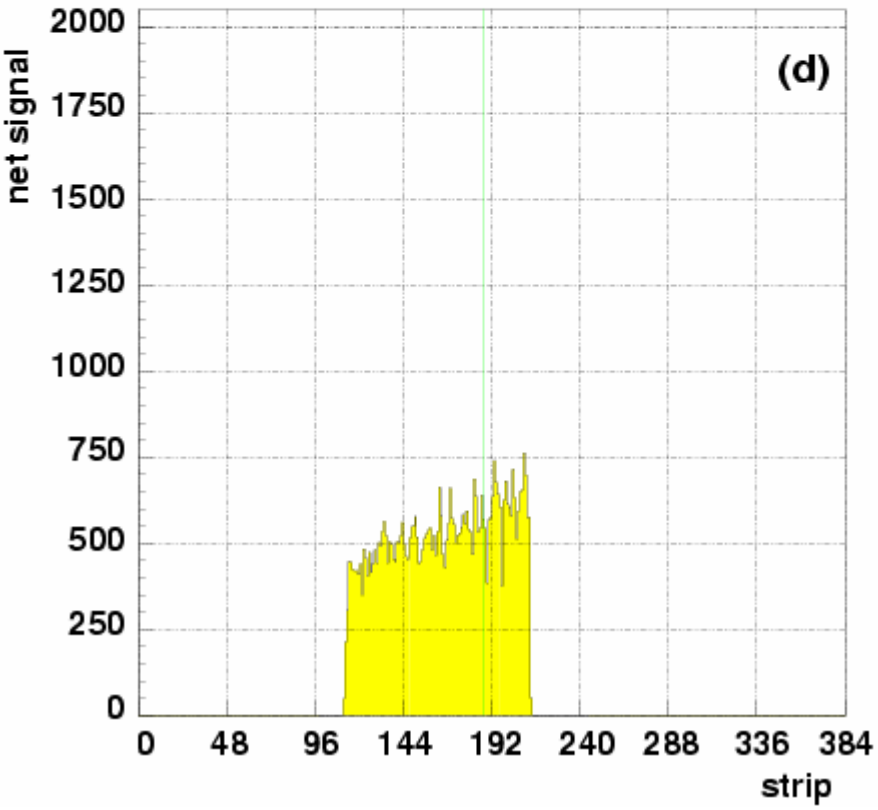
(b)

Figure 4.2 Continued.



(c)

Figure 4.2 Continued.



(d)

Figure 4.2 Continued.

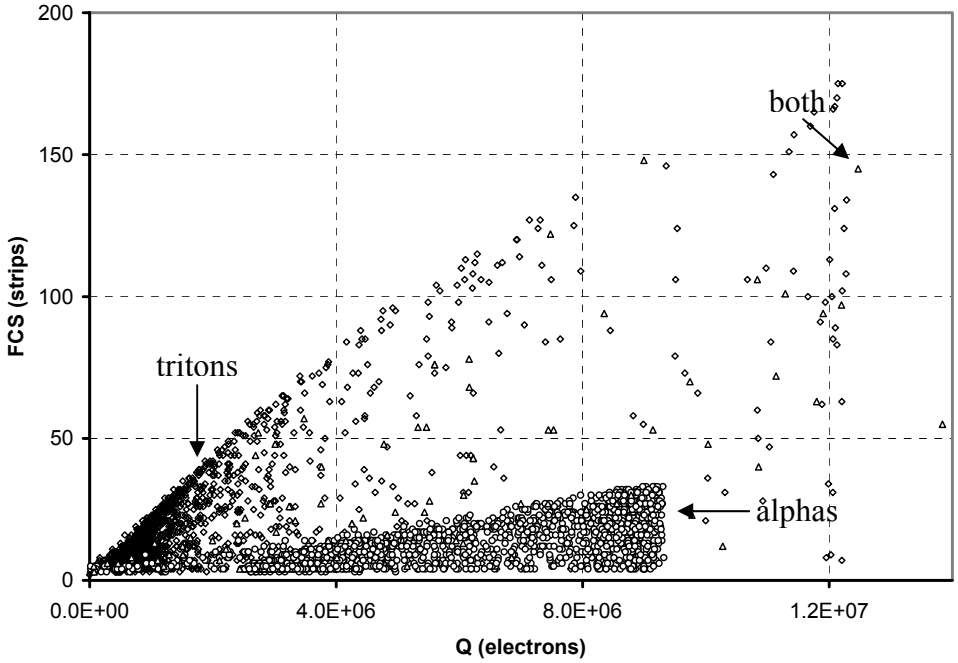


Figure 4.3. Simulated fitted cluster size (FCS) vs. cluster charge (Q) for ^{6}Li. Circles designate alpha particles, diamonds correspond to tritons, and events in which both particles entered the drift gap are marked by triangles. Compare to the experimental results in Figure 4.4.

events for thermal neutrons involve energy deposition in the drift gap by both ^{6}Li reaction products. Most of these events appear in the triton signal region and some are of greater amplitude than any of the pulses produced by a single reaction product.

Comparison of the simulated Q vs. FCS data in Figure 4.3 with the data from Andriamonje et al (reproduced in Figure 4.4) shows that the plots are quite similar [2]. One significant difference does appear in the alpha signal. For example, in the experimental data the minimum FCS increases with Q and is roughly half the maximum FCS, whereas the minimum FCS is constant in the simulation data. The minimum FCS in the data from Andriamonje et al, for an alpha particle that travels parallel to the anode strips and that deposits all of its energy in the drift gap, corresponds to an electron shower width of 4 mm perpendicular to the direction of particle travel. For a drift gap width of 3 mm and an electron diffusion coefficient in the gas of 370 microns/$\sqrt{}$cm, the minimum electron shower width should be much smaller than this, suggesting a significant overestimation of the FCS in this case. Two possible reasons for this are as follows. First, FCS, is an approximation to the cluster size that varies with the fitting method for a given anode strip signal topology; thus, the FCS can be overestimated (or underestimated) if a non-optimal fitting function is used. Second, it is known that capacitance between neighboring strips induces signals on anode strips that did not actually collect any electrons [2]. In this work a fitting function is used on the simulated anode strip topologies that corresponds to that used by Andriamonje et al, and it is found that it does not overestimate FCS, leading us to conclude that crosstalk between the anode strips is the responsible factor [15]. (If the number of anode strips affected by crosstalk is about the same for different FCS values, the overestimation in FCS is around 5-10% for large clusters.) A second small difference is that the simulated results suggest slightly longer maximum reaction product ranges in the gas since the very largest clusters span a wider number of anode strips than those of Andriamonje et al. This is easily explained by a small difference in

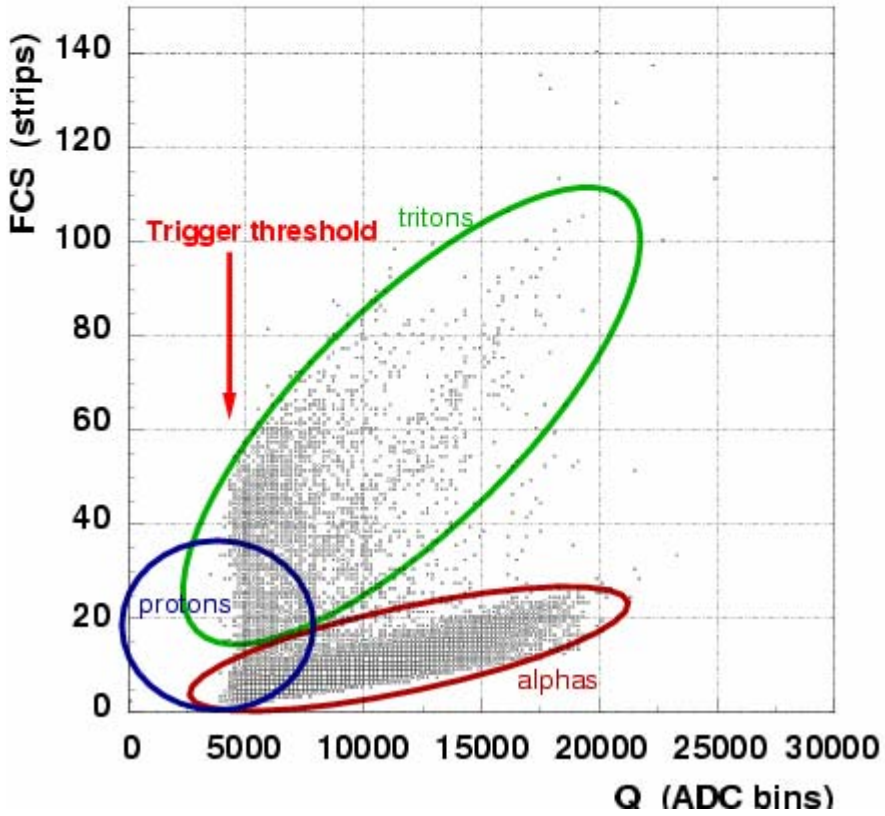


Figure 4.4. Measured fitted cluster size (FCS) vs. cluster charge (Q) for ^{6}Li. Courtesty Dr. Ioannis Papadopoulos, CERN [5].

gas pressure (e.g. altitude), error in the predicted reaction product range of a few percent, and possibly an underestimation of cluster size for large clusters by the fitting function.

Q/FCS Distribution

Average stopping power values for the reaction product motion perpendicular to the anode strips is estimated by dividing the total charge collected by the number of anode strips in the cluster, Q/FCS. (Note that this is only an approximation to the true stopping power as the reaction products have additional direction components that can be quite large.) The simulated Q/FCS distribution in Figure 4.5 is in good agreement with the experimental shape reproduced in Figure 4.6 [2]. Good separation is seen between the triton and alpha particles.

Comparison of simulation data in this work with published experimental results shows good agreement between the two. This indicates that simulation results can be useful for understanding aspects of Micromegas neutron detector behavior that are not easily deduced from experimental results and for behavior prediction to assist in Micromegas design optimization.

(c) Detection Efficiencies for ^{6}Li and ^{10}B Converters

Limitations Due to Self-Shielding

The detection efficiency of Micromegas neutron detectors is limited by the neutron absorption cross section of the neutron converter material and the range of the reaction products in the converter. Since the mean free path of neutrons in ^{10}B and ^{6}Li is much greater than the range of the reaction products from those materials for neutron energies of thermal and above, self-shielding of the reaction products by the converter foil limits the neutron detection efficiency that can be achieved. For ^{10}B and ^{6}Li, the maximum neutron detection efficiency is less than ten percent for thermal neutrons. This is suitable for neutron beam monitoring but is inadequate for many other applications.

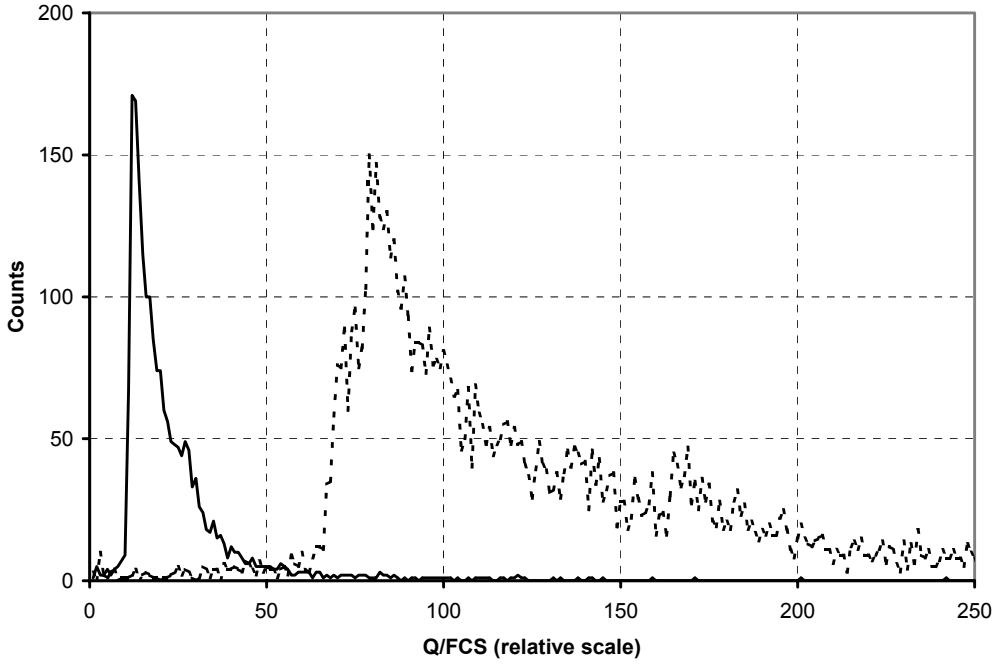


Figure 4.5. Simulated Q/FCS distribution for ^{6}Li: alphas (solid line) and tritons (dashed line) particles entering the drift gap. Compare to the experimental results in Figure 4.6. The x scale (Q/FCS or ADC bins/strip) is relative because it depends on the size of the ADC bin used.

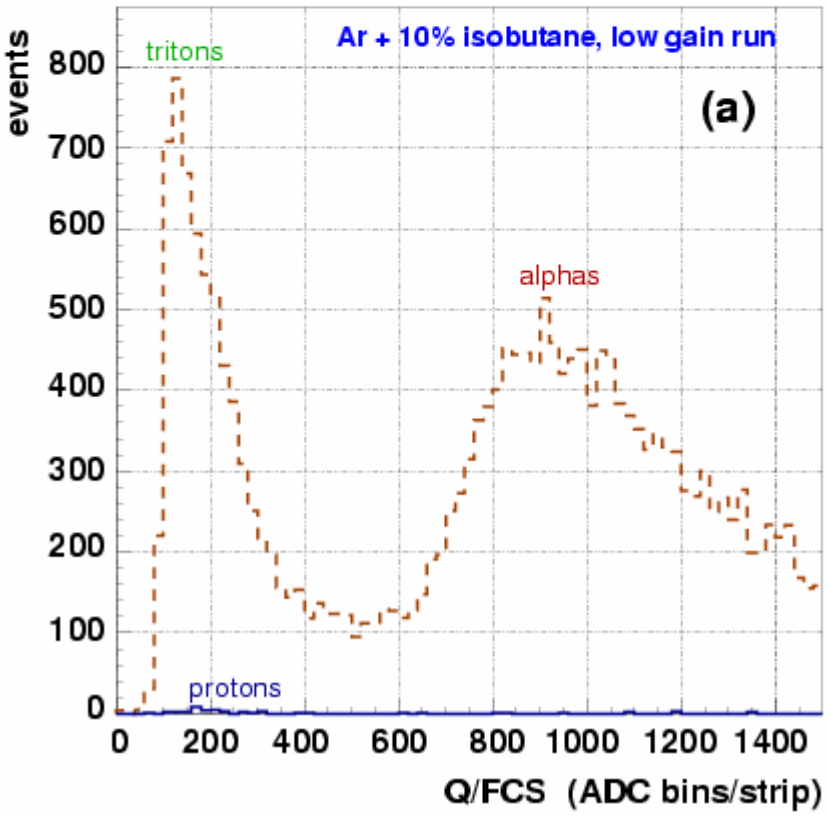


Figure 4.6. Measured Q/FCS distribution for ^{6}Li: proton recoils (solid line) and tritons and alphas (dashed line) with low gain and a 90:10 argon/isobutane gas mixture. Courtesty Dr. Ioannis Papadopoulos, CERN [5].

Dependence on Lower-Level Discriminator

Detection efficiency is also dependent on the lower level discriminator (LLD) setting used for distinguishing neutrons from gamma events since neutron events that do not deposit sufficient energy in the detector chamber to produce a pulse amplitude that clears the LLD will not be counted. Since the simulation does not include energetic (keV to MeV) electron transport, it could not be determined from simulations what the necessary LLD will be for different Micromegas neutron beam monitor designs and configurations. Experiments have shown that a minimum signal threshold equivalent to 235 keV of deposited energy will eliminate all discernable gamma contamination when a 3 mm drift gap of 90/10 Ar/isobutane is used [2]. Since the average energy to create an ion pair in gases is not greatly different for particles of widely differing LET values, neutrons and gamma rays depositing the same energy in the gas will produce nearly equal pulse amplitudes. Lower LLD thresholds may very well be possible and pulse topologic analysis for rejection of very low stopping power particles (e.g. electrons) will almost certainly allow significant reductions in threshold. (It is known that x-rays of at least 22 keV are capable of depositing all of their energy inside a Micromegas gas chamber and thus the LLD must be well above this unless pulse topologic analysis can reject these events [16].) Changes in detector design also change the LLD level necessary for elimination of the gamma signal. For example, increasing the gas chamber width or gas pressure will allow gamma-induced electrons to deposit greater energy in the gas chamber before they exit it. Similarly, decreasing the chamber width will decrease the necessary LLD. Since no data are currently available on how the minimum LLD for elimination of the gamma signal varies with Micromegas detector design, an assumption is made herein of a 235 keV LLD for all detection efficiency calculations. (The simulation is accurate down to at least 10 keV for energy deposited as a result of neutron interactions so good calculations of how LLD affects neutron detection efficiency can be made as soon as more information

on gamma pulse amplitudes becomes available.) The LLD primarily affects neutron counting when using ^{6}Li target material (as opposed to ^{10}B) due to the much lower stopping power of the triton reaction product from ^{6}Li as compared with the ^{10}B reaction products.

Calculations and Results

The probability of a neutron being absorbed in the converter material foil varies with neutron cross section and thus neutron energy and is easily calculated for various cases. For low energy neutrons (<10 keV), the probability of an absorption in the converter foil resulting in 235 keV or more of energy deposition in the drift gap, and thus exceeding the LLD and producing a count, will be quite sensitive to converter material type and to thickness but relatively insensitive to neutron energy. Simulations are therefore run for normally incident thermal neutrons impinging on ^{10}B and ^{6}Li converter layers between 1 nm and 300 microns thick at half-decade intervals. Figure 4.7 shows the fraction of neutrons absorbed that produce a useful pulse (one exceeding the LLD and thus producing a count), along with 95% confidence intervals. As can be seen in the graph, ^{10}B has a much lower useful thickness than ^{6}Li, with significant decreases in useful count probability beyond 300 nm versus almost 10 microns for ^{6}Li. (The thicknesses giving the maximum neutron detection efficiency occur beyond this due to gains from increased neutron capture initially exceeding losses from self-shielding as described in Section (g) of this chapter.) The greater useful thickness for ^{6}Li stems from the longer range of the ^{6}Li reaction products compared with those of ^{10}B due to the greater reaction Q value and the low stopping power of the ^{6}Li triton. The high fraction of ^{6}Li events not producing useful pulses at low converter layer thicknesses is due to the triton stopping power being so low that many tritons escape the drift gap before depositing 235 keV. Remedies for this include increasing the drift gap width and increasing the pressure of the gas.

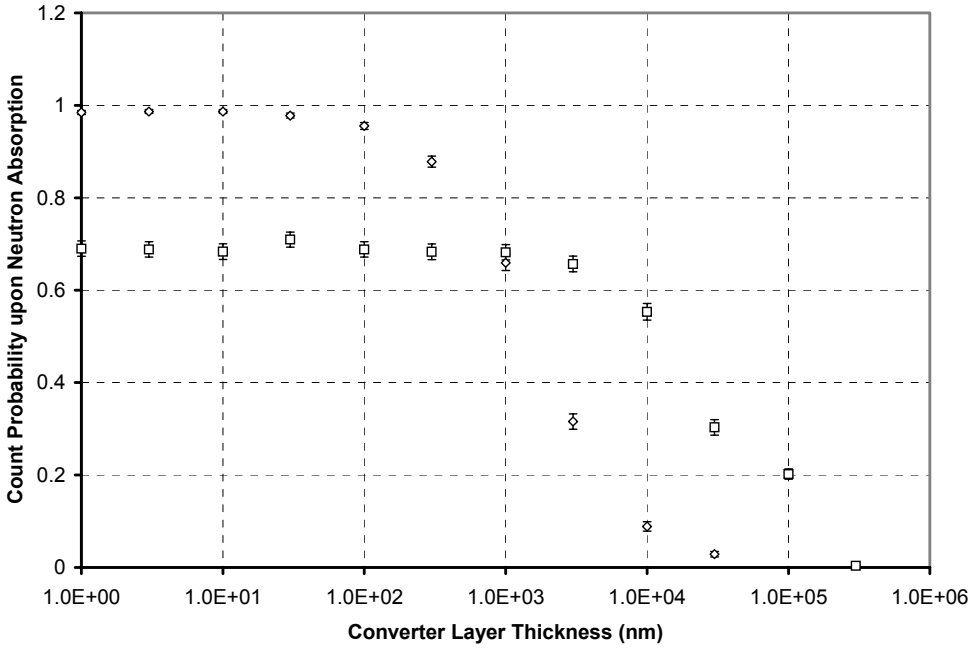


Figure 4.7. Count probability given a neutron absorption event (235 keV LLD). Values with ninety-five percent confidence intervals are shown for various converter layer thicknesses. ^{6}Li data are designated by squares and ^{10}B by diamonds.

(d) ^{10}B Converter Layer Event Characteristics

FCS Characteristics

^{10}B converter foils produce characteristic cluster size vs. charge distributions that are quite different from those from ^{6}Li converter foils. Figure 4.8 shows a plot of fitted cluster size vs. cluster charge for a 0.5 micron ^{10}B converter foil with the same design parameters as the ^{6}Li-based device discussed in Section (b) of this chapter. The two reaction products have very similar distribution patterns, unlike the ^{6}Li reaction products in Figure 4.3. Due to a lower Q, the pulses from the ^{10}B(n,alpha)^{7}Li reaction alpha particles are generally smaller than the pulses from alphas generated in the ^{6}Li(n,alpha)t reaction. In the case of the other reaction products, because of the differences in the stopping power ^{7}Li frequently produces a larger pulse than the tritons which often escape the chamber. The short range of ^{7}Li ions compared to tritons yields much better spatial resolution for ^{10}B converter foils than for ^{6}Li as evidenced by the larger clusters for ^{10}B being around 30 strips wide and by the largest ^{6}Li clusters around 180 strips wide. Finally, it is observed that ^{7}Li pulses are usually of lower amplitude than alpha pulses, but the cluster sizes do not decrease commensurately, yielding an apparently higher stopping power for alphas than ^{7}Li when using the Q/FCS stopping power approximation method, in contrast to the actual values. The reason for this disparity is that the horizontal diffusion of electrons while traversing the drift gap is independent of the incident particle type and that the measured FCS tends to increase by a fixed amount. Thus, the fractional increase in measured FCS (and thus the decrease in approximated stopping power) is largest for small cluster sizes since they are produced by particles with very short ranges, such as ^{7}Li. If no horizontal diffusion of electrons is considered, the stopping power for ^{7}Li ions, as approximated by the Q/FCS method, is more in keeping with what one would expect.

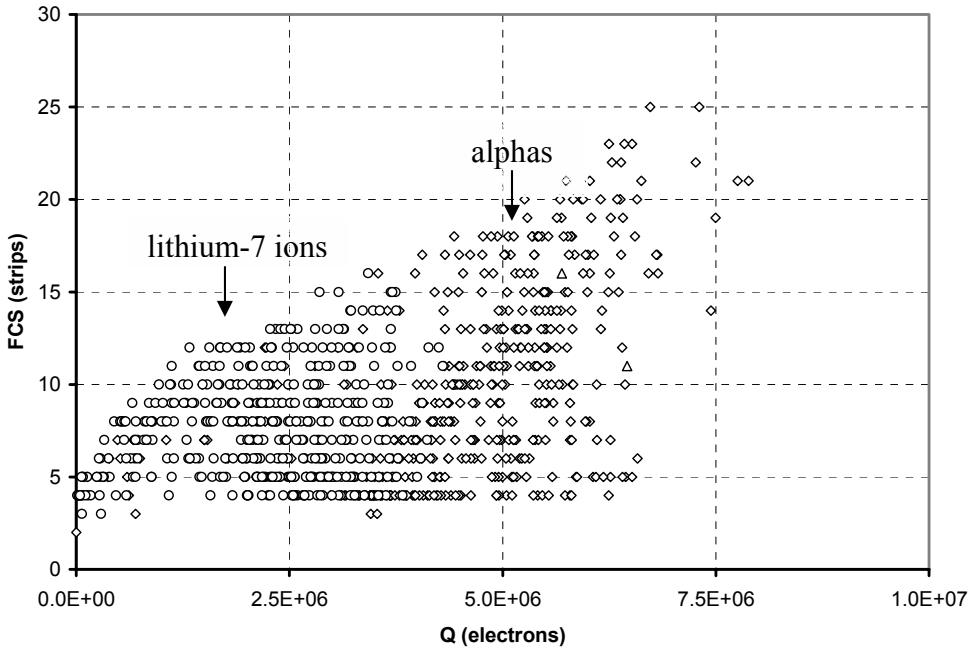
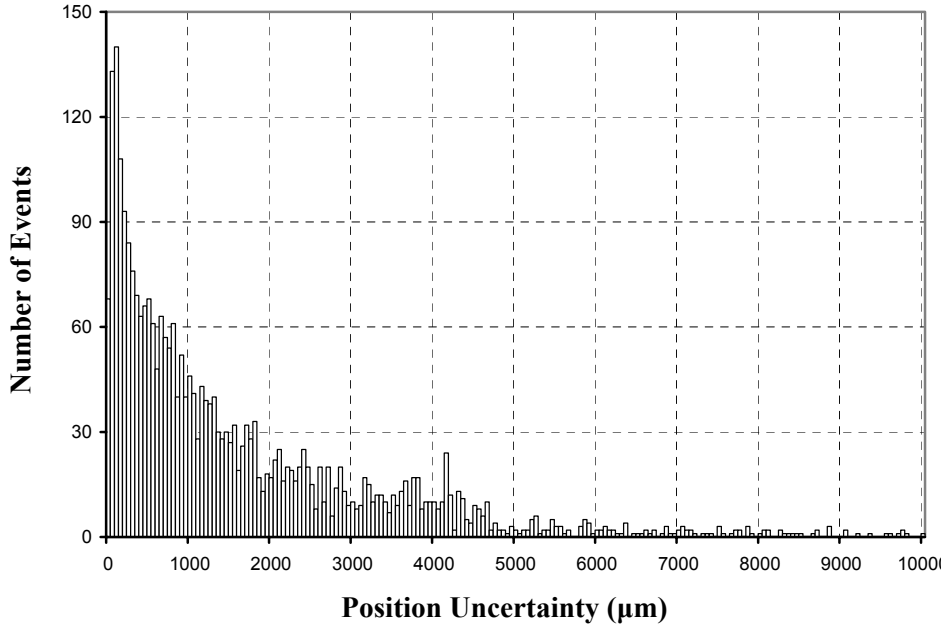


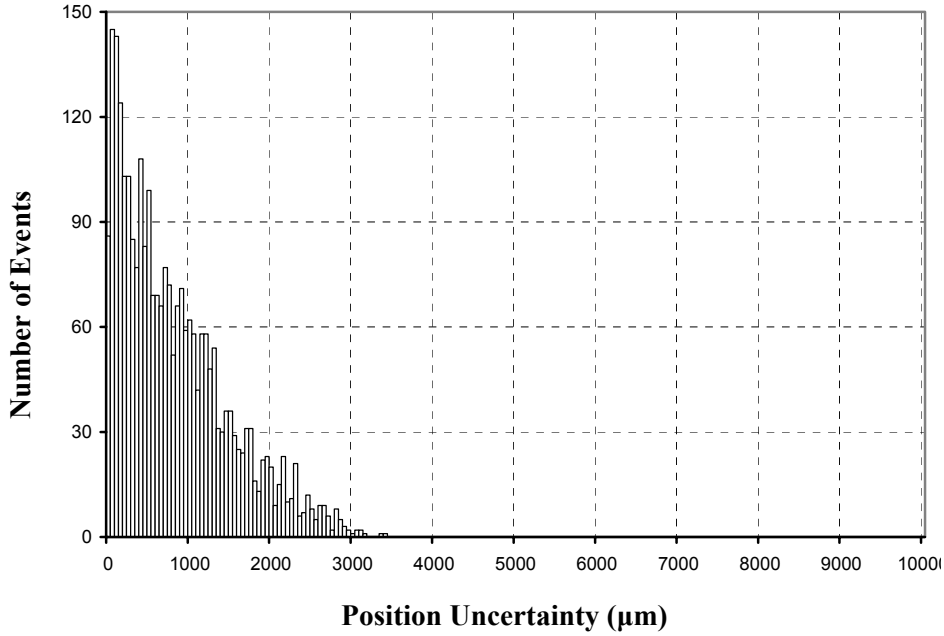
Figure 4.8. Simulated fitted cluster size (FCS) versus cluster charge for ^{10}B; 0.5 micron converter foil. Circles designate ^{7}Li ion events, diamonds correspond to alphas, and triangles events produced by both reaction products.

Position Resolution

Simulation of a Micromegas neutron detector is of considerable value in testing different methods for determining where a neutron absorption event took place. In a real detector the location of a specific incident neutron is not precisely known; instead, it is estimated based on the signal produced in the detector. Since position uncertainty (resolution) is the difference between the actual location and the estimated location, one is trying to estimate a value (position resolution) based on something that one inherently does not know with certainty (the actual location of the neutron interaction). By contrast, in a simulated detector both the precise event location and the detector response are known. If the simulation is accurate, one may compare different converter foils to see which ones yield the best position resolution. One may also test different methods for estimating the neutron interaction locations based on anode strip topology analysis and compare them to determine which methods work best. (The best method is that which minimizes the differences between estimated event locations and actual event locations.) To determine the difference in spatial resolution from the substitution of ^{10}B for ^{6}Li as the neutron converter material, simulations are run for both materials to generate sets of actual neutron absorption event locations and estimated neutron absorption event locations, the estimated location being at the midpoint of the anode strip cluster. Figure 4.9 shows the distributions of uncertainty of neutron interaction location (the difference between the true location and the estimated location) for 0.5 micron ^{6}Li and ^{10}B converter foils. While the distributions initially look very similar, the ^{10}B events rapidly disappear for uncertainties beyond 2000 microns, whereas the ^{6}Li events only slowly drop off. (An equal number of neutron capture events are used in both cases, but the number of events producing a signal is greater for ^{6}Li than for ^{10}B due to self-absorption of ^{10}B reaction products by the ^{10}B converter foil owing to their shorter range.) The long tail seen in the ^{6}Li converter foil case is produced by triton reaction products traveling at approximately right angles to the anode



(a)



(b)

Figure 4.9. Simulated neutron position uncertainty distributions. Distributions of neutron absorption event location uncertainty for (a) ^{6}Li and (b) ^{10}B 0.5 micron converter foils using the anode strip cluster center method.

strips and at shallow angles relative to the converter foil. (The tritons have a much longer range than alphas from the ^{6}Li reaction or either of the ^{10}B reaction products.) Variations in average position uncertainty reflect this difference in reaction path lengths, with a mean uncertainty for ^{10}B of 815 microns and with a mean uncertainty for ^{6}Li of 2034 microns. By rejecting a small fraction of events with the largest FCS (and thus the largest position uncertainty associated with them), average position resolution for ^{6}Li converter foils can be greatly reduced. For example, by rejecting the 5% of the ^{6}Li counts with the largest FCS values, mean position uncertainty can be reduced to 1445 microns.

(e) Effect of Drift Gap Width on Detector Performance

Another major determinant of Micromegas performance is the drift gap width. There are several reasons for this. When the drift gap width is less than the maximum reaction product range in the gas, narrower gaps lead to smaller pulses, but they also reduce the neutron capture event location uncertainty since the reaction product track length is constrained to a shorter distance around the event. Larger drift gaps lead to larger neutron pulses, but they also increase position uncertainty and contribute to larger gamma pulses that must be discriminated against. When the drift gap is larger than the range of the reaction products produced by the neutron absorption event, the primary effects of increasing the gap width are to increase the size of the gamma pulses and to slowly spread out the neutron signal over a larger number of anode strips due to electron diffusion. No improvement in signal amplitude occurs. In order to determine how the detector response varies with drift gap width, the effects on neutron counting efficiency and position uncertainty are examined. As no data are published on how the LLD required for gamma rejection varies with Micromegas drift gap width, a value of 235 keV is assumed for all drift gap widths as explained in Section (c) of this chapter. (In practice the

appropriate LLD will vary significantly with Micromegas design, especially drift gap width and gas pressure.)

Table 4.1 lists the average position resolution and probability of a neutron capture event that produces a pulse exceeding the LLD. When using a ^{10}B converter foil, shrinking the drift gap width from 3 mm to 1 mm almost doubles the neutron position resolution without any loss of counts. When using an LLD equivalent to 235 keV of energy deposited in the drift gap and when using Ar/isobutane at one atmosphere pressure, the only reasons for using a drift gap greater than 1 mm are construction, and possibly signal-to-noise issues. When ^{6}Li is used as the converter material however, raising the drift gap width to 5 mm increases the count probability (after a neutron capture event) from 0.69 to 0.85, but it also increases the average position uncertainty by 30%. The large increase in count probability results entirely from those triton particles whose increased path length in the drift gap prior to escape causes their energy deposited in the drift gap to cross the 235 keV LLD threshold. With the ^{6}Li converter, a tradeoff can be made between position resolution and counting efficiency by adjusting the width of the drift gap, or by the rejection of large cluster events as described in Section (c) of this chapter.

(f) Pressurization Effects

Pressurization of a Micromegas neutron detector is appealing because it increases the detector's spatial resolution by decreasing reaction product ranges in the gas, and it increases pulse sizes in those cases where reaction products escape the drift gap. The primary limitation when pressurizing a Micromegas is the need to keep the neutron scattering cross section as low as possible, which requires thin detector walls. To the extent that a Micromegas can be pressurized, it is helpful to do so. Table 4.2 lists the average uncertainty in neutron position and the count probability per neutron absorption for various gas pressures using the standard detector design parameters listed earlier in Section (b) of this chapter. Reaction product ranges

Table 4.1. Effects of drift gap width. Variation in average neutron position error and count probability given a neutron absorption event for various drift gap widths at one atmosphere of gas pressure (0.5 micron converter foil is assumed).

Drift gap (microns)	Average resolution (microns)		Count probability	
	^{6}Li	^{10}B	^{6}Li	^{10}B
5000	2689	909	0.848±0.022	0.828±0.024
4000	2520	894	0.784±0.026	0.825±0.024
3500	2344	878	0.726±0.028	0.824±0.024
3000	2034	815	0.691±0.029	0.806±0.025
2500	1731	745	0.667±0.029	0.817±0.024
2000	1685	647	0.644±0.030	0.802±0.025
1500	1287	598	0.611±0.030	0.800±0.025
1000	968	446	0.477±0.031	0.806±0.025
750	769	375	0.359±0.030	0.621±0.030
500	531	276	0.234±0.026	0.356±0.030

Table 4.2. Effects of gas pressure. Variation in average neutron position error and count probability given a neutron absorption event for various gas pressures (a 3 mm drift gap and 0.5 micron converter foil are assumed).

Pressure (atm)	Average resolution (microns)		Count probability	
	^{6}Li	^{10}B	^{6}Li	^{10}B
1	2034	815	0.690±0.029	0.821±0.024
1.5	1816	593	0.809±0.025	0.830±0.023
2	1624	465	0.896±0.019	0.835±0.023
2.5	1548	360	0.977±0.009	0.837±0.023
3	1337	320	0.992±0.006	0.833±0.023
4	1159	245	0.997±0.003	0.806±0.025

in a particular gas are inversely proportional to the gas pressure. While the simulation with ^{10}B indicates a position resolution that is almost proportional to the inverse of the pressure, a more complex behavior is predicted for ^{6}Li due to the long range of the tritons. Some additional small deviations from inverse proportionality occur and are related to lateral diffusion of electrons in the drift gap. The count probability for ^{10}B does not increase with pressurization, but the count probability for ^{6}Li increases rapidly, approaching one at a pressure of 2.5 atmospheres.

(g) Optimization of Detector Characteristics

Converter foil self-shielding restricts the Micromegas neutron converter foil design to applications in which the required neutron detection efficiency is quite low (e.g. a few percent at thermal). As previous work has shown, the design is well-suited to neutron beam monitoring for time-of-flight experiments, and future work can be anticipated in this area [2]. For a neutron beam monitor that must provide spatial information, a ^{10}B converter foil and a 1 mm drift gap width provide the best combination of resolution and counting probability per neutron absorption available without resorting to pressurization. If pressurization is possible without excessive detector wall thicknesses, the drift gap should be decreased in an inverse proportion to the increase in pressure. This will keep the drift gap thickness as low as possible, thus minimizing gamma pulse amplitudes and neutron position uncertainty, without decreasing the counting efficiency.

The optimal converter foil thickness (and thus detection efficiency) varies widely with application. For example, the desired detection efficiency for a neutron beam monitor on a high flux thermal neutron beam may only be 10^{-5} since neutron beam monitors should only capture enough neutrons as are needed to get good statistics on the beam characteristics, and beyond that they should have minimal impact on the neutron beam. In other applications the highest possible neutron detection efficiency may be desired. To determine the

converter foil thickness giving the highest neutron detection efficiency, the neutron capture probabilities for various thicknesses of ^{6}Li and ^{10}B are multiplied by the probability of a neutron capture event producing a useful count. Figure 4.7 shows the neutron detection efficiency as a function of converter foil thickness at half-decade intervals. It is found that a 3 micron ^{10}B layer results in a detection efficiency of 4.4% at thermal (0.0254 eV) and that a 100 micron ^{6}Li layer results in an efficiency of 7.1%, which are the highest efficiencies found among the converter foil thicknesses tested. It is possible to increase the neutron detection efficiency fractionally by having two layers of material in the converter foil; one of ^{10}B and one of ^{6}Li, the ^{10}B layer facing the drift gap.

(h) Recommendations for Future Work

Within the framework of Micromegas neutron detectors, several areas immediately lend themselves to further investigation. The gamma sensitivity of Micromegas neutron detectors is largely unknown and characterization of their gamma response will enable optimized designs for neutron detection without false counts from gammas. It is especially important to know how the LLD required for gamma rejection varies with drift gap width, the type of gas used, and the gas pressure. Gamma response characterization is probably the most important area for follow-up investigation. Current methods for neutron event location use the center of the anode strip signal cluster [2]. Many other methods are possible such as taking the anode strip with the largest signal or by taking one of the ends of the track as the event location. The second method will yield the best position resolution if reasonable criteria are used to determine which track end is most likely to be the one adjacent to the neutron capture event location. Simulation is a very useful tool for testing different methods of estimating the neutron absorption location to see which ones are likely to produce the best results when incorporated into an actual readout scheme. Anode strip topology analysis will also improve neutron-gamma

discrimination. Another worthy avenue of investigation is how best to pressurize a Micromegas detector without creating an excessive detector thickness in which to scatter neutrons. As demonstrated by this body of work, pressurization will increase the Micromegas position resolution and in some cases the counting efficiency given a neutron capture event and thus should be very useful for many applications.

CHAPTER 5
MICROMEGAS NEUTRON BEAM MONITOR NEUTRONICS

(a) Introduction

Neutron beam monitors are used for measuring the intensity and spatial and time distributions of neutron beams used in experiments such as neutron scattering research and neutron cross section measurements. Accurate measurement of these parameters is essential as they are used to normalize data from instruments (e.g. two dimensional position-sensitive neutron detectors) located downstream on the neutron beam. Other than capturing a small fraction of the neutrons for beam characterization purposes, a neutron beam monitor should interfere with a neutron beam as little as possible. Its purpose is to determine the characteristics of the beam, not to alter those characteristics. Scattering of neutrons (as opposed to unintended neutron capture in structural materials) is especially onerous as changes induced in the beam are difficult to predict and thus the effects on experiments downstream on the beamline are especially deleterious. It is preferred that the Micromegas neutron beam monitors to be used at the SNS have a maximum neutron scattering probability of 10^{-3}. [12]

Like the previous two chapters, this chapter focuses on the simulation of Micromegas neutron beam monitors as a tool for achieving an optimized design. However, unlike the previous two chapters, this chapter focuses on the relationship between Micromegas design and neutron scattering within the device. The radiation transport code MCNP is used for calculations of neutron interactions, as opposed to the custom simulation code used in the work of the last two chapters. In this chapter the contribution of different Micromegas structural elements to the total neutron scattering probability in the device are

examined and methods recommended for minimizing scattering, including an approximate best-case design.

(b) Monte Carlo Simulation Methods

Simulation Methods

The intent of this work is to examine the neutronics characteristics of different materials that may be used in Micromegas neutron beam monitors and provide suggestions as to how to minimize undesirable neutronics effects by Micromegas detector materials. To this end, calculations of neutron scattering and absorption probabilities are made for a number of possible detector materials over the range 10^{-4}–10^{6} eV at half-decade intervals using the computer code MCNP [3]. To give the greatest possible flexibility to the Micromegas neutron beam monitor designer, calculations are made for each type of possible design material individually, allowing a designer to easily determine the effect of different combinations of detector materials. Interaction probabilities are calculated on a per mm basis for gases as the gas chambers in Micromegas devices are usually several mm wide, and on a per 100 microns basis for solid materials and Micromegas detector walls will likely be on that order of thickness.

A custom code is written to automate en masse MCNP runs and to process their output into appropriate information. Each MCNP run uses either 200,000 or 500,000 monoenergetic neutrons in order to obtain good statistics. The PTRAC option in MCNP is used to generate data on neutron interactions inside the Micromegas and exit points from it. The custom code determines the number of neutrons achieving the following outcomes:

(1) unscattered transmission through the monitor,
(2) scattered transmission through the monitor,
(3) backscatter,
(4) and absorption in the monitor (rare).

The code also calculates the probability and statistical uncertainty associated with each event. Statistical uncertainties are minimized through the judicious selection of material thicknesses when calculating probabilities for unscattered neutron transmission. Unscattered probabilities in the appropriate units (per mm, per 100 microns) are then calculated from these and scattering probabilities found. (In the cases examined, neutron absorption was usually absent.)

Materials Examined

Neutron scattering and absorption probabilities are calculated for a variety of gases and solids. The non-hydrogenous gases Ar, He, Ne, and CF_4 and the hydrogenous gases methane, isobutane, and dimethylether (DME) are all considered as Micromegas devices using these gases are reported in the literature [1-2, 5, 17-21]. Neutron scattering and absorption probabilities for gas mixtures (e.g. P-10) may be determined by adding the contributions from the constituent gases. The hydrogenous detector wall materials epoxy and mylar and the non-hydrogenous materials Al, quartz (SiO_2), Cu, Ni, Au, Cr, W, Co, Zn, Sn, Ti, and Si are all considered, some of them are used in previous work and others not [1-2, 21]. It is found in all cases that neutron absorption in the materials investigated is negligible ($<10^{-6}$). Thus results are presented and calculations are made only for neutron scattering. Neutronics calculations are obtained using MCNP version 4c2 using the ENDF60 cross sections [3]. The densities of materials used in the calculations are given in Table 5.1 (gases) and Table 5.2 (solids).

Scattering Probabilities

To facilitate comparison of the neutronics properties of different materials, results are presented in the form of scattering probabilities for fixed material thicknesses (per mm for gases and per 100 microns for solids). The equations that describe the probability of a neutron scattering in the Micromegas detector are exponential in nature. However, if the sum of the scattering probabilities per unit thickness times the thicknesses of the materials

Table 5.1. Densities used for gases in MCNP simulations.

Gas	Density (g/cc)
Ar	1.67×10^{-3}
He	1.69×10^{-3}
Ne	8.53×10^{-4}
CF_4	1.25×10^{-3}
methane	6.80×10^{-4}
isobutane	2.51×10^{-3}
dimethylether	1.91×10^{-3}

Table 5.2. Densities used for solids in MCNP simulations.

Material	Density (g/cc)
cast epoxy	1.18
mylar	1.397
Al	2.702
quartz (SiO_2)	2.32
Cu	8.92
Ni	8.896
Au	19.311
Cr	7.20
W	19.35
Co	8.90
Zn	7.14
Sn	7.28
Ti	4.52
Si	2.32

in the Micromegas are small compared to one (e.g. <0.10), the scattering probability *p* may be very accurately approximated as

$$p = \sum_{i=1}^{n} s_i t_i \,, \qquad (5.1)$$

where n is the number of material layers in the detector, s_i is the scattering probability per unit path length for a particular layer, and t_i is the thickness of that layer (in the same units as those used in s_i).

(c) Results and Discussion

General Considerations

The Micromegas concept is essentially a parallel plate proportional counter with a Frisch grid (a center electrode, usually a nickel mesh) added part way between the two parallel plate anodes. (See Figure 3.1.) This divides the gas chamber into a drift gap and an amplification gap, and it allows the electric fields in the two zones to be set indepenendently. From the standpoint of neutron interactions in the Micromegas, this means that unwanted neutron interactions may occur in both detector walls, the gas chamber, the nickel mesh, and the anode readout strips. (The converter foil, which absorbs neutrons as part of the detection process, can also scatter neutrons, but this is not considered as its scattering probability will usually be inconsequential. Scattering probabilities for ^{10}B and ^{6}Li are three orders of magnitude less than absorption probabilities at thermal. Thus if the converter foil in a neutron beam monitor has a neutron detection efficiency of 10^{-3}, it will have a neutron scattering probability of 10^{-6} or less.) It is found that in all cases considered, neutron absorption probabilities are inconsequential in comparison to neutron scattering probabilities. (Neutron absorption probabilities in detector materials other than the converter foil were always less than 10^{-6}, whereas scattering probabilities were always several orders of magnitude or more higher.) Thus, only neutron scattering is considered from this point forward. The total

neutron scattering probability in the detector is a sum of the scattering probabilities in all of these layers. We will consider each of these in turn.

Detector Walls

Scattering probabilities per 100 microns of material thickness are given for the non-hydrogenous solids Al, quartz, Cr, Au, Cu, and Ni in Figure 5.1 and Si, Ti, Sn, Zn, Co, and W in Figure 5.2. Scattering probabilities for the hydrogenous solids mylar and cast epoxy are illustrated in Figure 5.3. Of all the materials tested, the lowest scattering probabilities are given by Al, Si, Sn, and quartz, respectively. At thermal energy (0.0254 eV), the scattering probability for Al is 1.0×10^{-3}/100 microns thickness. Si and quartz are of particular interest since both are etchable, and this makes them eminently suitable for use in the back detector wall where regular spaced posts are needed on the anode strip side of the detector wall to keep the mesh at a proper separation distance from the detector wall. The scattering probabilities for Si and quartz at thermal are 1.2×10^{-3}/100 microns and 2.4×10^{-3}/100 microns, respectively. Scattering probabilities for hydrogenous solids are an order of magnitude higher than those for Al. The highest scattering probability, however, is for Ni which has a probability about twice that of mylar and cast epoxy.

Prior experimental investigations have used a 1.6 mm epoxy and 1 mm quartz detector walls [2, 21]. The data in Figures 5.1 and 5.3 indicate that these will produce neutron scattering probabilities in excess of 10^{-1} and 10^{-2}, respectively, even without including the contributions from any other detector structural elements. Clearly, assessment of the neutronics characteristics of Micromegas neutron beam monitors for the purpose of minimizing neutron scattering is an important aspect of the design process.

In terms of minimizing neutron scattering probability, it is clearly best to use as a detector wall material one of the non-hydrogenous materials with a

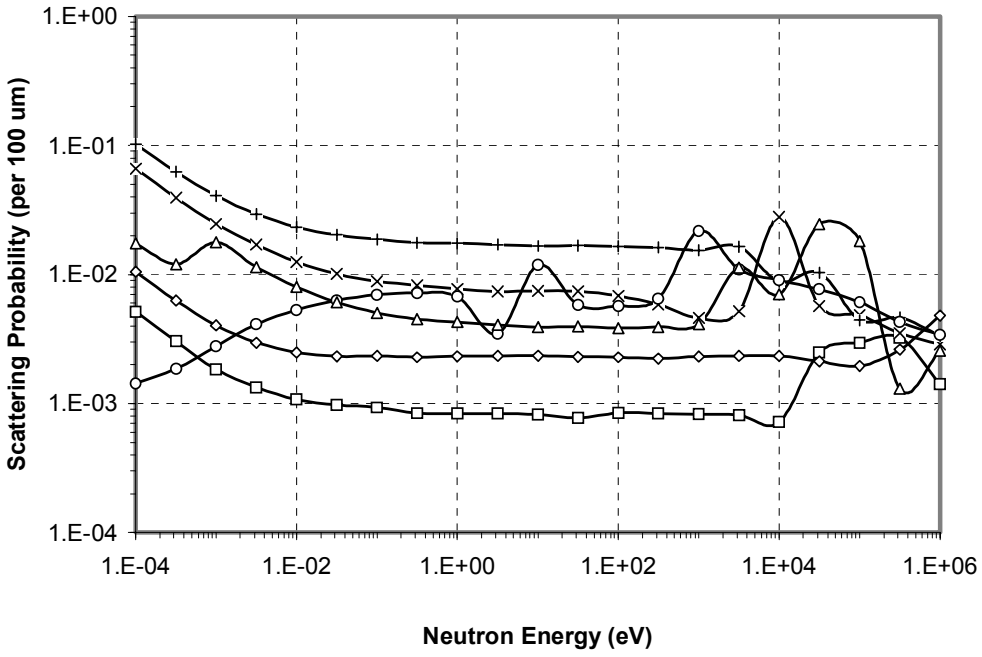


Figure 5.1. Neutron scattering in non-hydrogenous solids (I). Probabilities per 100 microns of travel: Al (squares), quartz (diamonds), Cr (triangles), Au (circles), Cu (x), and Ni (+).

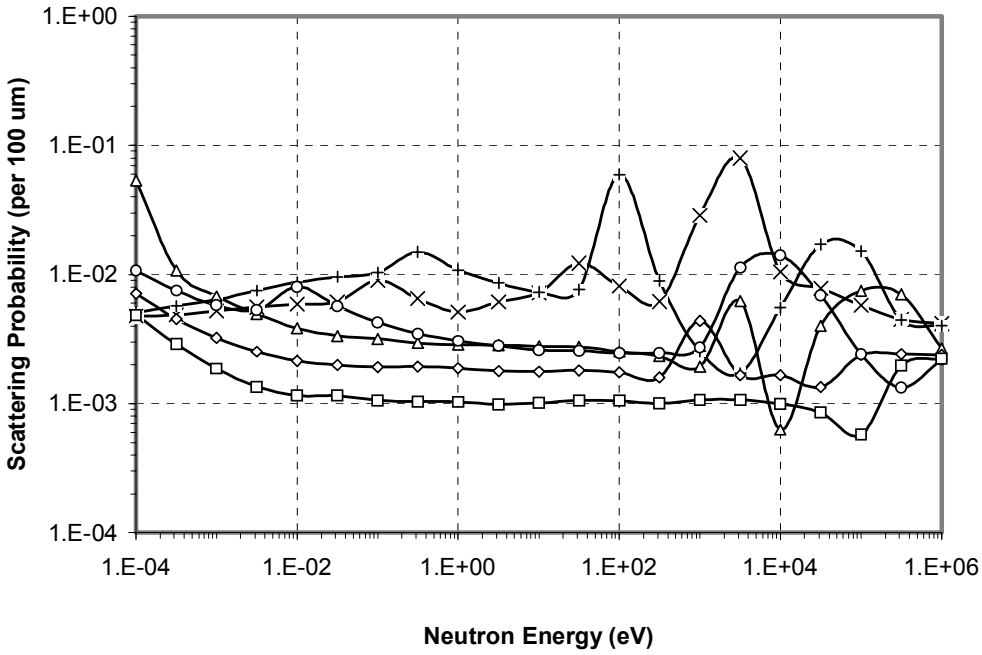


Figure 5.2. Neutron scattering in non-hydrogenous solids (II). Probabilities per 100 microns of travel: Si (squares), Ti (diamonds), Sn (triangles), Zn (circles), Co (x), and W (+).

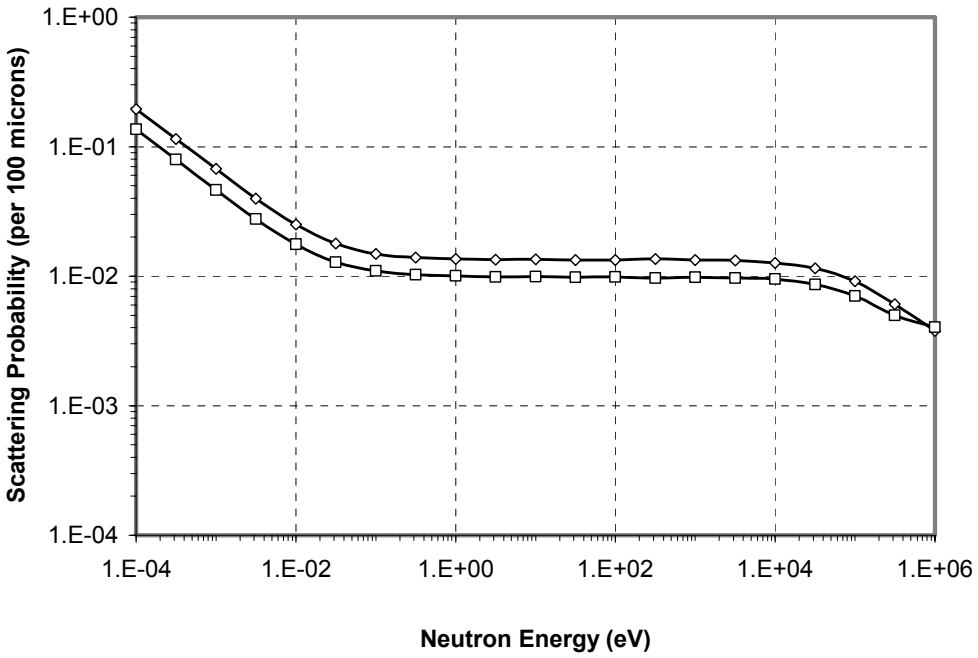


Figure 5.3. Neutron scattering in hydrogenous solids. Probabilities per 100 microns of travel: mylar (squares) and cast epoxy (diamonds).

very low neutron scattering probability. (Although there are other potential hydrogenous detector wall materials besides mylar and cast epoxy, such materials will necessarily have similar scattering probabilities as mylar and expoxy if they have significant hydrogen content.) One factor to consider is that since many non-hydrogenous materials such as Al are significantly stronger than hydrogenated materials per unit of thickness, hydrogenous materials will require greater wall thicknesses. Thus a straight comparison of scattering probability per unit thickness underestimates the extent to which hydrogenous materials will scatter neutrons as compared to the best non-hydrogenous materials. Al is well suited for use as the first detector wall where it can also double as the cathode. However, a non-conducting material should be used for the second detector wall on which the anode readout strips are placed. Although the oxide layer on the Al will resist the flow of current, the flow of charge through the anode readout strips will induce current flow in the Al.

Frisch Grid

Published research into Micromegas radiation detectors has, to this point, normally involved the use of a nickel mesh as the material for the Frisch grid that divides the gas chamber into the drift and amplification gaps [1-2, 6, 11, 16-21]. Of all the solids tested, nickel has the highest scattering probability for neutrons with energies in the vicinity of thermal. Nickel meshes with average areal thickness of 1.4 microns and 3.5 microns have been used in Micromegas neutron detectors [2, 21]. These thicknesses correspond to neutron scattering probabilities of 3.0×10^{-4} and 7.4×10^{-4}, respectively. While the scattering probability in the nickel mesh might not be as high as that of some other detector components, it is still quite significant. Replacing the nickel mesh with a mesh of more weakly scattering conducting material would yield a material improvement in detector performance. For example, the use of Au or Cr instead of Ni would approximately halve the average neutron scattering probability (given an equal mesh thickness) over the energy range

from cold to 10 eV. (See Figure 5.1.) Thus reducing scattering probability to around $1.5x10^{-4}$ at thermal seems realistic. A reduction in the thickness of the mesh will also be very helpful, with the caveat that the mesh must be sufficiently thick to retain its rigidity under the stresses of the electric fields to which it is exposed.

Anode Readout Strips

Micromegas devices also include anode strips for the readout of electron showers induced in the gas chamber by radiation interaction events. Prior work uses anode readout strips equivalent to 7 microns average areal coverage of Cu and 25 nm of Cr [2, 21]. The corresponding neutron scattering probabilities at thermal are $7.6x10^{-4}$ and $1.7x10^{-6}$. Among the materials assessed, Cu is a strong neutron scatterer and Cr a moderate scatterer (see Figure 5.1); although the thinness of the Cr anode strips prevents them from being a significant scatterer of neutrons. Using Cu or other strong neutron scatterers for anode readout strips is obviously problematic if this thicknesses is on the order of a micron but is acceptable is thicknesses for a few tenths of a micron or less.

Gas Chamber

Figure 5.4 shows the scattering probabilities per mm for the non-hydrogenous gases He, Ar, Ne, and CF_4 on a per mm basis. Figure 5.5 shows the scattering probabilities for the hydrogenous gases methane, isobutane, and dimethylether (DME). Unsurprisingly, the gases show lower scattering probabilities than the solids previously discussed due to their much lower densities despite the thicknesses used being ten times greater. Interestingly, the scattering probability of the hydrogenous gas methane is not very much higher than those of the non-hydrogenous gases He and Ar. (Methane has a much lower density at atmospheric pressure than that of the other hydrogenous gases considered and thus a relatively low hydrogen concentration per unit volume.) Thus, the use of P-10 gas (90/10 Ar/methane), a very common gas in gaseous radiation detectors that exhibits excellent properties for such

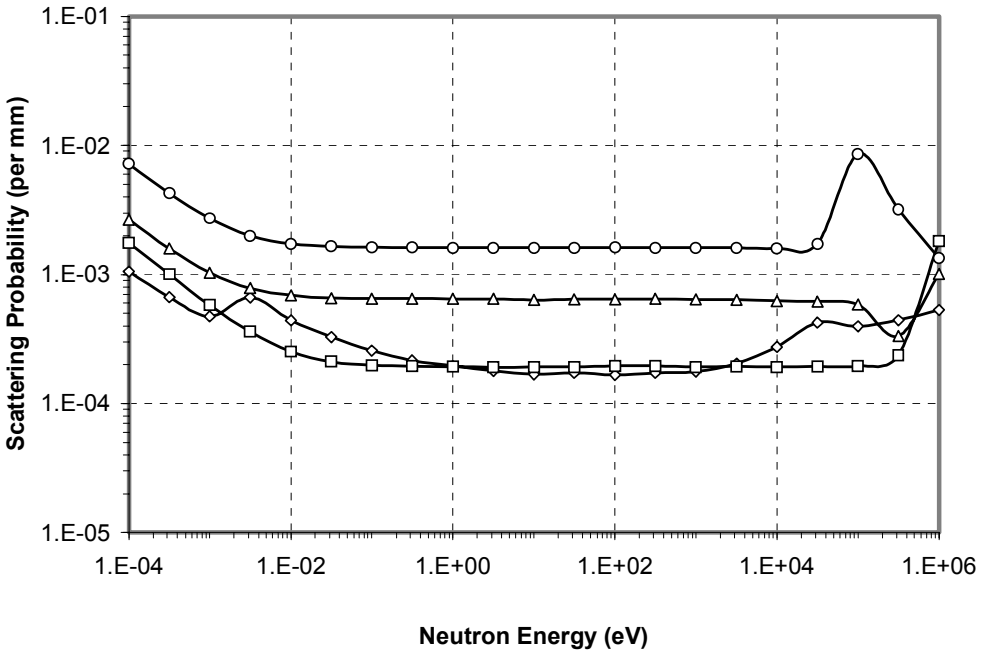


Figure 5.4. Neutron scattering in non-hydrogenous gases. Probabilities per mm of travel: He (squares), Ar (diamonds), Ne (triangles), and CF_4 (circles).

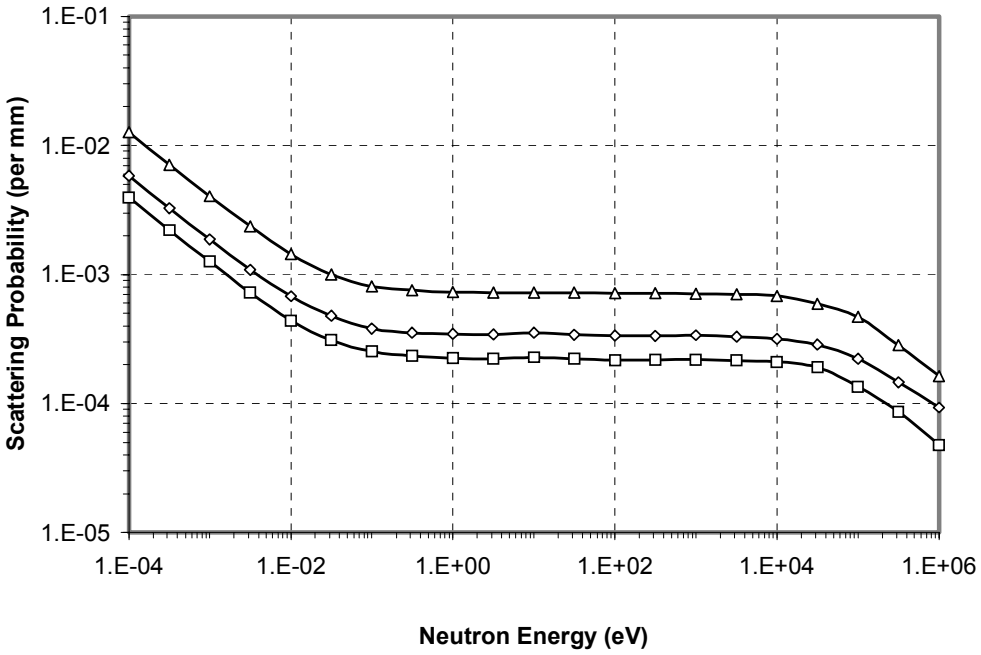


Figure 5.5. Neutron scattering in hydrogenous gases. Probabilities per mm of travel: methane (squares), dimethylether (diamonds), and isobutane (triangles).

applications, is consistent with minimizing neutron scattering. For neutrons at thermal energy (0.0254 eV), the scattering probability in P-10 gas at atmospheric pressure will be around 3.5×10^{-4}/mm. Previous experimental investigations of Micromegas neutron beam monitors have used 3 mm drift gap widths and 73-100 micron amplification gaps, giving a total scattering probability at thermal in the gas chamber of 1.4×10^{-3} for 90/10 Ar/isobutane and 1.1×10^{-3} for P-10 [2, 21].

Best-Case Scenario

The important parameters for determining the total neutron scattering probability are, as noted in Equation (5.1), the thickness of each layer of material in the detector and the neutron scattering cross section of each layer. The detector structural elements contributing the most to scattering probability are the two detector walls and the gas chamber. Minimizing the thickness of these two structural elements is therefore highly desirable. A realistic best case thickness (at atmospheric pressure) for detector walls is 25 microns, which gives a neutron scattering probability of 2.5×10^{-4} for Al, 3.0×10^{-4} for Si, and 6.0×10^{-4} for quartz. Assuming 75 micron high spacing posts for holding the mesh in place (that cover four percent of the area of the second detector wall), this brings the total scattering probabilities (walls and posts) to 3.4×10^{-4} for Si and 6.7×10^{-4} for quartz. The calculations described in Chapters 3 and 4 show that if ^{10}B is used as the neutron target material, a drift gap thickness of 1 mm may be used without appreciable loss of counting efficiency when using a lower level discriminator (LLD) setting of 235 keV. (Although 90/10 Ar/isobutane is used in those calculations, the density of P-10 is very similar and thus the results are comparable.) Furthermore, reducing the drift gap width from 3 mm to 1 mm will nearly double the position resolution of the detector. For a 1 mm drift gap and an amplification gap of around 75 microns, the total neutron scattering probability in the gas will be 3.8×10^{-4}.

We now have sufficient information to estimate a best-case (i.e. minimized) thermal neutron scattering probability for a Micromegas neutron

beam monitor using current design conventions. An equivalent expression to Equation (5.1) for the total neutron scattering probability p is

$$p = p_{walls} + p_{gas} + p_{mesh} + p_{anode}. \quad (5.2)$$

Given a 25 micron Al cathode/detector wall, a 25 micron Si second detector wall with associated spacer posts, a 1 mm drift gap and a 75 micron amplification gap filled with P-10 gas, a mesh made of relatively weakly scattering conducting material, and Cr anode strips of 25 nm average areal coverage, a best-case p is around 1.1×10^{-3} at thermal. (See Figure 3.1 for a diagram of the Micromegas layout.) If quartz is used in lieu of Si as the second detector wall material, the best-case p value is then 1.5×10^{-3} at thermal. Table 5.3 breaks down the neutron scattering probabilities for the various Micromegas structural elements.

(d) Conclusions

Calculations of the neutronics characteristics of Micromegas neutron beam monitors reported in the literature to date indicate that for thermal neutrons their scattering probabilities are in the range of 10^{-1}-10^{-2}. Ideally, neutron beam monitors should have scattering probabilities at or below their neutron detection probabilities. The calculations in this chapter demonstrate that in a realistic best-case design scenario, neutron scattering probabilities at thermal of 1.1×10^{-3} may be achieved. Since intended neutron detection probabilities of the Micromegas neutron beam monitors to be used on the SNS beamlines vary from 10^{-5}-10^{-3} depending on the intensity of the neutron beamline in question, a higher scattering probability than desired will have to be accepted on some beamlines.

To minimize the neutron scattering probability, the following design parameters are recommended. The cathode/first detector wall should be as thin as possible (25 microns is suggested) and consist of Al. The second detector wall should consist of either Si or quartz, both of those materials are non-

Table 5.3. Best-case Micromegas neutron beam monitor scenario. Breakdown of a best-case thermal neutron scattering probability by Micromegas neutron beam monitor structural element. The two values for the second detector wall (and for the total) are for Si and quartz.

Structural Element	**Scattering Probability**
1st detector wall	2.5×10^{-4}
2nd detector wall	$3.4 \times 10^{-4} / 6.7 \times 10^{-4}$
gas	3.8×10^{-4}
mesh	1.5×10^{-4}
anode	1.7×10^{-6}
Total	$1.1 \times 10^{-3} / 1.5 \times 10^{-3}$

conducting and readily etchable. (Again, a 25 micron thickness is suggested.) The neutron target material in converter foil should be ^{10}B. The use of ^{10}B will give a minimum gas chamber thickness of 1 mm for the drift gap and 75 microns for the amplification gap. P-10 gas is ideal for use in the detector. The mesh separating the two regions of the gas chamber should be made of relatively weakly scattering conducting material such as Cr, and it should be kept as thin as is consistent with structural integrity. Cr may be used for thin (tens of nm average areal coverage) anode readout strips. If these design parameters are followed, scattering probabilities at thermal of $1.1x10^{-3}$ and $1.5x10^{-3}$ may be anticipated using Si and quartz second detector walls, respectively.

CHAPTER 6
COMPOSITE NEUTRON SCINTILLATOR SIMULATION: MONTE CARLO METHODS

(a) Introduction to Composite Neutron Scintillators

Nature and Use of Scintillators

Scintillators have been used for the detection of ionizing radiation from the very early days of nuclear science. In the scintillation process in inorganic materials, radiation deposits energy in a material through the formation of electron-hole pairs in which an electron is raised into the conduction band of the material and a hole is left in the valence band. While the electrons and holes can move through the valence and conduction bands of the materials independently, most group together into excitons, an exciton being an electron and hole combination that are bound together. They have not actually recombined but are in a lower energy state than an unbound electron and hole. This characteristic of being bound causes the electron and hole in an exciton to move together and thus they can be considered a single entity for practical purposes. Excitons can only recombine at specific sites in a scintillator and scintillation light is emitted when this occurs at an appropriate recombination site. (Not all recombination sites result in light emission.) Organic scintillators work in a somewhat different fashion and this description is not included in this work since it is concerned primarily with inorganic scintillators. Information on both organic and inorganic scintillator theory is readily available elsewhere [7, 22].

Scintillators are one of the most widely used types of ionizing radiation detectors. When used for this purpose, scintillators are placed in a light-proof housing and connected to a photosensitive device, such as a photomultiplier tube (PMT), which produces an electrical signal upon detection of the light. Scintillators typically demonstrate much higher particle detection efficiencies

than gas-based detectors due to their higher density (particle interaction coefficients are generally proportional to detector material density) and high maximum count rates due to the speed of most scintillation pulses (most scintillators produce pulse durations under 1 microsecond, although a few are significantly slower).

Composite Scintillators

The major limitation in the use of scintillators for neutron detection is the lack of different types of scintillators available, meaning that scintillators suitable for a particular neutron detection application are often unavailable. As a means of ameliorating this problem, composite scintillator mixtures consisting of fluorescent dopant particles (e.g. ZnS:Ag) suspended in a matrix material loaded with neutron-sensitive material (e.g. ^{6}LiF) are widely used for neutron detection [23-25]. The major reason for the popularity of composite scintillators is that they allow fluorescent materials that are insensitive to neutrons to be used for neutron detection, which widens the range of scintillator characteristics (e.g. light emission wavelength, scintillation pulse duration, quantum yield) available for neutron detection.

Composite scintillators enable non-neutron sensitive fluorescent dopant particles to be used for neutron detection by placing neutron target material (e.g. ^{6}Li, ^{10}B, H) in a matrix material surrounding the dopant particles that are embedded throughout the matrix. When a neutron interacts with a target atom in the matrix, it produces one or more energetic charged particles that travel through the scintillator, releasing energy as they go. When an energetic charged particles passes through a fluorescent dopant particle, it deposits energy in the particle, which in turn emits scintillation light. It is assumed in this work that light emission by the scintillator is proportional to energy deposited in the dopant particles and only energy deposited in the dopant particles is effective in producing light emission. In other words, electrons, holes, and excitons are assumed not to travel across the boundary between the dopant particle and the matrix material, which is usually true of most material

boundaries. The large dopant particle sizes (and thus separation distances in most simulation cases) relative to typical mean free paths of electrons, ions, and excitons in most materials suggest that most energy carriers will never come into contact with a boundary between the matrix and dopant particles anyway.

Optimization of Composite Scintillators

Although composite neutron scintillators based on ZnS:Ag fluorescent material have been in use for many decades, there are still many questions pertaining to how to optimize such scintillators that have hitherto been unanswered. Factors to consider when optimizing composite scintillators include fluorescent dopant particle size, dopant particle concentration, composite scintillator optical transmission properties, and neutron target material density. As an example, fluorescent dopant particle size profoundly impacts neutron detection efficiency and the scintillator's neutron energy peak characteristics, in addition to the obvious effect on scintillator optical transparency if the dopant and matrix materials have ill-matched optical indices. If the particles are very large with a commensurate average separation distance between them, many neutron capture events in the matrix will fail to produce scintillation since self-shielding of the energetic reaction products will stop many products before they reach a fluorescent dopant particle. Large dopant particle sizes also lead to broad and poorly formed neutron energy peaks associated with energy deposition by the reaction products. (The energy peaks are created by the reaction products with the energy coming from the Q value of the neutron absorption reaction, the energy of the neutron usually being orders of magnitude less than the Q value. Although these energy peaks thus are not primarily due to the energy of the neutron, they are usually referred to as "neutron energy peaks." That naming convention will be followed in this work.) Successful optimization of composite neutron scintillators will be greatly aided by knowledge of how different design parameters affect scintillator behavior.

Simulated Cases

This chapter and Chapter 7 focus on the creation of, and results obtained from, a Monte Carlo simulation of a composite fluorescent dopant/lithiated matrix material neutron scintillator. The simulation is intended to predict the effects of dopant particle size, particle concentration, and matrix and particle density on the light emission characteristics of the scintillator. The complexity of calculating estimates of optical transmission properties precluded including this in the simulation. Two types of scintillators are examined, the first being ZnS:Ag particles in a lithium-loaded glass matrix and the second organic dopant particles in a lithium-loaded organic matrix, with the focus being on the former due to the author's involvement in a project using those materials. The results are intended to provide guidance to persons seeking to use composite scintillators for neutron detection.

(b) Qualitative Effect of Design Parameters on Scintillator Performance

The composite scintillator Monte Carlo simulation is developed to provide quantitative data concerning the effect of composite scintillator composition parameters (e.g. dopant particle size and materials density) on scintillator performance characteristics. General qualitative effects of such parameters on composite scintillator behavior are predictable given sufficient thought and knowledge of the underlying physics, but they are not sufficient when seeking to optimize a scintillator formulation. However, a qualitative understanding is important (if not essential) to properly comprehending quantitative data. This section discusses the qualitative effects of composite scintillator design parameters on behavior in order to prepare the reader for the quantitative data that will be presented in Chapter 7.

Effect of Fluorescent Dopant Particle Size

Fluorescent dopant particle size affects composite scintillator behavior in several ways. When the dopant particles are very small, the scintillator is

practically a homogeneous mixture from the standpoint of the ^{6}Li reaction products. Assuming some reasonable concentration of the dopant particles in the scintillator, the fraction of the ^{6}Li reaction energy deposited in the dopant particles will equal the weight fraction of the dopant particles in the composite scintillator. It is when dopant particles are smallest that the neutron energy peak produced by the scintillator is most narrow and clearly defined. This is due to minimization of the variability of the fraction of the reaction energy deposited in the dopant particles. As fluorescent dopant particle size increases, significant variations in pulse amplitude from one neutron capture event to another begin to appear. This is followed by self-shielding effects on the part of the matrix material containing neutron target atoms (e.g. ^{6}Li). When dopant particles become sufficiently large, many neutron absorptions may fail to produce scintillation pulses due to the reaction products stopping in the matrix material before entering a dopant particle. (Since the dopant particles only occupy a fraction of the total volume, average particle sizes necessarily lead to increased average particle separation distances when the fractional volume of the composite scintillator consisting of dopant particles is held constant.) Figure 6.1 illustrates some of the effects caused by variation in dopant particle size well. (Although fluorescent dopant particles are regarded as spherical and of constant size in the simulation, a cross section of the scintillator will show the dopants forming circles of different sizes; this effect is not shown in Figure 6.1.)

Effect of Dopant Concentration

Fluorescent dopant particle concentration in the composite scintillator also has a very significant influence on scintillator behavior. High dopant concentrations maximize pulse amplitudes but limit the concentration of neutron target material in the composite scintillator as such material is restricted to the matrix. Lower dopant concentrations reduce average pulse amplitude and increase pulse amplitude variability.

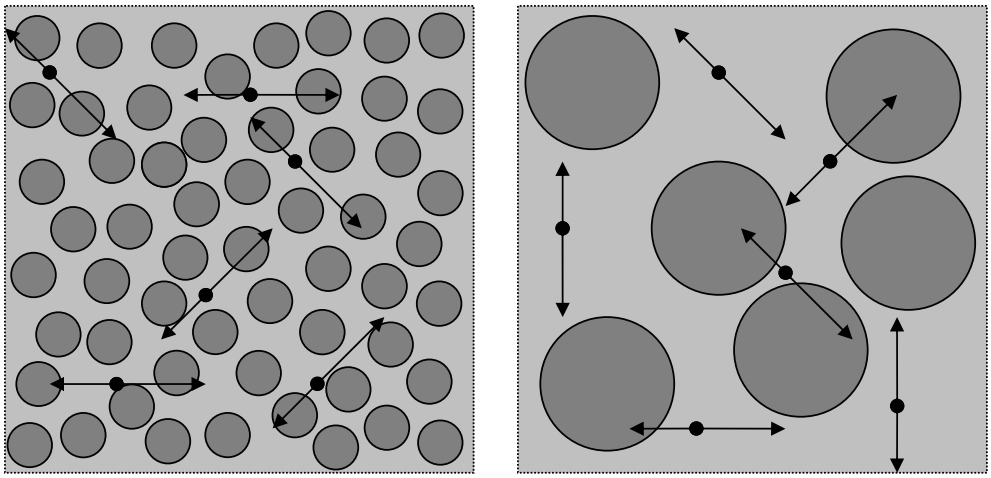


Figure 6.1. Dopant size and light emission. The effect of fluorescent dopant particle size on light emission: large particles result in much more variable pulse amplitudes and a lower probability of scintillation given a neutron absorption event.

Effect of Materials Density

Fluorescent dopant particle and matrix material density significantly affects scintillator characteristics. If the dopant particles are significantly denser than the matrix material energy deposition in the composite scintillator will be skewed towards the dopant particles and vice versa. Since the range of an energetic charged particle range is very nearly inversely proportional to the density of the material, changing the density of the dopant particles and matrix material by an equal fraction is equivalent from the standpoint of scintillation pulse characteristics to changing the dopant particle size. Thus increasing the densities is equivalent to increasing the dopant size and decreasing the densities shrinking the size.

Dependence of Parameters

It is important to note that parameters such as dopant particle size, dopant concentration, and materials densities are not truly independent of each other. While changing one parameter does not change another parameter, the qualitative effect of changing one parameter may be strongly influenced by the value of another parameter. For example, if the size of the dopant particles is greater than the combined track lengths of the reaction products, the shape of the neutron energy peak will be relatively insensitive to the dopant particle concentration. In lieu of an effect on the peak shape, the probability that a neutron capture will lead to a scintillation event will decrease. If the dopant particle sizes are significantly less than the combined reaction product track lengths but are still a significant fraction thereof, the neutron energy peak shape will change significantly with decreasing dopant concentration. (This is discussed in greater detail in Chapter 7.) Finally, if the dopant particles are very small compared to the reaction product track lengths, decreasing the dopant concentration will primarily affect the amplitude of the neutron energy peak (as opposed to its shape). Thus, we see that the effects of different parameters are indeed dependent on the values of other parameters.

(c) Methods Used in Simulation

The composite neutron scintillator simulation uses Monte Carlo methods in a series of calculations that cover: neutron absorption in the scintillator, creation of reaction products, and transport of the products in the composite scintillator. Transport of the reaction products in the scintillator necessarily includes: distribution of fluorescent dopant particles within the composite scintillator, determination of reaction product path through the matrix and the dopant particles (necessitating a geometry treatment including distinct volume regions and surface crossing calculations), and energy deposition in the matrix and in the dopant particles.

Besides Monte Carlo-type calculations, the simulation also contains significant geometry-handling elements. Since the composite scintillator is simulated in part through the use of spherical and cubic geometrical objects, and through a series of linear travel path segments describing reaction product paths, routines are included for finding surface crossing points and so forth.

Geometry-Handling Methods

A critical aspect of the simulation is how the composite scintillator geometry is modeled, including random distribution of dopant particles in the matrix material. (It is assumed that dopant particles do not clump together or do other things that tend to make their distribution non-random.)

An initial method of simulating the geometry involved consideration of a finite volume around the reaction product tracks and selection of random locations for dopant particles until the desired volume doping level was reached in the region. This proved unsatisfactory for several reasons. The computational time associated with checking to ensure a random location did not result in a particle overlapping previous particles and selecting a new location if it did became extremely large for small particle sizes. (Time counters were placed in the code to determine where CPU time was being consumed.) Once all the particles were in place, surface crossing calculations

consumed excessive computational time due to the need to check very large numbers of dopant particles to determine which one was the next one a reaction product entered. This is essentially the same problem as the computational time problem that occurs in MCNP when large numbers of smaller cells not in physical contact with each other reside within a larger cell [3]. Methods were implemented to speed up calculational time by eliminating distant dopant particles without actually calculating exact distances or track crossing points on the dopant particle surfaces. This helps, but it does not ultimately resolve the problem. Another difficulty is that particles tended to concentrate around the edge of the simulated volume where they could partially extrude out the side and fit in more easily, which reults in a lower particle concentration in the center of the volume where the reaction products give up much of their energy. Methods were developed to reduce the severity of the problem but ultimately it does not completely resolve the problem apart from simulating a very large volume of scintillator which only makes the computational time impractical. After constructing and testing a model using the first dopant particle random distribution method and after discovering problems that prove intractable, it became clear that a different method would be needed.

After some consideration, a second geometry handling method was devised. This second method uses a series of individual cubic cells that contain a single dopant particle. Reaction products are tracked through successive cubic cells. Figure 6.2 illustrates a cross section of a cubic cell that contains a fluorescent dopant particle along with some path segments of a particle traversing the cell. Figure 6.3 shows how a particle is tracked through successive cells. A cell contiguous with the previous cell (on the side a reaction product exits the first cell) is created when needed and is given a random offset from the exit point on the first cell and given a fluorescent dopant particle location. (Again, the dopant particles are spherical and randomly distributed, thus a cross section of the scintillator will show circles of

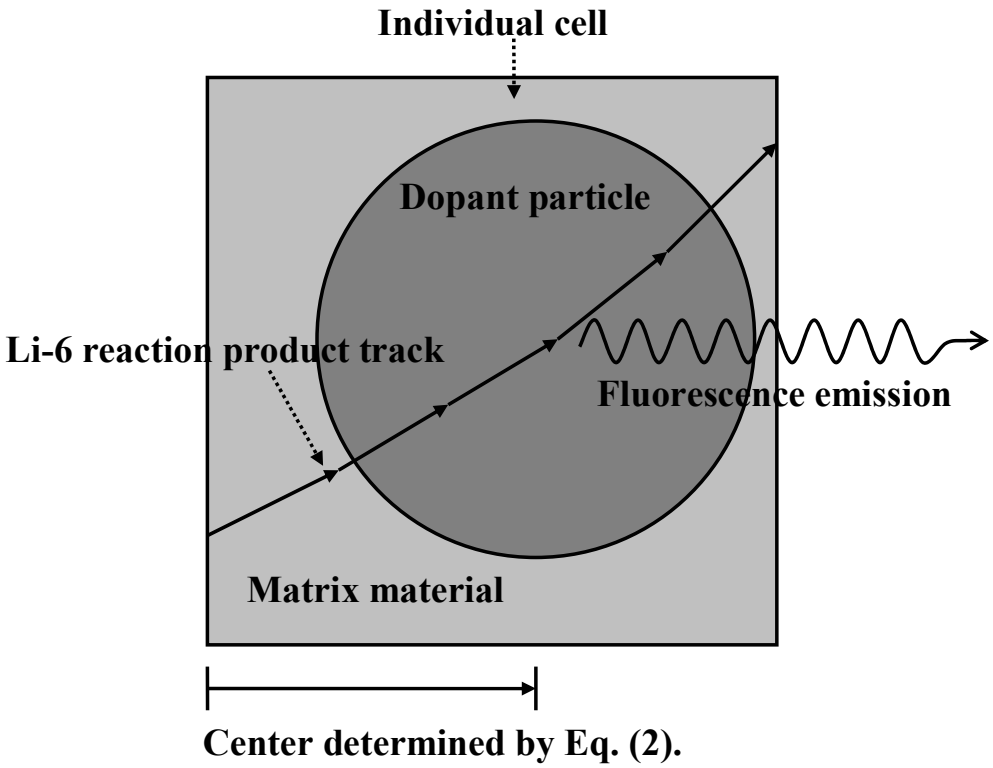


Figure 6.2. Cells for geometry handling. Diagram of an individual cell through which a ^{6}Li reaction product moves. The location of the fluorescent dopant particle within the cell is determined randomly using Equation (6.2). Energy deposition in the particle causes it to emit light.

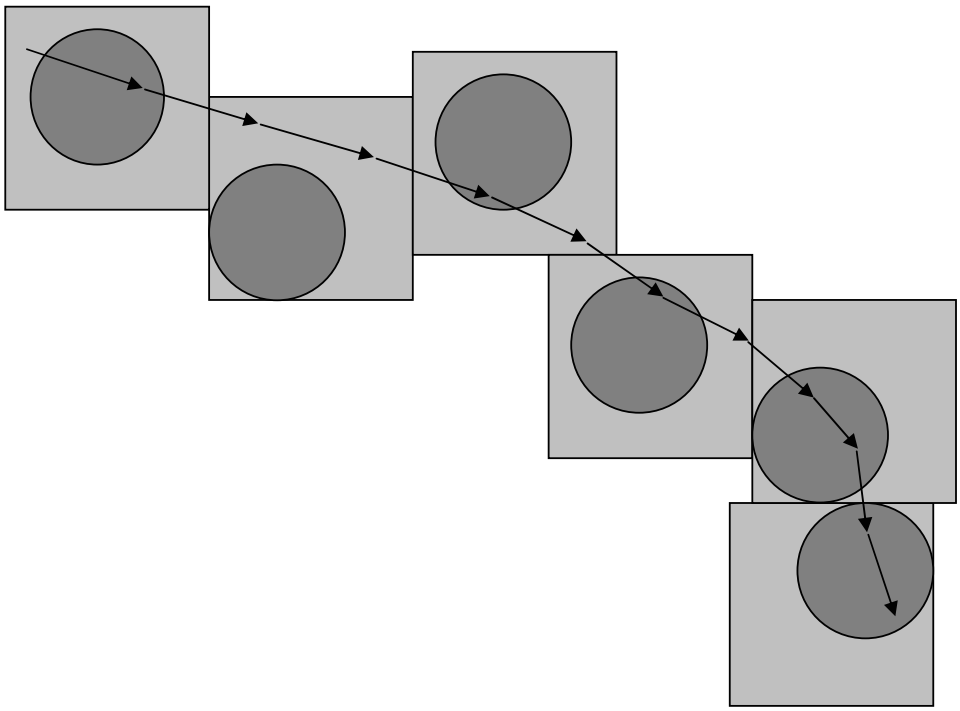


Figure 6.3. Creation of new cells. As a reaction product travels along its simulated path, a new cell containing a fluorescent dopant particle is created each time the reaction product leaves the cell it was in. When a boundary crossing occurs, a random offset is selected for new cell along with a particle location in it.

different sizes in the cells, and in some cases no dopant particle at all. This is not shown in Figure 6.3.)

Geometry Calculations

The Monte Carlo methods used in the simulation are described below. A cell size is selected such that the fraction of its volume occupied by the single fluorescent particle in it equals the volume doping fraction specified by the user. Thus, the side lengths of the cubic cell are

$$l = \left(\frac{4\pi r^3}{3f} \right)^{1/3}, \qquad (6.1)$$

where r is the radius of the fluorescent dopant particles and f is the volume doping fraction of the fluorescent particles in the composite scintillator. Since dopant particles are spherical and cells are cubic, the maximum volume doping fraction is approximately 0.5236.

The spherical dopant particle is distributed randomly within the cell with the restriction that it cannot extend beyond the cell boundaries. Thus the center of the sphere in any dimension (relative to the start of the cell in that dimension) is given by

$$c = r + \xi(l - 2r), \qquad (6.2)$$

where r is the dopant sphere radius, l is the length of the cell, and ξ is a random number between 0 and 1.

Reaction Product Creation

The equation for the ensuing reaction when ^{6}Li captures a neutron is

$$^6\text{Li} + \text{n} \rightarrow {}^4\text{He} + {}^3\text{H}, \text{ Q} = 4.78 \text{ MeV}. \qquad (6.3)$$

The reaction location is randomly selected within the portion of cell volume not occupied by the dopant particle. The simulation includes accurate modeling of reaction and neutron energy distribution between the two reaction products and between the directions of the products relative to each other and the incident neutron. When neutron energy is minimal (i.e. thermal or low epithermal), conservation of energy and momentum dictate that 2.73 MeV is

imparted to the alpha (^{4}He) product, 2.05 MeV to the triton (^{3}H), and that their initial directions are very nearly opposite. A more complete discussion of this may be found in Chapter 3, Section (b).

Reaction Product Transport

As discussed in Chapter 2, reaction product transport is simulated using position-energy pairs for approximately evenly distributed points along the reaction product paths generated using the TRIM code. Individual reaction product path segments (i.e. between two position-energy points) are processed sequentially and a constant energy distribution per unit path length is assumed within an individual segment. Changes in product LET with energy occur between path segments. Paths are generated using 1 g/cm^3 silicon as the target material. Silicon was chosen as it has a similar Z number to atoms such as S and C that are frequently the basis for neutron scintillators. As stopping powers for heavy charged particles are nearly proportional to the material density through which the charged particles travel (some variation from linearity with the material Z number does occur), the distance traveled in each path segment is scaled using the matrix and dopant material densities. A density of 1 g/cm^3 was used in the TRIM calculations as a convenient value from which to scale distances.

Considering a single cell at a time made the program faster and less complex than when simulating a large volume all at once containing the entire particle track. The simulation code processes a single travel path segment for a particular particle at a time and finds how much of the energy lost within the segment is released inside dopant particles and how much within the lithiated matrix material. This entails checking each segment to determine if and where it crosses the dopant particle and cell boundaries. When a cell boundary crossing occurs, a new cell is created with a boundary on the same plane as the boundary of the old cell through which the particle exited. The new cell also has a random displacement across the plane relative to the boundary crossing point. The random displacement is sampled from an interval that allows the

entrance point into the new cell to cover but not extend beyond the boundary of the new cell. The equation for the random offsets in the two dimensions across which the boundary extends is

$$\Delta = (l + \delta) - \xi(l - 2\delta), \tag{6.4}$$

where l is the length of the cell, and ξ is a random number between 0 and 1. The constant δ is a very small number ($\sim 10^{-5}$ was used when distances were in microns) included to prevent entrance points falling on two different perpendicular cell boundaries and potentially causing difficulties in the code.

Successive path segments are run from the point at which the last segment ended and both reaction products are simulated in order to get the total energy deposited in the fluorescent dopant particles as the result of a neutron capture event. Referring again to Figure 6.3 it may be seen how a set of reaction product tracks extend through successive cells until the product is stopped, with random offsets occurring at each cell boundary crossing. Since the products usually travel in approximately straight lines over most of their path lengths, they are unlikely to reenter areas they have previously left and any effects stemming from reentered areas having different cell and dopant particle locations the second time around are likely to be very small and thus are not considered.

Geometry handling in the simulation requires numerous calculations of surface crossing points for a reaction product travel path segment and determinations of whether or not a point falls within a dopant particle or a cell. Such calculations are not random but deterministic; thus, they do not use Monte Carlo methods. Nonetheless, the methods used for such calculations will be described below as they are integral to the successful implementation of a simulation.

Surface Crossing Points for Spheres

Whether or not a point lies within, on, or outside a sphere is determined as follows. Given a point (x, y, z) and a sphere (e.g. a dopant particle) centered

at location (x_c, y_c, z_c) with radius r, the distance d between the center of the sphere and the point is given by

$$d = ((x - x_c)^2 + (y - y_c)^2 + (z - z_c)^2)^{0.5} . \quad (6.5)$$

If $d < r$, the point lies within the sphere; if $d = r$, the point lies on the sphere; and if $d > r$, the point lies outside the sphere. In practice, when writing a computer simulation, small errors slip into location values due to the limited decimal precision of floating point number variables used in computer programs. To avoid very considerable errors that will arise during geometry treatments, it is necessary to include a small tolerance factor. Thus, if $d < r - \delta$, the point lies within the sphere; if $r - \delta < d < r + \delta$, the point lies on the edge of the sphere; and if $d > r + \delta$, the point lies outside the sphere. It is also very useful to use another variable to track whether a point has been previously determined to be outside, on, or in a sphere (the same is true for other geometrical shapes also) as an extra precaution against confusion of location around surfaces, and this approach is utilized in the simulation code.

Finding the crossing points of a travel path segment across the surface of a sphere is more complex and the methods used are best broken down by case. Since the travel path segment is on a line, the equation of the line is determined and the line is tested to see if it crosses the sphere. The first case to consider is that in which both the start and end points of the segment are outside the sphere. In this circumstance there are either two crossing points or none at all. To determine whether this is the case, a quadratic equation is solved involving seeking a set of points where the line equation and the sphere equation are equal. Thus,

$$a = (x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2 , \quad (6.6)$$

$$b = 2((x_2 - x_1)(x_1 - x_c) + (y_2 - y_1)(y_1 - y_c) + (z_2 - z_1)(z_1 - z_c)) , \quad (6.7)$$

$$c = x_c^2 + y_c^2 + z_c^2 + x_1^2 + y_1^2 + z_1^2 - 2(x_c x_1 + y_c y_1 + z_c z_1) - r^2 , \quad (6.8)$$

and

$$d = b^2 - 4ac , \quad (6.9)$$

where (x_1, y_1, z_1) is the start point of the path segment, (x_2, y_2, z_2) is the end point, (x_c, y_c, z_c) is the sphere center point, and r is the sphere radius. If $d < 0$, the path segment does not cross the surface of the sphere. (Since a is always positive, it is not necessary to divide by $2a$.)

If the path segment does cross the surface of the sphere, the fractions of the distance between the start point and the end point of the path segment at which the two crossing points occur are given by

$$f_1 = (-b - (b^2 - 4ac)^{0.5}) / 2a , \tag{6.10}$$

and

$$f_2 = (-b + (b^2 - 4ac)^{0.5}) / 2a . \tag{6.11}$$

Thus the crossing points are calculated after the manner

$$x_{cross,1} = x_1 + f_1(x_2 - x_1) , \tag{6.12}$$

with the obvious substitutions made for y and z and the second crossing point for the first. Once the two crossing points are found, they need to be tested to see if they fall beyond one of the two ends of the path segment (in which case the path segment does not actually cross the surface of the sphere) and to see if the surface crossings occur at one or both of the start and end path segment points. Again, care must be exercised to ensure that point locations relative to surfaces do not become confused as this will inevitably result in severe intractable problems in the simulation.

If one of the two points that defines the path segment falls within the sphere and the other outside of it, a single crossing point exists. To determine where that crossing point is, f_1 and f_2 are calculated after the manner of Equations (6.10) and (6.11), using the previous equations as needed. Whichever of f_1 and f_2 falls between zero and one is the correct f value to determine the location of the surface crossing.

A final case encountered in the simulation in which a sphere crossing point may exist is that in which the path segment start point sits on the sphere boundary and the end point sits out the sphere. In this case there can be one

crossing point or none. In this event the same procedure is used for that in which both end points sit outside there sphere. Any crossing points found are then tested to determine if they are the same as the point that is already sitting on the sphere boundary (a small delta is used to filter out essentially identical points) and if so are thrown out.

In addition there are cases in which no crossing points exist. These include when the start and end points are both inside the sphere and when one is on the surface of the sphere and the other is inside the sphere.

Surface Crossing Points for Cubes

Equivalent geometry handling methods are used for the cubic-shaped cells as for the spherical dopant particles. A point (x, y, z) sits inside a cubic-shaped cell if it satisfies the conditions $x_{lower} + \delta < x < x_{upper} - \delta$, $y_{lower} + \delta < y < y_{upper} - \delta$, and $z_{lower} + \delta < z < z_{upper} - \delta$, where δ is defined as earlier and x_{lower}, x_{upper}, and so forth are the locations of the edges of the cell in different directions. (Thus $x_{upper} = x_{lower} + l$, where l is the length of the cube on one side.) If the point satisfies the conditions $x_{lower} - \delta < x < x_{lower} + \delta$, $y_{lower} + \delta < y < y_{upper} - \delta$, and $z_{lower} + \delta < z < z_{upper} - \delta$, it is located on the side of the cell that sits on the plane $x = x_{lower}$. The conditions for determining if the point sits on one of the seven other bounding planes of the cell are obvious. If the point satisfies the conditions $x < x_{lower} - \delta$ or $x > x_{upper} + \delta$, $y < y_{lower} - \delta$ or $y > y_{upper} + \delta$, and $z < z_{lower} - \delta$ or $z > z_{upper} + \delta$, then it sits outside the cell.

If both the start and end points of a path segment lie outside the cell, either two crossing points exist or none at all. A simple check to see if crossing is possible may be performed before checking for actual crossing points. For example, if $x_1 < x_{lower}$ and $x_2 < x_{lower}$ or $x_1 > x_{upper}$ and $x_2 > x_{upper}$ (and vice versa for y and z) then crossing points on the surface of the cubic cell are clearly impossible. If surface crossing points are possible, potential crossing points on each side of the cell may be calculated as follows:

$$f = (x_{lower} - x_1)/(x_2 - x_1), \tag{6.13}$$

$$x_{cross,x-lower} = x_{lower}, \tag{6.14}$$

$$y_{cross,x-lower} = y_1 + f(y_2 - y_1), \tag{6.15}$$

and

$$z_{cross,x-lower} = z_1 + f(z_2 - z_1). \tag{6.16}$$

The values for $y_{cross,x\text{-}lower}$ and $z_{cross,x\text{-}lower}$ are then checked to see if they meet the requirements of falling with the y and z bounds of the cell. If they do, a real crossing point has been found. If not, then not. The substitutions for finding potential crossing points on the other seven sides of the cubic cell are obvious. If two crossing points are found (there will be either two or none), the order in which they occur may be ascertained by comparing the two f values; the lower f value will belong to the first crossing point and higher f value to the second.

If one path segment end point lies within the cell and the other one outside the cell, a single crossing point exists. The same methods are used as when two crossing points exist except with the exception that potential crossing points on some sides of the cell may be eliminated; for example, if (x_1, y_1, z_1) lies within the cell and $x_2 > x_{upper}$ then there cannot be a crossing point on cell boundary lying on the plane $x = x_{lower}$. If one path segment end point is on a cell boundary and the other is outside the cubic cell, either one crossing point exists or none at all. The same methods are used to calculate potential crossing points and all crossing points generated must be checked to ensure they are not the same as the segment end point that sits on a cell boundary. The cases in which no boundary crossings are possible are obvious and may be eliminated right away.

Reaction Product Location Tracking

It is important to keep track of a reaction product's location relative to the major geometric elements in the simulation (the cubic cell and the dopant particle it contains) and its previous location as the reaction product is tracked through the composite scintillator. There are several reasons for this. One is

that confusion easily results when finding surface crossing points. For example, if a point lies on a surface, the mathematical procedure for finding surface crossing points between that point and another point will often return a crossing point that is the same as the point lying on the surface. The precision of a point location is limited by the number of decimal places of accuracy in computer floating point numbers and two points that are essentially identical may have a small difference several decimal places out. Thus a situation may easily arise in which essentially the same surface crossing point is repeatedly returned and the reaction product is advanced each time to that same point (albeit with a small difference after three or four decimal places) and the reaction product becomes stuck at that location, the victim of an infinite loop in the computer code. It is important to check to see if the two points are essentially the same using a small δ as a minimum separation distance for them to be considered two separate points. Augmentation with a variable to track the location of the reaction product with respect to the surfaces of the geometrical objects (in the object, on the object's surface, outside the object) is also very helpful as the reaction product can be tested to see if a new location generated for it is consistent with its path to that point. Another important point is that the energy a reaction product releases in the composite scintillator needs to be added to the correct total for that particle. In other words, energy deposited in the fluorescent dopant particles and energy deposited in the matrix must be added to the correct tallies, since only energy deposited in the fluorescent dopant particles produces scintillation light emission. Reaction product path segments must also be scaled according to the density of the material through which the reaction product is traveling, the densities of the matrix and the dopants frequently being different from each other.

Scaling of Travel Distance and Calculation of Energy Deposition

Reaction product path segments are based on TRIM data calculated for travel through 1 g/cm^3 Si, as discussed earlier in this section. As a path segment is processed, the reaction product is started at the start point of the

segment, advanced to the first boundary crossing point along that segment, advanced to the second boundary crossing, and so forth until the end point is reached. It is found most convenient to keep remaining travel units after each step within the segment in the original units from travel in the Si to avoid scaling this value every time the material density of the region the reaction product is in changes. Since the densities of the matrix and the fluorescent dopant materials are frequently different than 1 g/cm^3, end points to test (i.e. to see if any surface crossing points occurred between the current point and the test end points) were calculated by scaling the remaining travel distance according to the material density and adding that to the current point. (Since the remaining travel distance was kept in the original units, scaled units for generating test points were kept as separate variables and were generated from the original units when needed.) Thus,

$$x_{test} = x_s + x_r / \rho , \tag{6.17}$$

where x_s is the starting x location, x_r is remaining x travel distance (based on travel through 1 g/cm^3 Si), and ρ is the density of the material being traveled through in units of g/cm^3. (The units for x_r are cm(g/cm^3), or g/cm^2. Dividing by the density of material being traveled through gives a range in cm.) The values of y_{test} and z_{test} are found using the obvious substitutions. When the first boundary crossing point the reaction product makes is established (if it crosses a boundary before reaching the end of the segment), the fraction f it traverses of the segment before reaching the crossing point is given by

$$f = (x_{cross} - x_s) / x_r . \tag{6.18}$$

This assumes that the reaction product has some component of motion in the x direction. If not, y or z values or distances between the points may be substituted in the obvious fashion. Given f, the values of the remaining travel distance and reaction product energy to expend in that path segment are given by

$$x_{r,new} = (1 - f) x_{r,old} \tag{6.19}$$

and

$$E_{r,new} = (1 - f)E_{r,old}\,, \tag{6.20}$$

where E is the reaction product energy relinquished in that path segment. The energy from the reaction product deposited in the region through which it passes is

$$\Delta E = fE_{r,old}\,. \tag{6.21}$$

(d) Organization of Simulation Code

In addition to calculating energy deposition in the matrix and dopant materials from a neutron capture event, the composite neutron scintillator simulation program includes other features such as the ability to simulate scintillation light transport in a composite scintillator that is somewhat opaque to its own light. These other features are a work in progress as this dissertation is being written and are not discussed in this dissertation or used in the calculation of any of the simulated results.

Figure 6.4 is a flowchart that gives an overview of the organization of the simulation program. The flowchart shows the major steps, logical decisions, and calculations made during the course of program execution. The subroutine that handles individual reaction product travel path segments is separated out into Figure 6.5.

Several points should be noted. The simulation is set up such that it can use output from MCNP to get neutron absorption locations and the energies and directions of the neutrons thus absorbed. This is useful for calculating the intensity of light emission at the surface of a piece of scintillator when the scintillator is at least somewhat opaque to its own light. This particular feature is available, but it is not used for any of the calculations presented in this work. When this feature is not used, effects related to scintillator size are omitted. It is also set up so that the user can calculate new reaction product tracks using TRIM or can use existing data. The program is

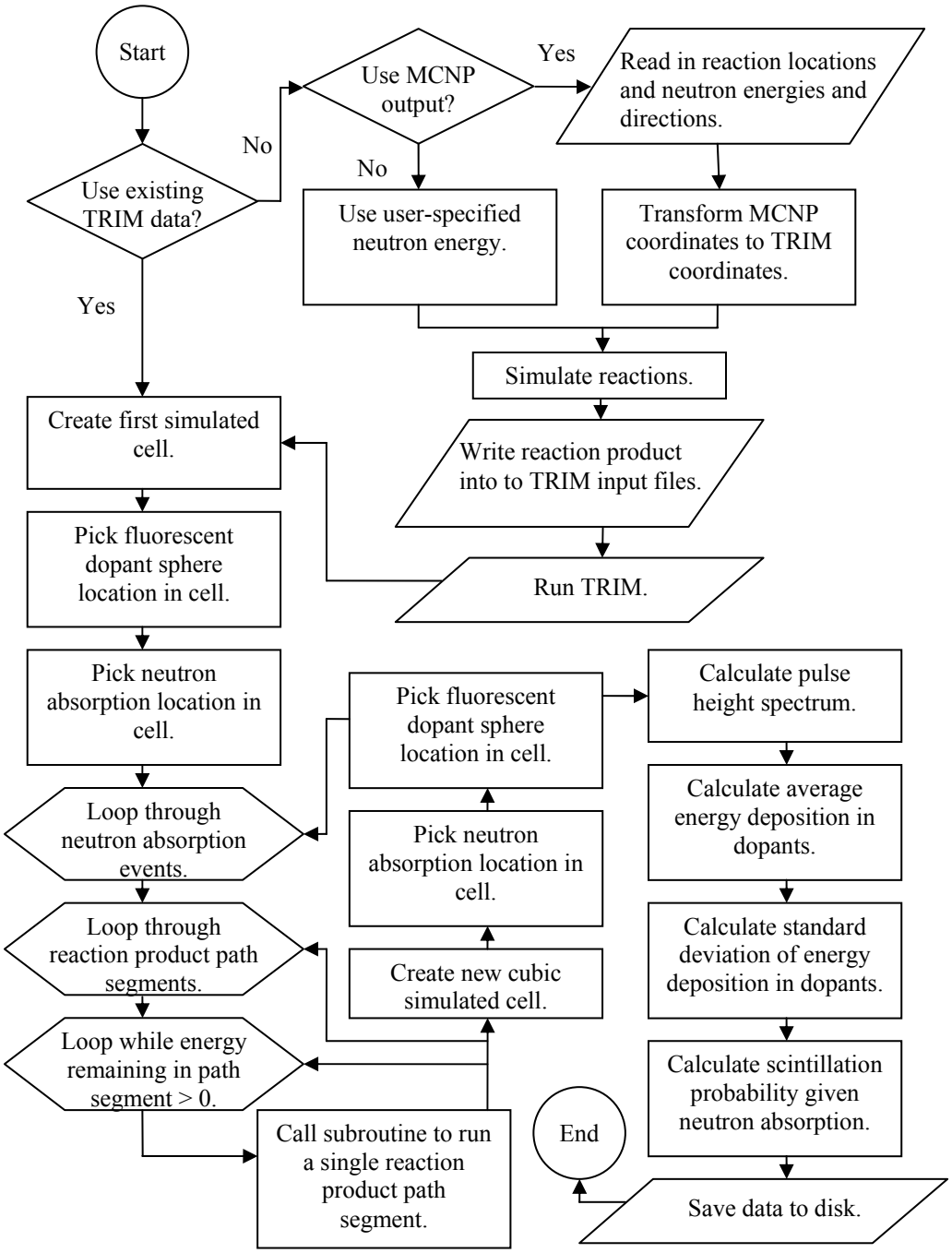


Figure 6.4. Flowchart of simulation program. Note the call to the subroutine that processes individual reaction product travel path segments.

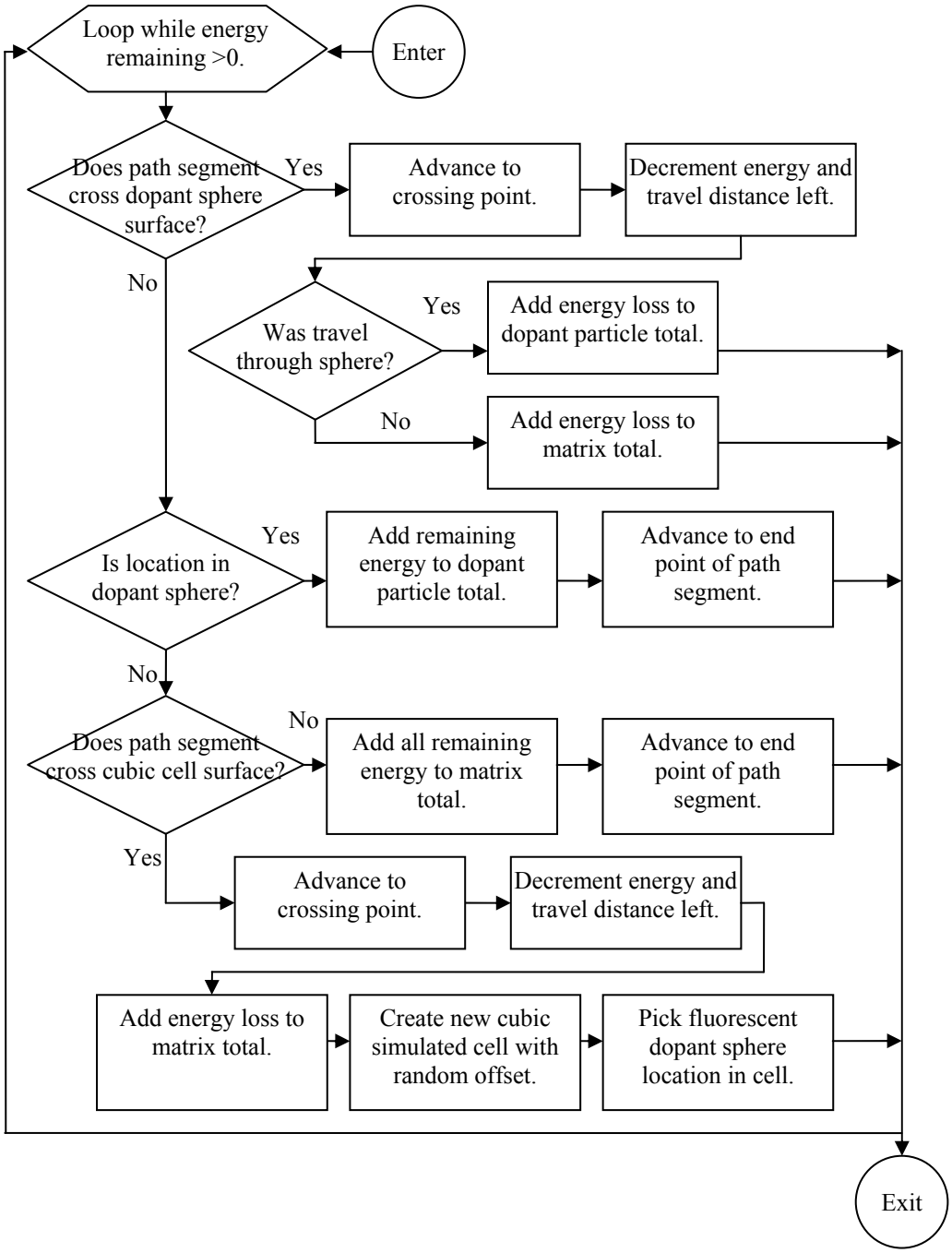


Figure 6.5. Flowchart of subroutine for handling path segments.

currently controlled by manually altering the values of constants and variables in the source code that specify the parameters needed for a simulation run. (The values that control each run are the dopant particle size, dopant and matrix density, dopant concentration, and number of events to run.) To date, no visual front-end has been constructed for the user. This scheme has the advantage of being very flexible; for example, the user can set up the code to run many different cases sequentially with no user intervention. However, it requires the user to have some knowledge of Visual Basic 6 (the computer language in which the simulation is coded) and access to the VB6 package.

(e) Code Verification by Testing Calculational Methods

There are two aspects to code validation. Verification involves testing of routines in the code to verify that they are indeed doing what they are intended to do. Validation involves comparing simulated results with real, measured data, or, in the absence of any such data, performing simulations for which expected results may be readily calculated by hand. In the latter case (since simulations are designed to predict results for problems that are too complex to calculate by hand) this entails the use of very simple or straightforward problems. Validation by this method is dealt with in Chapter 7.

Significant effort is expended in verifying the code by testing individual routines in the code dealing with critical aspects of the simulation calculations. For example, routines for finding crossing points on geometric surfaces given different line path segments (representing reaction product travel) were repeatedly tested to ensure the routines are functioning correctly. Cell and dopant particle locations were printed out along with the reaction product locations as the reaction products traveled along to verify that the locations of the cells and dopant particles were always consistent with that of the reaction products. Another verification method used is to sum the energy deposited in the dopant particles with the energy deposited in the matrix

material for an individual neutron capture event to make sure that the total is equal to the sum of the energy of the incident neutron and the Q value of the neutron capture reaction. Finally, checks are built into the code such that if unanticipated events or inconsistent variable values are encountered, an error indicating the nature of the problem is returned to the user and code execution ceases.

The simulation code was tested during its development to quickly expose errors when they occured (e.g. a reaction product becoming lost by exiting the cell under consideration without an appropriate new cell being created) and to allow problems to be corrected right away. Testing uncovers areas where particular care is needed (e.g. identification of two identical crossing points on a spherical surface as separate points due to small differences in numbers at several decimal places out or more) and enables precautions to be taken to avoid problems arising in areas where they are prone to occur.

One final item is noted. When dopant particle sizes become very small, errors occasionally occur due to rounding off of point location values as described earlier in Section (c) of this Chapter. This occurs almost exclusively at fluorescent dopant particle radii of 10 nm or less. (Normally these errors are avoided through the use of a small δ value as described earlier but eventually mathematical limits are reached.) If an error does occur, it is automatically detected and code execution ceases. When this happens, the user should rerun the code until an error-free run occurs.

Other than for the error just described (associated with mathematical limitations of computer simulations) that occasionally occurs at small dopant sizes (and that is automatically kept from infecting output data), the simulation routines in the code have proved extremely reliable when tested at correctly calculating results from specific inputs.

CHAPTER 7
COMPOSITE NEUTRON SCINTILLATOR SIMULATION: RESULTS AND DISCUSSION

(a) Introduction

The composite scintillator simulation is written to provide quantitative data concerning the effect of fundamental design parameters on composite neutron scintillator performance. Specific parameters of interest include fluorescent dopant particle size, dopant and matrix densities, and dopant concentrations in the scintillator. A qualitative description of how these parameters affect performance is found in Chapter 6 Section (b), and a description of the methods used in the simulation is found in Section (c) of that same chapter.

To briefly restate information discussed in Chapter 6, composite neutron scintillators consist of non-neutron sensitive fluorescent dopant particles mixed into a non-scintillating matrix material containing neutron-sensitive materials (e.g. ^{6}Li, ^{10}B). When a neutron is absorbed in the target material in the matrix, it creates one or more energetic reaction products. These reaction products travel through the composite scintillator, relinquishing their energy as they go, until they lose all of their energy and stop. Energy deposited by the reaction products in the fluorescent dopant particles results in scintillation light emission. A cursory examination of the nature of this operation will establish that the amount of energy deposited in the dopant particles by a specific event will depend on factors such as the dopant particle size, dopant concentration, variations in the path of the reaction products, distribution of the dopant particles in the composite scintillator, and other factors.

Almost all composite neutron scintillators in use today are based on ZnS:Ag doped into lithiated, boronated, or hydrogenated matrix material.

Lithiated and boronated scintillators are typically used for thermal neutron detection and hydrogenated scintillators for fast neutron detection. Of these three types of neutron target material, lithium (specifically the ^{6}Li isotope) is most common. Thus, it was decided to focus on lithium target material. Thus two basic scintillator types are examined: ZnS:Ag in lithiated glass (representative of inorganic cases generally) and an organic case consisting of organic fluorescent dopant particles in a lithium-bearing organic matrix.

(b) Simulation Cases

The two types of composite scintillators investigated are ZnS:Ag in a lithiated glass matrix, and organic fluorescent particles in a lithiated organic matrix. The material densities used are 4.1 g/cm^3 for the ZnS:Ag, 1.409 g/cm^3 for the lithium-bearing glass, and 1 g/cm^3 for both the organic dopant and matrix materials. A range of particle sizes from 5 nm to 100 microns in radius are simulated at volume concentrations in the composite scintillator from 0.02 to 0.4 for ZnS:Ag and 0.02 to 0.5 for the organic case.

The simulation code is used to calculate several different types of parameters relating to the predicted behavior of the composite scintillator. The parameters are as follows:

(1) the probability that a neutron absorption event will lead to fluorescent light emission (equivalent to the probability that one or both ^{6}Li reaction products will at some point enter a fluorescent dopant particle),

(2) the average and standard deviation of energy deposited in dopant particles by such events (both for all events and only for events producing light emission),

(3) simulated pulse height spectra based on the energy deposited in particles from each event.

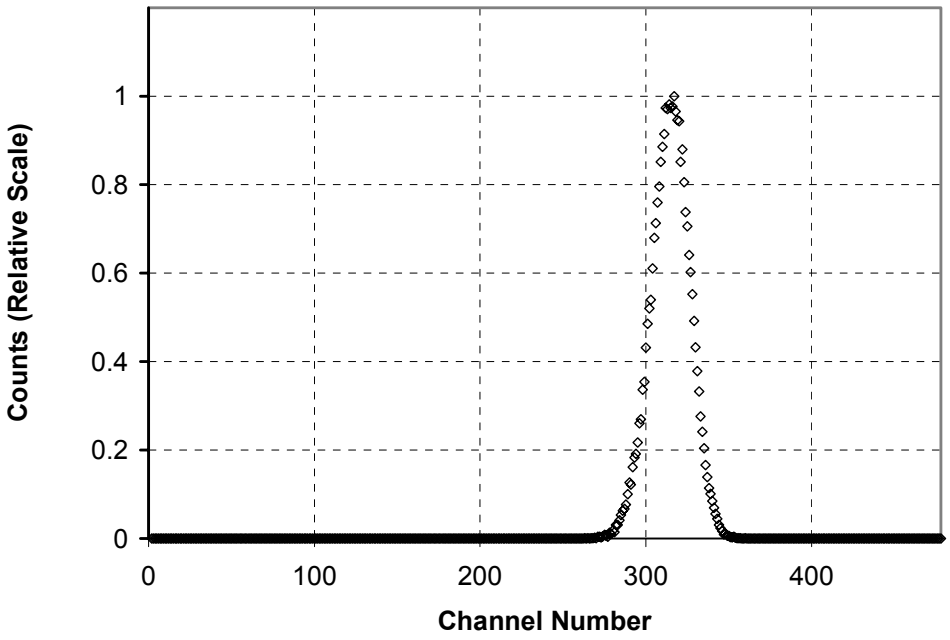
Together, these parameters give a very good indication of how changes in dopant particle size and concentration affect the behavior of the composite

scintillator. The second and third parameters are most meaningful if the number of photons emitted is directly proportional to the energy deposited in the fluorescent dopant particles. For high light output scintillators such as ZnS:Ag, where a neutron capture event may result in the emission of a hundred thousand photons, statistical variations in the number of photons emitted per unit energy deposited in the dopant particles are small (<1%) relative to variations in the amount of energy deposited in the dopant particles. Thus the simulated pulse height spectra are good indicators of what may be expected in actuality.

(c) ZnS:Ag in a Lithiated Glass Matrix

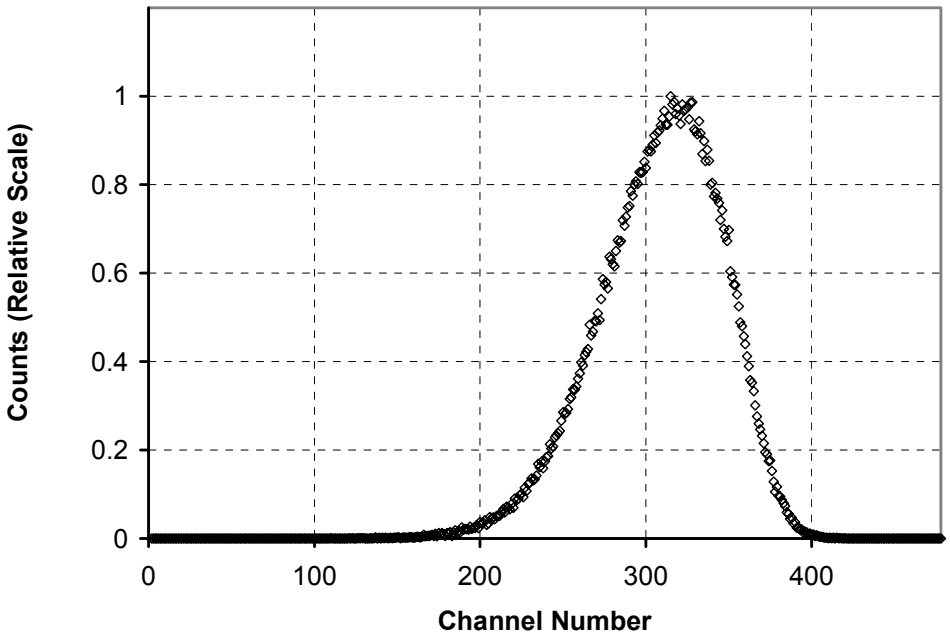
Pulse Height Spectra as a Function of Particle Size

Figure 7.1 shows simulated composite scintillator pulse height spectra containing fluorescent dopant particles from 50 nm to 10 microns in size at volume concentrations of 0.4 and is most instructive in the physical effects at work. In Figure 7.1(a), the very small particle size compared to the ^{6}Li reaction products track lengths (about 45 microns in total for the two products) makes the composite scintillator nearly homogeneous from the standpoint of reaction products traveling through it. Thus, there is little variation in the fraction of reaction energy deposited in the particles, and the simulated pulse height spectrum shows a very distinct neutron energy peak. As particle size is increased to 500 nm (Figure 7.1(b)), variability of energy deposition in the particles increases markedly. Increasing particle size into the micron range (Figure 7.1(c) and (d)) leads to a second peak below the main neutron energy peak first appearing and then becoming prominent. The lower peak seen clearly at a 2 micron particle radius (Figure 7.1(d)) is formed mainly by cases in which only one reaction product enters a dopant particle, the other product expends itself in the matrix material. In this case the total energy deposited in the particles is limited to the maximum energy carried by a single product; thus, separation between the peak produced by both reaction products entering



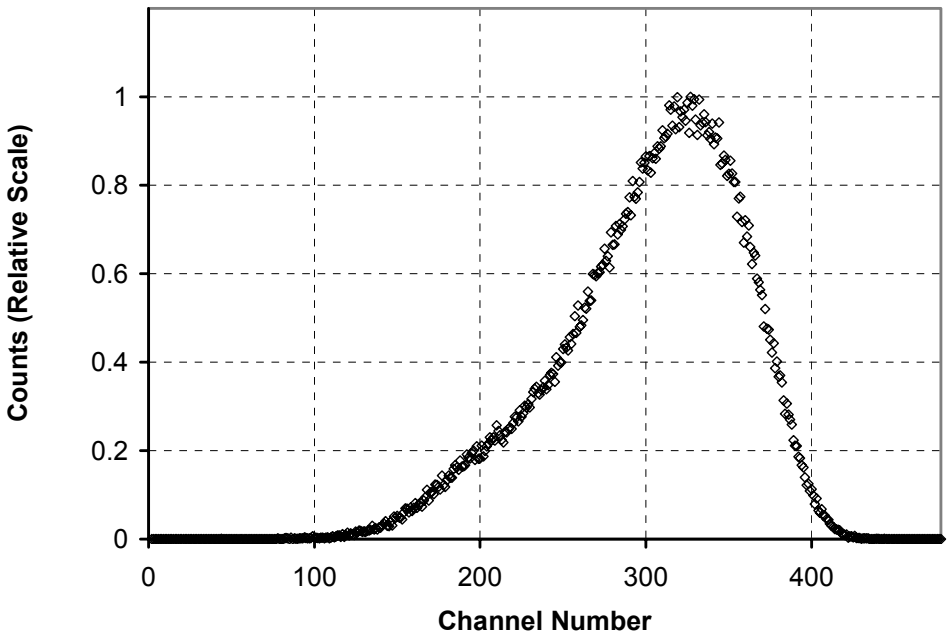
(a)

Figure 7.1. Simulated pulse height spectra for different dopant particle sizes. Simulated pulse height spectra for ZnS:Ag particles in lithiated glass at a 0.4 volume concentration for particle radii of (a) 50 nm, (b) 500 nm, (c) 1 micron, (d) 2 microns, (e) 5 microns, and (f) 10 microns. Each channel is equivalent to 10 keV of energy deposited in the particles. As particle size rises, the Gaussian shape is lost and effects resulting from the inhomogeneity of the composite material are increasingly apparent.

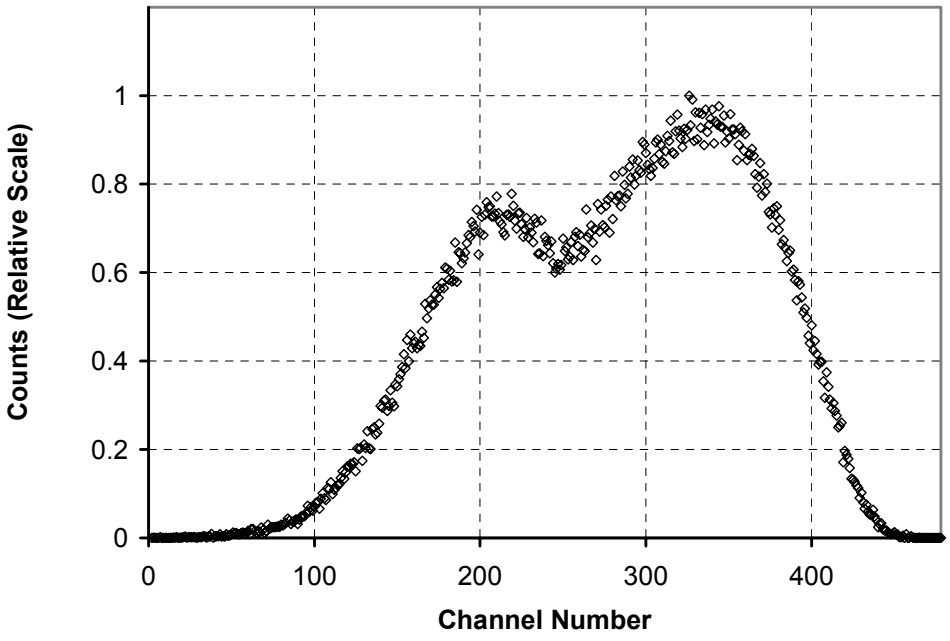


(b)

Figure 7.1 Continued.

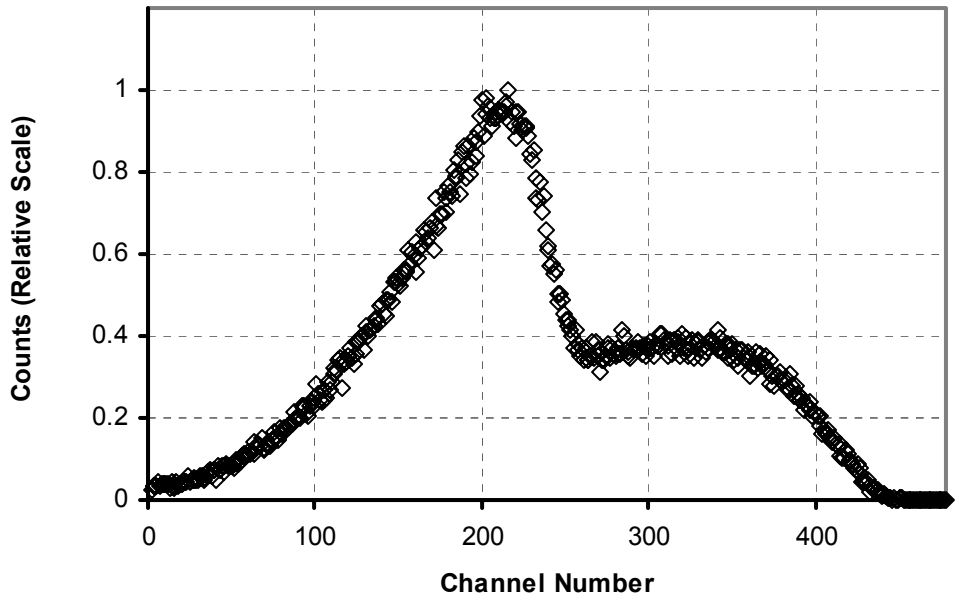


(c)

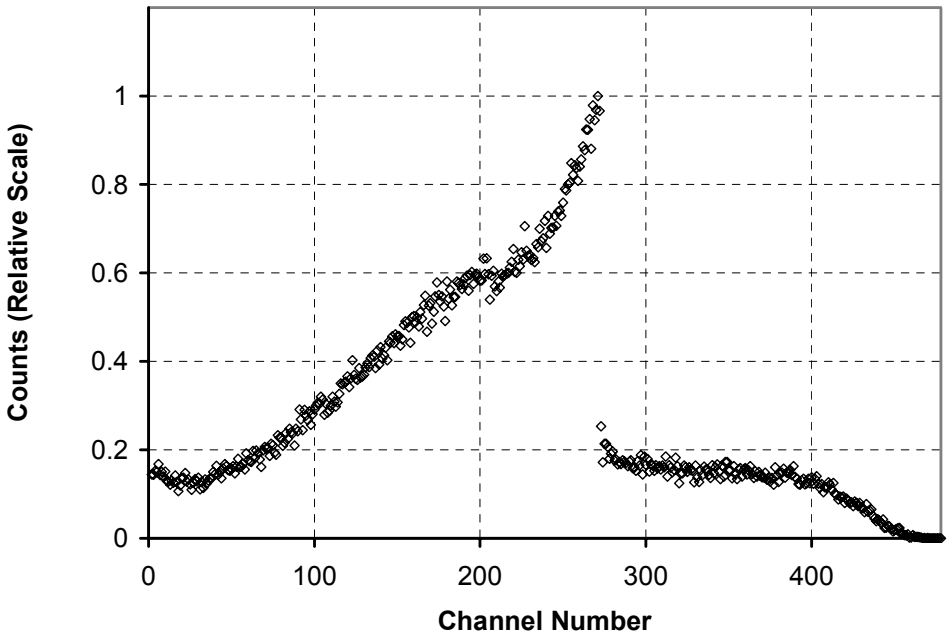


(d)

Figure 7.1 Continued.



(e)



(f)

Figure 7.1 Continued.

dopant particles and the peak from one product only is visible. Further increases in particle size (Figure 7.1(e) and (f)) make it unlikely that both products will enter a particle and the higher energy peak diminishes. Additionally, the large particle size makes it likely that a product entering a particle deposits all of its energy before exits the particle. This explains the rapid rise in counts in the lower peak up to an abrupt drop at a channel corresponding to energy imparted to the trition (H-3) product from the reaction in Equation (3.2). (In a real system noise and statistical variations in the number of scintillation photons counted from equivalent events smooth out rough areas in the pulse height spectra, especially around abrupt drops.)

Pulse Amplitude and Probability as a Function of Particle Size

Figure 7.2 shows the average and standard deviation of the energy deposited in the fluorescent dopant particles as a function of particle radius (events not depositing any energy in the particles are excluded), given a volumetric fluorescent dopant particle concentration of 0.4. Average energy deposited in the particles is increased and variability in deposited energy decreased when particles is decreased. As particle radius is increased, energy deposited in the particles first becomes more variable (10 nm – 1 micron), then it drops significantly with some additional increase in variability (1 micron – 50 microns), before leveling off (above 50 microns).

An initial increase in pulse amplitude variability with little decrease in average pulse amplitude is observed in Figure 7.2, which is understood from Figure 6.1. Larger particle sizes occur concomitantly with larger average particle separation distances (assuming a constant volumetric doping fraction), and this increases the likelihood a reaction product will either fail to enter a particle or remain in a particle for most of or all its travel path. In the regime up to 1 micron particle radius, ^{6}Li reaction product path lengths are large compared to the particle sizes and the reduction in average pulse amplitude illustrated in Figure 6.1(b), due to self-shielding of the products in the matrix, is still small. Above 1 micron particle radius, average particle separation

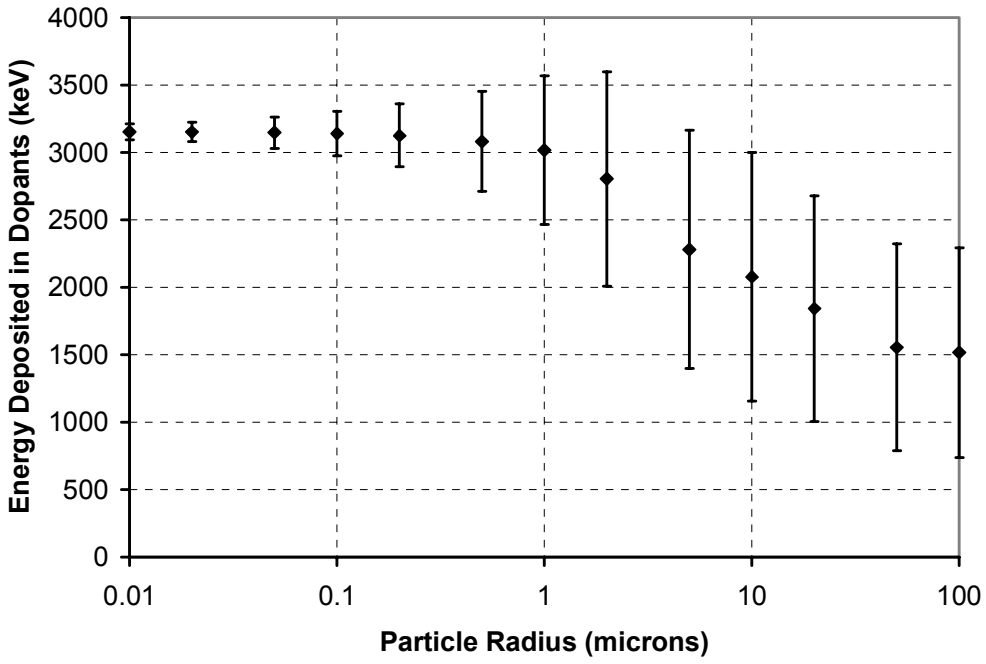


Figure 7.2. Pulse amplitude as a function of dopant particle size. Average and standard deviation of the energy deposited in the ZnS:Ag particles (proportional to scintillation intensity) for the reference case (0.4 volume dopant concentration) as a function of particle radius for events producing scintillation emission.

distances are sufficient for self-shielding to become significant. Eventually a point is reached where "dead" areas emerge between particles in which products originating in those areas are entirely self-shielded by the matrix material and unable to reach a particle and deposit energy. At this point particles are ordinarily so large that any product entering one is very unlikely to exit it. Further increases in particle size increase the dead areas between particles, which reduces the probability that a neutron capture event produces light emission; however, the characteristics of pulses which do occur remain largely unchanged.

Figure 7.3 illustrates the probability that a neutron capture results in energy deposition in the fluorescent dopant particles (and thus light emission) as a function of particle size, again given a volumetric concentration of 0.4 for the fluorescent dopant particles. The probability is very nearly one up to a particle radius of 5 microns. For particles with radius greater than 5 microns, the probability falls quickly with particle radius and drops to 0.21 at 100 microns radius. It asymptotically approaches zero for very large particle sizes. This drop is attributable to the creation and spread of previously described dead areas where products are entirely self-shielded by the matrix.

Effect of Dopant Concentration on Pulse Amplitude and Probability

Figures 7.4 and 7.5 show the average energy deposited in fluorescent dopant particles (events not depositing energy in the particles are excluded), and illustrate the probability of light emission (given a neutron capture event) for different ZnS:Ag particle doping levels in the composite scintillator as a function of particle radius from 100 nm to 100 microns. For very large particle sizes, the average energy deposited in the particles is about the same for all volume concentrations of particles in the composite scintillator. This stems from the dead area effect described above. Changing the concentration merely changes the dead area size. Even at higher volume concentrations (e.g. 0.4), average particle separation distances for very large particle sizes are sufficient that it is unusual for both products from a neutron capture event to enter a

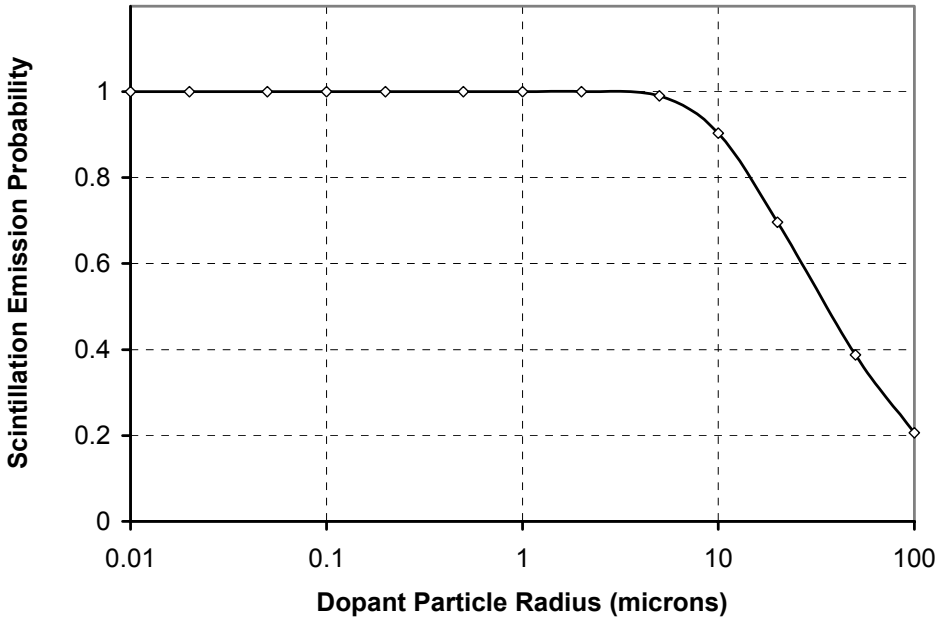


Figure 7.3. Scintillation probability as a function of dopant particle size. Probability of scintillation for the reference ZnS:Ag/lithiated glass case (0.4 volume dopant concentration) as a function of ZnS:Ag particle radius.

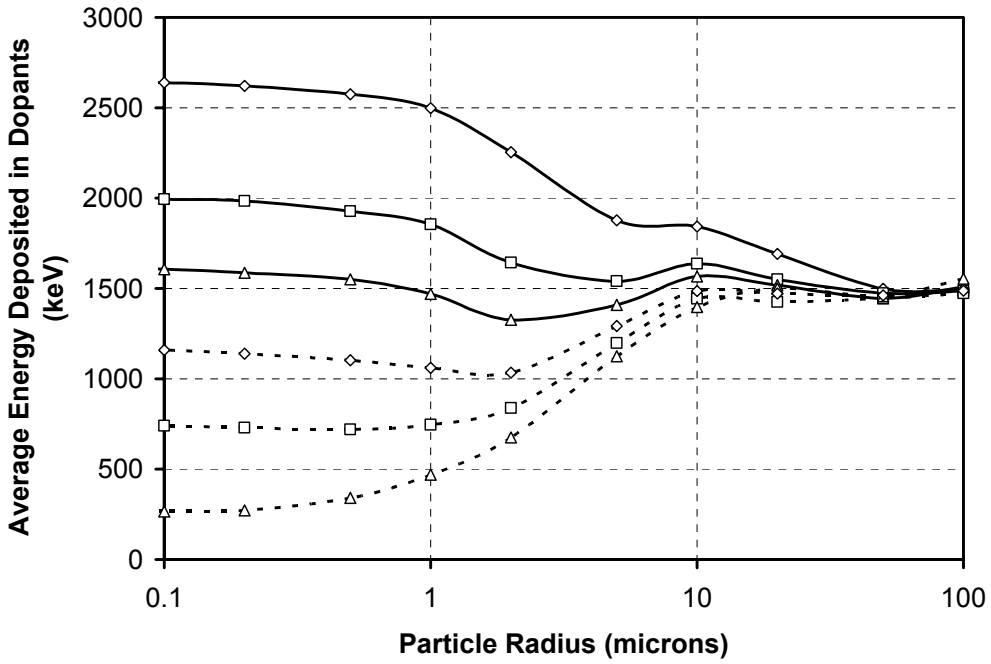


Figure 7.4. Pulse amplitudes for the inorganic case. Average energy deposited in the ZnS:Ag particles (proportional to scintillation intensity) as a function of dopant particle size with volume concentrations as follows: diamond, solid line: 0.3; square, solid line: 0.2; triangle, solid line: 0.15; diamond, dashed line: 0.1; square, dashed line: 0.06; triangle, dashed line: 0.02.

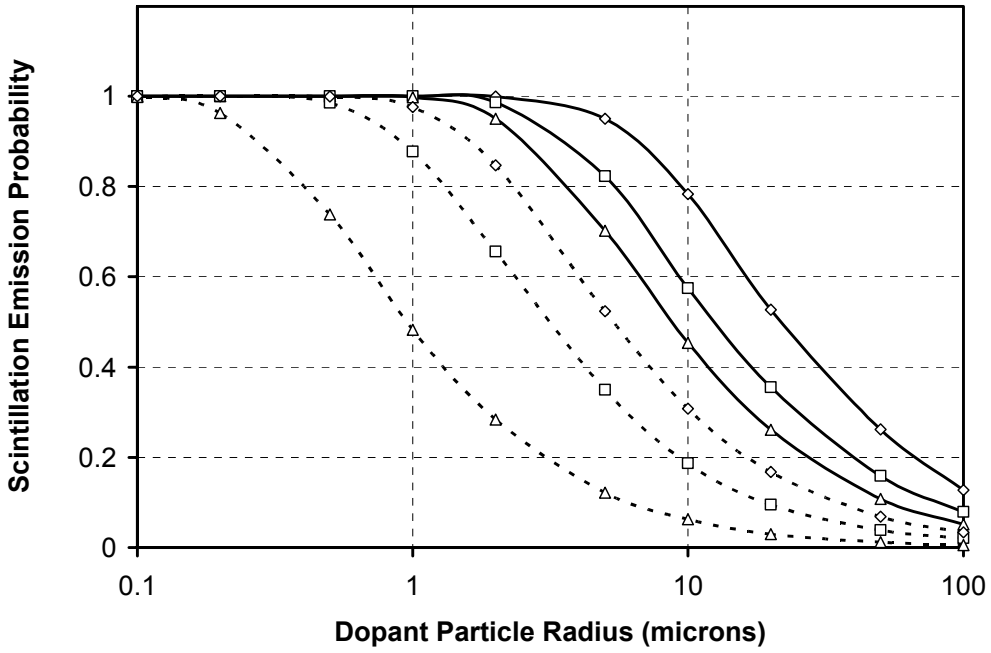


Figure 7.5. Scintillation probability for the inorganic case. Scintillation probability for the ZnS:Ag/lithiated glass case as a function of dopant particle size with volume doping concentrations as follows: diamond, solid line: 0.3; square, solid line: 0.2; triangle, solid line: 0.15; diamond, dashed line: 0.1; square, dashed line: 0.06; triangle, dashed line: 0.02.

particle.

Figure 7.5 shows the probability of scintillation light emission given a neutron capture event as a function of particle size for selected volume doping concentrations between 0.02 and 0.3. (The probability of light emission given a neutron capture event is the same as the probability that one or both reaction products will enter a fluorescent dopant particle.) As expected, the higher the particle volume concentration, the larger the particles must be before scintillation probability begins to drop.

Dependence of Parameters

As discussed in Chapter 6, the effect on scintillator performance characteristics by a change in one design parameter (dopant particle size, dopant volumetric concentration, dopant and matrix densities) is not independent of the values of the other parameters. Different design parameters interact in ways that are always obvious at first glance but at length may be understood and qualitatively predicted.

An excellent example of the effect of the values of parameters being held constant while another parameter is varied is the effect of dopant particle size on pulse amplitude while volumetric concentration is varied. In this case there are three regimes where different effects are seen. At small fluorescent dopant particle sizes, pulse amplitude varies in proportion to the volumetic dopant particle concentration. At large dopant particle sizes, pulse amplitudes are unaffected by dopant concentration; instead the scintillation probability varies with volumetric dopant concentration. At intermediate dopant particle sizes, pulse height spectra (including neutron energy peak shape and amplitude) and scintillation probability are both affected by volumetric concentration. The explanations for these effects are as follows.

An important observation for small fluorescent dopant particle sizes (under 0.5 microns radius) is that the fraction of the neutron capture reaction energy deposited in the particles is very nearly the weight fraction of the particles in the composite scintillator. When the particles are quite small

compared to the reaction product track lengths, individual reaction products typically pass in and out of many particles before stopping. This gives a low variability from one reaction product to the next in the fraction of reaction product path length spent traversing particles, and it gives an average fraction of reaction product length spent traversing particles that is nearly the volume doping fraction of the particles in the composite scintillator. Since reaction product energy loss per unit travel distance is proportional to the density of the material being traversed, it follows that the fraction of the initial reaction energy deposited in the particles is (on average) equal to the weight fraction of particles in the composite scintillator.

Low particle concentrations act to counter the variance reduction effect of small particles by increasing average particle separation, and thus decrease the probability per unit path length that a product will enter a particle. Thus, at low particle volume doping concentrations the total energy deposited in the particles is more sensitive to particle size than at high particle concentrations. This is seen in Figure 7.4 where a shift in particle radius from 100 nm to 1 microns changes the average energy deposited in the particles by a factor of two at a 0.02 volume concentration, whereas the shift is less than ten percent at a 0.4 volume concentration. For very small dopant particle sizes, this consequence is of minor import. However, as dopant particles become larger, the effect of low particle concentrations becomes more pronounced. This effect may also be seen in Figure 7.5 in which low dopant volumetric concentrations (e.g. 0.02) result in the drop in scintillation probability with increasing dopant particle size setting in at a much smaller dopant particle size than for high dopant concentrations (e.g. 0.4). This region of dopant particle size over which the variance reduction effect of smaller particle sizes breaks down and scintillation probability upon absorption of a neutron begins to drop forms a second, transitional regime.

At large dopant particle sizes (in essence, where dopant particle sizes are a significant fraction or larger than the reaction product track lengths),

average pulse amplitudes approach a common value regardless of dopant particle volumetric concentration. This is due to the development of "dead" areas between dopant particles in which neutron capture reactions have little or no likelihood of producing scintillation emissions because their reaction products are unlikely to reach and enter a fluorescent dopant particle. Thus changes in dopant particle volumetric concentration merely serve to alter the scintillation emission probability by changing the size of the dead areas and do not affect average pulse amplitudes.

In the case just described, we see that the effect of varying one parameter (fluorescent dopant particle volumetric concentration) is very highly dependent on the value of another parameter (dopant particle size). This type of dependence between parameters is most pronounced when the dopant particle sizes are a significant fraction of the total reaction product path lengths in the scintillator. When assessing the effects of varying one parameter, it is important to consider possible dependence on the value of other parameters.

(d) Organic Fluorescent Particles in an Organic Matrix

The largest difference between organic fluorescent particles doped into an organic matrix and ZnS:Ag doped into a lithiated matrix is that in the organic case the density of the materials is much lower than in the ZnS:Ag-lithiated glass case. This density difference implies that particle sizes in the ZnS:Ag-lithiated glass matrix are equivalent to larger particle sizes in the organic matrix when the same pulse amplitude distribution and scintillation probability are the basis for comparison. In the organic case, the fluorescent dopant particle and matrix densities are equal. In the inorganic case, the particles are denser than the matrix. Thus higher volume doping concentrations are necessitated in the organic case than the inorganic case to achieve a given average light emission intensity.

Figure 7.6 is the equivalent to Figure 7.4 for organics and it shows the average energy deposited in the particles for selected volume doping fractions

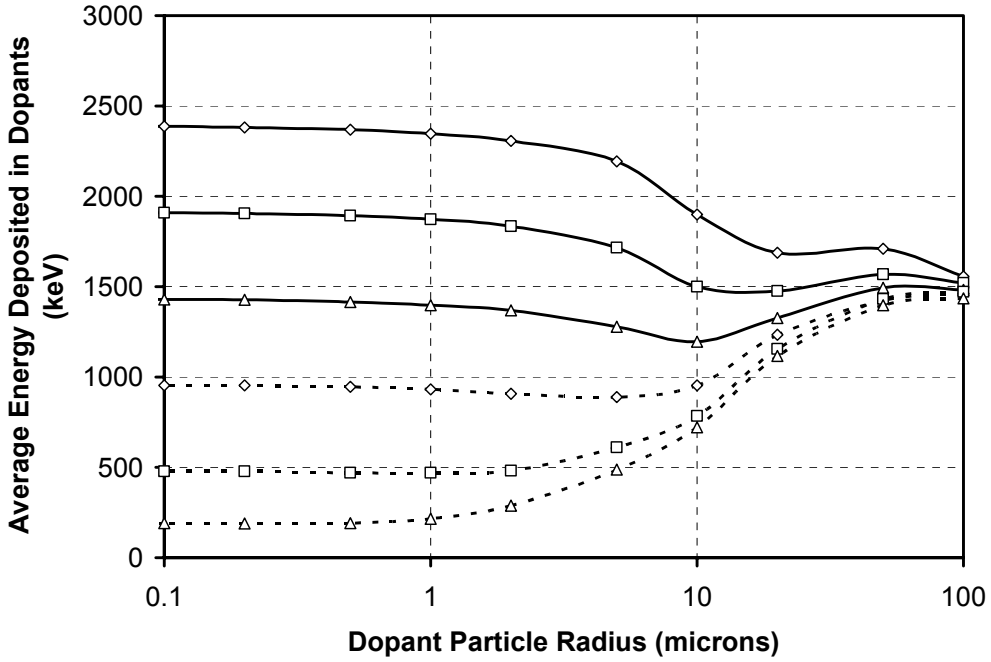


Figure 7.6. Pulse amplitudes for the organic case. Average energy deposited in the fluorescent particles (proportional to scintillation intensity) in the organic case as a function of particle radius with volume concentrations as follows: diamond, solid line: 0.5; square, solid line: 0.4; triangle, solid line: 0.3; diamond, dashed line: 0.2; square, dashed line: 0.1; triangle, dashed line: 0.04.

from 0.04 to 0.5 over a particle radius range of 100 nm to 100 microns. Figure 7.7 is the equivalent to Figure 7.5 for organics and it shows the scintillation probability for the same doping concentrations and dopant sizes. The same effects are at work in the organic case as in the inorganic case; thus, the same general ideas apply.

(e) Code Validation Using Expected Results

As discussed in Chapter 6, two methods of code validation are feasible for this code: testing of calculational methods within the code to ensure the code is functioning as designed, and comparing calculated results with correct solutions for simple cases in which the answer is determined from other methods. A description of actions taken relating to the former method is found in Chapter 6.

One validation method is to utilize very small dopant particle sizes relative to the track lengths of the ^{6}Li reaction products and to compare the average fraction of the reaction energy (Q value) deposited in the fluorescent dopant particles for various dopant sizes with the weight fraction of the scintillator consisting of dopant particles. For very small dopant particles, the scintillator is essentially a homogenous mixture of matrix and dopant materials so far as the reaction products are concerned. Since the linear energy transfer by a ^{6}Li reaction product is for all intents and purposes proportional to the density of the material through which it is traveling, the average fraction of the reaction energy deposited in the dopants should be equal to the weight fraction of the composite scintillator consisting of fluorescent dopant particles. To this end, simulations are run using particles of 5 nm radius for ZnS:Ag/lithiated glass and organic dopants/lithiated organic matrix cases with volume doping concentrations ranging from 0.02 to 0.4 and 0.02 to 0.5, respectively. In all cases, the values derived from simulation and hand calculation differed by less than one percent (the very small differences are attributed to statistical

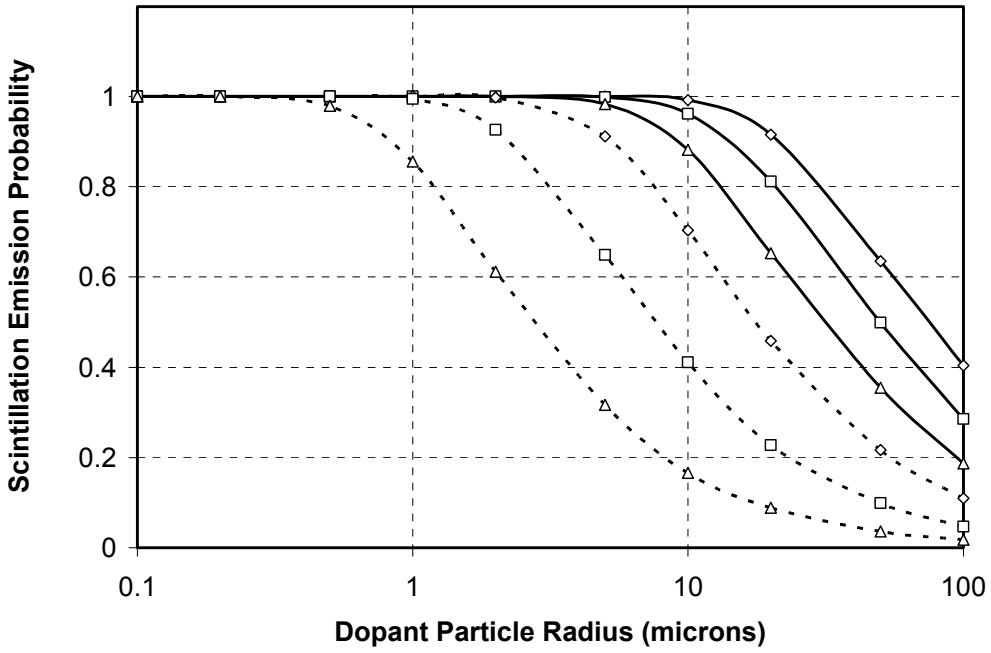


Figure 7.7. Scintillation probability for the organic case. Scintillation probability for the organic case as a function of particle radius with volume concentrations are as follows: diamond, solid line: 0.5; square, solid line: 0.4; triangle, solid line: 0.3; diamond, dashed line: 0.2; square, dashed line: 0.1; triangle, dashed line: 0.04.

uncertainties that necessarily occur when a finite number of events are simulated).

The behavior of a number of other scintillator characteristics as a function of design parameters (e.g. dopant particle size, dopant concentration) is estimated in a qualitative fashion. For example, for fairly high volume fraction concentrations of dopants (e.g. 0.4), the probability of scintillation emission upon neutron absorption will be one (or very nearly so) for dopant particle sizes that are small in comparison to the ^{6}Li reaction product track lengths. When the dopant particle sizes become a significant fraction of the track lengths, the probability of scintillation emission begins to drop quickly and asymptotically approach zero at particle sizes significantly larger than the track lengths. Figures 7.5 and 7.7 (discussed later in this Chapter) demonstrate this behavior in the simulation results. Comparisons of qualitative predictions of scintillator light emission characteristics as a function of design parameters with actual calculated results show consistent agreement, arguing that the composite neutron scintillator simulation is in fact producing reliable results.

There is one aspect to the simulation that should be mentioned at this point. It pertains to the methods used in the simulation (as opposed to accuracy in the implementation of those methods). The simulation uses cubic cells as geometric structures to contain individual fluorescent dopant particles as discussed in Chapter 6, Section (c). The use of these geometrical simulation methods has the potential to reduce the randomness in the particle distribution by introducing some amount of artificial regularity in their spacing. To this it should be added that for small dopant particle sizes, the reaction products enter and exit particles many times, and changes in spacing will have a minimal effect on product energy distribution between the matrix and the dopant particles. For large dopant particle sizes, there is sufficient freedom in the location of the particles that both reaction products may still easily enter particles and a single product may pass through two particles sequentially as a result of them being essentially contiguous in location. It is theorized that

reductions in the randomness of the particle distribution has the greatest effect on energy distribution for medium-sized particles. It is believed that any such reductions in energy distribution variability are minor.

(f) Recommendations for Future Work

A number of avenues for future work are of interest. ZnS:Ag-based composite neutron scintillators using at least three different neutron target materials (^{6}Li, ^{10}B, H) are regularly used in different neutron detection applications. Calculation and publication of data for the neutron target materials not covered in this work would be useful for construction of composite neutron scintillator-based systems.

This author is considering the geometrical aspects of composite neutron scintillators and expects that it can be demonstrated by mathematical proof and/or numerical simulation that cases with the same weight concentration of fluorescent dopant particles and with the same dopant particle mass (assuming spherical, uniformly sized dopant particles) are essentially equivalent. This mathematical equivalence will likely hold for cases in which the densities of the matrix and dopant materials are reasonably similar (say, within an order of magnitude of each other) and break down when they are very dissimilar. If this is correct, values such as the probability of light emission (given neutron absorption) and the average energy deposited in the dopant particles can be calculated as a function of those two variables and expressed in single plots from which values for any particular case can then be obtained. Neutron energy peak shapes vary substantially, and in many cases conventional measurement parameters (e.g. average amplitude, standard deviation of amplitude) associated with Gaussian-shaped curves give only a very vague idea of the shape and nature of the peak. In conjunction with reducing scintillator performance parameters to a dependence on only two variables, development of a method for systematizing peak characteristics (e.g. amplitude

and distribution) and then characterizing them as a function of these same two scintillator description variables would be very useful.

Finally, it would be both interesting and useful to create a simulation that predicts optical transmission properties for composite neutron scintillators based on the optical indices of the dopant and matrix materials and on their light transmission properties. The reason for this is that optical transmission is a major issue for composite neutron scintillators and the dependence of light attenuation lengths on such factors as dopant particle size and concentration in ZnS:Ag-based scintillators is not very well characterized. Being able to predict light attenuation characteristics would assist in the optimization of such scintillators in a number of ways, including clarifying less obvious issues such as where to set an LLD for neutron-gamma discrimination. (ZnS:Ag-based scintillators typically show a continuum of counts instead of a well-defined neutron energy peak, making LLD selection difficult.)

CHAPTER 8
CONCLUSIONS AND RECOMMENDATIONS FOR FUTURE WORK

This dissertation is concerned with the simulation of neutron detectors using Monte Carlo techniques. The value of creating such simulations lies in their ability to give insight into how such detectors work and how they may be optimized for various applications. Three types of simulations are considered in this dissertation. The first is simulation of the internal workings of a Micromegas neutron beam monitor using a custom code in order to understand how the different design parameters affect the performance of the device, with a view to determining which set of design parameters gives the best performance characteristics for neutron beam monitoring. The second is simulation of the neutronics of Micromegas neutron beam monitors with the aim of minimizing neutron scattering in the device. The third is simulation of composite neutron scintillators using a custom code with a view to determining how design parameters such as fluorescent dopant particle size and dopant particle concentration affect the performance characteristics of the scintillator. Specific conclusions regarding each of these three topical areas are as follows.

Micromegas Simulation

A Micromegas neutron detector Monte Carlo simulation has been developed and simulated neutron responses are compared with published experimental data. Theoretical and experimental results show good agreement. Modeling of various detector design parameter changes enables quantification of the effects of varying neutron target material, neutron converter foil thickness, drift gap width, and gas pressure on spatial resolution and count probability per neutron absorbed. It is found that for neutron beam monitors operating at one atmosphere of 90/10 Ar/isobutane gas or P-10 gas, ^{10}B neutron converter material and a 1 mm drift gap produce the best results among the designs tested. Optimal design parameters will change with

application requirements and the calculated results for variations in different design parameters will provide guidance for future investigations.

Suggestions for future work on the Micromegas are as follows. Investigations should be made of gamma sensitivity and methods for gamma rejection, including readout strip pulse topology analysis. Pulse topology analysis should also be used to improve neutron position resolution. Detector pressurization should be investigated as a means of improving neutron position resolution. (Note that the increase in neutron scattering probability with pressurization due to thicker detector walls may limit pressurized detectors to particular applications.)

Micromegas Neutron Beam Monitor Neutronics

Calculations of the neutronics characteristics of Micromegas neutron beam monitors reported in the literature to date indicate that for thermal neutrons their scattering probabilities are in the range of 10^{-1}-10^{-2}. Ideally, neutron beam monitors should have scattering probabilities at or below their neutron detection probabilities. In a realistic best-case design scenario, neutron scattering probabilities at thermal of 1.1×10^{-3} may be achieved. Since intended neutron detection probabilities of the Micromegas neutron beam monitors to be used on the SNS beamlines vary from 10^{-5}-10^{-3} depending on the intensity of the neutron beamline in question, a higher scattering probability than desired will have to be accepted on some beamlines.

To minimize the neutron scattering probability, the following design parameters are recommended. The cathode/first detector wall should be as thin as possible (25 microns is assumed) and consist of Al. The second detector wall should consist of either Si or quartz, both of those materials being non-conducting and readily etchable. (Again, a 25 micron thickness is assumed.) The neutron target material in converter foil should be ^{10}B. The use of ^{10}B will give a minimum gas chamber thickness of 1 mm for the drift gap and 75 microns for the amplification gap. Both P-10 and 90/10 Ar/isobutane gas are ideal for use in the detector. The mesh separating the two regions of the gas

chamber should be made of relatively weakly scattering conducting material such as Cr and be kept as thin as is consistent with structural integrity. Cr may be used for thin (tens of nm average areal coverage) anode readout strips. If these design parameters are followed, scattering probabilities at thermal of $1.1x10^{-3}$ and $1.5x10^{-3}$ may be anticipated using Si and quartz second detector walls, respectively.

Composite Neutron Scintillator Simulation

Simulation of a composite neutron scintillator consisting of fluorescent dopant particles embedded in a ^{6}Li-bearing matrix material is accomplished using a custom simulation based on Monte Carlo techniques. This simulation allows examination of the effects of particle size, particle concentration, and materials density on the important scintillator qualities of scintillation intensity and probability given absorption of a neutron. The simulation has been used to investigate cases representative of both inorganic and organic scintillators, with a particular focus on ZnS:Ag fluorescent dopant particles in a lithiated glass matrix.

For the ZnS:Ag/lithiated glass case with ZnS:Ag composing forty percent of the composite scintillator by volume, it was found that ZnS:Ag particle radii of 1 micron or less give a reasonable Gaussian neutron energy peak shape, with the peak first splitting into two peaks and then becoming ramp shaped for larger particle sizes. For this same case, scintillation probability is very nearly one for particle radii up to 5 microns, beyond which it drops quickly. At lower fluorescent dopant particle concentrations light emission is more variable and lower in intensity and the scintillation probability is more sensitive to particle size. In the organic case the same dependence on particle size and concentration is seen with particular effects appearing at larger particle sizes due to the lower materials density and consequently longer ^{6}Li reaction product track lengths in the composite scintillator.

Some opportunities for future work are as follows. Other neutron target materials besides ^{6}Li (most notably ^{10}B and H) may be studied using the simulation to determine what characteristics they will have if used in composite neutron scintillators. A study may be done to determine what types of cases are essentially equivalent to each other (though they may have different sets of design parameters, the sets produce the same results). Based on this, results may be reported in a format from which predicted results can be found for arbitrary sets of design parameters. The composite neutron scintillator simulation may be modified to predict optical transmission properties for composite scintillators in which the refractive indices of the matrix and dopant materials are ill-matched (e.g. ZnS:Ag in lithiated plastic).

REFERENCES

[1] Y. Giomataris, Ph. Rebourgeard, J. P. Robert, G. Charpak, Nuclear Instruments and Methods A 376, 29 (1996).

[2] S. Andriamonje, D. Cano-Ott, A. Delbart, J. Derre, S. Diez, I. Giomataris, E. M. Gonzalez-Romero, F. Jeanneau, D. Karamanis, A. Lepretre, I. Papadopoulos, P. Pavlopoulos, D. Villamarin, Nuclear Instruments and Methods A **481**, 120 (2002).

[3] J. F. Briemeister (Ed.), 2001. MCNP - A General Monte Carlo N-Particle Transport Code, Version 4C2. LA-12625-M, Los Alamos National Laboratory, Los Alamos, NM.

[4] J. F. Ziegler, J. P., Biersack, U. Littmark. The Stopping and Range of Ions in Matter. Pergamon Press, New York, 1985. Calculations were performed with SRIM-2000 version 40 (see http://www.SRIM.org).

[5] G. Barouch, A. Bay, S. Bouchigny, G. Charpak, J. Derre, F. Didierjean, J.-C. Faivre, Y. Giomataris, C. Kochowski, F. Kunne, J.-M. Le Goff, F. Lehar, Y. Lemoigne, S. Loucatos, J.-C. Lugol, A. Magnon, B. Mayer, J.-P. Perroud, S. Platchkov, G. Puill, Ph. Rebourgeard, Y. Terrien, D. Thers, H. Zaccone, Nuclear Instruments and Methods A **423**, 32 (1999).

[6] J. Derre, Y. Giomataris, H. Zaccone, A. Bay, J.-P. Perroud, F. Ronga, Nuclear Instruments and Methods A **459**, 523 (2001).

[7] G. F. Knoll, Radiation Detection and Measurement, 3rd Edition. John Wiley & Sons, New York, 1999.

[8] J. P. Biersack, L. Haggmark, Nuclear Instruments and Methods **174**, 257 (1980).

[9] J. F. Ziegler, SRIM Instruction Manual, available at www.srim.org.

[10] J. F. Ziegler, private communication.

[11] G. Charpak, J. Derre, A. Giganon, Y. Giomataris, D. Jourde, C. Kochowski, S. Loucatos, G. Puill, Ph. Rebourgeard, J. P. Robert, Nuclear Instruments and Methods A **412**, 47 (1998).

[12] R. G. Cooper, private communication.

[13] C. L. Britton, Jr., private communication.

[14] I. K. Bronic, B. Grosswendt, Nuclear Instruments and Methods B **117**, 5 (1996).

[15] I. H. Papadopoulos, private communication.

[16] G. Charpak, J. Derre, Y. Giomataris, Ph. Rebourgeard, Nuclear Instruments and Methods A **478**, 26 (2002).

[17] A. Delbart, R. De Oliveira, J. Derré, Y. Giomataris, F. Jeanneau, Y. Papadopoulos Ph. Rebourgeard, Nuclear Instruments and Methods A **461**, 84 (2001).

[18] J. Derre, Y. Giomataris, Ph. Rebourgeard, H. Zaccone, J. P. Perroud, G. Charpak, Nuclear Instruments and Methods A **449**, 314 (2000).

[19] D. Thers, Ph. Abbon, J. Ball, Y. Bedfer, C. Bernet, C. Carasco, E. Delagnes, D. Durand, J.-C. Faivre, H. Fonvieille, A. Giganon, F. Kunne, J.-M.

Le Goff, F. Lehar, A. Magnon, D. Neyret, E. Pasquetto, H. Pereira, S. Platchkov, E. Poisson, Ph. Rebourgeard, Nuclear Instruments and Methods A **469**, 133 (2001).

[20] F. Jeanneau, Y. Benhammou, R. Blaes, J. M. Brom, F. Charles, F. Didierjean, F. Drouhin, J. C. Fontaine, J. M. Helleboid, D. Huss, A. Pallares, I. Ripp-Baudot, P. VanLaer, A. Zghiche, Nuclear Instruments and Methods A **450**, 313 (2000).

[21] C. L. Britton, Jr., W. L. Bryan, A. L. Wintenberg, R. J. Warmack, T. E. McKnight, S. S. Frank, R. G. Cooper, N. J. Dudney, G. M. Veith, A. C. Stephan, "A Detector for Neutron Imaging," Presented at the 2003 Nuclear Science Symposium, Portland, OR, October 19-25, 2003.

[22] J. B. Birks, The Theory and Practice of Scintillation Counting, Macmillan, New York (1964).

[23] R. Stedman, Review of Scientific Instruments **31**, 1156 (1960).

[24] R. H. Bossi, A. H. Robinson, Transactions of the American Nuclear Society **22**, 153 (1975).

[25] F.Mantler-Niederstatter, F. Bensch, F. Grass, Nuclear Instruments and Methods **142**, 463 (1977).

APPENDICES

APPENDIX A:
SAMPLE TRIM INPUT AND OUTPUT FILES

```
==> SRIM-2000.41 <== TRIM.IN is for TRIM(DOS), TRIM(Win) uses TRIM_WIN.IN ===
Ion: Z1 ,  M1,  Energy (keV), Angle,Number,Bragg Corr,AutoSave Number.
   2   4.002   20000   0   99999   0   10000
Cascades(1=No;2=Full;3=Sputt;4-5=Ions;6-7=Neutrons), Random Number Seed,
Reminders
    4   996749   0
Diskfiles (0=no,1=yes): Ranges, Backscatt, Transmit, Sputtered, Recoils (2=Alt-
C)
    1   1   1   0   0
Target material : Number of Elements & Layers
"Target into layer"    1    1
Target Energies (eV): Binding, Surface, Individual Displacement
            3     2      20
PlotType (0-5); Plot Depths: Xmin, Xmax(Ang.) [=0 0 for Viewing Full Target]
       5                    0            0
Target Elements:    Z Mass(amu)
Atom 1 = Li =        3   6.0
Layer   Layer Name /              Width Density    Li(3)
Numb.   Description               (Ang) (g/cm3)    Stoich
 1      "Lithium"           10000  0.462        1
0  Target layer phases (0=Solid, 1=Gas)
0
Target Compound Corrections (Bragg)
0
Target atom displacement energies (eV)
20
Target atom lattice binding energies (eV)
3
Target atom surface binding energies (eV)
2
Stopping Power Version (2=1996, 0=2000, 1=2000 interpolated)
0
```

Figure A.1. Sample TRIM trim_win.in file.

```
Ûßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßß
ÛßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßÛ
```

```
Ûßßßßßßßßß TRIM with various Incident Ion Energies/Angles and Depths ßßßßßßßÛ
Û Top  10 lines are user comments,  with  line  #8 describing experiment.    Û
Û Line #8 will be written into all TRIM output files ( various files:  *.TXT).Û
Û Data Table line consist of: EventName(5 char)+8 numbers separated by spaces.Û
Û The Event Name consists of any 5  characters  to identify that line.       Û
Û Cos(X) = 1 for normal incidence, and Cos(X) = -1  for backwards.           Û
ßßßßßßßßßßßßßßßß Typical Data File is shown below  ßßßßßßßßßßßßßßßßßßßßßßßßßßß
ÉÍ Reaction Product Ions into   ( A thick) (Various Energies and Angles) Í»
Event  Atom  Energy  Depth   Lateral-Position   ----- Atom Direction ----
Name   Numb   (eV)    _X_(A)    _Y_(A)  _Z_(A)   Cos(X)   Cos(Y)   Cos(Z)
1       4.002 1470753 7835 2500000 118664265 -0.6780 0.0626 0.7324
2       4.002 1470853 15417 2500000 453457415 0.7389 -0.4398 0.5106
3       4.002 1470840 18028 2500000 197436422 0.5678 0.5574 -0.6057
4       4.002 1470822 7084 2500000 174762368 0.3115 0.6809 0.6629
5       4.002 1470866 17549 2500000 487622648 0.9375 -0.0916 -0.3356
6       4.002 1470816 17515 2500000 323993593 0.2297 0.8354 0.4993
7       4.002 1470854 10073 2500000 215316981 0.7607 -0.2940 -0.5787
8       4.002 1470846 17204 2500000 429846644 0.6455 -0.3302 -0.6887
9       4.002 1470731 2830 2500000 117182106 -0.9848 -0.0706 0.1585
10      4.002 1470839 19881 2500000 364880413 0.5565 -0.5056 0.6593
11      4.002 1470810 8450 2500000 123073578 0.1382 0.9705 0.1978
12      4.002 1777696 8858 2500000 380516112 0.0137 0.8236 -0.5671
13      4.002 1470742 9878 2500000 333836168 -0.8310 -0.3398 0.4404
14      4.002 1470788 17459 2500000 329581022 -0.1781 0.7625 -0.6220
15      4.002 1470858 18914 2500000 196955711 0.8242 -0.3647 0.4333
16      4.002 1470748 12883 2500000 11478752 -0.7456 0.3640 0.5583
17      4.002 1470863 14001 2500000 281198651 0.8979 0.4095 0.1617
18      4.002 1470870 9359 2500000 389665186 0.9949 -0.0912 0.0428
19      4.002 1470824 8221 2500000 167865872 0.3415 -0.9210 -0.1876
20      4.002 1470829 12327 2500000 85395724 0.4048 0.4514 -0.7952
```

Figure A.2. Sample TRIM trim.dat file.

```
Calc. Date= 10-12-2003, Time= 01:13:45
OOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOO SRIM-2000.40 OOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOO
 ======================================================================
 =                    Ion Energy vs Position File                     =
 ======================================================================
 =   AXIS DEFINITIONS: X=Depth, Y,Z= Lateral plane of target surface.=
 =   (If beam enters target at an angle, this tilt is in Y direction)=
 =   Shown are: Ion Number, Energy (keV), X, Y, Z Position           =
 ==================  CALCULATION   DATA  ==============================
  Ion Data: Name, Mass,   Energy , Energy Interval
            Li     007.01  20000keV    25000eV
 ======================================================================
Ion       Energy     Depth (X)     Y           Z       Energy Lost in
Number    (keV)    (Angstrom)  (Angstrom)  (Angstrom)  Last Collision (eV)
------- ----------- ---------- ----------- ----------- ------------------
0000001 6.2883E+01  0.0000E+00  2.8000E+08  1.2330E+08  0.0000E+00
0000001 4.9515E+01  1.8878E+06  2.8031E+08  1.2338E+08  1.0490E+03
0000001 2.4389E+01  3.1604E+06  2.8118E+08  1.2518E+08  6.7951E+02
0000002 2.3332E+01  0.0000E+00  2.8000E+08  4.9410E+08  5.2535E-01
0000003 6.6573E+01  0.0000E+00  2.8000E+08  7.9540E+07  5.1813E-01
0000003 4.9956E+01  2.7195E+06  2.7870E+08  7.9538E+07  5.2509E+02
0000003 2.3314E+01  6.3782E+06  2.7725E+08  7.9105E+07  2.6972E+03
0000004 6.1954E+02  0.0000E+00  2.8000E+08  4.4340E+08  3.7057E-01
0000004 5.8914E+02  6.9039E+05  2.7994E+08  4.4260E+08  1.3528E+04
0000004 5.5619E+02  1.3591E+06  2.7989E+08  4.4182E+08  1.9907E+04
0000004 5.3958E+02  1.6830E+06  2.7987E+08  4.4144E+08  1.6610E+04
0000004 5.2103E+02  2.0002E+06  2.7985E+08  4.4107E+08  1.8550E+04
0000004 4.9190E+02  2.6158E+06  2.7981E+08  4.4036E+08  1.2501E+04
0000004 4.7348E+02  2.9105E+06  2.7979E+08  4.4001E+08  1.8426E+04
0000004 4.3724E+02  3.7250E+06  2.7965E+08  4.3898E+08  1.5790E+04
0000004 4.2339E+02  3.9855E+06  2.7960E+08  4.3865E+08  1.3850E+04
0000004 3.9698E+02  4.4879E+06  2.7951E+08  4.3801E+08  1.3944E+04
0000004 3.7013E+02  5.1894E+06  2.7939E+08  4.3707E+08  7.3178E+03
0000004 3.4290E+02  5.8593E+06  2.7930E+08  4.3618E+08  1.1381E+04
0000004 3.1678E+02  6.4916E+06  2.7920E+08  4.3533E+08  9.2281E+03
0000004 2.9421E+02  7.0854E+06  2.7910E+08  4.3451E+08  8.8247E+03
0000004 2.6983E+02  7.6361E+06  2.7899E+08  4.3373E+08  8.2941E+03
0000004 2.4694E+02  8.3233E+06  2.7890E+08  4.3274E+08  4.6403E+03
0000004 2.2385E+02  9.2954E+06  2.7877E+08  4.3133E+08  6.3983E+03
0000004 1.9908E+02  1.0321E+07  2.7855E+08  4.2979E+08  4.3453E+03
0000004 1.7267E+02  1.1430E+07  2.7830E+08  4.2820E+08  2.9548E+03
0000004 1.4860E+02  1.2419E+07  2.7797E+08  4.2647E+08  4.1766E+03
0000004 1.2405E+02  1.3099E+07  2.7774E+08  4.2460E+08  1.4967E+03
0000004 9.7474E+01  1.3665E+07  2.7743E+08  4.2201E+08  3.2052E+03
0000004 7.4290E+01  1.4134E+07  2.7730E+08  4.1930E+08  1.6773E+03
0000004 4.9757E+01  1.4445E+07  2.7703E+08  4.1618E+08  1.4278E+03
0000004 2.4891E+01  1.6790E+07  2.7714E+08  4.1343E+08  6.9505E+02
0000005 1.1242E+01  0.0000E+00  2.8000E+08  4.2610E+08  8.7259E-01
0000006 3.9540E+02  0.0000E+00  2.8000E+08  9.9010E+07  4.1971E+00
0000006 3.7050E+02  6.4810E+05  2.8038E+08  9.8787E+07  1.1918E+04
0000006 3.4549E+02  1.5711E+06  2.8094E+08  9.8475E+07  7.1476E+03
0000006 3.1698E+02  2.7419E+06  2.8164E+08  9.8086E+07  8.5448E+03
0000006 2.9712E+02  3.5732E+06  2.8212E+08  9.7747E+07  6.9242E+03
0000006 2.7145E+02  4.6438E+06  2.8272E+08  9.7357E+07  4.6222E+03
0000006 2.4557E+02  5.9039E+06  2.8343E+08  9.6895E+07  6.9990E+03
0000006 2.2490E+02  6.6098E+06  2.8383E+08  9.6620E+07  6.6329E+03
0000006 1.9540E+02  7.6797E+06  2.8445E+08  9.6137E+07  6.3472E+03
0000006 1.7433E+02  8.6333E+06  2.8503E+08  9.5635E+07  4.3869E+03
0000006 1.4778E+02  1.0266E+07  2.8599E+08  9.4886E+07  5.6195E+03
0000006 1.2282E+02  1.1516E+07  2.8684E+08  9.4361E+07  3.7476E+03
0000006 9.8653E+01  1.3619E+07  2.8810E+08  9.3308E+07  2.7189E+03
0000006 7.4380E+01  1.5193E+07  2.8881E+08  9.2642E+07  2.3636E+03
0000006 4.9003E+01  1.8670E+07  2.9029E+08  9.1772E+07  1.4714E+03
0000006 2.4986E+01  2.2061E+07  2.9105E+08  8.9968E+07  1.4301E+01
```

Figure A.3. Sample TRIM exyz.txt file.

```
ÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛ SRIM-2000.40 ÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛ
ßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßß
Ûßßßßßßßßßßßßßßßß  TRANSMIT.TXT : File of Transmitted Ions  ßßßßßßßßßßßßßßßßÛ
Û  This file tabulates the kinetics of ions or atoms leaving the target.    Û
Û  Column #1: S= Sputtered Atom, B= Backscattered Ion, T= Transmitted Ion.  Û
Û  Col.#2: Ion Number, Col.#3: Z of atom leaving, Col.#4: Atom energy (eV).  Û
Û  Col.#5-7: Last location:  X= Depth into target, Y,Z= Transverse axes.    Û
Û  Col.#8-10: Cosines of final trajectory.  NOTE: Use hotkey  C  for details.Û
Û *** This data file is in the same format as TRIM.DAT (see manual for uses).Û
ÉÍ Reaction Product Ions into   ( A thick) (Various Energies and Angles) Í»
 IonÍ»AtomÉÍ  Energy  ÍÍ»  ÉÍ  Depth  ÍÍÍ»ÉÍLateral-PositionÍÍÍ»ÉÍÍ  Atom Direction
ÍÍÍÍ»
 NumbºNumbÌÍ   (eV)   ÍÍ¹  ÌÍ   X(A)  ÍÍÍ¹ÌÍÍY(A)ÍÍÍÍËÍÍZ(A)ÍÍÍ¹Ì  Cos(X)   Cos(Y)
Cos(Z)¹
T    2  2 .1234172E+07 2016608E-02 .2497E+07 .4535E+09 .74317 -.46583 .48032
T    3  2 .1327819E+07 2006057E-02 .2502E+07 .1974E+09 .57133 .55594 -.60374
T    5  2 .1366663E+07 2001192E-02 .2500E+07 .4876E+09 .93821 -.09307 -.33330
T    6  2 .9903606E+06 2012024E-02 .2510E+07 .3240E+09 .20802 .82098 .53171
T    7  2 .9479762E+06 2007366E-02 .2496E+07 .2153E+09 .75546 -.29093 -.58706
T    8  2 .1309036E+07 2007197E-02 .2499E+07 .4298E+09 .64172 -.32931 -.69264
T   10  2 .1462227E+07 2003667E-02 .2500E+07 .3649E+09 .55650 -.50560 .65930
T   15  2 .1419501E+07 2006116E-02 .2499E+07 .1970E+09 .82430 -.36654 .43148
T   17  2 .1217003E+07 2008151E-02 .2503E+07 .2812E+09 .89870 .40949 .15702
T   18  2 .1046115E+07 2000394E-02 .2499E+07 .3897E+09 .99894 -.04076 .02134
T   19  2 .1024332E+05 2018001E-02 .2467E+07 .1679E+09 .43200 -.81068 -.39519
T   20  2 .6852582E+06 2006681E-02 .2509E+07 .8538E+08 .41100 .44696 -.79455
T   23  2 .1032338E+07 2000222E-02 .2508E+07 .4297E+09 .54913 .72403 -.41742
T   26  2 .3174355E+06 2007118E-02 .2482E+07 .3670E+09 .58574 -.65947 -.47119
T   31  2 .1469609E+07 2002485E-02 .2500E+07 .4812E+09 .74600 -.61040 -.26630
T   32  2 .1412709E+07 2010991E-02 .2490E+07 .2843E+09 .18635 -.94659 -.26313
T   37  2 .2027961E+06 2008284E-02 .2524E+07 .2001E+09 .25500 .83034 -.49549
T   39  2 .8216005E+06 2001652E-02 .2505E+07 .1378E+09 .29706 .18766 -.93624
T   40  2 .1243731E+07 2003280E-02 .2499E+07 .4355E+08 .83461 -.19742 -.51425
T   42  2 .9616522E+06 2004768E-02 .2500E+07 .2828E+09 .73699 .02877 -.67529
T   44  2 .8599330E+06 2009759E-02 .2504E+07 .4000E+08 .62819 .26818 -.73038
T   47  2 .7900586E+06 2001388E-02 .2506E+07 .3589E+09 .79386 .31987 .51717
T   48  2 .1660893E+06 2007152E-02 .2497E+07 .4626E+09 .60001 -.01905 -.79977
T   49  2 .1344635E+07 2001490E-02 .2501E+07 .3526E+09 .72345 .38686 .57181
T   51  2 .2398482E+06 2012677E-02 .2526E+07 .3594E+09 .25509 .92046 -.29610
T   52  2 .9177303E+06 2016692E-02 .2509E+07 .3051E+09 .67961 .67645 .28379
T   56  2 .1260341E+07 2004704E-02 .2501E+07 .3169E+09 .95584 .24829 .15721
T   58  2 .2293161E+06 2013301E-02 .2476E+07 .3665E+09 .35621 -.84103 -.40717
T   61  2 .4796082E+06 2003446E-02 .2483E+07 .3300E+09 .62472 -.75471 -.20033
T   63  2 .5567826E+06 2004346E-02 .2498E+07 .2806E+09 .80293 -.09547 -.58838
T   66  2 .1051951E+07 2001160E-02 .2505E+07 .3933E+09 .62904 .45457 .63062
T   67  2 .1287209E+07 2014000E-02 .2504E+07 .4416E+09 .37231 .86846 -.32737
T   68  2 .7088414E+06 2008333E-02 .2510E+07 .7364E+08 .75231 .52746 -.39474
T   70  2 .9395633E+06 2008043E-02 .2496E+07 .3351E+09 .94946 -.31059 .04530
```

Figure A.4. Sample TRIM transmit.txt file.

```
ÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛ SRIM-2000.40 ÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛ
ßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßßß
Ûßßßßßßßßßßßßßßß BACKSCAT.TXT : File of Backscattered Ions ßßßßßßßßßßßßßßßÛ
Û  This file tabulates the kinetics of ions or atoms leaving the target.    Û
Û  Column #1: S= Sputtered Atom, B= Backscattered Ion, T= Transmitted Ion.  Û
Û  Col.#2: Ion Number, Col.#3: Z of atom leaving, Col.#4: Atom energy (eV).  Û
Û  Col.#5-7: Last location:  X= Depth into target, Y,Z= Transverse axes.    Û
Û  Col.#8-10: Cosines of final trajectory.  NOTE: Use hotkey  C  for details.Û
Û *** This data file is in the same format as TRIM.DAT (see manual for uses).Û
ÉÍ Reaction Product Ions into   ( A thick) (Various Energies and Angles) Í»
 IonÍ»AtomÉÍ  Energy  ÍÍ»  ÉÍ  Depth  ÍÍÍ»ÉÍLateral-PositionÍÍÍ»ÉÍÍ  Atom Direction
ÍÍÍÍ»
 NumbºNumbÌÍ    (eV)    ÍÍ¹  ÌÍ    X(A)  ÍÍÍ¹ÌÍÍY(A)ÍÍÍÍËÍÍZ(A)ÍÍÍ¹Ì  Cos(X)    Cos(Y)
Cos(Z)¹
B    1  2 .9925333E+06 -9269364E-05 .2501E+07 .1187E+09 -.66264 .05997 .74654
B    9  2 .1359722E+07 -1030178E-04 .2500E+07 .1172E+09 -.98470 -.06926 .15993
B   13  2 .9923094E+06 -3734766E-05 .2496E+07 .3338E+09 -.83744 -.34408 .42462
B   16  2 .7489829E+06 -4546755E-05 .2506E+07 .1149E+08 -.74823 .35151 .56266
B   21  2 .9248910E+06 -6020406E-06 .2491E+07 .1967E+09 -.52864 -.76722 -.36320
B   27  2 .1311117E+07 -2596345E-05 .2496E+07 .4453E+09 -.46282 -.86231 -.20545
B   28  2 .9667162E+06 -9957196E-05 .2504E+07 .3397E+09 -.83768 .37141 -.40044
B   29  2 .9768323E+06 -1903836E-05 .2501E+07 .2004E+09 -.30862 .05811 -.94941
B   30  2 .1345709E+07 -1967503E-05 .2501E+07 .4309E+09 -.75422 .35960 .54940
B   35  2 .1094605E+07 -4766374E-05 .2510E+07 .3133E+09 -.77107 .50261 -.39095
B   41  2 .1044095E+07 -2427933E-05 .2496E+07 .1704E+09 -.92288 -.36863 .11137
B   43  2 .1277016E+07 -4150514E-05 .2503E+07 .2316E+09 -.46701 .49808 .73063
B   54  2 .1421022E+07 -1021440E-04 .2500E+07 .9787E+08 -.19573 .35324 -.91483
B   55  2 .2280066E+06 -1204990E-04 .2512E+07 .3988E+09 -.37603 .38019 .84502
B   57  2 .1286331E+07 -2545292E-05 .2503E+07 .3298E+09 -.58965 .64032 -.49225
B   59  2 .7238323E+06 -7129454E-05 .2504E+07 .3717E+09 -.94636 .22801 .22893
B   60  2 .4301133E+06 -7167688E-05 .2483E+07 .3459E+09 -.47744 -.69810 -.53358
B   62  2 .7207179E+06 -7721362E-05 .2499E+07 .3756E+08 -.58627 -.10647 -.80309
B   64  2 .9246665E+06 -3136679E-05 .2502E+07 .3527E+09 -.27797 .16823 .94574
B   65  2 .1518438E+07 -1274310E-04 .2506E+07 .4986E+09 -.47061 .78205 .40857
B   72  2 .1016486E+07 -4353903E-05 .2499E+07 .1538E+09 -.57037 -.13139 .81081
B   73  2 .1309886E+07 -1029572E-04 .2498E+07 .4314E+09 -.78832 -.50314 -.35412
B   74  2 .1158370E+07 -9204563E-05 .2497E+07 .1887E+09 -.24811 -.39145 .88612
B   78  2 .1215741E+07 -9663445E-05 .2498E+07 .3658E+09 -.81521 -.22210 -.53489
B   80  2 .7186035E+06 -1098205E-04 .2515E+07 .4518E+09 -.58971 .55097 -.59048
```

Figure A.5. Sample TRIM backscat.txt file.

APPENDIX B:
MICROMEGAS SIMULATION CODE LISTING

(a) Introduction

This appendix contains the computer code listings for the Micromegas neutron beam monitor simulation in the Visual Basic 6 programming language. The program also has a visual front-end that for obvious reasons cannot be included here. While the code does make reference to elements of the visual front-end (for example, text boxes containing user-defined values), the important details of how the simulation works may be found in this code listing.

This program is a work in progress in the sense that it is periodically modified in order to improve it or to produce new output data. As such, it is contains comments regarding how it may be modified to do this or that, many lines are more than 80 characters long (and thus wrap onto the next line in this document), and so forth. Further, while user control is normally accomplished by interacting with the visual front-end, the user may control the program by modifying the code directly and some variables (e.g. isotopic enrichment and gas density) are currently only adjustable via this method.

(b) MicromegasSim.frm

```
Option Explicit

'*************************************************************************
'*                                                                       *
'*                          MONTE CARLO SIMULATION OF                    *
'*                                                                       *
'*                       MICROMEGAS NEUTRON BEAM MONITOR                 *
'*                                                                       *
'*                                Andrew Stephan                         *
'*                                                                       *
'*************************************************************************
'*                                                                       *
'*                             MicromegasSim.frm                         *
'*                                                                       *
'*              Gets user input and sets up simulation runs.             *
'*                                                                       *
'*************************************************************************

Dim MacroCrossSection As Single
Dim TravelDistance As Single
Dim Interacted As Boolean
'Energies are in eV.
Dim NeutronEnergy As Single
Dim IonsInTargetArray(3, 6, 67, 3) As Single
'1st value is particle type, 2nd is target type, 3rd gives the order, 4th gives
'energy and range info
'Dim EnergyDepositionInGas(51, 5) As Single
'The first marker gives the section number since the path is being cut into
sections.
'It starts at 1 and doesn't correspond to any number from anything else.
'The 5 items addressable by the second marker are, respectively:
'(1) The energy deposition (in eV) during that path section.
'(2) The x point (in microns) at the start of the section.
'(3) The z point (in microns) at the start of the section.
'(4) The x point (in microns) at the end of the section.
'(5) The z point (in microns) at the end of the section.
Dim EnergyWhenEnteringGas(150) As Long
'Cross section arrays give the energy in eV and the microscopic cross section
in barns.
Dim B10XSection(2, 673) As Single
Dim Li6XSection(2, 498) As Single
Dim He3XSection(2, 342) As Single
'Fractional enrichment, can be used or ignored as needed.
Dim B10Enrichment As Single
Dim Li6Enrichment As Single
Dim He3Enrichment As Single

Dim FixedYTarget As Boolean
Dim FixedZTarget As Boolean
Dim FixedYTargetPoint As Single ' enter in cm
Dim FixedZTargetPoint As Single ' enter in cm

'*************************************************************************
'*                                                                       *
'*                  Code for Setting Things Up and Quitting              *
'*                                                                       *
'*************************************************************************

'Codes for particle type:
'1 = triton (H-3), 2 = alpha (He-4), 3 = lithium-7
'Codes for material type:
'1 is B, 2 is Li, 3 is Al, 4 is Ar, 5 is 90% Ar/10% isobutane, 6 is P-10 (90%
Ar/10% methane)
'7 is LiF, 8 is Si, 9 is glass (SiO2), 10 is doped glass - 20 vol% B-10, 11 is
doped
'glass - 20 vol% Li-6, 12 is inert lithium, 13 is inert boron

Private Sub Form_Load()
   '
   Dim i As Long

   For i = 0 To (MaxNumberOfLayers - 1) Step 1
      comboMaterial(i).AddItem "<none>", 0
```

```
      comboMaterial(i).AddItem "Boron-10", 1
      comboMaterial(i).AddItem "Lithium-6", 2
      comboMaterial(i).AddItem "Aluminium", 3
      comboMaterial(i).AddItem "Argon gas", 4
      comboMaterial(i).AddItem "90/10 Ar/isobutane gas", 5
      comboMaterial(i).AddItem "P-10 gas (Ar/methane)", 6
      comboMaterial(i).AddItem "Li-F (100% Li-6)", 7
      comboMaterial(i).AddItem "Silicon", 8
      comboMaterial(i).AddItem "Glass (SiO2)", 9
      comboMaterial(i).AddItem "Doped glass (20 vol% B-10)", 10
      comboMaterial(i).AddItem "Doped glass (20 vol% Li-6)", 11
      comboMaterial(i).AddItem "Lithium (inert)", 12
      comboMaterial(i).AddItem "Boron (inert)", 13

   Next i

   MicromegasSim.Show
   optTotalDeposition.Value = True
   optAllowTransmissions.Value = True
   checkNormallyIncident.Value = 1
   Randomize Timer
   B10Enrichment = 1#
   Li6Enrichment = 1#
   He3Enrichment = 1#

   SingleTargetStrip = False
   SingleTargetStripNumber = 0

   Call ReadInXSections

   Call RunFromCode
   'Call RunInputFileFromCode
End Sub

Private Sub QuitButton_Click()
   End
End Sub

'*****************************************************************************
'*                                                                           *
'*                Code that Controls Running Simulation Cases                *
'*                                                                           *
'*****************************************************************************

Private Sub RunButton_Click()
   '
   Dim i As Long
   Dim ii As Long
   Dim temp$
   Dim ENeutron As Single
   Dim xstartpos As Long
   Dim ystartpos As Long
   Dim zstartpos As Long
   Dim spacestring$

   Dim combostring$

   Dim currentrnd As Single
   Dim secondrnd As Single
   Dim ratio As Single
   Dim opticallength As Single
   Dim opticallength1 As Single
   Dim opticallength2 As Single
   Dim TravelDistance As Single
   Dim TL1path As Single
   Dim TL2path As Single
   Dim probability1 As Single
   Dim probability2 As Single
   Dim totalprobability As Single
   Dim TargetLayerInteractedIn As Long
   Dim TLEN As Long

   Dim nxdir As Single
   Dim nydir As Single
   Dim nzdir As Single
   Dim normalizer As Single
```

```
   'Added in as fix.  Different gases have different densities; thus this
should
   'eventually automated in being set by hand in the code.
   GasDensity = 0.00175

   'Determine the characteristics of the material layers as defined by the
user.
   i = 0
   For ii = 0 To (MaxNumberOfLayers - 1) Step 1
      If comboMaterial(ii).ListIndex > 0 Then i = i + 1
   Next ii

   NumberOfLayers = i

   If (NumberOfLayers = 0) Then
      MsgBox "You must enter at least one layer.", , "Error:"
      Exit Sub
   End If

   ReDim LayerThickness(NumberOfLayers)
   ReDim LayerMaterial(NumberOfLayers)

   i = 1
   TargetLayer = -1
   SecondTargetLayer = -1
   LayerOfInterest = -1

   For ii = 0 To (MaxNumberOfLayers - 1) Step 1
      If comboMaterial(ii).ListIndex > 0 Then
         LayerMaterial(i) = comboMaterial(ii).ListIndex
         LayerThickness(i) = CSng(txtThickness(ii).Text)
         If LayerThickness(i) <= 0# Then
            MsgBox "Material layer thickness must be greater than zero.", ,
"Error:"
            Exit Sub
         End If
         If (TargetLayer = -1) Then
            If (LayerMaterial(i) = 1 Or LayerMaterial(i) = 2 Or
LayerMaterial(i) = 7 Or LayerMaterial(i) = 10 Or LayerMaterial(i) = 11) Then
               TargetLayer = ii + 1
            End If
         Else
            If (LayerMaterial(i) = 1 Or LayerMaterial(i) = 2 Or
LayerMaterial(i) = 7 Or LayerMaterial(i) = 10 Or LayerMaterial(i) = 11) Then
               SecondTargetLayer = ii + 1
            End If
         End If
         If (optLOI(ii).Value = True) Then LayerOfInterest = ii + 1
         i = i + 1
      End If
   Next ii

   If (TargetLayer = -1) Then
      MsgBox "The detector must contain neutron-sensitive materials.", ,
"Error:"
      Exit Sub
   End If
   If (LayerOfInterest = -1) Then
      MsgBox "You must select a layer of interest.", , "Error:"
      Exit Sub
   End If

   'Convert the layer thicknesses from microns to angstroms.
   For i = 1 To NumberOfLayers Step 1
      LayerThickness(i) = LayerThickness(i) * 10000#
   Next i

   TargetThickness = LayerThickness(TargetLayer)
   If (SecondTargetLayer > 0) Then
      SecondTargetThickness = LayerThickness(SecondTargetLayer)
   End If

   AllowTransmissions = True

   If (optTotalDeposition = True) Then
      GeneratePulse = False
   Else
```

```
      GeneratePulse = True
   End If
   If (checkSemiconductorVariance.Value = 1) Then
      SemiconductorVariance = True
   Else
      SemiconductorVariance = False
   End If
   If (checkSaveReactionLocations.Value = 1) Then
      SaveReactionLocations = True
   Else
      SaveReactionLocations = False
   End If
   If (checkReactionProductInfo.Value = 1) Then
      IncludeReactionProductInfo = True
   Else
      IncludeReactionProductInfo = False
   End If
   If (MicromegasSim.checkElectronToAnodeSave.Value = 1) Then
      SaveAnodeSignals = True
   End If
   If (MicromegasSim.checkQFCSExcel.Value = 1) Then
      SaveQFCSToExcel = True
   End If
   If (MicromegasSim.checkSaveQFCS.Value = 1) Then
      SaveQFCS = True
   End If
   If (MicromegasSim.checkSaveEnergyDeposition.Value = 1) Then
      SaveEnergyDeposition = True
   End If
   If (optAllowTransmissions.Value = True) Then
      AllowTransmissions = True
   Else
      AllowTransmissions = False
   End If
   If (checkNormallyIncident.Value = 1) Then
      nxdir = 1#
      nydir = 0#
      nzdir = 0#
   Else
      nxdir = CSng(txtxdir.Text)
      nydir = CSng(txtydir.Text)
      nzdir = CSng(txtzdir.Text)
      If (nxdir <= 0#) Then
         MsgBox "The x direction vector must be positive.", , "Error:"
         Exit Sub
      End If
      normalizer = (nxdir ^ 2# + nydir ^ 2# + nzdir ^ 2#)
      normalizer = normalizer ^ 0.5
      nxdir = nxdir / normalizer
      nydir = nydir / normalizer
      nzdir = nzdir / normalizer
   End If
   eVperIonization = CSng(txtIonizationEnergy.Text)
   AnodeGranularity = CSng(txtAnodePitch.Text)
   DiffusionCoefficient = CSng(txtDiffusionCoefficient.Text)
   GasGain = CSng(txtGasGain.Text)
   FittingRatio = CSng(txtFittingRatio.Text)
   NumberOfRuns = CLng(txtNumberOfRuns.Text)
   YDimension = CSng(txtYlength.Text) * 10000# 'convert to microns
   ZDimension = CSng(txtZlength.Text) * 10000# 'convert to microns
   EnergyBinSize = CSng(txtEnergyBinSize.Text)
   ENeutron = txtNeutronEnergy.Text

   EnergyBinSize = EnergyBinSize * 1000# 'Convert from keV to eV.

   'Delete all files in the OutputFiles folder so errors won't occur later on
as a
   'result of trying to copy files over files that already exist.
   If (Dir$(path$ & "\OutputFiles\*.*") = "") Then
      'There isn't anything in the folder.
   Else
      Kill (path$ & "\OutputFiles\*.*")
   End If

   DoEvents

   'Create a TRIM.DAT file for each type of reaction product.
```

```
   Open path$ & "OLDTRIM.DAT" For Input As #1
   'Create one TRIM.DAT type of file for each type of reaction product.
   Open path$ & "TRIM1.DAT" For Output As #2
   Open path$ & "TRIM2.DAT" For Output As #3
   If (SecondTargetLayer > 0) Then
      Open path$ & "2TRIM1.DAT" For Output As #4
      Open path$ & "2TRIM2.DAT" For Output As #5
   End If

   'Insert boilerplate at the start of the file.
   For i = 1 To 10 Step 1
      Line Input #1, temp$
      Print #2, temp$
      Print #3, temp$
      If (SecondTargetLayer > 0) Then
         Print #4, temp$
         Print #5, temp$
      End If
   Next i

   Close #1

   If (ENeutron <= 0#) Then
      MsgBox "Neutrons must have energy greater than zero.", , "Error:"
      End
   End If

   'The reaction locations need to be selected for all neutrons, tallies must
be kept
   'of many neutrons went through and how many didn't, how many produced pulses
out of
   'those that were absorbed, and if there are multiple layers of neutron
sensitive
   'material, the simplest way to treat them is run them as entirely separate
   'simulations *but* make sure that they draw on the reaction data generated
as the
   'first thing in the simulation.

   TL1EventNum = 0
   TL2EventNum = 0

   If (SaveReactionLocations = True) Then
      Open path$ & "OutputFiles/1REACTION_LOCATIONS.TXT" For Output As #6
      If (SecondTargetLayer > 0) Then
         Open path$ & "OutputFiles/2REACTION_LOCATIONS.TXT" For Output As #7
      End If
   End If

   opticallength1 = MacroscopicXSection(ENeutron, LayerMaterial(TargetLayer)) *
(LayerThickness(TargetLayer) / nxdir) * (0.00000001)
   TL1path = LayerThickness(TargetLayer) / nxdir
   If (SecondTargetLayer > 0) Then
      opticallength2 = MacroscopicXSection(ENeutron,
LayerMaterial(SecondTargetLayer)) * (LayerThickness(SecondTargetLayer) / nxdir)
* (0.00000001)
      opticallength = opticallength1 + opticallength2
      TL2path = LayerThickness(SecondTargetLayer) / nxdir
   Else
      opticallength = opticallength1
   End If

   If (AllowTransmissions = True) Then
      'Collisions occur or do not occur based on the cross section of the
neutron
      'target material layers.
      If (SecondTargetLayer > 0) Then
         'There are two layers of neutron target material.
         totalprobability = 1 - ee ^ (-opticallength)
         For i = 1 To NumberOfRuns Step 1
            TargetLayerInteractedIn = 0
            currentrnd = Rnd
            If (currentrnd < totalprobability) Then
               'It interacted somewhere in the two layers.
               TravelDistance = -Log(1 - currentrnd)
               If (TravelDistance > opticallength1) Then
                  'It interacted in the second layer.
```

```
                  TravelDistance = TravelDistance - opticallength1
                  Call CalculateAllNeutronReactionInfo(ENeutron,
LayerMaterial(SecondTargetLayer), nxdir, nydir, nzdir)
                  xstartpos = (TravelDistance / opticallength2) *
LayerThickness(SecondTargetLayer)
                  TargetLayerInteractedIn = 2
                  TL2EventNum = TL2EventNum + 1
                  TLEN = TL2EventNum
               Else
                  'It interacted in the first layer.
                  Call CalculateAllNeutronReactionInfo(ENeutron,
LayerMaterial(TargetLayer), nxdir, nydir, nzdir)
                  xstartpos = (TravelDistance / opticallength1) *
LayerThickness(TargetLayer)
                  TargetLayerInteractedIn = 1
                  TL1EventNum = TL1EventNum + 1
                  TLEN = TL1EventNum
               End If

               If (FixedYTarget = True) Then
                  ystartpos = FixedYTargetPoint * 10000# * 10000#  'convert
from cm to angstroms
               Else
                  ystartpos = Rnd * YDimension * 10000# 'convert from microns
to angstroms
               End If
               If (FixedZTarget = True) Then
                  zstartpos = FixedZTargetPoint * 10000# * 10000#  'convert
from cm to angstroms
               Else
                  zstartpos = Rnd * ZDimension * 10000#  'convert from microns
to angstroms
               End If

               spacestring$ = ""
               If (TLEN < 10) Then
                  spacestring$ = "       "
               ElseIf (TLEN < 100) Then
                  spacestring$ = "      "
               ElseIf (TLEN < 1000) Then
                  spacestring$ = "     "
               ElseIf (TLEN < 10000) Then
                  spacestring$ = "    "
               ElseIf (TLEN < 100000) Then
                  spacestring$ = "   "
               Else
                  spacestring$ = "  "
               End If
               If (TargetLayerInteractedIn = 1) Then
                  Print #2, TLEN & spacestring$ & P1M & " " & E1 & " " &
xstartpos & " " & ystartpos & " " & zstartpos & " " & Format(x1, "0.0000") & "
" & Format(y1, "0.0000") & " " & Format(z1, "0.0000")
                  Print #3, TLEN & spacestring$ & P2M & " " & E2 & " " &
xstartpos & " " & ystartpos & " " & zstartpos & " " & Format(x2, "0.0000") & "
" & Format(y2, "0.0000") & " " & Format(z2, "0.0000")
                  If (SaveReactionLocations = True) Then
                     'Convert the location to microns and save it.
                     Print #6, (xstartpos / 10000#) & Chr$(9) & (ystartpos /
10000#) & Chr$(9) & (zstartpos / 10000#)
                  End If
               ElseIf (TargetLayerInteractedIn = 2) Then
                  Print #4, TLEN & spacestring$ & P1M & " " & E1 & " " &
xstartpos & " " & ystartpos & " " & zstartpos & " " & Format(x1, "0.0000") & "
" & Format(y1, "0.0000") & " " & Format(z1, "0.0000")
                  Print #5, TLEN & spacestring$ & P2M & " " & E2 & " " &
xstartpos & " " & ystartpos & " " & zstartpos & " " & Format(x2, "0.0000") & "
" & Format(y2, "0.0000") & " " & Format(z2, "0.0000")
                  If (SaveReactionLocations = True) Then
                     'Convert the location to microns and save it.
                     Print #7, (xstartpos / 10000#) & Chr$(9) & (ystartpos /
10000#) & Chr$(9) & (zstartpos / 10000#)
                  End If
               End If
            End If
         Next i
      Else
         'There is only one layer of neutron target material.
```

```
         probability1 = 1 - ee ^ (-opticallength1)
         For i = 1 To NumberOfRuns Step 1
            currentrnd = Rnd
            If (currentrnd < probability1) Then
               'It interacted in the target layer.
               TravelDistance = -Log(1 - currentrnd)
               Call CalculateAllNeutronReactionInfo(ENeutron,
LayerMaterial(TargetLayer), nxdir, nydir, nzdir)
               xstartpos = (TravelDistance / opticallength1) *
LayerThickness(TargetLayer)
               TL1EventNum = TL1EventNum + 1
               spacestring$ = ""
               If (TL1EventNum < 10) Then
                  spacestring$ = "       "
               ElseIf (TL1EventNum < 100) Then
                  spacestring$ = "      "
               ElseIf (TL1EventNum < 1000) Then
                  spacestring$ = "     "
               ElseIf (TL1EventNum < 10000) Then
                  spacestring$ = "    "
               ElseIf (TL1EventNum < 100000) Then
                  spacestring$ = "   "
               Else
                  spacestring$ = "  "
               End If

               If (FixedYTarget = True) Then
                  ystartpos = FixedYTargetPoint * 10000# * 10000#  'convert
from cm to angstroms
               Else
                  ystartpos = Rnd * YDimension * 10000# 'convert from microns
to angstroms
               End If
               If (FixedZTarget = True) Then
                  zstartpos = FixedZTargetPoint * 10000# * 10000#  'convert
from cm to angstroms
               Else
                  zstartpos = Rnd * ZDimension * 10000#  'convert from microns
to angstroms
               End If

               Print #2, TL1EventNum & spacestring$ & P1M & " " & E1 & " " &
xstartpos & " " & ystartpos & " " & zstartpos & " " & Format(x1, "0.0000") & "
" & Format(y1, "0.0000") & " " & Format(z1, "0.0000")
               Print #3, TL1EventNum & spacestring$ & P2M & " " & E2 & " " &
xstartpos & " " & ystartpos & " " & zstartpos & " " & Format(x2, "0.0000") & "
" & Format(y2, "0.0000") & " " & Format(z2, "0.0000")
               If (SaveReactionLocations = True) Then
                  'Convert the location to microns and save it.
                  Print #6, (xstartpos / 10000#) & Chr$(9) & (ystartpos /
10000#) & Chr$(9) & (zstartpos / 10000#)
               End If
            End If
         Next i
      End If
   Else
      'A collision should be forced.
      If (SecondTargetLayer > 0) Then
         'There are two layers of neutron target material.
         For i = 1 To NumberOfRuns Step 1
            probability1 = 1 - ee ^ (-opticallength1)
            totalprobability = 1 - ee ^ (-opticallength)
            'Now normalize everything.
            probability1 = probability1 / totalprobability

            currentrnd = Rnd
            secondrnd = Rnd
            If (currentrnd < probability1) Then
               probability1 = 1 - ee ^ -opticallength1
               ratio = 1# / probability1
               TravelDistance = -Log(1 - secondrnd / ratio)
               Call CalculateAllNeutronReactionInfo(ENeutron,
LayerMaterial(TargetLayer), nxdir, nydir, nzdir)
               xstartpos = (TravelDistance / opticallength1) *
LayerThickness(TargetLayer)
               TargetLayerInteractedIn = 1
               TL1EventNum = TL1EventNum + 1
```

```
               TLEN = TL1EventNum
            Else
               probability2 = 1 - ee ^ -opticallength2
               ratio = 1# / probability2
               TravelDistance = -Log(1 - secondrnd / ratio)
               Call CalculateAllNeutronReactionInfo(ENeutron,
LayerMaterial(SecondTargetLayer), nxdir, nydir, nzdir)
               xstartpos = (TravelDistance / opticallength2) *
LayerThickness(SecondTargetLayer)
               TargetLayerInteractedIn = 2
               TL2EventNum = TL2EventNum + 1
               TLEN = TL2EventNum
            End If

            If (FixedYTarget = True) Then
               ystartpos = FixedYTargetPoint * 10000# * 10000#  'convert from
cm to angstroms
            Else
               ystartpos = Rnd * YDimension * 10000# 'convert from microns to
angstroms
            End If
            If (FixedZTarget = True) Then
               zstartpos = FixedZTargetPoint * 10000# * 10000#  'convert from
cm to angstroms
            Else
               zstartpos = Rnd * ZDimension * 10000#  'convert from microns to
angstroms
            End If

            spacestring$ = ""
            If (TLEN < 10) Then
               spacestring$ = "       "
            ElseIf (TLEN < 100) Then
               spacestring$ = "      "
            ElseIf (TLEN < 1000) Then
               spacestring$ = "     "
            ElseIf (TLEN < 10000) Then
               spacestring$ = "    "
            ElseIf (TLEN < 100000) Then
               spacestring$ = "   "
            Else
               spacestring$ = "  "
            End If
            If (TargetLayerInteractedIn = 1) Then
               Print #2, TLEN & spacestring$ & P1M & " " & E1 & " " & xstartpos
& " " & ystartpos & " " & zstartpos & " " & Format(x1, "0.0000") & " " &
Format(y1, "0.0000") & " " & Format(z1, "0.0000")
               Print #3, TLEN & spacestring$ & P2M & " " & E2 & " " & xstartpos
& " " & ystartpos & " " & zstartpos & " " & Format(x2, "0.0000") & " " &
Format(y2, "0.0000") & " " & Format(z2, "0.0000")
               If (SaveReactionLocations = True) Then
                  'Convert the location to microns and save it.
                  Print #6, (xstartpos / 10000#) & Chr$(9) & (ystartpos /
10000#) & Chr$(9) & (zstartpos / 10000#)
               End If
            ElseIf (TargetLayerInteractedIn = 2) Then
               Print #4, TLEN & spacestring$ & P1M & " " & E1 & " " & xstartpos
& " " & ystartpos & " " & zstartpos & " " & Format(x1, "0.0000") & " " &
Format(y1, "0.0000") & " " & Format(z1, "0.0000")
               Print #5, TLEN & spacestring$ & P2M & " " & E2 & " " & xstartpos
& " " & ystartpos & " " & zstartpos & " " & Format(x2, "0.0000") & " " &
Format(y2, "0.0000") & " " & Format(z2, "0.0000")
               If (SaveReactionLocations = True) Then
                  'Convert the location to microns and save it.
                  Print #7, (xstartpos / 10000#) & Chr$(9) & (ystartpos /
10000#) & Chr$(9) & (zstartpos / 10000#)
               End If
            End If
         Next i
      Else
         'There is only one layer of neutron target material.
         For i = 1 To NumberOfRuns Step 1
            currentrnd = Rnd
            probability1 = 1 - ee ^ -opticallength1
            ratio = 1# / probability1
            TravelDistance = -Log(1 - currentrnd / ratio)
```

```
            Call CalculateAllNeutronReactionInfo(ENeutron,
LayerMaterial(TargetLayer), nxdir, nydir, nzdir)
            xstartpos = (TravelDistance / opticallength1) *
LayerThickness(TargetLayer)
            TargetLayerInteractedIn = 1
            TL1EventNum = TL1EventNum + 1
            TLEN = TL1EventNum

            If (FixedYTarget = True) Then
               ystartpos = FixedYTargetPoint * 10000# * 10000#  'convert from
cm to angstroms
            Else
               ystartpos = Rnd * YDimension * 10000# 'convert from microns to
angstroms
            End If
            If (FixedZTarget = True) Then
               zstartpos = FixedZTargetPoint * 10000# * 10000#  'convert from
cm to angstroms
            Else
               zstartpos = Rnd * ZDimension * 10000#  'convert from microns to
angstroms
            End If

            spacestring$ = ""
            If (TLEN < 10) Then
               spacestring$ = "       "
            ElseIf (TLEN < 100) Then
               spacestring$ = "      "
            ElseIf (TLEN < 1000) Then
               spacestring$ = "     "
            ElseIf (TLEN < 10000) Then
               spacestring$ = "    "
            ElseIf (TLEN < 100000) Then
               spacestring$ = "   "
            Else
               spacestring$ = "  "
            End If
            Print #2, TLEN & spacestring$ & P1M & " " & E1 & " " & xstartpos &
" " & ystartpos & " " & zstartpos & " " & Format(x1, "0.0000") & " " &
Format(y1, "0.0000") & " " & Format(z1, "0.0000")
            Print #3, TLEN & spacestring$ & P2M & " " & E2 & " " & xstartpos &
" " & ystartpos & " " & zstartpos & " " & Format(x2, "0.0000") & " " &
Format(y2, "0.0000") & " " & Format(z2, "0.0000")
            If (SaveReactionLocations = True) Then
               'Convert the location to microns and save it.
               Print #6, (xstartpos / 10000#) & Chr$(9) & (ystartpos / 10000#)
& Chr$(9) & (zstartpos / 10000#)
            End If
         Next i
      End If
   End If

   'Insert an end-of-file character at the end of each file (this *must* be
done) and then
   'close it.
   Print #2, Chr$(26)
   Print #3, Chr$(26)
   Close #2
   Close #3
   If (SaveReactionLocations = True) Then
      Print #6, Chr$(26)
      Close #6
   End If
   If (SecondTargetLayer > 0) Then
      Print #4, Chr$(26)
      Print #5, Chr$(26)
      Close #4
      Close #5
      If (SaveReactionLocations = True) Then
         Print #7, Chr$(26)
         Close #7
      End If
   End If

   DoEvents

   If (LayerMaterial(TargetLayer) = 1) Then
```

```
      MaximumPossibleEnergy = 2790000 + ENeutron
   ElseIf (LayerMaterial(TargetLayer) = 2 Or LayerMaterial(TargetLayer) = 7)
Then
      MaximumPossibleEnergy = 4780000 + ENeutron
   End If

   'Run the first target layer in the usual way.
   Call TrimAutomation.NewRunCase
   DoEvents
   Call AnalyzeTrimData.AnalyzeTrimData

   'Put the files from the first layer in the OutputFiles folder so they won't
be lost,
   'renaming them along the way and deleting the old copies.
   If Dir$(path$ & "TOTAL_ENERGY_DEPOSITION.TXT") = "" Then
      'The file does not exist.
   Else
      FileCopy (path$ & "TOTAL_ENERGY_DEPOSITION.TXT"), (path$ &
"OutputFiles\1TOTAL_ENERGY_DEPOSITION.TXT")
      Kill (path$ & "TOTAL_ENERGY_DEPOSITION.TXT")
   End If
   If Dir$(path$ & "BINNED_ENERGY_DEPOSITION.TXT") = "" Then
      'The file does not exist.
   Else
      FileCopy (path$ & "BINNED_ENERGY_DEPOSITION.TXT"), (path$ &
"OutputFiles\1BINNED_ENERGY_DEPOSITION.TXT")
      Kill (path$ & "BINNED_ENERGY_DEPOSITION.TXT")
   End If
   For i = 1 To 3 Step 1
      If Dir$(path$ & "BINNED_ENERGY_DEPOSITION-" & i & ".TXT") = "" Then
         'The file does not exist.
      Else
         FileCopy (path$ & "BINNED_ENERGY_DEPOSITION-" & i & ".TXT"), (path$ &
"OutputFiles\1BINNED_ENERGY_DEPOSITION-" & i & ".TXT")
         Kill (path$ & "BINNED_ENERGY_DEPOSITION-" & i & ".TXT")
      End If
   Next i
   If Dir$(path$ & "QFCS_DATA.TXT") = "" Then
      'The file does not exist.
   Else
      FileCopy (path$ & "QFCS_DATA.TXT"), (path$ &
"OutputFiles\1QFCS_DATA.TXT")
      Kill (path$ & "QFCS_DATA.TXT")
   End If
   For i = 1 To 3 Step 1
      If Dir$(path$ & "QFCS_DATA_" & i & ".TXT") = "" Then
         'The file does not exist.
      Else
         FileCopy (path$ & "QFCS_DATA_" & i & ".TXT"), (path$ &
"OutputFiles\1QFCS_DATA_" & i & ".TXT")
         Kill (path$ & "QFCS_DATA_" & i & ".TXT")
      End If
   Next i
   If (TL1EventNum > 0 And SaveEnergyDeposition = True) Then
      For i = 1 To TL1EventNum Step 1
         If (Dir$(path$ & "ANODE_STRIP_DEPOSITION_" & i & ".TXT") = "") Then
            'The file does not exist.
         Else
            FileCopy (path$ & "ANODE_STRIP_DEPOSITION_" & i & ".TXT"), (path$ &
"OutputFiles\1ANODE_STRIP_DEPOSITION_" & i & ".TXT")
            Kill (path$ & "ANODE_STRIP_DEPOSITION_" & i & ".TXT")
         End If
      Next i
   End If

   If (SecondTargetLayer > 0) Then
      'Delete the old input files from the first layer so the input files for
the
      'second layer can be assigned the same name.
      If (Dir$(path$ & "TRIM1.DAT") = "") Then
         'File doesn't exist.
      Else
         Kill (path$ & "TRIM1.DAT")
      End If
      If (Dir$(path$ & "TRIM2.DAT") = "") Then
         'File doesn't exist.
      Else
```

```
         Kill (path$ & "TRIM2.DAT")
      End If

      'Next, copy the input files for the second target layer over to the
correct names.
      If (Dir$(path$ & "2TRIM1.DAT") = "") Then
         'File doesn't exist.
      Else
         Name (path$ & "2TRIM1.DAT") As (path$ & "TRIM1.DAT")
      End If
      If (Dir$(path$ & "2TRIM2.DAT") = "") Then
         'File doesn't exist.
      Else
         Name (path$ & "2TRIM2.DAT") As (path$ & "TRIM2.DAT")
      End If

      DoEvents

      'Change any variables that need changing.
      TargetLayer = SecondTargetLayer
      If (LayerMaterial(TargetLayer) = 1) Then
         MaximumPossibleEnergy = 2790000 + ENeutron
      ElseIf (LayerMaterial(TargetLayer) = 2 Or LayerMaterial(TargetLayer) = 7)
Then
         MaximumPossibleEnergy = 4780000 + ENeutron
      End If

      'Now run the second target layer.
      Call TrimAutomation.NewRunCase
      DoEvents
      Call AnalyzeTrimData.AnalyzeTrimData

      'Rename the output files appropriately if they exist.
      If Dir$(path$ & "TOTAL_ENERGY_DEPOSITION.TXT") = "" Then
         'The file does not exist.
      Else
         FileCopy (path$ & "TOTAL_ENERGY_DEPOSITION.TXT"), (path$ &
"OutputFiles\2TOTAL_ENERGY_DEPOSITION.TXT")
         Kill (path$ & "TOTAL_ENERGY_DEPOSITION.TXT")
      End If
      If Dir$(path$ & "BINNED_ENERGY_DEPOSITION.TXT") = "" Then
         'The file does not exist.
      Else
         FileCopy (path$ & "BINNED_ENERGY_DEPOSITION.TXT"), (path$ &
"OutputFiles\2BINNED_ENERGY_DEPOSITION.TXT")
         Kill (path$ & "BINNED_ENERGY_DEPOSITION.TXT")
      End If
      For i = 1 To 3 Step 1
         If Dir$(path$ & "BINNED_ENERGY_DEPOSITION-" & i & ".TXT") = "" Then
            'The file does not exist.
         Else
            FileCopy (path$ & "BINNED_ENERGY_DEPOSITION-" & i & ".TXT"), (path$
& "OutputFiles\2BINNED_ENERGY_DEPOSITION-" & i & ".TXT")
            Kill (path$ & "BINNED_ENERGY_DEPOSITION-" & i & ".TXT")
         End If
      Next i
      If Dir$(path$ & "QFCS_DATA.TXT") = "" Then
         'The file does not exist.
      Else
         FileCopy (path$ & "QFCS_DATA.TXT"), (path$ &
"OutputFiles\2QFCS_DATA.TXT")
         Kill (path$ & "QFCS_DATA.TXT")
      End If
      For i = 1 To 3 Step 1
         If Dir$(path$ & "QFCS_DATA_" & i & ".TXT") = "" Then
            'The file does not exist.
         Else
            FileCopy (path$ & "QFCS_DATA_" & i & ".TXT"), (path$ &
"OutputFiles\2QFCS_DATA_" & i & ".TXT")
            Kill (path$ & "QFCS_DATA_" & i & ".TXT")
         End If
      Next i
      If (TL2EventNum > 0 And MicromegasSim.checkSaveEnergyDeposition.Value =
1) Then
         For i = 1 To TL2EventNum Step 1
            If (Dir$(path$ & "ANODE_STRIP_DEPOSITION_" & i & ".TXT") = "") Then
               'The file does not exist.
```

```
            Else
               FileCopy (path$ & "ANODE_STRIP_DEPOSITION_" & i & ".TXT"),
(path$ & "OutputFiles\2ANODE_STRIP_DEPOSITION_" & i & ".TXT")
               Kill (path$ & "ANODE_STRIP_DEPOSITION_" & i & ".TXT")
            End If
         Next i
      End If
   End If

   End

End Sub

Public Sub RunFromCode()
   'Rather than going through all the data entry stuff, instead enter the
commands and
   'parameters here and then run a bunch of things sequentially without any
user
   'involvement being needed.

   'WARNING: Don't go fiddling with this sub unless you have a reasonable idea
of what
   'is going on.  Not all of the error checking is present.

   Dim i As Long
   Dim ii As Long
   Dim iii As Long
   Dim iv As Long
   Dim temp$
   Dim ENeutron As Single
   Dim xstartpos As Long
   Dim ystartpos As Long
   Dim zstartpos As Long
   Dim spacestring$

   Dim combostring$

   Dim currentrnd As Single
   Dim secondrnd As Single
   Dim ratio As Single
   Dim opticallength As Single
   Dim opticallength1 As Single
   Dim opticallength2 As Single
   Dim TravelDistance As Single
   Dim TL1path As Single
   Dim TL2path As Single
   Dim probability1 As Single
   Dim probability2 As Single
   Dim totalprobability As Single
   Dim TargetLayerInteractedIn As Long
   Dim TLEN As Long

   Dim nxdir As Single
   Dim nydir As Single
   Dim nzdir As Single
   Dim normalizer As Single

   Dim copypath$

   Dim StripCenterPoint As Single

   Dim FileExists() As Boolean
   Dim FileEOF() As Boolean
   Dim inp_num As Long
   Dim WriteString As String
   Dim numelectrons As Single
   Dim AnyFilesNotEOF As Boolean

   Dim StripStart As Single
   Dim StripCenter As Single
   Dim StripEnd As Single

   Call AdjustCrossSections '----- Put in equivalent for somewhere else

   AllowTransmissions = False
   SemiconductorVariance = False
   IncludeReactionProductInfo = True
```

```
   nxdir = 1#
   nydir = 0#
   nzdir = 0#
   If (nxdir <= 0#) Then
      MsgBox "The x direction vector must be positive.", , "Error:"
      End
   End If
   normalizer = (nxdir ^ 2# + nydir ^ 2# + nzdir ^ 2#)
   normalizer = normalizer ^ 0.5
   If (normalizer <> 1#) Then
      nxdir = nxdir / normalizer
      nydir = nydir / normalizer
      nzdir = nzdir / normalizer
   End If
   eVperIonization = 22.4
   AnodeGranularity = 1000 '317.5
   DiffusionCoefficient = 200# '370
   GasGain = 1 '100
   FittingRatio = 1000
   NumberOfRuns = 1000
   YDimension = 5# * 10000# '10#
   ZDimension = 5# * 10000# '10#
   EnergyBinSize = 10# * 1000#
   ENeutron = 1000000# '0.025

   SaveReactionLocations = True
   GeneratePulse = True
   NumberOfLayers = 4
   ReDim LayerThickness(NumberOfLayers)
   ReDim LayerMaterial(NumberOfLayers)
   TargetLayer = 2
   SecondTargetLayer = -1
   LayerOfInterest = 3

   For iii = 1 To 25 Step 4 '25 Step 1 ' strip being looked at
      For iv = 1 To 199 Step 1 '199 Step 1 ' discrete starting location being
looked at
         GasDensity = 0.00175
         DiffusionCoefficient = 370# '200#

         LayerThickness(1) = 50# * 10000#
         LayerThickness(2) = 2# * 10000#
         LayerThickness(3) = 3000# * 10000# '3000
         LayerThickness(4) = 50# * 10000#
         '1 is B-10, 2 is Li-6, 3 is Al, 5 is Ar/isobutane
         LayerMaterial(1) = 3
         LayerMaterial(2) = 1
         LayerMaterial(3) = 5
         LayerMaterial(4) = 3
         NumberOfRuns = 400
         SaveReactionLocations = False
         GeneratePulse = True
         SaveAnodeSignals = False
         SaveQFCSToExcel = False
         SaveQFCS = False
         SaveEnergyDeposition = False
         copypath$ = "C:\Andrew\MicromegasSim\CurrentResults\"

         SingleTargetStrip = True
         SingleTargetStripNumber = iii

         FixedYTarget = True
         FixedYTargetPoint = iv * 0.025 ' use discrete starting points with 250
microns spacing
         FixedZTarget = False
         FixedZTargetPoint = 0#

         ' If discrete starting point is within 1 cm of one of the edges of the
strip,
         ' run the case, otherwise assume no effect.
         StripCenterPoint = (iii - 1) * 0.1 + 0.05

         If Abs(StripCenterPoint - FixedYTargetPoint) <= 0.35 Then

            '================ Code for running the case - can be moved around
=========
```

```
            ChDir (path$)

            'Delete all files in the OutputFiles folder so errors won't occur
later on as a
            'result of trying to copy files over files that already exist.
            If (Dir$(path$ & "\OutputFiles\*.*") = "") Then
               'There isn't anything in the folder.
            Else
               Kill (path$ & "\OutputFiles\*.*")
            End If

            'Create a TRIM.DAT file for each type of reaction product.

            Open path$ & "OLDTRIM.DAT" For Input As #1
            'Create one TRIM.DAT type of file for each type of reaction
product.
            Open path$ & "TRIM1.DAT" For Output As #2
            Open path$ & "TRIM2.DAT" For Output As #3
            If (SecondTargetLayer > 0) Then
               Open path$ & "2TRIM1.DAT" For Output As #4
               Open path$ & "2TRIM2.DAT" For Output As #5
            End If

            'Insert boilerplate at the start of the file.
            For i = 1 To 10 Step 1
               Line Input #1, temp$
               Print #2, temp$
               Print #3, temp$
               If (SecondTargetLayer > 0) Then
                  Print #4, temp$
                  Print #5, temp$
               End If
            Next i

            Close #1

            If (ENeutron <= 0#) Then
               MsgBox "Neutrons must have energy greater than zero.", ,
"Error:"
               End
            End If

            'The reaction locations need to be selected for all neutrons,
tallies must be kept
            'of many neutrons went through and how many didn't, how many
produced pulses out of
            'those that were absorbed, and if there are multiple layers of
neutron sensitive
            'material, the simplest way to treat them is run them as entirely
separate
            'simulations *but* make sure that they draw on the reaction data
generated as the
            'first thing in the simulation.

            TL1EventNum = 0
            TL2EventNum = 0

            If (SaveReactionLocations = True) Then
               Open path$ & "OutputFiles/1REACTION_LOCATIONS.TXT" For Output As
#6
               If (SecondTargetLayer > 0) Then
                  Open path$ & "OutputFiles/2REACTION_LOCATIONS.TXT" For Output
As #7
               End If
            End If

            opticallength1 = MacroscopicXSection(ENeutron,
LayerMaterial(TargetLayer)) * (LayerThickness(TargetLayer) / nxdir) *
(0.00000001)
            TL1path = LayerThickness(TargetLayer) / nxdir
            If (SecondTargetLayer > 0) Then
               opticallength2 = MacroscopicXSection(ENeutron,
LayerMaterial(SecondTargetLayer)) * (LayerThickness(SecondTargetLayer) / nxdir)
* (0.00000001)
               opticallength = opticallength1 + opticallength2
               TL2path = LayerThickness(SecondTargetLayer) / nxdir
            Else
```

```
            opticallength = opticallength1
         End If

         If (AllowTransmissions = True) Then
            'Collisions occur or do not occur based on the cross section of
the neutron
            'target material layers.
            If (SecondTargetLayer > 0) Then
               'There are two layers of neutron target material.
               totalprobability = 1 - ee ^ (-opticallength)
               For i = 1 To NumberOfRuns Step 1
                  TargetLayerInteractedIn = 0
                  currentrnd = Rnd
                  If (currentrnd < totalprobability) Then
                     'It interacted somewhere in the two layers.
                     TravelDistance = -Log(1 - currentrnd)
                     If (TravelDistance > opticallength1) Then
                        'It interacted in the second layer.
                        TravelDistance = TravelDistance - opticallength1
                        Call CalculateAllNeutronReactionInfo(ENeutron,
LayerMaterial(SecondTargetLayer), nxdir, nydir, nzdir)
                        xstartpos = (TravelDistance / opticallength2) *
LayerThickness(SecondTargetLayer)
                        TargetLayerInteractedIn = 2
                        TL2EventNum = TL2EventNum + 1
                        TLEN = TL2EventNum
                     Else
                        'It interacted in the first layer.
                        Call CalculateAllNeutronReactionInfo(ENeutron,
LayerMaterial(TargetLayer), nxdir, nydir, nzdir)
                        xstartpos = (TravelDistance / opticallength1) *
LayerThickness(TargetLayer)
                        TargetLayerInteractedIn = 1
                        TL1EventNum = TL1EventNum + 1
                        TLEN = TL1EventNum
                     End If

                     If (FixedYTarget = True) Then
                        ystartpos = FixedYTargetPoint * 10000# * 10000#
'convert from cm to angstroms
                     Else
                        ystartpos = Rnd * YDimension * 10000# 'convert from
microns to angstroms
                     End If
                     If (FixedZTarget = True) Then
                        zstartpos = FixedZTargetPoint * 10000# * 10000#
'convert from cm to angstroms
                     Else
                        zstartpos = Rnd * ZDimension * 10000#  'convert from
microns to angstroms
                     End If

                     'ystartpos = Rnd * YDimension * 10000#  'convert to
Angstroms
                     'zstartpos = Rnd * ZDimension * 10000#  'convert to
Angstroms

                     spacestring$ = ""
                     If (TLEN < 10) Then
                        spacestring$ = "       "
                     ElseIf (TLEN < 100) Then
                        spacestring$ = "      "
                     ElseIf (TLEN < 1000) Then
                        spacestring$ = "     "
                     ElseIf (TLEN < 10000) Then
                        spacestring$ = "    "
                     ElseIf (TLEN < 100000) Then
                        spacestring$ = "   "
                     Else
                        spacestring$ = "  "
                     End If
                     If (TargetLayerInteractedIn = 1) Then
                        Print #2, TLEN & spacestring$ & P1M & " " & E1 & " "
& xstartpos & " " & ystartpos & " " & zstartpos & " " & Format(x1, "0.0000") &
" " & Format(y1, "0.0000") & " " & Format(z1, "0.0000")
```

```
                        Print #3, TLEN & spacestring$ & P2M & " " & E2 & " "
& xstartpos & " " & ystartpos & " " & zstartpos & " " & Format(x2, "0.0000") &
" " & Format(y2, "0.0000") & " " & Format(z2, "0.0000")
                        If (SaveReactionLocations = True) Then
                           'Convert the x location to microns and the y and
z to cm and save them
                           Print #6, (xstartpos / 10000#) & Chr$(9) &
(ystartpos / 100000000#) & Chr$(9) & (zstartpos / 100000000#)
                        End If
                     ElseIf (TargetLayerInteractedIn = 2) Then
                        Print #4, TLEN & spacestring$ & P1M & " " & E1 & " "
& xstartpos & " " & ystartpos & " " & zstartpos & " " & Format(x1, "0.0000") &
" " & Format(y1, "0.0000") & " " & Format(z1, "0.0000")
                        Print #5, TLEN & spacestring$ & P2M & " " & E2 & " "
& xstartpos & " " & ystartpos & " " & zstartpos & " " & Format(x2, "0.0000") &
" " & Format(y2, "0.0000") & " " & Format(z2, "0.0000")
                        If (SaveReactionLocations = True) Then
                           'Convert the x location to microns and the y and
z to cm and save them
                           Print #7, (xstartpos / 10000#) & Chr$(9) &
(ystartpos / 100000000#) & Chr$(9) & (zstartpos / 100000000#)
                        End If
                     End If
                  End If
               Next i
            Else
               'There is only one layer of neutron target material.
               probability1 = 1 - ee ^ (-opticallength1)
               For i = 1 To NumberOfRuns Step 1
                  currentrnd = Rnd
                  If (currentrnd < probability1) Then
                     'It interacted in the target layer.
                     TravelDistance = -Log(1 - currentrnd)
                     Call CalculateAllNeutronReactionInfo(ENeutron,
LayerMaterial(TargetLayer), nxdir, nydir, nzdir)
                     xstartpos = (TravelDistance / opticallength1) *
LayerThickness(TargetLayer)
                     TL1EventNum = TL1EventNum + 1
                     spacestring$ = ""
                     If (TL1EventNum < 10) Then
                        spacestring$ = "       "
                     ElseIf (TL1EventNum < 100) Then
                        spacestring$ = "      "
                     ElseIf (TL1EventNum < 1000) Then
                        spacestring$ = "     "
                     ElseIf (TL1EventNum < 10000) Then
                        spacestring$ = "    "
                     ElseIf (TL1EventNum < 100000) Then
                        spacestring$ = "   "
                     Else
                        spacestring$ = "  "
                     End If

                     If (FixedYTarget = True) Then
                        ystartpos = FixedYTargetPoint * 10000# * 10000#
'convert from cm to angstroms
                     Else
                        ystartpos = Rnd * YDimension * 10000# 'convert from
microns to angstroms
                     End If
                     If (FixedZTarget = True) Then
                        zstartpos = FixedZTargetPoint * 10000# * 10000#
'convert from cm to angstroms
                     Else
                        zstartpos = Rnd * ZDimension * 10000#  'convert from
microns to angstroms
                     End If

                     'ystartpos = Rnd * YDimension * 10000#  'convert to
Angstroms
                     'zstartpos = Rnd * ZDimension * 10000#  'convert to
Angstroms

                     Print #2, TL1EventNum & spacestring$ & P1M & " " & E1 &
" " & xstartpos & " " & ystartpos & " " & zstartpos & " " & Format(x1,
"0.0000") & " " & Format(y1, "0.0000") & " " & Format(z1, "0.0000")
```

```
                        Print #3, TL1EventNum & spacestring$ & P2M & " " & E2 &
" " & xstartpos & " " & ystartpos & " " & zstartpos & " " & Format(x2,
"0.0000") & " " & Format(y2, "0.0000") & " " & Format(z2, "0.0000")
                        If (SaveReactionLocations = True) Then
                           'Convert the x location to microns and the y and z
to cm and save them
                           Print #6, (xstartpos / 10000#) & Chr$(9) &
(ystartpos / 100000000#) & Chr$(9) & (zstartpos / 100000000#)
                        End If
                     End If
                  Next i
               End If
            Else
               'A collision should be forced.
               If (SecondTargetLayer > 0) Then
                  'There are two layers of neutron target material.
                  For i = 1 To NumberOfRuns Step 1
                     probability1 = 1 - ee ^ (-opticallength1)
                     totalprobability = 1 - ee ^ (-opticallength)
                     'Now normalize everything.
                     probability1 = probability1 / totalprobability

                     currentrnd = Rnd
                     secondrnd = Rnd
                     If (currentrnd < probability1) Then
                        probability1 = 1 - ee ^ -opticallength1
                        ratio = 1# / probability1
                        TravelDistance = -Log(1 - secondrnd / ratio)
                        Call CalculateAllNeutronReactionInfo(ENeutron,
LayerMaterial(TargetLayer), nxdir, nydir, nzdir)
                        xstartpos = (TravelDistance / opticallength1) *
LayerThickness(TargetLayer)
                        TargetLayerInteractedIn = 1
                        TL1EventNum = TL1EventNum + 1
                        TLEN = TL1EventNum
                     Else
                        probability2 = 1 - ee ^ -opticallength2
                        ratio = 1# / probability2
                        TravelDistance = -Log(1 - secondrnd / ratio)
                        Call CalculateAllNeutronReactionInfo(ENeutron,
LayerMaterial(SecondTargetLayer), nxdir, nydir, nzdir)
                        xstartpos = (TravelDistance / opticallength2) *
LayerThickness(SecondTargetLayer)
                        TargetLayerInteractedIn = 2
                        TL2EventNum = TL2EventNum + 1
                        TLEN = TL2EventNum
                     End If

                     If (FixedYTarget = True) Then
                        ystartpos = FixedYTargetPoint * 10000# * 10000#
'convert from cm to angstroms
                     Else
                        ystartpos = Rnd * YDimension * 10000# 'convert from
microns to angstroms
                     End If
                     If (FixedZTarget = True) Then
                        zstartpos = FixedZTargetPoint * 10000# * 10000#
'convert from cm to angstroms
                     Else
                        zstartpos = Rnd * ZDimension * 10000#  'convert from
microns to angstroms
                     End If

                     'ystartpos = Rnd * YDimension * 10000#  'convert to
Angstroms
                     'zstartpos = Rnd * ZDimension * 10000#  'convert to
Angstroms

                     spacestring$ = ""
                     If (TLEN < 10) Then
                        spacestring$ = "       "
                     ElseIf (TLEN < 100) Then
                        spacestring$ = "      "
                     ElseIf (TLEN < 1000) Then
                        spacestring$ = "     "
                     ElseIf (TLEN < 10000) Then
                        spacestring$ = "    "
```

```
                    ElseIf (TLEN < 100000) Then
                       spacestring$ = "    "
                    Else
                       spacestring$ = "  "
                    End If
                    If (TargetLayerInteractedIn = 1) Then
                       Print #2, TLEN & spacestring$ & P1M & " " & E1 & " " &
xstartpos & " " & ystartpos & " " & zstartpos & " " & Format(x1, "0.0000") & "
" & Format(y1, "0.0000") & " " & Format(z1, "0.0000")
                       Print #3, TLEN & spacestring$ & P2M & " " & E2 & " " &
xstartpos & " " & ystartpos & " " & zstartpos & " " & Format(x2, "0.0000") & "
" & Format(y2, "0.0000") & " " & Format(z2, "0.0000")
                       If (SaveReactionLocations = True) Then
                          'Convert the x location to microns and the y and z
to cm and save them
                          Print #6, (xstartpos / 10000#) & Chr$(9) &
(ystartpos / 100000000#) & Chr$(9) & (zstartpos / 100000000#)
                       End If
                    ElseIf (TargetLayerInteractedIn = 2) Then
                       Print #4, TLEN & spacestring$ & P1M & " " & E1 & " " &
xstartpos & " " & ystartpos & " " & zstartpos & " " & Format(x1, "0.0000") & "
" & Format(y1, "0.0000") & " " & Format(z1, "0.0000")
                       Print #5, TLEN & spacestring$ & P2M & " " & E2 & " " &
xstartpos & " " & ystartpos & " " & zstartpos & " " & Format(x2, "0.0000") & "
" & Format(y2, "0.0000") & " " & Format(z2, "0.0000")
                       If (SaveReactionLocations = True) Then
                          'Convert the x location to microns and the y and z
to cm and save them
                          Print #7, (xstartpos / 10000#) & Chr$(9) &
(ystartpos / 100000000#) & Chr$(9) & (zstartpos / 100000000#)
                       End If
                    End If
                 Next i
              Else
                 'There is only one layer of neutron target material.
                 For i = 1 To NumberOfRuns Step 1
                    currentrnd = Rnd
                    probability1 = 1 - ee ^ -opticallength1
                    ratio = 1# / probability1
                    TravelDistance = -Log(1 - currentrnd / ratio)
                    Call CalculateAllNeutronReactionInfo(ENeutron,
LayerMaterial(TargetLayer), nxdir, nydir, nzdir)
                    xstartpos = (TravelDistance / opticallength1) *
LayerThickness(TargetLayer)
                    TargetLayerInteractedIn = 1
                    TL1EventNum = TL1EventNum + 1
                    TLEN = TL1EventNum

                    If (FixedYTarget = True) Then
                       ystartpos = FixedYTargetPoint * 10000# * 10000#
'convert from cm to angstroms
                    Else
                       ystartpos = Rnd * YDimension * 10000# 'convert from
microns to angstroms
                    End If
                    If (FixedZTarget = True) Then
                       zstartpos = FixedZTargetPoint * 10000# * 10000#
'convert from cm to angstroms
                    Else
                       zstartpos = Rnd * ZDimension * 10000#  'convert from
microns to angstroms
                    End If

                    'ystartpos = Rnd * YDimension * 10000#  'convert to
Angstroms
                    'zstartpos = Rnd * ZDimension * 10000#  'convert to
Angstroms

                    spacestring$ = ""
                    If (TLEN < 10) Then
                       spacestring$ = "       "
                    ElseIf (TLEN < 100) Then
                       spacestring$ = "      "
                    ElseIf (TLEN < 1000) Then
                       spacestring$ = "     "
                    ElseIf (TLEN < 10000) Then
                       spacestring$ = "    "
```

```
                     ElseIf (TLEN < 100000) Then
                        spacestring$ = "    "
                     Else
                        spacestring$ = "  "
                     End If
                     Print #2, TLEN & spacestring$ & P1M & " " & E1 & " " &
xstartpos & " " & ystartpos & " " & zstartpos & " " & Format(x1, "0.0000") & "
" & Format(y1, "0.0000") & " " & Format(z1, "0.0000")
                     Print #3, TLEN & spacestring$ & P2M & " " & E2 & " " &
xstartpos & " " & ystartpos & " " & zstartpos & " " & Format(x2, "0.0000") & "
" & Format(y2, "0.0000") & " " & Format(z2, "0.0000")
                     If (SaveReactionLocations = True) Then
                        'Convert the x location to microns and the y and z to
cm and save them
                        Print #6, (xstartpos / 10000#) & Chr$(9) & (ystartpos /
100000000#) & Chr$(9) & (zstartpos / 100000000#)
                     End If
                  Next i
               End If
            End If

            'Insert an end-of-file character at the end of each file (this
*must* be done) and then
            'close it.
            Print #2, Chr$(26)
            Print #3, Chr$(26)
            Close #2
            Close #3
            If (SaveReactionLocations = True) Then
               Print #6, Chr$(26)
               Close #6
            End If
            If (SecondTargetLayer > 0) Then
               Print #4, Chr$(26)
               Print #5, Chr$(26)
               Close #4
               Close #5
               If (SaveReactionLocations = True) Then
                  Print #7, Chr$(26)
                  Close #7
               End If
            End If

            DoEvents

            If (LayerMaterial(TargetLayer) = 1) Then
               MaximumPossibleEnergy = 2790000 + ENeutron
            ElseIf (LayerMaterial(TargetLayer) = 2 Or
LayerMaterial(TargetLayer) = 7) Then
               MaximumPossibleEnergy = 4780000 + ENeutron
            End If

            'Run the first target layer in the usual way.
            Call TrimAutomation.NewRunCase
            DoEvents
            Call AnalyzeTrimData.AnalyzeTrimData

            'Put the files from the first layer in the OutputFiles folder so
they won't be lost,
            'renaming them along the way and deleting the old copies.
            If Dir$(path$ & "TOTAL_ENERGY_DEPOSITION.TXT") = "" Then
               'The file does not exist.
            Else
               FileCopy (path$ & "TOTAL_ENERGY_DEPOSITION.TXT"), (copypath$ &
"1TOTAL_ENERGY_DEPOSITION.TXT")
               Kill (path$ & "TOTAL_ENERGY_DEPOSITION.TXT")
            End If
            If Dir$(path$ & "BINNED_ENERGY_DEPOSITION.TXT") = "" Then
               'The file does not exist.
            Else
               FileCopy (path$ & "BINNED_ENERGY_DEPOSITION.TXT"), (copypath$ &
"1BINNED_ENERGY_DEPOSITION.TXT")
               Kill (path$ & "BINNED_ENERGY_DEPOSITION.TXT")
            End If
            For i = 1 To 3 Step 1
               If Dir$(path$ & "BINNED_ENERGY_DEPOSITION-" & i & ".TXT") = ""
Then
```

```
                'The file does not exist.
            Else
                FileCopy (path$ & "BINNED_ENERGY_DEPOSITION-" & i & ".TXT"),
(copypath$ & "1BINNED_ENERGY_DEPOSITION-" & i & ".TXT")
                Kill (path$ & "BINNED_ENERGY_DEPOSITION-" & i & ".TXT")
            End If
        Next i
        If Dir$(path$ & "QFCS_DATA.TXT") = "" Then
            'The file does not exist.
        Else
            FileCopy (path$ & "QFCS_DATA.TXT"), (copypath$ &
"1QFCS_DATA.TXT")
            Kill (path$ & "QFCS_DATA.TXT")
        End If
        For i = 1 To 3 Step 1
            If Dir$(path$ & "QFCS_DATA_" & i & ".TXT") = "" Then
                'The file does not exist.
            Else
                FileCopy (path$ & "QFCS_DATA_" & i & ".TXT"), (copypath$ &
"1QFCS_DATA_" & i & ".TXT")
                Kill (path$ & "QFCS_DATA_" & i & ".TXT")
            End If
        Next i
        If (TL1EventNum > 0 And SaveEnergyDeposition = True) Then
            For i = 1 To TL1EventNum Step 1
                If (Dir$(path$ & "ANODE_STRIP_DEPOSITION_" & i & ".TXT") =
"") Then
                    'The file does not exist.
                Else
                    FileCopy (path$ & "ANODE_STRIP_DEPOSITION_" & i & ".TXT"),
(copypath$ & "1ANODE_STRIP_DEPOSITION_" & i & ".TXT")
                    Kill (path$ & "ANODE_STRIP_DEPOSITION_" & i & ".TXT")
                End If
            Next i
        End If
        If (SaveReactionLocations = True) Then
            If (Dir$(path$ & "\OutputFiles\1REACTION_LOCATIONS.TXT") = "")
Then
                'The file does not exist.
            Else
                FileCopy (path$ & "\OutputFiles\1REACTION_LOCATIONS.TXT"),
(copypath$ & "1REACTION_LOCATIONS.TXT")
                Kill (path$ & "\OutputFiles\1REACTION_LOCATIONS.TXT")
            End If
        End If

        'IMPORTANT NOTE:
        'The code pertaining to handling a second layer (if one exists) has
not been
        'fixed yet!

        If (SecondTargetLayer > 0) Then
            'Delete the old input files from the first layer so the input
files for the
            'second layer can be assigned the same name.
            If (Dir$(path$ & "TRIM1.DAT") = "") Then
                'File doesn't exist.
            Else
                Kill (path$ & "TRIM1.DAT")
            End If
            If (Dir$(path$ & "TRIM2.DAT") = "") Then
                'File doesn't exist.
            Else
                Kill (path$ & "TRIM2.DAT")
            End If

            'Next, copy the input files for the second target layer over to
the correct names.
            If (Dir$(path$ & "2TRIM1.DAT") = "") Then
                'File doesn't exist.
            Else
                Name (path$ & "2TRIM1.DAT") As (path$ & "TRIM1.DAT")
            End If
            If (Dir$(path$ & "2TRIM2.DAT") = "") Then
                'File doesn't exist.
            Else
                Name (path$ & "2TRIM2.DAT") As (path$ & "TRIM2.DAT")
```

```
                End If

                DoEvents

                'Change any variables that need changing.
                TargetLayer = SecondTargetLayer
                If (LayerMaterial(TargetLayer) = 1) Then
                   MaximumPossibleEnergy = 2790000 + ENeutron
                ElseIf (LayerMaterial(TargetLayer) = 2 Or
LayerMaterial(TargetLayer) = 7) Then
                   MaximumPossibleEnergy = 4780000 + ENeutron
                End If

                'Now run the second target layer.
                Call TrimAutomation.NewRunCase
                DoEvents
                Call AnalyzeTrimData.AnalyzeTrimData

                'Rename the output files appropriately if they exist.
                If Dir$(path$ & "TOTAL_ENERGY_DEPOSITION.TXT") = "" Then
                   'The file does not exist.
                Else
                   FileCopy (path$ & "TOTAL_ENERGY_DEPOSITION.TXT"), (copypath$
& "2TOTAL_ENERGY_DEPOSITION.TXT")
                   Kill (path$ & "TOTAL_ENERGY_DEPOSITION.TXT")
                End If
                If Dir$(path$ & "BINNED_ENERGY_DEPOSITION.TXT") = "" Then
                   'The file does not exist.
                Else
                   FileCopy (path$ & "BINNED_ENERGY_DEPOSITION.TXT"), (copypath$
& "2BINNED_ENERGY_DEPOSITION.TXT")
                   Kill (path$ & "BINNED_ENERGY_DEPOSITION.TXT")
                End If
                For i = 1 To 3 Step 1
                   If Dir$(path$ & "BINNED_ENERGY_DEPOSITION-" & i & ".TXT") =
"" Then
                      'The file does not exist.
                   Else
                      FileCopy (path$ & "BINNED_ENERGY_DEPOSITION-" & i &
".TXT"), (copypath$ & "2BINNED_ENERGY_DEPOSITION-" & i & ".TXT")
                      Kill (path$ & "BINNED_ENERGY_DEPOSITION-" & i & ".TXT")
                   End If
                Next i
                If Dir$(path$ & "QFCS_DATA.TXT") = "" Then
                   'The file does not exist.
                Else
                   FileCopy (path$ & "QFCS_DATA.TXT"), (copypath$ &
"2QFCS_DATA.TXT")
                   Kill (path$ & "QFCS_DATA.TXT")
                End If
                For i = 1 To 3 Step 1
                   If Dir$(path$ & "QFCS_DATA_" & i & ".TXT") = "" Then
                      'The file does not exist.
                   Else
                      FileCopy (path$ & "QFCS_DATA_" & i & ".TXT"), (copypath$ &
"2QFCS_DATA_" & i & ".TXT")
                      Kill (path$ & "QFCS_DATA_" & i & ".TXT")
                   End If
                Next i
                If (TL2EventNum > 0 And
MicromegasSim.checkSaveEnergyDeposition.Value = 1) Then
                   For i = 1 To TL2EventNum Step 1
                      If (Dir$(path$ & "ANODE_STRIP_DEPOSITION_" & i & ".TXT") =
"") Then
                         'The file does not exist.
                      Else
                         FileCopy (path$ & "ANODE_STRIP_DEPOSITION_" & i &
".TXT"), (copypath$ & "2ANODE_STRIP_DEPOSITION_" & i & ".TXT")
                         Kill (path$ & "ANODE_STRIP_DEPOSITION_" & i & ".TXT")
                      End If
                   Next i
                End If
             End If

             'Rename the reaction location file
```

```
            If
Dir$("C:\Andrew\MicromegasSim\CurrentResults\1REACTION_LOCATIONS.TXT") <> ""
Then
               FileCopy
("C:\Andrew\MicromegasSim\CurrentResults\1REACTION_LOCATIONS.TXT"), _
               ("C:\Andrew\MicromegasSim\CurrentResults\ReactionLocations_" &
SingleTargetStripNumber _
               & "_" & FixedYTargetPoint & ".txt")
               Kill
("C:\Andrew\MicromegasSim\CurrentResults\1REACTION_LOCATIONS.TXT")
            End If

            'Copy over and rename the electrons on the strip file
            If Dir$("C:\Program Files\SRIM
2000\ElectronsOnSingleTargetStrip.txt") <> "" Then
               FileCopy ("C:\Program Files\SRIM
2000\ElectronsOnSingleTargetStrip.txt"), _

("C:\Andrew\MicromegasSim\CurrentResults\ElectronsOnTargetStrip_" &
SingleTargetStripNumber _
               & "_" & FixedYTargetPoint & ".txt")
               Kill ("C:\Program Files\SRIM
2000\ElectronsOnSingleTargetStrip.txt")
            End If

         End If

      Next iv

   Next iii

   'We've run all the cases.  Go through files, see which ones are present, and
collect all
   'appropriate data for each different strip and put it all into one file for
importation
   'into MS Excel.

   For iii = 1 To 25 Step 1 '25 Step 1 ' strip being looked at
      Open "C:\Andrew\MicromegasSim\CurrentResults\Electrons_On_Strip_" & iii _
       & ".txt" For Output As #507

      ReDim FileExists(199)
      ReDim FileOpen(199)
      ReDim FileEOF(199)

      For iv = 1 To 199 Step 1 '199 Step 1 ' discrete starting location being
looked at
         FixedYTargetPoint = iv * 0.025
         inp_num = 100 + iv
         If
Dir$("C:\Andrew\MicromegasSim\CurrentResults\ElectronsOnTargetStrip_" & iii _
          & "_" & FixedYTargetPoint & ".txt") <> "" Then
            FileExists(iv) = True
            Open
"C:\Andrew\MicromegasSim\CurrentResults\ElectronsOnTargetStrip_" & iii _
             & "_" & FixedYTargetPoint & ".txt" For Input As #inp_num
            If EOF(inp_num) = True Then
               FileEOF(inp_num) = True
            End If
         Else
            FileExists(iv) = False
         End If
      Next iv

      StripStart = CSng(iii - 1) * 0.1
      StripCenter = StripStart + 0.05
      StripEnd = StripStart + 0.1

      Print #507, "Micromegas Simulation"
      Print #507, ""
      Print #507, "Electrons on single strip number " & iii & "."
      Print #507, "Strip start = " & StripStart & " cm, strip center = " &
StripCenter _
       & " cm, strip end = " & StripEnd & " cm"
      Print #507, ""
      Print #507, "Target locations in y direction follow in units of cm."
```

```
      Print #507, "Number of electrons in individual events are below that in
columns."
      Print #507, ""
      Print #507, ""
      Print #507, ""

      WriteString = ""

      For iv = 1 To 199 Step 1
         FixedYTargetPoint = iv * 0.025
         If FileExists(iv) = True Then
            WriteString = WriteString & FixedYTargetPoint & " cm" & Chr$(9)
         End If
      Next iv

      Print #507, WriteString
      WriteString = ""
      Print #507, ""

      'QuitLoopFlag = False
      AnyFilesNotEOF = True

      Do While (AnyFilesNotEOF = True) '(QuitLoopFlag = False)
         AnyFilesNotEOF = False
         For iv = 1 To 199 Step 1
            If FileExists(iv) = True Then
               If FileEOF(iv) = False Then
                  inp_num = 100 + iv
                  Input #inp_num, numelectrons
                  WriteString = WriteString & numelectrons & Chr$(9)
                  If EOF(inp_num) = True Then
                     FileEOF(iv) = True
                  Else
                     AnyFilesNotEOF = True
                  End If
               Else
                  WriteString = WriteString & Chr$(9)
               End If
            End If
         Next iv

         Print #507, WriteString
         WriteString = ""
      Loop

      For iv = 1 To 199 Step 1
         If FileExists(iv) = True Then
            inp_num = 100 + iv
            Close #inp_num
         End If
      Next iv

      Close #507
   Next iii

   'Now get rid of all unnecessary data files - remove if needed
   For iii = 1 To 25 Step 1
      For iv = 1 To 199 Step 1
         FixedYTargetPoint = iv * 0.025
         If
Dir$("C:\Andrew\MicromegasSim\CurrentResults\ElectronsOnTargetStrip_" & iii _
          & "_" & FixedYTargetPoint & ".txt") <> "" Then
            Kill
("C:\Andrew\MicromegasSim\CurrentResults\ElectronsOnTargetStrip_" & iii _
           & "_" & FixedYTargetPoint & ".txt")
         End If
      Next iv
   Next iii


'can do a select case
'      GasDensity = 0.00175
'      DiffusionCoefficient = 370# '200#
'      Select Case iii
'      Case 1:
'         LayerThickness(1) = 50# * 10000#
'         LayerThickness(2) = 2# * 10000#
```

```
'         LayerThickness(3) = 3000# * 10000# '3000
'         LayerThickness(4) = 50# * 10000#
'         '1 is B-10, 2 is Li-6, 3 is Al, 5 is Ar/isobutane
'         LayerMaterial(1) = 3
'         LayerMaterial(2) = 1
'         LayerMaterial(3) = 5
'         LayerMaterial(4) = 3
'         NumberOfRuns = 100
'         SaveReactionLocations = True
'         GeneratePulse = True
'         SaveAnodeSignals = False
'         SaveQFCSToExcel = False
'         SaveQFCS = False
'         SaveEnergyDeposition = False
'         copypath$ = "C:\Andrew\MicromegasSim\CurrentResults\"
'
'         SingleTargetStrip = True
'         SingleTargetStripNumber = 25
'
'         FixedYTarget = False
'         FixedYTargetPoint = 2.5
'         FixedZTarget = True
'         FixedZTargetPoint = 2.5
''end of case
'
'      Case 2:
'         LayerThickness(1) = 50 * 10000#
'         LayerThickness(2) = 0.5 * 10000#
'         LayerThickness(3) = 2500 * 10000#
'         LayerThickness(4) = 50 * 10000#
'         '1 is B-10, 2 is Li-6, 3 is Al, 5 is Ar/isobutane
'         LayerMaterial(1) = 3
'         LayerMaterial(2) = 2
'         LayerMaterial(3) = 5
'         LayerMaterial(4) = 3
'         NumberOfRuns = 1000
'         SaveReactionLocations = True
'         GeneratePulse = True
'         SaveAnodeSignals = True
'         SaveQFCSToExcel = False
'         SaveQFCS = True
'         SaveEnergyDeposition = True
'         copypath$ =
"C:\AndrewStephan\SNSInstruments\Simulations\MicromegasSim\Results-
FirstSimulationPaper\0.5umLi6-2500umdriftgap\"
''end of case
''end select

   End

End Sub

Public Sub RunInputFileFromCode()
   'Routine for running a simulation case based on incoming neutrons whose
   'information (e.g. location) is in a file.  This feature is not yet
finished.
   'It is currently set to simulate a semiconductor.  After creating the code,
   'I found that it could be very easily modified to simulate a semiconductor
   'with adjacent Li-6 or B-10 converter material with the semiconductor taking
   'the place of the gas.  I briefly did some calculations and compared them
   'with data I had taken some time back (a few years ago) from an SBD with
   'an adjacent boronated and lithiated glass layer.  The good agreement I saw
   'helped to give me confidence the simulation was working correctly.
   Dim i As Long
   Dim ii As Long
   Dim iii As Long
   Dim temp$
   Dim ENeutron As Single
   Dim xstartpos As Long
   Dim ystartpos As Long
   Dim zstartpos As Long
   Dim spacestring$
   Dim combostring$
   Dim currentrnd As Single
   Dim secondrnd As Single
   Dim ratio As Single
   Dim opticallength As Single
```

```
Dim opticallength1 As Single
Dim opticallength2 As Single
Dim TravelDistance As Single
Dim TL1path As Single
Dim TL2path As Single
Dim probability1 As Single
Dim probability2 As Single
Dim totalprobability As Single
Dim TargetLayerInteractedIn As Long
Dim TLEN As Long

Dim nxdir As Single
Dim nydir As Single
Dim nzdir As Single
Dim normalizer As Single

Dim copypath$

Dim weight As Single
Dim maxENeutron As Single

AllowTransmissions = False
ReducedWeight = True
SemiconductorVariance = False
IncludeReactionProductInfo = True
'nxdir = 1#
'nydir = 0#
'nzdir = 0#
'If (nxdir <= 0#) Then
'   MsgBox "The x direction vector must be positive.", , "Error:"
'   End
'End If
'normalizer = (nxdir ^ 2# + nydir ^ 2# + nzdir ^ 2#)
'normalizer = normalizer ^ 0.5
'If (normalizer <> 1#) Then
'   nxdir = nxdir / normalizer
'   nydir = nydir / normalizer
'   nzdir = nzdir / normalizer
'End If
'eVperIonization = 22.4
'AnodeGranularity = 317.5
'DiffusionCoefficient = 370#
'GasGain = 100
'FittingRatio = 1000
YDimension = 10# * 10000#
ZDimension = 10# * 10000#
EnergyBinSize = 10# * 1000#
'ENeutron = 0.025

SaveReactionLocations = False
GeneratePulse = False
NumberOfLayers = 2
ReDim LayerThickness(NumberOfLayers)
ReDim LayerMaterial(NumberOfLayers)
TargetLayer = 1
SecondTargetLayer = -1
LayerOfInterest = 2

eVperIonization = 4#
SemiconductorVariance = True

LayerThickness(1) = 50 * 10000#
LayerThickness(2) = 300 * 10000#
LayerMaterial(1) = 2 '11 'Li-doped sol-gel glass
LayerMaterial(2) = 8 'Si
SaveEnergyDeposition = True
copypath$ = "C:\Andrew\MicromegasSim\NeutronInputFileOutput\"

NumberOfRuns = 0

DoEvents

Open "C:\Andrew\MicromegasSim\IncomingNeutrons.txt" For Input As #511
'Create a TRIM.DAT file for each type of reaction product.
Open path$ & "OLDTRIM.DAT" For Input As #1
'Create one TRIM.DAT type of file for each type of reaction product.
Open path$ & "TRIM1.DAT" For Output As #2
```

```
   Open path$ & "TRIM2.DAT" For Output As #3
   If (SaveReactionLocations = True) Then
      Open path$ & "OutputFiles/1REACTION_LOCATIONS.TXT" For Output As #6
   End If

   'Insert boilerplate at the start of the file.
   For i = 1 To 10 Step 1
      Line Input #1, temp$
      Print #2, temp$
      Print #3, temp$
   Next i

   Open path$ & "PARTICLE_WEIGHTS.TXT" For Output As #510

   ystartpos = 0#
   zstartpos = 0#

   Do While EOF(511) = False

      Input #511, xstartpos, ystartpos, zstartpos, nxdir, nydir, nzdir,
ENeutron

      NumberOfRuns = NumberOfRuns + 1

      If (ENeutron <= 0#) Then
         MsgBox "Neutron energy must be greater than zero.", , "Error:"
      End If

      If (ENeutron > maxENeutron) Then
         maxENeutron = ENeutron
      End If

      normalizer = (nxdir ^ 2# + nydir ^ 2# + nzdir ^ 2#)
      normalizer = normalizer ^ 0.5
      If (normalizer <> 1#) Then
         nxdir = nxdir / normalizer
         nydir = nydir / normalizer
         nzdir = nzdir / normalizer
      End If

      'Note: As of right now, don't worry about neutrons or reaction products
exiting through the sides of the
      'detector.

      opticallength1 = MacroscopicXSection(ENeutron,
LayerMaterial(TargetLayer)) * (LayerThickness(TargetLayer) / nxdir) *
(0.00000001)
      opticallength = opticallength1
      TL1path = LayerThickness(TargetLayer) / nxdir

      'There is only one layer of neutron target material.
      currentrnd = Rnd
      probability1 = 1 - ee ^ -opticallength1
      weight = probability1
      ratio = 1# / probability1
      TravelDistance = -Log(1 - currentrnd / ratio)
      Call CalculateAllNeutronReactionInfo(ENeutron,
LayerMaterial(TargetLayer), nxdir, nydir, nzdir)
      xstartpos = (TravelDistance / opticallength1) *
LayerThickness(TargetLayer)
      TargetLayerInteractedIn = 1
      TL1EventNum = TL1EventNum + 1
      TLEN = TL1EventNum

      spacestring$ = ""
      If (TLEN < 10) Then
         spacestring$ = "       "
      ElseIf (TLEN < 100) Then
         spacestring$ = "      "
      ElseIf (TLEN < 1000) Then
         spacestring$ = "     "
      ElseIf (TLEN < 10000) Then
         spacestring$ = "    "
      ElseIf (TLEN < 100000) Then
         spacestring$ = "   "
      Else
         spacestring$ = "  "
```

```
      End If
      Print #2, TLEN & spacestring$ & P1M & " " & E1 & " " & xstartpos & " " &
ystartpos & " " & zstartpos & " " & Format(x1, "0.0000") & " " & Format(y1,
"0.0000") & " " & Format(z1, "0.0000")
      Print #3, TLEN & spacestring$ & P2M & " " & E2 & " " & xstartpos & " " &
ystartpos & " " & zstartpos & " " & Format(x2, "0.0000") & " " & Format(y2,
"0.0000") & " " & Format(z2, "0.0000")

      Print #510, TL1EventNum & " " & weight

   Loop

   Print #2, Chr$(26)
   Print #3, Chr$(26)
   Print #510, Chr$(26)
   Close #2
   Close #3

   Close #1
   Close #510
   Close #511

   If (LayerMaterial(TargetLayer) = 1 Or LayerMaterial(TargetLayer) = 10) Then
      MaximumPossibleEnergy = 2790000 + maxENeutron
   ElseIf (LayerMaterial(TargetLayer) = 2 Or LayerMaterial(TargetLayer) = 7 Or
LayerMaterial(TargetLayer) = 11) Then
      MaximumPossibleEnergy = 4780000 + maxENeutron
   End If

   DoEvents

   'Read back in the particle weights from the weight file now that we now how
many events there are.

   i = 0
   ReDim ParticleWeights(NumberOfRuns)

   Open path$ & "PARTICLE_WEIGHTS.TXT" For Input As #510

   Do While EOF(510) = False
      Input #510, ii, weight
      i = i + 1
      ParticleWeights(i) = weight
   Loop

   Close #510

   'Run the first target layer in the usual way.
   Call TrimAutomation.NewRunCase
   DoEvents
   Call AnalyzeTrimData.AnalyzeTrimData

   'Put the files from the first layer in the OutputFiles folder so they won't
be lost,
   'renaming them along the way and deleting the old copies.
   If Dir$(path$ & "TOTAL_ENERGY_DEPOSITION.TXT") = "" Then
      'The file does not exist.
   Else
      FileCopy (path$ & "TOTAL_ENERGY_DEPOSITION.TXT"), (copypath$ &
"1TOTAL_ENERGY_DEPOSITION.TXT")
      Kill (path$ & "TOTAL_ENERGY_DEPOSITION.TXT")
   End If
   If Dir$(path$ & "BINNED_ENERGY_DEPOSITION.TXT") = "" Then
      'The file does not exist.
   Else
      FileCopy (path$ & "BINNED_ENERGY_DEPOSITION.TXT"), (copypath$ &
"1BINNED_ENERGY_DEPOSITION.TXT")
      Kill (path$ & "BINNED_ENERGY_DEPOSITION.TXT")
   End If
   For i = 1 To 3 Step 1
      If Dir$(path$ & "BINNED_ENERGY_DEPOSITION-" & i & ".TXT") = "" Then
         'The file does not exist.
      Else
         FileCopy (path$ & "BINNED_ENERGY_DEPOSITION-" & i & ".TXT"),
(copypath$ & "1BINNED_ENERGY_DEPOSITION-" & i & ".TXT")
         Kill (path$ & "BINNED_ENERGY_DEPOSITION-" & i & ".TXT")
      End If
```

```
   Next i

   End

End Sub

'*****************************************************************************
'*                                                                           *
'*                 Neutron Absorption Reaction Simulation Code               *
'*                                                                           *
'*****************************************************************************

Public Sub CalculateAllNeutronReactionInfo(ByVal ENeutron As Single, ByVal
TargetLayerMaterial As Long, nxdir As Single, nydir As Single, nzdir As Single)
   'This sub is intended to cover all aspects of the reaction product creation.

   'The code below has been pasted in from other places and has not yet been
altered as
   'necessary.
   Dim sumofsquares As Single
   Dim sqrofsumofsquares As Single
   Dim flag As Boolean
   Dim xdir As Single
   Dim ydir As Single
   Dim zdir As Single
   Dim Va As Single 'velocity of neutron (particle "a" in reaction)
   Dim Vax As Single
   Dim Vay As Single
   Dim Vaz As Single
   Dim Vb As Single 'velocity of target atom (particle "b" in reaction)
   Dim Vbx As Single
   Dim Vby As Single
   Dim Vbz As Single
   Dim Q As Single 'in MeV
   Dim Vcom As Single 'velocity of center-of-mass system relative to lab system
   Dim Vcomx As Single
   Dim Vcomy As Single
   Dim Vcomz As Single
   Dim TM As Single 'target mass in amu
   Dim Vc As Single 'velocity of first reaction product
   Dim Vcx As Single
   Dim Vcy As Single
   Dim Vcz As Single
   Dim Vd As Single 'velocity of first reaction product
   Dim Vdx As Single
   Dim Vdy As Single
   Dim Vdz As Single
   Dim Ecom As Single 'energy of neutron and target in COM system prior to
   'reaction

   If (TargetLayerMaterial = 1 Or TargetLayerMaterial = 10) Then
      P1 = 2
      P1M = 4.002
      P1Z = 2
      P2 = 3
      P2M = 7.014
      P2Z = 3
   ElseIf (TargetLayerMaterial = 2 Or TargetLayerMaterial = 7 Or
TargetLayerMaterial = 11) Then
      P1 = 1
      P1M = 3.016
      P1Z = 1
      P2 = 2
      P2M = 4.002
      P2Z = 2
   Else
      MsgBox "Unrecognized neutron target type.", , "Error:"
      End
   End If

   'Switch neutron energy from eV to MeV.
   ENeutron = ENeutron / 1000000#

   'Get particle types and energies.
   If (TargetLayerMaterial = 1 Or TargetLayerMaterial = 10) Then
      If (Rnd < 0.92) Then
         E1 = 1.47
```

```
         E2 = 0.84
         Q = 2.31
      Else
         E1 = 1.777
         E2 = 1.015
         Q = 2.792
      End If
      TM = 10.007
   ElseIf (TargetLayerMaterial = 2 Or TargetLayerMaterial = 7 Or
TargetLayerMaterial = 11) Then
      E1 = 2.73
      E2 = 2.05
      Q = 4.78
      TM = 6.009
   Else
      MsgBox "Unrecognized neutron target type.", , "Error:"
      End
   End If

   flag = False

   'Returns the x, y, and z components of a randomly directed unit vector.
   Do While (flag = False)
      xdir = Rnd * 2 - 1
      ydir = Rnd * 2 - 1
      zdir = Rnd * 2 - 1

      sumofsquares = xdir ^ 2 + ydir ^ 2 + zdir ^ 2
      sqrofsumofsquares = sumofsquares ^ 0.5

      If (sumofsquares <= 1) And (sumofsquares > 0) Then
         'Enter if the point in inside the sphere.
         'Normalize to a unit vector.
         xdir = xdir / sqrofsumofsquares
         ydir = ydir / sqrofsumofsquares
         zdir = zdir / sqrofsumofsquares
         flag = True
      End If
   Loop

   'Randomly select which one gets which set of directions.  This is done as a
   'precaution to help keep the directions totally random.
   If (Rnd < 0.5) Then
      x1 = xdir
      y1 = ydir
      z1 = zdir
      x2 = -xdir
      y2 = -ydir
      z2 = -zdir
   Else
      x1 = -xdir
      y1 = -ydir
      z1 = -zdir
      x2 = xdir
      y2 = ydir
      z2 = zdir
   End If

   'Determine the initial speeds of the incident particle and target and the
closing
   'speed in the lab frame as a fraction of c, the speed of light.

   Va = (2# * ENeutron / (1.008664 * 931.5)) ^ 0.5
   Vax = Va * nxdir
   Vay = Va * nydir
   Vaz = Va * nzdir

   Vb = 0#
   Vbx = 0#
   Vby = 0#
   Vbz = 0#

   Vcom = (1.008664 * Va) / (1.008664 + TM)
   Vcomx = Vcom * nxdir
   Vcomy = Vcom * nydir
   Vcomz = Vcom * nzdir
```

```
   'Va = Va - Vcom 'in the x direction

   Vax = Vax - Vcomx
   Vay = Vay - Vcomy
   Vaz = Vaz - Vcomz
   Va = (Vax ^ 2# + Vay ^ 2# + Vaz ^ 2#) ^ 0.5

   'Vb = Vb + Vcom 'in the minus x direction

   Vbx = Vbx - Vcomx ' +
   Vby = Vby - Vcomy ' +
   Vbz = Vbz - Vcomz ' +
   Vb = (Vbx ^ 2# + Vby ^ 2# + Vbz ^ 2#) ^ 0.5

   'Find the total kinetic energy in the COM system and add the Q value.
   Ecom = (0.5 * 1.008664 * 931.5 * (Va ^ 2)) + (0.5 * TM * 931.5 * (Vb ^ 2)) +
Q

   'Find the velocities of the two reaction products in the COM system.  (The
directions
   'don't matter at this point.)
   Vd = ((2# * Ecom * P1M) / (931.5 * (P2M ^ 2 + P1M * P2M))) ^ 0.5
   Vc = (P2M / P1M) * Vd

   E1 = (0.5 * P1M * 931.5 * (Vc ^ 2))
   E2 = (0.5 * P2M * 931.5 * (Vd ^ 2))

   'Find the individual directional components and add the COM speed.
   Vcx = Vc * x1
   Vcy = Vc * y1
   Vcz = Vc * z1
   Vdx = Vd * x2
   Vdy = Vd * y2
   Vdz = Vd * z2

   Vcx = Vcx + Vcomx
   Vcy = Vcy + Vcomy
   Vcz = Vcz + Vcomz
   Vdx = Vdx + Vcomx
   Vdy = Vdy + Vcomy
   Vdz = Vdz + Vcomz

   'Calculate the total velocities.
   Vc = (Vcx ^ 2 + Vcy ^ 2 + Vcz ^ 2) ^ 0.5
   Vd = (Vdx ^ 2 + Vdy ^ 2 + Vdz ^ 2) ^ 0.5

   'Calculate the new direction unit vectors.
   x1 = Vcx / Vc
   y1 = Vcy / Vc
   z1 = Vcz / Vc
   x2 = Vdx / Vd
   y2 = Vdy / Vd
   z2 = Vdz / Vd

   'Calculate the new energies.
   E1 = (0.5 * P1M * 931.5 * (Vc ^ 2))
   E2 = (0.5 * P2M * 931.5 * (Vd ^ 2))

   'Convert the energies to eV.
   E1 = E1 * 1000000#
   E2 = E2 * 1000000#

End Sub

'*****************************************************************************
'*                                                                           *
'*                       Math-Related Subs and Functions                     *
'*                                                                           *
'*****************************************************************************

Public Sub ReadInXSections()
   'Reads in microscopic cross sections from the microscopic cross section
files.
   Dim energy As Single
   Dim xsection As Single
   Dim i As Long
```

```
   Open "C:\Andrew\MicromegasSim\B-10xsections.txt" For Input As #1
   For i = 1 To 673 Step 1
      Input #1, energy, xsection
      B10XSection(1, i) = energy * 1000000# 'convert to eV from MeV
      B10XSection(2, i) = xsection * 1E-24 'convert to cm^2 from barns
   Next i
   Close #1

   Open "C:\Andrew\MicromegasSim\Li-6xsections.txt" For Input As #1
   For i = 1 To 498 Step 1
      Input #1, energy, xsection
      Li6XSection(1, i) = energy * 1000000#
      Li6XSection(2, i) = xsection * 1E-24
   Next i
   Close #1

   Open "C:\Andrew\MicromegasSim\He-3xsections.txt" For Input As #1
   For i = 1 To 342 Step 1
      Input #1, energy, xsection
      He3XSection(1, i) = energy * 1000000#
      He3XSection(2, i) = xsection * 1E-24
   Next i
   Close #1

End Sub

Public Sub AdjustCrossSections()
   'Adjust neutron absorption cross sections using enrichment specified by user
   Dim energy As Single
   Dim xsection As Single
   Dim i As Long

   For i = 1 To 673 Step 1
      B10XSection(2, i) = B10Enrichment * B10XSection(2, i)
   Next i

   For i = 1 To 498 Step 1
      Li6XSection(2, i) = Li6Enrichment * Li6XSection(2, i)
   Next i

   For i = 1 To 342 Step 1
      He3XSection(2, i) = He3Enrichment * He3XSection(2, i)
   Next i
End Sub

Public Function MacroscopicXSection(ENeutron As Single, MaterialType As Long)
   'Returns macroscopic neutron absorption cross sections for arbitrary neutron
   'energies for the various neutron target materials.
   Dim TargetAtomsPerCC As Single
   Dim MicroscopicXSection As Single
   Dim TargetAtomType As Long '1 is B-10, 2 is Li-6, 3 is He-3
   Dim currenergy As Single
   Dim prevenergy As Single
   Dim currxsection As Single
   Dim prevxsection As Single
   Dim i As Long

   Select Case MaterialType
   Case 1
      TargetAtomType = 1
      TargetAtomsPerCC = 1.309E+23
   Case 2
      TargetAtomType = 2
      TargetAtomsPerCC = 4.633E+22
   Case 7
      TargetAtomType = 2
      TargetAtomsPerCC = 2.001E+22
   Case 10
      TargetAtomType = 1
      TargetAtomsPerCC = 2.618E+22
   Case 11
      TargetAtomType = 2
      TargetAtomsPerCC = 9.266E+21
   End Select

   i = 0
   If (TargetAtomType = 1) Then
```

```
      'Target atom type is B-10
      i = 1
      currenergy = B10XSection(1, i)
      currxsection = B10XSection(2, i)
      If (ENeutron < currenergy) Then
         'Note: we could cause it to use the cross section of the minimum
energy supported even
         'if its energy is below this.  This would be an extremely simple
change to make.
         MsgBox "Neutron energy below supported range.", , "Error:"
         End
      End If
      Do While (currenergy < ENeutron And i < 673)
         i = i + 1
         prevenergy = currenergy
         prevxsection = currxsection
         currenergy = B10XSection(1, i)
         currxsection = B10XSection(2, i)
      Loop
      If (ENeutron > currenergy) Then
         'Note: we could cause it to use the cross section of the maximum
energy supported even
         'if its energy is above this.  This would be an extremely simple
change to make.
         MsgBox "Neutron energy above supported range.", , "Error:"
         End
      End If
      MicroscopicXSection = prevxsection + (currxsection - prevxsection) *
((ENeutron - prevenergy) / (currenergy - prevenergy))
   ElseIf (TargetAtomType = 2) Then
      'Target atom type is Li-6
      i = 1
      currenergy = Li6XSection(1, i)
      currxsection = Li6XSection(2, i)
      If (ENeutron < currenergy) Then
         'Note: we could cause it to use the cross section of the minimum
energy supported even
         'if its energy is below this.  This would be an extremely simple
change to make.
         MsgBox "Neutron energy below supported range.", , "Error:"
         End
      End If
      Do While (currenergy < ENeutron And i < 498)
         i = i + 1
         prevenergy = currenergy
         prevxsection = currxsection
         currenergy = Li6XSection(1, i)
         currxsection = Li6XSection(2, i)
      Loop
      If (ENeutron > currenergy) Then
         'Note: we could cause it to use the cross section of the maximum
energy supported even
         'if its energy is above this.  This would be an extremely simple
change to make.
         MsgBox "Neutron energy above supported range.", , "Error:"
         End
      End If
      MicroscopicXSection = prevxsection + (currxsection - prevxsection) *
((ENeutron - prevenergy) / (currenergy - prevenergy))
   ElseIf (TargetAtomType = 3) Then
      'Target atom type is He-3
      i = 1
      currenergy = He3XSection(1, i)
      currxsection = He3XSection(2, i)
      If (ENeutron < currenergy) Then
         'Note: we could cause it to use the cross section of the minimum
energy supported even
         'if its energy is below this.  This would be an extremely simple
change to make.
         MsgBox "Neutron energy below supported range.", , "Error:"
         End
      End If
      Do While (currenergy < ENeutron And i < 342)
         i = i + 1
         prevenergy = currenergy
         prevxsection = currxsection
         currenergy = He3XSection(1, i)
```

```
         currxsection = He3XSection(2, i)
      Loop
      If (ENeutron > currenergy) Then
         'Note: we could cause it to use the cross section of the maximum
energy supported even
         'if its energy is above this.  This would be an extremely simple
change to make.
         MsgBox "Neutron energy above supported range.", , "Error:"
         End
      End If
      MicroscopicXSection = prevxsection + (currxsection - prevxsection) *
((ENeutron - prevenergy) / (currenergy - prevenergy))
   Else
      MsgBox "Unknown material target type.", , "Error:"
   End If
   MacroscopicXSection = MicroscopicXSection * TargetAtomsPerCC

End Function

'*****************************************************************************
'*                                                                           *
'*                       Math-Related Subs and Functions                     *
'*                                                                           *
'*****************************************************************************

Public Function GetRandomNormal()
   'Returns a random number selected from a normal distribution.
   Dim fac As Double
   Dim r As Double
   Dim v1 As Double
   Dim v2 As Double
   Dim flag As Boolean

   flag = False

   Do While (flag = False)
      v1 = 2 * Rnd - 1
      v2 = 2 * Rnd - 1
      r = v1 ^ 2 + v2 ^ 2
      If (r < 1) Then
         flag = True
      End If
   Loop
   fac = Sqr(-2 * Log(r) / r)
   GetRandomNormal = v2 * fac
End Function

Public Sub GetRandomDirection(xdir As Single, ydir As Single, zdir As Single)
   'Returns the x, y, and z components of a randomly directed unit vector.
   Dim sumofsquares As Single
   Dim sqrofsumofsquares As Single
   Dim flag As Boolean

   flag = False

   Do While (flag = False)
      xdir = Rnd * 2 - 1
      ydir = Rnd * 2 - 1
      zdir = Rnd * 2 - 1

      sumofsquares = xdir ^ 2 + ydir ^ 2 + zdir ^ 2
      sqrofsumofsquares = sumofsquares ^ 0.5

      If (sumofsquares <= 1) And (sumofsquares > 0) Then
         'Enter if the point in inside the sphere.
         'Normalize to a unit vector.
         xdir = xdir / sqrofsumofsquares
         ydir = ydir / sqrofsumofsquares
         zdir = zdir / sqrofsumofsquares
         flag = True
      End If
   Loop

End Sub
```

(c) AnalyzeTrimData.bas

```
Option Explicit

'*************************************************************************************
'*                                                                                   *
'*                              AnalyzeTrimData.bas                                  *
'*                                                                                   *
'*          Create Simulated Electrons and Drift Them Through the Gas                *
'*                                                                                   *
'*************************************************************************************

Public Sub AnalyzeTrimData()
   'This is the main sub for this module.  Call this sub to do the other
things.
   Call SumEnergyDepositionData

   If (GeneratePulse = True) Then
      Call CombineEnergyDepositionFiles
      DoEvents
      Call CycleThroughEnergyDepositionData
      DoEvents
   Else
      Call PulsesUserEditable.NoPulsesCall
      DoEvents
   End If
End Sub

Private Sub SumEnergyDepositionData()
   'The first task is to fill up the array TotalEnergyDeposition() using the
Trim
   'output data.  This will mean that the data can then be saved by the user if
   'desired by putting an appropriate routine in the PulsesUserEditable module.
   Dim EventNum As Long
   Dim s1 As Single
   Dim s2 As Single
   Dim s3 As Single
   Dim s4 As Single
   Dim s5 As Single
   Dim s6 As Single
   Dim deltaE As Single
   Dim i As Long
   Dim ii As Long
   Dim fullpath$

   If (IncludeReactionProductInfo = True) Then
      If (TargetLayer = SecondTargetLayer) Then
         ReDim ContributingParticles(TL2EventNum)
      Else
         ReDim ContributingParticles(TL1EventNum)
      End If
   End If

   ReDim TotalEnergyDeposition(NumberOfRuns)
   'It is necessary to see if the GeneratePulse option is selected because the
   'syntax in the data file is different if the option is not selected.
   If (GeneratePulse = False) Then
      'Add up the total amounts of energy deposited in the layer of interest.
      For i = 1 To v Step 1
         For ii = 1 To 2 Step 1
            fullpath$ = path$ & "LOI_PARTICLE_DATA_" & i & "-" & ii & ".OUT"
            If Dir$(fullpath$) = "" Then
               'The file doesn't exist.
            Else
               'The file does exist.
               Open fullpath$ For Input As #1
               Do While EOF(1) = False
                  Input #1, EventNum, deltaE
                  If (IncludeReactionProductInfo = True) Then
                     If (ContributingParticles(EventNum) = 0) Then
                        ContributingParticles(EventNum) = ii
                     ElseIf (ContributingParticles(EventNum) <> ii) Then
                        ContributingParticles(EventNum) = 3
                     End If
                  End If
```

```
                  TotalEnergyDeposition(EventNum) =
TotalEnergyDeposition(EventNum) + deltaE
               Loop
               Close #1
            End If
         Next ii
      Next i
   Else
      'Add up the total amounts of energy deposited in the layer of interest.
      For i = 1 To v Step 1
         For ii = 1 To 2 Step 1
            fullpath$ = path$ & "LOI_PARTICLE_DATA_" & i & "-" & ii & ".OUT"
            If Dir$(fullpath$) = "" Then
               'The file doesn't exist.
            Else
               'The file does exist.
               Open fullpath$ For Input As #1
               Do While EOF(1) = False
                  Input #1, EventNum, s1, s2, s3, s4, s5, s6, deltaE
                  If (IncludeReactionProductInfo = True) Then
                     If (ContributingParticles(EventNum) = 0) Then
                        ContributingParticles(EventNum) = ii
                     ElseIf (ContributingParticles(EventNum) <> ii) Then
                        ContributingParticles(EventNum) = 3
                     End If
                  End If
                  TotalEnergyDeposition(EventNum) =
TotalEnergyDeposition(EventNum) + deltaE
               Loop
               Close #1
            End If
         Next ii
      Next i
   End If

   For i = 1 To UBound(TotalEnergyDeposition) Step 1
      If (TotalEnergyDeposition(i) < 0#) Then
         MsgBox "Total energy deposition less than zero - bug in the code.", ,
"Error:"
         MsgBox "Ignore all output from this run.", , "Note:"
         End
      End If
   Next i

   DoEvents
End Sub

Private Sub CombineEnergyDepositionFiles()
   'Before running this sub, the energy deposition information is potentially
   'scattered across many files having a name along the lines of
   'LOI_PARTICLE_DATA_x_y, where x and y are integers.  This data should be
combined
   'into one file with everything being put in order by event number; that will
make
   'it easier to process all of the data later on.
   'Dim exists(v, 2) As Boolean
   Dim i As Long
   Dim ii As Long
   Dim counter As Long
   Dim currentstring$()
   Dim currenteventnum() As Long
   Dim flag As Boolean
   Dim endoffile() As Long
   Dim location As Long
   Dim lowest As Long
   Dim lowvalue As Long
   Dim stringempty() As Boolean
   Dim fullpath$
   Dim temp$

   For i = 1 To v Step 1
      For ii = 1 To 2 Step 1
         fullpath$ = path$ & "LOI_PARTICLE_DATA_" & i & "-" & ii & ".OUT"
         If Dir$(fullpath$) = "" Then
            'The file doesn't exist.
         Else
            'The file does exist.
```

```
            flag = False
            Open fullpath$ For Input As #1
            If EOF(1) = True Then
               flag = True
            End If
            Close #1
            If flag = True Then
               Kill (fullpath$)
            End If
         End If
      Next ii
   Next i

   'The main part of the routine follows.
   counter = 0
   For i = 1 To v Step 1
      For ii = 1 To 2 Step 1
         fullpath$ = path$ & "LOI_PARTICLE_DATA_" & i & "-" & ii & ".OUT"
         If Dir$(fullpath$) = "" Then
            'The file doesn't exist.
         Else
            'The file does exist.
            counter = counter + 1
            Open fullpath$ For Input As #counter
         End If
      Next ii
   Next i

   If (counter = 0) Then
      'Maybe we should quit the program?
      Exit Sub
   End If

   DoEvents

   ReDim currentstring$(counter)
   ReDim currenteventnum(counter)
   ReDim endoffile(counter)
   ReDim stringempty(counter)

   flag = False

   For i = 1 To counter Step 1
      If EOF(counter) = True Then
         endoffile(counter) = True
      End If
   Next i
   endoffile(0) = True

   For i = 1 To counter Step 1
      If endoffile(counter) = False Then
         Line Input #i, currentstring$(i)
         location = InStr(currentstring$(i), Chr$(9))
         currenteventnum(i) = CLng(Left(currentstring$(i), location - 1))
      End If
   Next i

   lowvalue = 2147483647
   lowest = 0
   For i = 1 To counter Step 1
      If (currenteventnum(i) < lowvalue And currenteventnum(i) > 0 And
endoffile(i) = False) Then
         lowvalue = currenteventnum(i)
         lowest = i
      End If
   Next i

   For i = 1 To counter Step 1
      If EOF(i) = True Then
         endoffile(i) = True
      End If
   Next i

   Open path$ & "LOI_PARTICLE_DATA.OUT" For Output As #511

   DoEvents
```

```
   Do While (flag = False)
      Print #511, currentstring$(lowest)

      If (endoffile(lowest) = False) Then
         Line Input #lowest, currentstring$(lowest)
         location = InStr(currentstring$(lowest), Chr$(9))
         currenteventnum(lowest) = CLng(Left(currentstring$(lowest), location -
1))
         If EOF(lowest) = True Then
            endoffile(lowest) = True
         End If

      Else
         stringempty(lowest) = True
      End If

      'Make sure that there are still some data strings that haven't been
written to
      'the output file.  If there aren't any left, it's time to quit this do
loop.
      flag = True
      For i = 1 To counter Step 1
         If (stringempty(i) = False) Then
            flag = False
            DoEvents
         End If
      Next i

      'Make sure that strings marked as not empty really aren't empty.
      For i = 1 To counter Step 1
         If (currentstring$(i) = "") Then
            stringempty(i) = True
         End If
      Next i

      'Determine the next source from which to write data to the output file.
      lowvalue = 2147483647
      lowest = 0
      For i = 1 To counter Step 1
         If (stringempty(i) = False) Then
            If (currenteventnum(i) < lowvalue And currenteventnum(i) > 0) Then
               lowvalue = currenteventnum(i)
               lowest = i
            End If
         End If
      Next i
   Loop

   Print #511, Chr$(26)
   Close #511
   For i = 1 To counter Step 1
      Close #i
   Next i

   'Delete LOI files that are now unused.
   For i = 1 To v Step 1
      For ii = 1 To 2 Step 1
         fullpath$ = path$ & "LOI_PARTICLE_DATA_" & i & "-" & ii & ".OUT"
         If Dir$(fullpath$) = "" Then
            'The file doesn't exist.
         Else
            'The file does exist.
            Kill (fullpath$)
         End If
      Next ii
   Next i

   DoEvents
End Sub

Private Sub CycleThroughEnergyDepositionData()
   '
   Dim previouseventnum As Long
   Dim EventNum As Long
   Dim xstart As Single
   Dim ystart As Single
   Dim zstart As Single
```

```
   Dim xfinish As Single
   Dim yfinish As Single
   Dim zfinish As Single
   Dim deltaE As Single

   'Put in a call to FirstCall so that the user can put code in that subroutine
to
   'initialize an array for collecting data or whatever else is desired.
   Call PulsesUserEditable.FirstCall

   previouseventnum = 0
   Open path$ & "LOI_PARTICLE_DATA.OUT" For Input As #1

   Do While (EOF(1) = False)

      Input #1, EventNum, xstart, ystart, zstart, xfinish, yfinish, zfinish,
deltaE

      If (EOF(1) = True) Then
         'If appropriate, finish the last event.
         If (previouseventnum <> EventNum And previouseventnum > 0) Then
            Call PulsesUserEditable.ProcessElectronsAtAnode(previouseventnum)
            DoEvents
         End If
         'Finish the current event.
         If (EventNum > 0) Then
            Call TrackElectronsThroughGas(EventNum, xstart, ystart, zstart,
xfinish, yfinish, zfinish, deltaE)
            DoEvents
            Call PulsesUserEditable.ProcessElectronsAtAnode(EventNum)
            DoEvents
         End If
         'Quit.
         Exit Do

      End If

      If (EventNum <> previouseventnum) Then
         'A new event has been started.
         If (previouseventnum > 0) Then
            'Call whatever routine processes the binned electrons at the anode.
            'Beyond this, things are handled by whatever code the user puts in
            'the PulsesUserEditable module.
            Call PulsesUserEditable.ProcessElectronsAtAnode(previouseventnum)
            DoEvents
         End If
         previouseventnum = EventNum
      End If
      'Pass the data to the routine for generating and drifting electrons.
      Call TrackElectronsThroughGas(EventNum, xstart, ystart, zstart, xfinish,
yfinish, zfinish, deltaE)
      DoEvents

   Loop

   Close #1

   'Put in a call to LastCall so that the user can put code in that subroutine
to
   'do things like process the data into a useful form and display it, save it,
or
   'do whatever is desired.
   Call PulsesUserEditable.LastCall
   DoEvents
End Sub

Public Sub TrackElectronsThroughGas(reactionnum As Long, xPathStart As Single,
yPathStart As Single, zPathStart As Single, xPathFinish As Single, yPathFinish
As Single, zPathFinish As Single, EnergyLostInStep As Single)
   Dim xStepDistance As Single
   Dim yStepDistance As Single
   Dim zStepDistance As Single
   Dim temp As Single
   Dim temp2 As Single
   Dim NumberOfIonizations As Long
   Dim xStepBetweenIonizations As Single
   Dim yStepBetweenIonizations As Single
```

```
   Dim zStepBetweenIonizations As Single
   Dim j As Long
   Dim previousj As Long
   Dim a As Single
   Dim xElectronStart As Single
   Dim yElectronStart As Single
   Dim zElectronStart As Single
   Dim ElectronDriftDistance As Single
   Dim MeanDiffusionDistance As Single
   Dim yElectronFinish As Single
   Dim zElectronFinish As Single
   Dim currentrnd As Single

   'The particle track position data needs to be converted from Angstroms to
   'microns in order to match the other values that are used.
   xPathStart = xPathStart / 10000#
   yPathStart = yPathStart / 10000#
   zPathStart = zPathStart / 10000#
   xPathFinish = xPathFinish / 10000#
   yPathFinish = yPathFinish / 10000#
   zPathFinish = zPathFinish / 10000#

   xStepDistance = xPathFinish - xPathStart
   yStepDistance = yPathFinish - yPathStart
   zStepDistance = zPathFinish - zPathStart

   'There is some randomness in the number electrons created.  I am assuming
that
   'the standard deviation of the number of electrons is the square root of the
   'number of electrons.
   temp = EnergyLostInStep / eVperIonization
   NumberOfIonizations = CLng(temp)
   temp2 = temp ^ 0.5
   temp2 = temp2 * MicromegasSim.GetRandomNormal
   NumberOfIonizations = NumberOfIonizations + CLng(temp2)
   If (NumberOfIonizations < 0) Then
      NumberOfIonizations = 0
   End If

   DoEvents

   If (NumberOfIonizations >= 1) Then
      previousj = 0

      'Treat each electron individually.
      For j = 1 To NumberOfIonizations Step 1
         currentrnd = Rnd
         xElectronStart = xPathStart + currentrnd * xStepDistance
         yElectronStart = yPathStart + currentrnd * yStepDistance
         zElectronStart = zPathStart + currentrnd * zStepDistance

         'Only process electrons that started within the boundary of the
detector.
         If (yElectronStart >= 0# And yElectronStart <= YDimension And
zElectronStart >= 0# And zElectronStart <= ZDimension) Then
            ElectronDriftDistance = (LayerThickness(LayerOfInterest) / 10000#)
- xElectronStart

            If (ElectronDriftDistance > 0#) Then
               MeanDiffusionDistance = DiffusionCoefficient / ((10000# /
ElectronDriftDistance) ^ 0.5)
            Else
               MeanDiffusionDistance = 0#
            End If

            'The calculation below gives how far the electron drifts in the y-z
plane.
            'Multiplying by the sine of a random angle gives the amount of
drift in the
            'y direction and multiplying by the cosine gives the drift in the
z.
            a = MicromegasSim.GetRandomNormal
            currentrnd = Rnd
            yElectronFinish = yElectronStart + MeanDiffusionDistance * a *
Sin(currentrnd * 2# * Pi)
            zElectronFinish = zElectronStart + MeanDiffusionDistance * a *
Cos(currentrnd * 2# * Pi)
```

```
            'Reflect back in any electrons that drift beyond the detector
bounds.
            If (yElectronFinish < 0#) Then
               yElectronFinish = 0 - yElectronFinish
            End If
            If (yElectronFinish > YDimension) Then
               yElectronFinish = 2# * YDimension - yElectronFinish
            End If
            If (zElectronFinish < 0#) Then
               zElectronFinish = 0 - zElectronFinish
            End If
            If (zElectronFinish > ZDimension) Then
               zElectronFinish = 2# * ZDimension - zElectronFinish
            End If

            If (j - previousj > 50) Then
               DoEvents
               previousj = j
            End If

            'Call whatever routine bins the electrons to the anode features.
            Call PulsesUserEditable.CollectElectronsAtAnode(yElectronFinish,
zElectronFinish, reactionnum)
         End If
      Next j
   End If
End Sub
```

(d) MicromegasSimModule.bas

```
Option Explicit

'*****************************************************************************
'*                                                                           *
'*                          MicromegasSimModule.bas                          *
'*                                                                           *
'* Contains Variables and Constants Used Throughout the Micromegas Simulation *
'*                                                                           *
'*****************************************************************************

'Note: Some variables are no longer in use, although most are.

'GEOMETRY NOTES:
'The program geometry has been set to match that of TRIM.  This means that the
X
'axis is the one that goes into the detector and the Y and Z axes both go
across
'the detector at right angles to each other.  The Micromegas anode strips are
'assumed to run parallel to the Z axis (i.e. Y position determines anode
collection
'strip).

'Physical constants
Global Const Pi = 3.14159265
'Global Const JPereV = 1.6 * 10 ^ -19
'Global Const kgPeramu = 1.674 * 10 ^ -27
'Global Const eVPerJ = 6.24 * 10 ^ 18

'Mathematical constants
Global Const ee = 2.71828182846

'User-defined variables
Global NeutronTargetType As Long '1 is Li-6, 2 is B-10
Global TargetThickness As Single
Global SecondTargetThickness As Single
Global LayerOfInterestThickness As Single
Global eVperIonization As Single
Global AnodeGranularity As Single
Global DiffusionCoefficient As Single
Global GasGain As Single
Global FittingRatio As Single
Global NumberOfRuns As Long
Global YDimension As Single
Global ZDimension As Single

'Variables pertaining to the different types of layers and their qualities
'and what the simulation should do.
Global NumberOfLayers As Long
Global LayerThickness() As Single
Global LayerMaterial() As Long
Global LayerOfInterest As Long
Global GeneratePulse As Boolean 'false - give total deposition, true - do full
pulse
Global SaveReactionLocations As Boolean 'true - save the neutron locations in a
text file
Global SaveAnodeSignals As Boolean
Global SaveQFCSToExcel As Boolean
Global SaveQFCS As Boolean
Global SaveEnergyDeposition As Boolean
Global SingleTargetStrip As Boolean
Global SingleTargetStripNumber As Long
Global TargetLayer As Long 'Which layer is the neutron target layer?
Global SecondTargetLayer As Long 'If there is a second neutron target layer.
Global TotalEnergyDeposition() As Single
Global PHABins() As Long
Global EnergyBinSize As Single
Global ParticleWeights() As Single

'Variables pertaining to the characteristics of the reaction products.
Global E1 As Single
Global P1 As Long
Global P1M As Single
```

```
Global P1Z As Long
Global x1 As Single
Global y1 As Single
Global z1 As Single
Global E2 As Single
Global P2 As Long
Global P2M As Single
Global P2Z As Long
Global x2 As Single
Global y2 As Single
Global z2 As Single

Global MaximumPossibleEnergy As Single
Global ContributingParticles() As Long
Global v As Long 'used as a counter but the final value generated in
TrimAutomation
'is needed in the AnalyzeTrimData module

Global TL1EventNum As Long
Global TL2EventNum As Long

Global AllowTransmissions As Boolean
Global ReducedWeight As Boolean 'Use when you have neutrons of different
energies or different directions and you
'force neutron absorption (i.e. AllowTransmissions is false)
Global IncludeReactionProductInfo As Boolean
Global SemiconductorVariance As Boolean

'Variables set here (but could be made to depend on user input)
'Global Const MaxParticleRangeGas = 600000#
Global Const EnergyStepSize = 25000# '(eV)  The step size for the exyz.txt
file.  The
'energy step size is for use with the modified version of Trim that Dr. Ziegler
made
'especially for use with this code.  The EnergyStepSize gives the approximate
interval
'(in terms of change in energy) for which Trim is to report back the position
and
'energy of a particle as it travels along through a material.

Global Const adjustfactor = 8# 'This is the multiplier for the variance in the
gas gain.
'Software simulation has shown it to be in the neighborhood of 8, that is, for
a gain
'of 100 and an amplification gap, the standard deviation of the number of
electrons
'liberated in the electron shower is pretty consistently about eight times the
square
'root of the total number of electrons in the shower.
Global Const semiconductoradjustfactor = 1#
'Multiply this number by the square root of the number of electrons liberated
in the
'semiconductor and by a random number from a normal distribution to get the
number of
'electrons actually liberated (what produces the signal).

'Array for tracking information on particles as they pass through the layer of
interest.
Global ParticlesLayerOfInterest() As Single
'ParticlesLayerOfInterest(particlenumber, position, attributenumber)
'The following is the scheme for the second subscript:
'0 - entering the layer
'1 - on the negative side
'2 - on the positive side
'The following is the readout scheme for the third subscript:
'1 - particle x position
'2 - particle y position
'3 - particle z position
'4 - particle energy

'Create an array that gives unique identifying information for each reaction
particle
'being tracked, thus allowing correspondence to be made between each individual
'reaction and what happens inside the "detector" part (layer of interest) of
the
'detector.  Each particle can be identified by which group it belongs to (the
first or
```

```
'second type of reaction product), the layer it just exited and the particle
number it
'had in that run, or by the layer it is just entering and the particle number
it had
'in that run.
Global ParticleIdentifier() As Long
'Key to interpretation: three subscripts - the first is which group it is in
(the
'first or second), the second is its number (in the total NumberOfRuns), and
the third
'selects the following items:
' 1 - the layer it is associated with,
' 2 - the particle number it has in the input deck for that layer.
'It would be preferable to go from layer and number in the deck for that layer
to the
'original particle number rather than going the other way (necessitating
searching
'through lists of particle information), but I haven't yet come up with a slick
way
'to do this.
Global ReadIn() As Boolean
'If a particle corresponding to a particular reaction isn't read in during a
TRIM run
'cycle, zero out the particle identification values so that you know that there
is no
'longer a particle in circulation that stems from that particular reaction.

Global Const path$ = "C:\Program Files\SRIM 2000\"
Global Const MaxNumberOfLayers = 6

Global zposition As Single
Global xposition As Single
Global yposition As Single

'These variables can't be addressed in the main form unless they are made
global.
'Figure out what is needed and what isn't.
Dim MacroCrossSection As Single
Dim TravelDistance As Single
Dim Interacted As Boolean
Dim NeutronEnergy As Single
Dim IonsInTargetArray(3, 5, 52, 3) As Single
'1st value is particle type, 2nd is target type, 3rd gives the order, 4th gives
energy and range info
'Dim EnergyDepositionInGas(51, 5) As Single
'The first marker gives the section number since the path is being cut into
sections.
'It starts at 1 and doesn't correspond to any number from anything else.
'The 5 items addressable by the second marker are, respectively:
'(1) The energy deposition (in eV) during that path section.
'(2) The x point (in microns) at the start of the section.
'(3) The z point (in microns) at the start of the section.
'(4) The x point (in microns) at the end of the section.
'(5) The z point (in microns) at the end of the section.
Dim EnergyWhenEnteringGas(150) As Long
Global GasDensity As Single 'for use in changing the gas pressure for
Ar/isobutane when
'doing long automated runs
```

(e) PulsesUserEditable.bas

```
Option Explicit

'*************************************************************************************
'*                                                                                   *
'*                            PulsesUserEditable.bas                                 *
'*                                                                                   *
'*        Calculations for the simulation of pulses go in this module.               *
'*                                                                                   *
'* This module is designed to be easily edited by the user so that different        *
'* types of things that aren't done right now can be done relatively easily         *
'* (e.g. square electrode pixelation for 2D neutron position sensitivity).          *
'*                                                                                   *
'*************************************************************************************

'=====================================================================================
'                            IMPORTANT NOTES TO USER:
'
'=====================================================================================
'If you use file access in this module, be sure to number the file numbers from
511 on
'down when you do a for input as #x or for output as #x.  The reason for this
is that
'#1 and possibly numbers right above it (2, 3, 4...) are in use as file numbers
in open
'files elsewhere.

Public ElectronFinish() As Long
Dim QFCS() As Long

Public Sub FirstCall()
   'This subroutine is to be called before any of the electron data starts to
arrive.
   'This allows the user to program in initialization of data arrays or
whatever else
   'is desired.
   ReDim ElectronFinish(0 To (CLng(YDimension / AnodeGranularity) + 1))

   If (AllowTransmissions = False) Then
      ReDim QFCS(NumberOfRuns, 2)
      'For the second subscript:
      '1 - total number of electrons
      '2 - fitted cluster size
   Else
      If (TargetLayer = SecondTargetLayer) Then
         ReDim QFCS(TL2EventNum, 2)
      Else
         'It must be the first target layer.
         ReDim QFCS(TL1EventNum, 2)
      End If
   End If

   If (SingleTargetStrip = True) Then
      Open path$ & "ElectronsOnSingleTargetStrip.txt" For Output As #508
   End If
End Sub

Public Sub CollectElectronsAtAnode(ByVal yElectronFinish As Single, ByVal
zElectronFinish As Single, ByVal reactionnum As Long)
   'This is programmable by the user to do whatever he/she wants upon finding
out where
   'individual electrons arrive in the y-z plane upon crossing the gas chamber.
I
   'suggest that a record of some kind be kept of where the electrons arrive so
that
   'simulated pulses can be generated whenever ProcessElectronsAtAnode() gets
called
   '(which is every time a simulated reaction is finished).  Note that
electrons are
   'already reflected back into the detector volume if they try to diffuse out
of it.
```

```
   '(That is done in the AnalyzeTrimData module that feeds data into this
module.)
   Dim k As Long
   Dim StripStart As Single
   Dim StripFinish As Single
   Dim StripNumber As Long
   Dim Anode As Long

   ' The electron must be sent to the correct strip.
   k = CLng(yElectronFinish / AnodeGranularity)
   k = k + 1
   StripStart = CSng(k - 1) * AnodeGranularity
   StripFinish = CSng(k) * AnodeGranularity
   If (yElectronFinish < StripStart) Then
      Anode = k - 1
   Else
      Anode = k
   End If
   ElectronFinish(Anode) = ElectronFinish(Anode) + 1
End Sub

Public Sub ProcessElectronsAtAnode(reactionnum As Long)
   'This subroutine gets called whenever all the electrons for a particle event
have
   'been drifted to the anode side of the layer of interest (presumably the gas
charge
   'drift region).  The user should write whatever code is desired to handle
this and
   'process the collected electrons into useful data.
   Dim maxelectrons As Long
   Dim partmaxelectrons As Long
   Dim totalelectrons As Long
   Dim FCS As Long
   Dim i As Long

   Call CalculateElectronShower

   For i = LBound(ElectronFinish) To UBound(ElectronFinish) Step 1
      If (ElectronFinish(i) > maxelectrons) Then
         maxelectrons = ElectronFinish(i)
      End If
      totalelectrons = totalelectrons + ElectronFinish(i)
   Next i

   FCS = 0
   If (maxelectrons > 0) Then
      partmaxelectrons = (maxelectrons \ FittingRatio)
      If (partmaxelectrons = 0) Then
         partmaxelectrons = 1
      End If
      For i = LBound(ElectronFinish) To UBound(ElectronFinish) Step 1
         If (ElectronFinish(i) >= partmaxelectrons) Then
            FCS = FCS + 1
         End If
      Next i
   End If

   QFCS(reactionnum, 1) = totalelectrons
   QFCS(reactionnum, 2) = FCS
   If (SaveAnodeSignals = True) Then
      Call SaveElectronToAnodeData(reactionnum)
   End If
   If (SingleTargetStrip = True) Then
      Print #508, ElectronFinish(SingleTargetStripNumber)
   End If

   'Clear the ElectronFinish() array in preparation for the next event.
   ReDim ElectronFinish(0 To (CLng(YDimension / AnodeGranularity) + 1))
End Sub

Public Sub LastCall()
   'This subroutine gets called whenever all the particle data has been
processed.
   'This is the final external call to anything in this module.  The user can
put all
   'sorts of things in here such as code that takes the results from the
individual
```

```
   'event cases and calculates statistics regarding how much energy gets
deposited per
   'event or, as another example, code that starts up Microsoft Excel, dumps
all the
   'data from the different runs into an Excel spreadsheet, and then plots the
data
   'using Excel's graphing capabilities.
   'As part of this, an array containing the total amount of energy deposited
in
   'the detection layer/layer of interest for each individual event is passed
to this
   'sub.
   If (SaveQFCSToExcel = True) Then
      Call DrawDataChart
   End If
   If (SaveQFCS = True) Then
      Call SaveQFCSData
   End If
   If (SaveEnergyDeposition = True) Then
      Call SaveTotalEnergyDepositionData
   End If
   If (SingleTargetStrip = True) Then
      Call CloseTargetStripData
   End If
End Sub

Public Sub NoPulsesCall()
   'There is where it goes if the user doesn't choose to simulate full pulses.
   If (SaveEnergyDeposition = True) Then
      Call SaveTotalEnergyDepositionData
   End If
End Sub

Public Sub CalculateElectronShower()
   'Factor in gas gain and statistical variations to the number of electrons
headed
   'at each anode strip.
   Dim i As Long
   Dim NumberOfElectrons As Long
   Dim AddedElectrons As Long
   Dim randomvar As Single

   For i = LBound(ElectronFinish) To UBound(ElectronFinish) Step 1
      NumberOfElectrons = ElectronFinish(i)
      If (NumberOfElectrons > 0) Then
         AddedElectrons = NumberOfElectrons * GasGain
         randomvar = CSng(AddedElectrons) ^ 0.5
         randomvar = adjustfactor * randomvar * MicromegasSim.GetRandomNormal
         AddedElectrons = AddedElectrons + CLng(randomvar)
         If (AddedElectrons < 0) Then
            AddedElectrons = 0
         End If
         ElectronFinish(i) = NumberOfElectrons + AddedElectrons
      End If
   Next i
End Sub

Public Sub SaveElectronToAnodeData(reactionnum As Long)
   'Call this if you want to save the data for the collection of electrons on
   'individual data strips.  You have to call this for each individual
particle.
   Dim i As Long

   Open path$ & "ANODE_STRIP_DEPOSITION_" & reactionnum & ".TXT" For Output As
#511
   For i = LBound(ElectronFinish) To UBound(ElectronFinish) Step 1
      If (ElectronFinish(i) > 0) Then
         Print #511, i & Chr$(9) & ElectronFinish(i)
      End If
   Next i
   Print #511, Chr$(26)
   Close #511
End Sub

Public Sub SaveQFCSData()
   Dim i As Long
   Dim flag(3) As Boolean
```

```
   Open path$ & "QFCS_DATA.TXT" For Output As #511
   For i = 1 To UBound(QFCS) Step 1
      If (QFCS(i, 1) > 0) Then
         Print #511, QFCS(i, 1) & Chr$(9) & QFCS(i, 2)
      End If
   Next i
   Print #511, Chr$(26)
   Close #511

   If (IncludeReactionProductInfo = True) Then
      For i = 1 To 3 Step 1
         flag(i) = False
      Next i

      Open path$ & "QFCS_DATA_1.TXT" For Output As #509
      Open path$ & "QFCS_DATA_2.TXT" For Output As #510
      Open path$ & "QFCS_DATA_3.TXT" For Output As #511

      For i = 1 To UBound(QFCS) Step 1
         If (QFCS(i, 1) > 0) Then
            If (ContributingParticles(i) = 1) Then
               Print #509, QFCS(i, 1) & Chr$(9) & QFCS(i, 2)
               flag(1) = True
            ElseIf (ContributingParticles(i) = 2) Then
               Print #510, QFCS(i, 1) & Chr$(9) & QFCS(i, 2)
               flag(2) = True
            ElseIf (ContributingParticles(i) = 3) Then
               Print #511, QFCS(i, 1) & Chr$(9) & QFCS(i, 2)
               flag(3) = True
            End If
         End If
      Next i

      Print #509, Chr$(26)
      Print #510, Chr$(26)
      Print #511, Chr$(26)
      Close #509
      Close #510
      Close #511

      For i = 1 To 3 Step 1
         If (flag(i) = False) Then
            Kill (path$ & "QFCS_DATA_" & i & ".TXT")
         End If
      Next
   End If

End Sub

Public Sub CloseTargetStripData()
   '
   Close #508
End Sub

Public Sub DrawDataChart()
   'Some of this code is taken from Microsoft's coding examples.
   Dim oXL As Object        'Excel application
   Dim oBook As Object      'Excel workbook
   Dim oSheet As Object     'Excel Worksheet
   Dim oChart As Object     'Excel Chart
   Dim aTemp() As Long      'Data array
   Dim iRow As Integer      'Index variable for the current row
   Dim iCol As Integer      'Index variable for the current column

   Const cNumCols = 2       'Number of points in each Series
   Dim cNumRows As Long

   Dim i As Long

   'Only copy over actual events.
   cNumRows = 0
   For i = 1 To UBound(QFCS) Step 1
      If (QFCS(i, 2) > 0) Then
         cNumRows = cNumRows + 1
      End If
   Next i
```

```
   ReDim aTemp(1 To cNumRows, 1 To cNumCols)

   'Start Excel and create a new workbook
   Set oXL = CreateObject("Excel.application")
   Set oBook = oXL.Workbooks.Add
   Set oSheet = oBook.Worksheets.Item(1)

   'Insert data into cells for the two series:
   iRow = 0
   For i = 1 To UBound(QFCS) Step 1
      If (QFCS(i, 2) > 0) Then
         iRow = iRow + 1
         aTemp(iRow, 1) = QFCS(i, 1)
         aTemp(iRow, 2) = QFCS(i, 2)
      End If
   Next i

   oSheet.Range("A1").Resize(cNumRows, cNumCols).Value = aTemp

   'Make Excel Visible:
   oXL.Visible = True
   oXL.UserControl = True
End Sub

Public Sub SaveTotalEnergyDepositionData()
   'Save the total amounts of energy deposited in the layer of
interest/detection
   'layer for each individual event.
   Dim i As Long
   Dim ii As Long
   Dim iii As Long
   Dim EventNum As Long
   Dim flag As Boolean
   Dim VarianceDeposition As Single
   'Note: If the user checks the semiconductor variance check box, the amount
of energy
   'deposited will not change, but the magnitude of the signal it produces
will.  The
   'energy will be converted into the corresponding number of electrons using
the
   'ionization energy and this will then be varied statistically and converted
back
   'into a kind of equivalent energy based on the average ionization energy and
then
   'binned.  This is not meant to imply that the actual amount of energy
deposited
   'changes.
   Dim numelectrons As Single

   If (ReducedWeight = True And AllowTransmissions = False) Then
      Call SaveWeightedTotalEnergyDepositionData
      Exit Sub
   End If

   Open path$ & "TOTAL_ENERGY_DEPOSITION.TXT" For Output As #1
   If (IncludeReactionProductInfo = False) Then
      ReDim PHABins(CLng(MaximumPossibleEnergy / EnergyBinSize))
      If (AllowTransmissions = False) Then
         'Save the total energy deposition for all events in this layer.
         For i = 1 To UBound(TotalEnergyDeposition) Step 1 'relevantevents Step
1
            If (TotalEnergyDeposition(i) > 0#) Then
               Print #1, i & Chr$(9) & TotalEnergyDeposition(i)
            End If
         Next i
         Close #1
         'Bin the total energy deposited.
         For i = 1 To UBound(TotalEnergyDeposition) Step 1 'relevantevents Step
1
            If (SemiconductorVariance = True) Then
               numelectrons = CLng(TotalEnergyDeposition(i) / eVperIonization)
               numelectrons = numelectrons + semiconductoradjustfactor *
MicromegasSim.GetRandomNormal * (numelectrons ^ 0.5)
               If (numelectrons < 0#) Then
                  numelectrons = 0#
               End If
```

```
               VarianceDeposition = numelectrons * eVperIonization
               ii = CLng(VarianceDeposition / EnergyBinSize)
               PHABins(ii) = PHABins(ii) + 1
            Else
               ii = CLng(TotalEnergyDeposition(i) / EnergyBinSize)
               PHABins(ii) = PHABins(ii) + 1
            End If
         Next i
         'Save the binned pulse information.
         Open path$ & "BINNED_ENERGY_DEPOSITION.TXT" For Output As #1
         For i = 0 To UBound(PHABins) Step 1
            Print #1, i & Chr$(9) & PHABins(i)
         Next i
      Else
         'Save the total energy deposition for all events in this layer.
         For i = 1 To UBound(TotalEnergyDeposition) Step 1 'relevantevents Step
1
            If (TotalEnergyDeposition(i) > 0#) Then
               Print #1, i & Chr$(9) & TotalEnergyDeposition(i)
            End If
         Next i
         Close #1
         'Bin the total energy deposited.
         For i = 1 To UBound(TotalEnergyDeposition) Step 1 'relevantevents Step
1
            If (SemiconductorVariance = True) Then
               numelectrons = CLng(TotalEnergyDeposition(i) / eVperIonization)
               numelectrons = numelectrons + semiconductoradjustfactor *
MicromegasSim.GetRandomNormal * (numelectrons ^ 0.5)
               If (numelectrons < 0#) Then
                  numelectrons = 0#
               End If
               VarianceDeposition = numelectrons * eVperIonization
               ii = CLng(VarianceDeposition / EnergyBinSize)
               PHABins(ii) = PHABins(ii) + 1
            Else
               ii = CLng(TotalEnergyDeposition(i) / EnergyBinSize)
               PHABins(ii) = PHABins(ii) + 1
            End If
         Next i
         'Save the binned pulse information.
         Open path$ & "BINNED_ENERGY_DEPOSITION.TXT" For Output As #1
         For i = 0 To UBound(PHABins) Step 1
            Print #1, i & Chr$(9) & PHABins(i)
         Next i
      End If
      Close #1
   Else
      If (AllowTransmissions = False) Then
         'Save the total energy deposition for all events in this layer.
         For i = 1 To UBound(TotalEnergyDeposition) Step 1 'relevantevents Step
1
            If (TotalEnergyDeposition(i) > 0#) Then
               Print #1, i & Chr$(9) & TotalEnergyDeposition(i)
            End If
         Next i
         Close #1
         'Save the binned pulses separately for the different reaction product
cases.
         For iii = 1 To 3 Step 1
            flag = False
            ReDim PHABins(CLng(MaximumPossibleEnergy / EnergyBinSize))
            'Bin the total energy deposited for relevant events.
            For i = 1 To UBound(ContributingParticles) Step 1 'relevantevents
Step 1
               If (ContributingParticles(i) = iii) Then
                  flag = True
                  ii = CLng(TotalEnergyDeposition(i) / EnergyBinSize)
                  PHABins(ii) = PHABins(ii) + 1
               End If
            Next i
            'Save the binned pulse information.
            Open path$ & "BINNED_ENERGY_DEPOSITION-" & iii & ".TXT" For Output
As #1
            For i = 0 To UBound(PHABins) Step 1
               Print #1, i & Chr$(9) & PHABins(i)
            Next i
```

```
            Close #1
            If (flag = False) Then
               Kill (path$ & "BINNED_ENERGY_DEPOSITION-" & iii & ".TXT")
            End If
         Next iii
         'Save the binned pulses for all reaction product cases together.
         flag = False
         ReDim PHABins(CLng(MaximumPossibleEnergy / EnergyBinSize))
         'Bin the total energy deposited.
         For i = 1 To UBound(TotalEnergyDeposition) Step 1 'relevantevents Step
1
            flag = True
            If (SemiconductorVariance = True) Then
               numelectrons = CLng(TotalEnergyDeposition(i) / eVperIonization)
               numelectrons = numelectrons + semiconductoradjustfactor *
MicromegasSim.GetRandomNormal * (numelectrons ^ 0.5)
               If (numelectrons < 0#) Then
                  numelectrons = 0#
               End If
               VarianceDeposition = numelectrons * eVperIonization
               ii = CLng(VarianceDeposition / EnergyBinSize)
               PHABins(ii) = PHABins(ii) + 1
            Else
               ii = CLng(TotalEnergyDeposition(i) / EnergyBinSize)
               PHABins(ii) = PHABins(ii) + 1
            End If
         Next i
         'Save the binned pulse information.
         Open path$ & "BINNED_ENERGY_DEPOSITION.TXT" For Output As #1
         For i = 0 To UBound(PHABins) Step 1
            Print #1, i & Chr$(9) & PHABins(i)
         Next i
         Close #1
         If (flag = False) Then
            Kill (path$ & "BINNED_ENERGY_DEPOSITION.TXT")
         End If
      Else
         'Save the total energy deposition for all events in this layer.
         For i = 1 To UBound(TotalEnergyDeposition) Step 1 'relevantevents Step
1
            If (TotalEnergyDeposition(i) > 0#) Then
               Print #1, i & Chr$(9) & TotalEnergyDeposition(i)
            End If
         Next i
         Close #1
         'Save the binned pulses separately for the different reaction product
cases.
         For iii = 1 To 3 Step 1
            flag = False
            ReDim PHABins(CLng(MaximumPossibleEnergy / EnergyBinSize))
            'Bin the total energy deposited for relevant events.
            For i = 1 To UBound(ContributingParticles) Step 1 'relevantevents
Step 1
               If (ContributingParticles(i) = iii) Then
                  flag = True
                  If (SemiconductorVariance = True) Then
                     numelectrons = CLng(TotalEnergyDeposition(i) /
eVperIonization)
                     numelectrons = numelectrons + semiconductoradjustfactor *
MicromegasSim.GetRandomNormal * (numelectrons ^ 0.5)
                     If (numelectrons < 0#) Then
                        numelectrons = 0#
                     End If
                     VarianceDeposition = numelectrons * eVperIonization
                     ii = CLng(VarianceDeposition / EnergyBinSize)
                     PHABins(ii) = PHABins(ii) + 1
                  Else
                     ii = CLng(TotalEnergyDeposition(i) / EnergyBinSize)
                     PHABins(ii) = PHABins(ii) + 1
                  End If
               End If
            Next i
            'Save the binned pulse information.
            Open path$ & "BINNED_ENERGY_DEPOSITION-" & iii & ".TXT" For Output
As #1
            For i = 0 To UBound(PHABins) Step 1
               Print #1, i & Chr$(9) & PHABins(i)
```

```
            Next i
            Close #1
            If (flag = False) Then
               Kill (path$ & "BINNED_ENERGY_DEPOSITION-" & iii & ".TXT")
            End If
         Next iii
         'Save the binned pulses for all reaction product cases together.
         flag = False
         ReDim PHABins(CLng(MaximumPossibleEnergy / EnergyBinSize))
         'Bin the total energy deposited.
         For i = 1 To UBound(TotalEnergyDeposition) Step 1 'relevantevents Step
1
            If (TotalEnergyDeposition(i) > 0#) Then
               flag = True
               If (SemiconductorVariance = True) Then
                  numelectrons = CLng(TotalEnergyDeposition(i) /
eVperIonization)
                  numelectrons = numelectrons + semiconductoradjustfactor *
MicromegasSim.GetRandomNormal * (numelectrons ^ 0.5)
                  If (numelectrons < 0#) Then
                     numelectrons = 0#
                  End If
                  VarianceDeposition = numelectrons * eVperIonization
                  ii = CLng(VarianceDeposition / EnergyBinSize)
                  PHABins(ii) = PHABins(ii) + 1
               Else
                  ii = CLng(TotalEnergyDeposition(i) / EnergyBinSize)
                  PHABins(ii) = PHABins(ii) + 1
               End If
            End If
         Next i
         'Save the binned pulse information.
         Open path$ & "BINNED_ENERGY_DEPOSITION.TXT" For Output As #1
         For i = 0 To UBound(PHABins) Step 1
            Print #1, i & Chr$(9) & PHABins(i)
         Next i
         Close #1
         If (flag = False) Then
            Kill (path$ & "BINNED_ENERGY_DEPOSITION.TXT")
         End If
      End If
   End If
End Sub

Public Sub SaveWeightedTotalEnergyDepositionData()
   'Save the total amounts of energy deposited in the layer of
interest/detection
   'layer for each individual event.
   'The strategy for this one is going to be a bit different than that used by
the
   'other more-or-less equivalent routing.  This strategy is probably more
correct
   'when variance in signal amplitude is being used.
   Dim i As Long
   Dim ii As Long
   Dim iii As Long
   Dim EventNum As Long
   Dim flag As Boolean
   Dim VarianceDeposition As Single
   'Note: If the user checks the semiconductor variance check box, the amount
of energy
   'deposited will not change, but the magnitude of the signal it produces
will.  The
   'energy will be converted into the corresponding number of electrons using
the
   'ionization energy and this will then be varied statistically and converted
back
   'into a kind of equivalent energy based on the average ionization energy and
then
   'binned.  This is not meant to imply that the actual amount of energy
deposited
   'changes.
   Dim numelectrons As Single
   Dim WeightedPHABins() As Single
   Dim TotalWeightedPHABins() As Single

   Open path$ & "TOTAL_ENERGY_DEPOSITION.TXT" For Output As #1
```

```
   If (IncludeReactionProductInfo = False) Then
      ReDim TotalWeightedPHABins(CLng(MaximumPossibleEnergy / EnergyBinSize))
      For i = 1 To UBound(TotalEnergyDeposition) Step 1
         If (SemiconductorVariance = True) Then
            numelectrons = CLng(TotalEnergyDeposition(i) / eVperIonization)
            numelectrons = numelectrons + semiconductoradjustfactor *
EntryForm.GetRandomNormal * (numelectrons ^ 0.5)
            If (numelectrons < 0#) Then
               numelectrons = 0#
            End If
            VarianceDeposition = numelectrons * eVperIonization
            ii = CLng(VarianceDeposition / EnergyBinSize)
            TotalWeightedPHABins(ii) = TotalWeightedPHABins(ii) +
ParticleWeights(i)
         Else
            ii = CLng(TotalEnergyDeposition(i) / EnergyBinSize)
            TotalWeightedPHABins(ii) = TotalWeightedPHABins(ii) +
ParticleWeights(i)
         End If
      Next i
      For ii = 0 To UBound(TotalWeightedPHABins) Step 1
         Print #1, ii & Chr$(9) & TotalWeightedPHABins(ii)
      Next ii
   Else
      ReDim TotalWeightedPHABins(CLng(MaximumPossibleEnergy / EnergyBinSize))
      For i = 1 To 3 Step 1
         ReDim WeightedPHABins(CLng(MaximumPossibleEnergy / EnergyBinSize))
         flag = False
         For ii = 1 To UBound(TotalEnergyDeposition) Step 1
            If ContributingParticles(ii) = i Then
               flag = True
               If (SemiconductorVariance = True) Then
                  numelectrons = CLng(TotalEnergyDeposition(ii) /
eVperIonization)
                  numelectrons = numelectrons + semiconductoradjustfactor *
EntryForm.GetRandomNormal * (numelectrons ^ 0.5)
                  If (numelectrons < 0#) Then
                     numelectrons = 0#
                  End If
                  VarianceDeposition = numelectrons * eVperIonization
                  iii = CLng(VarianceDeposition / EnergyBinSize)
                  WeightedPHABins(iii) = WeightedPHABins(iii) +
ParticleWeights(ii)
                  TotalWeightedPHABins(iii) = TotalWeightedPHABins(iii) +
ParticleWeights(ii)
               Else
                  iii = CLng(TotalEnergyDeposition(ii) / EnergyBinSize)
                  TotalWeightedPHABins(iii) = TotalWeightedPHABins(iii) +
ParticleWeights(ii)
               End If
            End If
         Next ii
         If (flag = True) Then
            Open path$ & "BINNED_ENERGY_DEPOSITION-" & i & ".TXT" For Output As
#2
            For iii = 0 To UBound(WeightedPHABins) Step 1
               Print #2, iii & Chr$(9) & WeightedPHABins(iii)
            Next iii
            Close #2
         End If
      Next i
      For ii = 0 To UBound(TotalWeightedPHABins) Step 1
         Print #1, ii & Chr$(9) & TotalWeightedPHABins(ii)
      Next ii
   End If
   Close #1
End Sub
```

(f) TrimAutomation.bas

```
Option Explicit

'*************************************************************************************
'*                                                                                   *
'*                                TrimAutomation.bas                                 *
'*                                                                                   *
'*        All the code for automating TRIM.  Input from the main form                *
'*                          (MicromegasSim.frm) is used.                             *
'*                                                                                   *
'* This module is designed to be easily edited by the user so that different        *
'* types of things that aren't done right now can be done relatively easily         *
'* (e.g. square electrode pixelation for 2D neutron position sensitivity).          *
'*                                                                                   *
'*************************************************************************************

'*************************************************************************************
'*                          WHAT THIS MODULE'S CODE DOES:                            *
'*                                                                                   *
'*  The code in this module is a collection of the code that pertains to             *
'*  automating Trim.  It is not intended to create the initial particles or          *
'*  to analyze the data they produce when run through Trim; rather, it is            *
'*  intended to automate the running of Trim input decks and keep following          *
'*  all the particles until they either stop or completely escape all the            *
'*  material layers in the simulation.  As the code goes along it stores             *
'*  certain information in files for later analysis.  The information relates        *
'*  primarily to the deposition of energy in the layer that is defined by the        *
'*  user to be the layer of interest, this being what was needed to fulfill          *
'*  the original purpose of writing this code.  However, the code can be             *
'*  relatively easily modified to store additional or alternate information          *
'*  from the Trim runs if desired.                                                   *
'*************************************************************************************


'*************************************************************************************
'*                                                                                   *
'*            Code Pertaining to Running TRIM as a Shell Process                     *
'*                                                                                   *
'*************************************************************************************

'The next section on running a shell process and then detecting when it is
finished is
'taken from and/or patterned on example Q129796 from the Microsoft Knowledge
Base.

Private Type STARTUPINFO
   cb As Long
   lpReserved As String
   lpDesktop As String
   lpTitle As String
   dwX As Long
   dwY As Long
   dwXSize As Long
   dwYSize As Long
   dwXCountChars As Long
   dwYCountChars As Long
   dwFillAttribute As Long
   dwFlags As Long
   wShowWindow As Integer
   cbReserved2 As Integer
   lpReserved2 As Long
   hStdInput As Long
   hStdOutput As Long
   hStdError As Long
End Type

Private Type PROCESS_INFORMATION
   hProcess As Long
   hThread As Long
   dwProcessID As Long
   dwThreadID As Long
End Type
```

```
Private Declare Function WaitForSingleObject Lib "kernel32" (ByVal hHandle As
Long, ByVal dwMilliseconds As Long) As Long
Private Declare Function CreateProcessA Lib "kernel32" (ByVal lpApplicationName
As String, ByVal lpCommandLine As String, _
ByVal lpProcessAttributes As Long, ByVal lpThreadAttributes As Long, ByVal
bInheritHandles As Long, ByVal dwCreationFlags _
As Long, ByVal lpEnvironment As Long, ByVal lpCurrentDirectory As String,
lpStartupInfo As STARTUPINFO, lpProcessInformation _
As PROCESS_INFORMATION) As Long
Private Declare Function CloseHandle Lib "kernel32" (ByVal hObject As Long) As
Long
Private Declare Function GetExitCodeProcess Lib "kernel32" (ByVal hProcess As
Long, lpExitCode As Long) As Long

Private Const NORMAL_PRIORITY_CLASS = &H20&
Private Const INFINITE = -1&

Public Function ExecCmd(cmdline$)
   Dim proc As PROCESS_INFORMATION
   Dim start As STARTUPINFO
   Dim ret&

   ' Initialize the STARTUPINFO structure:
   start.cb = Len(start)

   ' Start the shelled application:
   ret& = CreateProcessA(vbNullString, cmdline$, 0&, 0&, 1&,
NORMAL_PRIORITY_CLASS, 0&, vbNullString, start, proc)

   ' Wait for the shelled application to finish:
   ret& = WaitForSingleObject(proc.hProcess, INFINITE)
   Call GetExitCodeProcess(proc.hProcess, ret&)
   Call CloseHandle(proc.hThread)
   Call CloseHandle(proc.hProcess)
   ExecCmd = ret&
End Function

Public Sub Run_TRIM()
   Dim retval As Long
   retval = ExecCmd(path$ & "trim_win_regular")
End Sub

Public Sub Run_Modified_Trim()
   Dim retval As Long
   retval = ExecCmd(path$ & "trim_win_modified")
End Sub

'*****************************************************************************
'*                                                                           *
'*                    Code Pertaining to Automating TRIM                     *
'*                                                                           *
'*****************************************************************************

Public Sub NewRunCase()
   Dim E As Single
   Dim x As Single
   Dim y As Single
   Dim z As Single
   Dim temp$
   Dim temp2$
   Dim i As Long
   Dim ii As Long
   Dim iii As Long
   Dim iv As Long
   'Dim v As Long 'Declared elsewhere as a global so it can be accessed in the
   'AnalyzeTrimData module.
   Dim vi As Long
   Dim atomicnumber As Single
   Dim fullpath$
   Dim spacestring$
   Dim currentlayer As Long
   Dim exittype$
   Dim dummy As Long
   Dim dummy2 As Long
   Dim strlength As Long
   Dim spacepos As Long
   Dim quitflag As Boolean
```

```
   Dim negativeflag As Boolean
   Dim mainloopflag As Boolean
   Dim ionsintolayer() As Boolean
   Dim particlenum As Long
   Dim currproduct As Long
   Dim flag As Boolean
   Dim readZ As Long
   Dim RandomSeed As Long
   Dim elic As Single
   Dim highestnumber As Long
   Dim previousparticlenum As Long
   Dim LeftLayer() As Boolean
   Dim eventnumber As Long
   Dim full() As Boolean
   Dim Dfraction As Single
   Dim Efraction As Single

   Dim xx1 As Single
   Dim yy1 As Single
   Dim zz1 As Single
   Dim EE1 As Single
   Dim xx2 As Single
   Dim yy2 As Single
   Dim zz2 As Single
   Dim EE2 As Single
   Dim xx3 As Single
   Dim yy3 As Single
   Dim zz3 As Single
   Dim EE3 As Single

   ReDim ionsintolayer(NumberOfLayers)
   ReDim ParticleIdentifier(2, NumberOfRuns, 2)
   ReDim ReadIn(NumberOfRuns)
   ReDim ParticlesLayerOfInterest(NumberOfRuns, 2, 4)

   For i = 1 To NumberOfRuns Step 1
      ParticleIdentifier(1, i, 1) = TargetLayer
      ParticleIdentifier(1, i, 2) = i
      ParticleIdentifier(2, i, 1) = TargetLayer
      ParticleIdentifier(2, i, 2) = i
   Next i

   currentlayer = TargetLayer
   'Run this first layer - the neutron target material - to see what escapes.
   'Call the command to make TRIM_WIN run using the input deck.

   'Before running any input decks, get rid of all old files.
   'File naming scheme is INPUT_x.IN, TRANSMIT_x.OUT, and BACKSCAT_x.OUT, where
x is
   'the layer number.

   For i = 1 To NumberOfLayers Step 1
      fullpath$ = path$ & "INPUT_" & i & ".IN"
      If Dir$(fullpath$) = "" Then
         'File doesn't exist.
      Else
         'File does exist.
         Kill (fullpath$)
      End If
      fullpath$ = path$ & "TRANSMIT_" & i & ".OUT"
      If Dir$(fullpath$) = "" Then
         'File doesn't exist.
      Else
         'File does exist.
         Kill (fullpath$)
      End If
      fullpath$ = path$ & "BACKSCAT_" & i & ".OUT"
      If Dir$(fullpath$) = "" Then
         'File doesn't exist.
      Else
         'File does exist.
         Kill (fullpath$)
      End If
      fullpath$ = path$ & "EXYZ_" & i & ".TXT"
      If Dir$(fullpath$) = "" Then
         'File doesn't exist.
      Else
```

```
         'File does exist.
         Kill (fullpath$)
      End If
   Next i

   DoEvents

   'Get rid of old particle data files.
   For i = 1 To 20 Step 1
      For ii = 1 To 2 Step 1
         fullpath$ = path$ & "LOI_PARTICLE_DATA_" & i & "-" & ii & ".OUT"
         If Dir$(fullpath$) = "" Then
            'The file doesn't exist.
         Else
            'The file does exist.
            Kill (fullpath$)
         End If
      Next ii
   Next i

   DoEvents

   For currproduct = 1 To 2 Step 1
      'Prepare to run a reaction product.
      mainloopflag = False

      If (currproduct = 1) Then
         'Do the first reaction product.
         FileCopy path$ & "TRIM1.DAT", path$ & "INPUT_" & currentlayer & ".IN"
      ElseIf (currproduct = 2) Then
         'Do the second reaction product.
         FileCopy path$ & "TRIM2.DAT", path$ & "INPUT_" & currentlayer & ".IN"
      Else
         MsgBox "Non-existent reaction product selected.", , "Error:"
      End If

      ChDir path$
      v = 0

      Do While (mainloopflag = False)
         RandomSeed = Int(Rnd * 999999#)
         'Run existing input decks and save their output in temporary files.
         For i = 1 To NumberOfLayers Step 1
            fullpath$ = path$ & "INPUT_" & i & ".IN"
            If Dir$(fullpath$) = "" Then
               'The file doesn't exist.
            Else
               'The file does exist.
               If (i = LayerOfInterest And GeneratePulse = True) Then
                  If (currproduct = 1) Then
                     'Do the first reaction product.
                     Call SetupTrim_Win_In(P1Z, P1M, False, LayerMaterial(i),
LayerThickness(i), RandomSeed)
                  ElseIf (currproduct = 2) Then
                     'Do the second reaction product.
                     Call SetupTrim_Win_In(P2Z, P2M, False, LayerMaterial(i),
LayerThickness(i), RandomSeed)
                  Else
                     MsgBox "Non-existent reaction product selected.", ,
"Error:"
                  End If
                  FileCopy fullpath$, path$ & "TRIM.DAT"
                  ChDir path$
                  Call Run_TRIM

                  DoEvents

                  FileCopy path$ & "BACKSCAT.TXT", path$ & "BACKSCAT_" & i &
".OUT"
                  FileCopy path$ & "TRANSMIT.TXT", path$ & "TRANSMIT_" & i &
".OUT"

                  'The code below sets up Trim to run again and produce the
file
                  'exyz.txt, a file that gives (position, energy) points for
the
                  'reaction products travelling through the detection layer.
```

```
                  If (currproduct = 1) Then
                     'Do the first reaction product.
                     Call SetupModifiedTrim_Win_In(P1Z, P1M, False,
LayerMaterial(i), LayerThickness(i), RandomSeed)
                  ElseIf (currproduct = 2) Then
                     'Do the second reaction product.
                     Call SetupModifiedTrim_Win_In(P2Z, P2M, False,
LayerMaterial(i), LayerThickness(i), RandomSeed)
                  Else
                     MsgBox "Non-existent reaction product selected.", ,
"Error:"
                  End If
                  ChDir path$
                  Call Run_Modified_Trim

                  DoEvents

                  FileCopy path$ & "EXYZ.TXT", path$ & "EXYZ_" & i & ".TXT"
               Else
                  If (currproduct = 1) Then
                     'Do the first reaction product.
                     Call SetupTrim_Win_In(P1Z, P1M, False, LayerMaterial(i),
LayerThickness(i), RandomSeed)
                  ElseIf (currproduct = 2) Then
                     'Do the second reaction product.
                     Call SetupTrim_Win_In(P2Z, P2M, False, LayerMaterial(i),
LayerThickness(i), RandomSeed)
                  Else
                     MsgBox "Non-existent reaction product selected.", ,
"Error:"
                  End If
                  FileCopy fullpath$, path$ & "TRIM.DAT"
                  ChDir path$
                  Call Run_TRIM

                  DoEvents

                  FileCopy path$ & "BACKSCAT.TXT", path$ & "BACKSCAT_" & i &
".OUT"
                  FileCopy path$ & "TRANSMIT.TXT", path$ & "TRANSMIT_" & i &
".OUT"
               End If
            End If
         Next i

         'Zero out all the ParticlesLayerOfInterest() values.
         ReDim ParticlesLayerOfInterest(NumberOfRuns, 2, 4)

         If (GeneratePulse = False) Then
            'Gather data from input and output decks on particle energy losses
in the
            'layer of interest.
            'Do the input deck first.
            fullpath$ = path$ & "INPUT_" & LayerOfInterest & ".IN"
            If Dir$(fullpath$) = "" Then
               'There wasn't any input for the layer of interest this time.
            Else
               Open fullpath$ For Input As #1
               For i = 1 To 10 Step 1
                  Line Input #1, temp$
               Next i
               'The program will think it is hitting a final particle right
before it
               'hits the end of file but I think it will make all the variables
be
               'zero and be okay.  I need to check this to be sure.
               Do While (EOF(1) = False)
                  Input #1, particlenum, readZ, E, xposition, yposition,
zposition, x, y, z
                  ParticlesLayerOfInterest(particlenum, 0, 1) = xposition
                  ParticlesLayerOfInterest(particlenum, 0, 2) = yposition
                  ParticlesLayerOfInterest(particlenum, 0, 3) = zposition
                  ParticlesLayerOfInterest(particlenum, 0, 4) = E
               Loop
               Close #1
            End If
            DoEvents
```

```
            'Do the TRANSMIT output file next.
            fullpath$ = path$ & "TRANSMIT_" & LayerOfInterest & ".OUT"
            If Dir$(fullpath$) = "" Then
               'The file doesn't exist.
            Else
               Open fullpath$ For Input As #1
               For i = 1 To 12 Step 1
                  Line Input #1, temp$
               Next i
               'See note above.
               Do While (EOF(1) = False)
                  Input #1, dummy, particlenum, readZ, E, xposition, yposition,
zposition, x, y, z
                  ParticlesLayerOfInterest(particlenum, 2, 1) = xposition
                  ParticlesLayerOfInterest(particlenum, 2, 2) = yposition
                  ParticlesLayerOfInterest(particlenum, 2, 3) = zposition
                  ParticlesLayerOfInterest(particlenum, 2, 4) = E
               Loop
               Close #1
            End If
            DoEvents
            'Do the BACKSCAT output file next.
            fullpath$ = path$ & "BACKSCAT_" & LayerOfInterest & ".OUT"
            If Dir$(fullpath$) = "" Then
               'The file doesn't exist.
            Else
               Open fullpath$ For Input As #1
               For i = 1 To 12 Step 1
                  Line Input #1, temp$
               Next i
               'See note above.
               Do While (EOF(1) = False)
                  Input #1, dummy, particlenum, readZ, E, dummy2, xposition,
yposition, zposition, x, y, z
                  ParticlesLayerOfInterest(particlenum, 1, 1) = xposition
                  ParticlesLayerOfInterest(particlenum, 1, 2) = yposition
                  ParticlesLayerOfInterest(particlenum, 1, 3) = zposition
                  ParticlesLayerOfInterest(particlenum, 1, 4) = E
               Loop
               Close #1
            End If
            DoEvents
            'Now put all the data into a file so I can see if it is working
correctly.
            fullpath$ = path$ & "LOI_PARTICLE_DATA_" & v & "-" & currproduct &
".OUT"
            Open fullpath$ For Output As #1
            flag = False
            For i = 1 To NumberOfRuns Step 1
               If (ParticlesLayerOfInterest(i, 0, 4) > 0#) Then
                  For ii = 1 To NumberOfRuns Step 1
                     If (ParticleIdentifier(currproduct, ii, 1) =
LayerOfInterest) And (ParticleIdentifier(currproduct, ii, 2) = i) Then
                        particlenum = ii
                     End If
                  Next ii
                  If ParticlesLayerOfInterest(i, 1, 4) > 0# Then
                     Print #1, particlenum, (ParticlesLayerOfInterest(i, 0, 4)
- ParticlesLayerOfInterest(i, 1, 4))
                  ElseIf ParticlesLayerOfInterest(i, 2, 4) > 0# Then
                     Print #1, particlenum, (ParticlesLayerOfInterest(i, 0, 4)
- ParticlesLayerOfInterest(i, 2, 4))
                  Else
                     Print #1, particlenum, ParticlesLayerOfInterest(i, 0, 4)
                  End If
                  flag = True
               End If
            Next i
            Close #1
            DoEvents
            'Trash the datafile if it is empty.
            If (flag = False) Then
               Kill (fullpath$)
            End If
         ElseIf (GeneratePulse = True) Then
            xx1 = 0#
            yy1 = 0#
```

```
            zz1 = 0#
            EE1 = 0#
            xx2 = 0#
            yy2 = 0#
            zz2 = 0#
            EE2 = 0#
            xx3 = 0#
            yy3 = 0#
            zz3 = 0#
            EE3 = 0#

            'Gather data from input and output decks on particle energy losses
in the
            'layer of interest.
            'Do the input deck first.
            fullpath$ = path$ & "INPUT_" & LayerOfInterest & ".IN"
            If Dir$(fullpath$) = "" Then
               'There wasn't any input for the layer of interest this time.
            Else
               'The first subscript in the redim statement actually only needs
               'to be however many particles were run in that particular input
               'file.  This could be added later by doing a Redim Preserve on
               'ParticlesLayerOfInterest() once the number of actual particles
               'is known and their information is loaded into the array.
               ReDim ParticlesLayerOfInterest(NumberOfRuns, 2, 4)
               'We are going to do this differently from before.  1 in the
second
               'subscript will give the values upon entering the layer and 2 in
the
               'second subscript will give the values upon exiting the layer.

               ReDim LeftLayer(NumberOfRuns)

               'Open the particle transmission file.
               Open path$ & "TRANSMIT_" & LayerOfInterest & ".OUT" For Input As
#1
               For i = 1 To 12 Step 1
                  Line Input #1, temp$
               Next i
               'The program will think it is hitting a final particle right
before it
               'hits the end of file but it will make all the variables be zero
and be
               'okay.
               Do While (EOF(1) = False)
                  Input #1, dummy, particlenum, readZ, E, xposition, yposition,
zposition, x, y, z

                  ParticlesLayerOfInterest(particlenum, 2, 1) =
LayerThickness(LayerOfInterest)
                  ParticlesLayerOfInterest(particlenum, 2, 2) = yposition
                  ParticlesLayerOfInterest(particlenum, 2, 3) = zposition
                  ParticlesLayerOfInterest(particlenum, 2, 4) = E
                  LeftLayer(particlenum) = True
               Loop
               Close #1
               DoEvents
               'Open the particle backscatter file.
               Open path$ & "BACKSCAT_" & LayerOfInterest & ".OUT" For Input As
#1
               For i = 1 To 12 Step 1
                  Line Input #1, temp$
               Next i
               Do While (EOF(1) = False)
                  Input #1, dummy, particlenum, readZ, E, dummy2, xposition,
yposition, zposition, x, y, z

                  ParticlesLayerOfInterest(particlenum, 2, 1) = 0#
                  ParticlesLayerOfInterest(particlenum, 2, 2) = yposition
                  ParticlesLayerOfInterest(particlenum, 2, 3) = zposition
                  ParticlesLayerOfInterest(particlenum, 2, 4) = E
                  LeftLayer(particlenum) = True
               Loop
               Close #1
               DoEvents

               'FORMAT FOR LOI_PARTICLE_DATA_x.OUT
```

```
            'It is being changed a bit from what it used to be when full
pulses are
            'generated.  The new format is event number, xstart, ystart,
zstart,
            'xfinish, yfinish, zfinish, energy loss (eV) between those two
positions

            'Open the LOI energy deposition file for writing data to.
            Open path$ & "LOI_PARTICLE_DATA_" & v & "-" & currproduct &
".OUT" For Output As #2
            'Open the EXYZ.TXT file.  This gives both the point of entry and
the
            'steps along the way.
            Open path$ & "EXYZ_" & LayerOfInterest & ".TXT" For Input As #1
            For i = 1 To 15 Step 1
               Line Input #1, temp$
            Next i

            previousparticlenum = 0
            ReDim full(3)

            Do While (EOF(1) = False)
               'STRATEGY: Read in data, wait until the end of the data for a
specific
               'particle is reached, and then write all the data for that
particle
               'to the output file.
               Input #1, particlenum, EE1, xx1, yy1, zz1, elic
               EE1 = EE1 * 1000#

               If (particlenum <> previousparticlenum) Then
                  'Get the exit position and energy (if it exists) and write
it to
                  'the file (if there was a previous particle).
                  If (previousparticlenum > 0) Then
                     If (LeftLayer(previousparticlenum) = True) Then
                        If ((EE2 -
ParticlesLayerOfInterest(previousparticlenum, 2, 4)) > 0#) Then
                           Print #2, eventnumber & Chr$(9) & xx2 & Chr$(9) &
yy2 & Chr$(9) & zz2 & Chr$(9) & ParticlesLayerOfInterest(previousparticlenum,
2, 1) & Chr$(9) & ParticlesLayerOfInterest(previousparticlenum, 2, 2) & Chr$(9)
& ParticlesLayerOfInterest(previousparticlenum, 2, 3) & Chr$(9) & (EE2 -
ParticlesLayerOfInterest(previousparticlenum, 2, 4))
                        End If

                        If ((EE2 -
ParticlesLayerOfInterest(previousparticlenum, 2, 4)) < 0) Then
                           'The program is ending right here.
                           'End
                        End If
                     Else
                        'If there isn't an exit location, extrapolate a
finishing point
                        'from the previous position data (again assuming
that there was a
                        'previous particle).  This requires the previous two
(position,
                        'energy) data sets.

                        'The TWO points prior to exiting (the first might be
the entry
                        'point) are both needed in order to do an
extrapolation.  Currently
                        'we are only saving one in a variable - that could
be changed or
                        'some other scheme could be devised.

                        'NOTE: I now know that I can get end points for
particles from
                        'backscat.txt.  The code can (eventually should) be
changed to
                        'use this.

                        If (full(3) = True) Then
                           E = 0#
                           Efraction = (EE2 - E) / (EE3 - EE2)
                           x = xx2 + (xx2 - xx3) * Efraction
```

```
                              y = yy2 + (yy2 - yy3) * Efraction
                              z = zz2 + (zz2 - zz3) * Efraction

                              'Now put in something to check and correct if the
particle
                              'goes out of bounds.
                              If (x > LayerThickness(LayerOfInterest)) Then
                                 Dfraction = (LayerThickness(LayerOfInterest) -
xx2) / (x - xx2)
                                 x = LayerThickness(LayerOfInterest)
                                 y = yy2 + Dfraction * (yy2 - yy3) * Efraction
                                 z = zz2 + Dfraction * (zz2 - zz3) * Efraction
                                 E = EE2 - Dfraction * (EE3 - EE2)
                              ElseIf (x < 0#) Then
                                 Dfraction = (0# - xx2) / (x - xx2)
                                 x = 0#
                                 y = yy2 + Dfraction * (yy2 - yy3) * Efraction
                                 z = zz2 + Dfraction * (zz2 - zz3) * Efraction
                                 E = EE2 - Dfraction * (EE3 - EE2)
                              End If

                              If ((EE2 - E) > 0#) Then
                                 Print #2, eventnumber & Chr$(9) & xx2 &
Chr$(9) & yy2 & Chr$(9) & zz2 & Chr$(9) & x & Chr$(9) & y & Chr$(9) & z &
Chr$(9) & (EE2 - E)
                              End If

                              If (EE2 - E) < 0 Then
                                 End
                              End If
                           Else
                              'We don't have and can't get enough points to
extrapolate
                              'it with.  We'll just have to forget about it and
drop it.
                           End If
                        End If
                     Else
                        'There isn't any data to save for previous particle
since this
                        'is the first particle.
                     End If

                     DoEvents

                     For i = 1 To NumberOfRuns Step 1
                        For ii = 1 To NumberOfRuns Step 1
                           If (ParticleIdentifier(currproduct, ii, 1) =
LayerOfInterest) And (ParticleIdentifier(currproduct, ii, 2) = particlenum)
Then
                              eventnumber = ii
                           End If
                        Next ii
                     Next i

                     'Start the next particle.
                     ReDim full(3)

                     'I changed the second subscript from 2 to 1 in all of the
array
                     'calls immediately following this.
                     If (xposition < 0#) Then
                        ParticlesLayerOfInterest(particlenum, 1, 1) = 0#
                     ElseIf (xposition > LayerThickness(LayerOfInterest)) Then
                        ParticlesLayerOfInterest(particlenum, 1, 1) =
LayerThickness(LayerOfInterest)
                     Else
                        ParticlesLayerOfInterest(particlenum, 1, 1) = xx1
                     End If
                     ParticlesLayerOfInterest(particlenum, 1, 2) = yy1
                     ParticlesLayerOfInterest(particlenum, 1, 3) = zz1
                     ParticlesLayerOfInterest(particlenum, 1, 4) = EE1

                     full(1) = True
                     previousparticlenum = particlenum
                  Else
                     'Add the position and energy data to the l
```

```
                     If ((EE2 - EE1) > 0#) Then
                        Print #2, eventnumber & Chr$(9) & xx2 & Chr$(9) & yy2 &
Chr$(9) & zz2 & Chr$(9) & xx1 & Chr$(9) & yy1 & Chr$(9) & zz1 & Chr$(9) & (EE2
- EE1)
                     End If

                     If (full(2) = True) Then
                        full(3) = True
                     End If
                     If (full(1) = True) Then
                        full(2) = True
                     End If

                     DoEvents
                  End If

                  EE3 = EE2
                  xx3 = xx2
                  yy3 = yy2
                  zz3 = zz2

                  EE2 = EE1
                  xx2 = xx1
                  yy2 = yy1
                  zz2 = zz1
               Loop
               Close #1
               Close #2
            End If
         End If

         'Set the ReadIn() values to false so they can be individually set to
true
         'when a particle corresponding to them is read in.
         For i = 1 To NumberOfRuns Step 1
            ReadIn(i) = False
         Next i
         'Begin creating the new input decks.
         For i = 1 To NumberOfLayers Step 1
            Open path$ & "TEMP_INPUT_" & i & ".IN" For Output As #i
         Next i
         'Insert boilerplate at the start of each file.
         Open path$ & "OLDTRIM.DAT" For Input As #510
         For i = 1 To 10 Step 1
            Line Input #510, temp$
            For iii = 1 To NumberOfLayers Step 1
               Print #iii, temp$
            Next iii
         Next i
         Close #510
         'Set all layers as not having any ions going into them until such ions
are found.
         For i = 1 To NumberOfLayers Step 1
            ionsintolayer(i) = False
         Next i
         ' Give CPU time to other things.
         DoEvents
         'Read in the output decks and use their data in the new input decks.
         For i = 1 To NumberOfLayers Step 1
            ii = 1
            If (i < NumberOfLayers) Then
               'Look for backscatters from the layer ahead of this layer.
               fullpath$ = path$ & "BACKSCAT_" & (i + 1) & ".OUT"
               If Dir$(fullpath$) = "" Then
                  'An output file for backscatters does not exist.
               Else
                  'An output file for backscatters exists (but it may be
empty).
                  Open fullpath$ For Input As #511
                  'Skip the first dozen lines as they are boilerplate type of
stuff.
                  For iii = 1 To 12 Step 1
                     Line Input #511, temp$
                  Next iii
                  vi = 0
                  Do While (EOF(511) = False)
                     'Read in the particle data.
```

```
                    Input #511, dummy, particlenum, readZ, E, dummy2,
xposition, yposition, zposition, x, y, z
                    'Note that dummy2 is added because there is a space
between the
                    'negative sign and the zposition number and the input
function
                    'doesn't realize these are supposed to be the same thing.

                    'Don't bother with particles that don't have at least 1
keV of
                    'energy.
                    If (E > 1000#) Then
                       'Find the old ParticleIdentifier() value for this
individual
                       'particle and update it.
                       For iv = 1 To NumberOfRuns Step 1
                          If (ParticleIdentifier(currproduct, iv, 1) = (i +
1)) And (ParticleIdentifier(currproduct, iv, 2) = particlenum) Then
                             ParticleIdentifier(currproduct, iv, 1) = i
                             ParticleIdentifier(currproduct, iv, 2) = ii
                             ReadIn(iv) = True
                             vi = vi + 1
                          End If
                       Next iv
                       'Write the particle data to the appropriate layer input
file.
                       spacestring$ = ""
                       If (ii < 10) Then
                          spacestring$ = "       "
                       ElseIf (ii < 100) Then
                          spacestring$ = "      "
                       ElseIf (ii < 1000) Then
                          spacestring$ = "     "
                       ElseIf (ii < 10000) Then
                          spacestring$ = "    "
                       ElseIf (ii < 100000) Then
                          spacestring$ = "   "
                       Else
                          spacestring$ = "  "
                       End If
                       'Write the particle info the new input file and update
variables.
                       If (currproduct = 1) Then
                          Print #i, ii & spacestring$ & P1Z & " " & E & " " &
LayerThickness(i) & " " & yposition & " " & zposition & " " & Format(x,
"0.0000") & " " & Format(y, "0.0000") & " " & Format(z, "0.0000")
                       ElseIf (currproduct = 2) Then
                          Print #i, ii & spacestring$ & P2Z & " " & E & " " &
LayerThickness(i) & " " & yposition & " " & zposition & " " & Format(x,
"0.0000") & " " & Format(y, "0.0000") & " " & Format(z, "0.0000")
                       Else
                          MsgBox "Unrecognized reaction product type.", ,
"Error:"
                          End
                       End If
                       ionsintolayer(i) = True
                       ii = ii + 1
                    Else
                       'Find the old ParticleIdentifier() value for this
individual
                       'particle and update it.
                       For iv = 1 To NumberOfRuns Step 1
                          If (ParticleIdentifier(currproduct, iv, 1) = (i +
1)) And (ParticleIdentifier(currproduct, iv, 2) = particlenum) Then
                             ParticleIdentifier(currproduct, iv, 1) = 0
                             ParticleIdentifier(currproduct, iv, 2) = 0
                             ReadIn(iv) = True
                             vi = vi + 1
                          End If
                       Next iv
                    End If
                 Loop
                 Close #511
              End If
           End If

           DoEvents
```

```
            If (i > 1) Then
               'Look for transmissions from the layer behind this layer.
               fullpath$ = path$ & "TRANSMIT_" & (i - 1) & ".OUT"
               If Dir$(fullpath$) = "" Then
                  'An output file for transmissions does not exist.
               Else
                  'An output file for transmissions exists (but it may be
empty).
                  Open fullpath$ For Input As #511
                  'Skip the first dozen lines as they are boilerplate type of
stuff.
                  For iii = 1 To 12 Step 1
                     Line Input #511, temp$
                  Next iii
                  vi = 0
                  Do While (EOF(511) = False)
                     'Read in the particle data.
                     Input #511, dummy, particlenum, readZ, E, xposition,
yposition, zposition, x, y, z
                     'Find the old ParticleIdentifier() value for this
individual
                     'particle and update it.
                     'Don't bother with particles that don't have at least 1
keV of
                     'energy.
                     If (E > 1000#) Then
                        For iv = 1 To NumberOfRuns Step 1
                           If (ParticleIdentifier(currproduct, iv, 1) = (i -
1)) And (ParticleIdentifier(currproduct, iv, 2) = particlenum) Then
                              ParticleIdentifier(currproduct, iv, 1) = i
                              ParticleIdentifier(currproduct, iv, 2) = ii
                              ReadIn(iv) = True
                              vi = vi + 1
                           End If
                        Next iv
                        'Write the particle data to the appropriate layer input
file.
                        spacestring$ = ""
                        If (ii < 10) Then
                           spacestring$ = "       "
                        ElseIf (ii < 100) Then
                           spacestring$ = "      "
                        ElseIf (ii < 1000) Then
                           spacestring$ = "     "
                        ElseIf (ii < 10000) Then
                           spacestring$ = "    "
                        ElseIf (ii < 100000) Then
                           spacestring$ = "   "
                        Else
                           spacestring$ = "  "
                        End If
                        'Write the particle info the new input file and update
variables.
                        If (currproduct = 1) Then
                           Print #i, ii & spacestring$ & P1Z & " " & E & " " &
0# & " " & yposition & " " & zposition & " " & Format(x, "0.0000") & " " &
Format(y, "0.0000") & " " & Format(z, "0.0000")
                        ElseIf (currproduct = 2) Then
                           Print #i, ii & spacestring$ & P2Z & " " & E & " " &
0# & " " & yposition & " " & zposition & " " & Format(x, "0.0000") & " " &
Format(y, "0.0000") & " " & Format(z, "0.0000")
                        Else
                           MsgBox "Unrecognized reaction product type.", ,
"Error:"
                           End
                        End If
                        ionsintolayer(i) = True
                        ii = ii + 1
                     Else
                        'Find the old ParticleIdentifier() value for this
individual
                        'particle and update it.
                        For iv = 1 To NumberOfRuns Step 1
                           If (ParticleIdentifier(currproduct, iv, 1) = (i +
1)) And (ParticleIdentifier(currproduct, iv, 2) = particlenum) Then
                              ParticleIdentifier(currproduct, iv, 1) = 0
```

```
                              ParticleIdentifier(currproduct, iv, 2) = 0
                              ReadIn(iv) = True
                              vi = vi + 1
                           End If
                        Next iv
                     End If
                  Loop
                  Close #511
               End If
            End If

            DoEvents
         Next i
         For i = 1 To NumberOfLayers Step 1
            Print #i, Chr$(26)
            Close #i
         Next i

         'Delete the old output decks.
         For i = 1 To NumberOfLayers Step 1
            fullpath$ = path$ & "BACKSCAT_" & i & ".OUT"
            If Dir$(fullpath$) = "" Then
               'File does not exist.
            Else
               Kill (fullpath$)
            End If
            fullpath$ = path$ & "TRANSMIT_" & i & ".OUT"
            If Dir$(fullpath$) = "" Then
               'File does not exist.
            Else
               Kill (fullpath$)
            End If
         Next i

         DoEvents

         'Delete all old input decks.
         For i = 1 To NumberOfLayers Step 1
            fullpath$ = path$ & "INPUT_" & i & ".IN"
            If Dir$(fullpath$) = "" Then
               'The file doesn't exist.
            Else
               'The file does exist.
               Kill (fullpath$)
            End If
         Next i

         DoEvents

         'NOTE: This whole use of temp_input_x.in was to deal with a particular
issue
         'involving collection and analysis of data for the particles
travelling through
         'the layer of interest.  I believe that I have another way to do that
now but
         'will leave the temp_input_x.in stuff alone for right now in case it's
needed
         'later for some reason.

         'Get rid of any input files that are empty, rename the others that
aren't.
         For i = 1 To NumberOfLayers Step 1
            fullpath$ = path$ & "TEMP_INPUT_" & i & ".IN"
            If (ionsintolayer(i) = False) Then
               Kill (fullpath$)
            Else
               Name fullpath$ As path$ & "INPUT_" & i & ".IN"
            End If
         Next i

         DoEvents

         'Zero out any particle identifier values for particles that are no
longer in
         'circulation (i.e. they have been stopped).
         For i = 1 To NumberOfRuns Step 1
            If (ReadIn(i) = False) Then
```

```
               ParticleIdentifier(currproduct, i, 1) = 0
               ParticleIdentifier(currproduct, i, 2) = 0
            End If
         Next i

         'See if there are any input files to run.  If not, exit the loop.
         mainloopflag = True
         For i = 1 To NumberOfLayers Step 1
            If ionsintolayer(i) = True Then
               mainloopflag = False
            End If
         Next

         v = v + 1
      Loop
   Next currproduct

   DoEvents

   'Delete all old input decks.
   For i = 1 To NumberOfLayers Step 1
      fullpath$ = path$ & "INPUT_" & i & ".IN"
      If Dir$(fullpath$) = "" Then
         'The file doesn't exist.
      Else
         'The file does exist.
         Kill (fullpath$)
      End If
   Next i

   DoEvents
End Sub

Public Sub SetupTrim_Win_In(ByVal z As Long, ByVal M As Single, ByVal
pulselayer As Boolean, ByVal MaterialType, ByVal Thickness As Long, ByVal
RandomSeed As Long)
   'Thickness needs to be in Angstroms (either that or put something in to
convert to Angstroms)
   Open path$ & "trim_win.in" For Output As #1

   Print #1, "==> SRIM-2000.41 <== TRIM.IN is for TRIM(DOS), TRIM(Win) uses
TRIM_WIN.IN ==="
   Print #1, "Ion: Z1 ,  M1,  Energy (keV), Angle,Number,Bragg Corr,AutoSave
Number."
   Print #1, "   " & z & "   " & M & "   20000   0   99999   0   10000"
   Print #1, "Cascades(1=No;2=Full;3=Sputt;4-5=Ions;6-7=Neutrons), Random
Number Seed, Reminders"
   Print #1, "    4   " & RandomSeed & "   0"
   Print #1, "Diskfiles (0=no,1=yes): Ranges, Backscatt, Transmit, Sputtered,
Recoils (2=Alt-C)"
   If (pulselayer = True) Then
      Print #1, "    1   1   1   0   2"
   Else
      Print #1, "    1   1   1   0   0"
   End If
   Print #1, "Target material : Number of Elements & Layers"
   '1 is B, 2 is Li, 3 is Al, 4 is Ar, 5 is 90% Ar/10% isobutane, 6 is P-10
(90% Ar/10% methane)
   Select Case MaterialType
   Case 1:
      Print #1, Chr$(34) & "Target into layer" & Chr$(34) & "    1    1"
      Print #1, "Target Energies (eV): Binding, Surface, Individual
Displacement"
      Print #1, "           3     2      20"
   Case 2:
      Print #1, Chr$(34) & "Target into layer" & Chr$(34) & "    1    1"
      Print #1, "Target Energies (eV): Binding, Surface, Individual
Displacement"
      Print #1, "           3     2      20"  'Change to appropriate values.
   Case 3:
      Print #1, Chr$(34) & "Target into layer" & Chr$(34) & "    1    1"
      Print #1, "Target Energies (eV): Binding, Surface, Individual
Displacement"
      Print #1, "           3     4      20"
   Case 4:
      Print #1, Chr$(34) & "Target into layer" & Chr$(34) & "    1    1"
```

```
      Print #1, "Target Energies (eV): Binding, Surface, Individual
Displacement"
      Print #1, "          3.3   3.5    13"
   Case 5:
      Print #1, Chr$(34) & "Target into layer" & Chr$(34) & "    3    1"
      Print #1, "Target Energies (eV): Binding, Surface, Individual
Displacement"
      Print #1, "           3.5   4      15     15     15"
   Case 6:
      Print #1, Chr$(34) & "Target into layer" & Chr$(34) & "    3    1"
      Print #1, "Target Energies (eV): Binding, Surface, Individual
Displacement"
      Print #1, "           3.5   4      15     15     15"
   Case 7:
      Print #1, Chr$(34) & "Target into layer" & Chr$(34) & "    2    1"
      Print #1, "Target Energies (eV): Binding, Surface, Individual
Displacement"
      Print #1, "           3.5   4      15     15"
   Case 8:
      Print #1, Chr$(34) & "Target into layer" & Chr$(34) & "    1    1"
      Print #1, "Target Energies (eV): Binding, Surface, Individual
Displacement"
      Print #1, "           3.5   4      15"
   Case 9:
      Print #1, Chr$(34) & "Target into layer" & Chr$(34) & "    2    1"
      Print #1, "Target Energies (eV): Binding, Surface, Individual
Displacement"
      Print #1, "           3.5   4      15"
   Case 10:
      Print #1, Chr$(34) & "Target into layer" & Chr$(34) & "    3    1"
      Print #1, "Target Energies (eV): Binding, Surface, Individual
Displacement"
      Print #1, "           3     3      15"
   Case 11:
      Print #1, Chr$(34) & "Target into layer" & Chr$(34) & "    3    1"
      Print #1, "Target Energies (eV): Binding, Surface, Individual
Displacement"
      Print #1, "           3     3      15"
   Case 12:
      Print #1, Chr$(34) & "Target into layer" & Chr$(34) & "    1    1"
      Print #1, "Target Energies (eV): Binding, Surface, Individual
Displacement"
      Print #1, "           3.5   4      15"
   Case 13:
      Print #1, Chr$(34) & "Target into layer" & Chr$(34) & "    1    1"
      Print #1, "Target Energies (eV): Binding, Surface, Individual
Displacement"
      Print #1, "           3     2      20"
   End Select

   Print #1, "PlotType (0-5); Plot Depths: Xmin, Xmax(Ang.) [=0 0 for Viewing
Full Target]"
   Print #1, "       5                          0            0"
   If (MaterialType = 1) Then
      Print #1, "Target Elements:    Z Mass(amu)"
      Print #1, "Atom 1 = B  =        5  10.0"
      Print #1, "Layer   Layer Name /               Width Density    B(5)"
      Print #1, "Numb.   Description                 (Ang) (g/cm3)    Stoich"
      Print #1, " 1      " & Chr$(34) & "Boron" & Chr$(34) & "          " &
Str$(Thickness) & "  2.174        1"
      Print #1, "0  Target layer phases (0=Solid, 1=Gas)"
      Print #1, "0"
      Print #1, "Target Compound Corrections (Bragg)"
      Print #1, "0"
      Print #1, "Target atom displacement energies (eV)"
      Print #1, "20"
      Print #1, "Target atom lattice binding energies (eV)"
      Print #1, "3"
      Print #1, "Target atom surface binding energies (eV)"
      Print #1, "2"
   ElseIf (MaterialType = 2) Then
      Print #1, "Target Elements:    Z Mass(amu)"
      Print #1, "Atom 1 = Li =        3   6.0"
      Print #1, "Layer   Layer Name /               Width Density    Li(3)"
      Print #1, "Numb.   Description                 (Ang) (g/cm3)    Stoich"
      Print #1, " 1      " & Chr$(34) & "Lithium" & Chr$(34) & "          " &
Str$(Thickness) & "  0.462        1"
```

```
      Print #1, "0  Target layer phases (0=Solid, 1=Gas)"
      Print #1, "0"
      Print #1, "Target Compound Corrections (Bragg)"
      Print #1, "0"
      Print #1, "Target atom displacement energies (eV)"
      Print #1, "20"
      Print #1, "Target atom lattice binding energies (eV)"
      Print #1, "3"
      Print #1, "Target atom surface binding energies (eV)"
      Print #1, "2"
   ElseIf (MaterialType = 3) Then
      Print #1, "Target Elements:    Z Mass(amu)"
      Print #1, "Atom 1 = Al =       13  26.982"
      Print #1, "Layer   Layer Name /               Width Density    Al(13)"
      Print #1, "Numb.   Description                 (Ang) (g/cm3)    Stoich"
      Print #1, " 1      " & Chr$(34) & "Aluminium" & Chr$(34) & "          " &
Str$(Thickness) & "  2.702        1"
      Print #1, "0  Target layer phases (0=Solid, 1=Gas)"
      Print #1, "0"
      Print #1, "Target Compound Corrections (Bragg)"
      Print #1, "0"
      Print #1, "Target atom displacement energies (eV)"
      Print #1, "20"
      Print #1, "Target atom lattice binding energies (eV)"
      Print #1, "3"
      Print #1, "Target atom surface binding energies (eV)"
      Print #1, "4"
   ElseIf (MaterialType = 4) Then
      Print #1, "Target Elements:    Z Mass(amu)"
      Print #1, "Atom 1 = Ar  =       18  39.948"
      Print #1, "Layer   Layer Name /               Width Density    Ar(18)"
      Print #1, "Numb.   Description                 (Ang) (g/cm3)    Stoich"
      Print #1, " 1      " & Chr$(34) & "Argon gas" & Chr$(34) & "          " &
Str$(Thickness) & "  .00166        1.0"
      Print #1, "0  Target layer phases (0=Solid, 1=Gas)"
      Print #1, "1"
      Print #1, "Target Compound Corrections (Bragg)"
      Print #1, "0"
      Print #1, "Target atom displacement energies (eV)"
      Print #1, "       13"
      Print #1, "Target atom lattice binding energies (eV)"
      Print #1, "       3.3"
      Print #1, "Target atom surface binding energies (eV)"
      Print #1, "       3.5"
   ElseIf (MaterialType = 5) Then
      Print #1, "Target Elements:    Z Mass(amu)"
      Print #1, "Atom 1 = Ar  =       18  39.948"
      Print #1, "Atom 2 = C   =        6  12.011"
      Print #1, "Atom 3 = H   =        1   1.008"
      Print #1, "Layer   Layer Name /               Width Density    Ar(18)
C(6)     H(1)"
      Print #1, "Numb.   Description                 (Ang) (g/cm3)    Stoich
Stoich   Stoich"
      Print #1, " 1      " & Chr$(34) & "Ar/isobutane gas" & Chr$(34) & "
" & Str$(Thickness) & "  .00175        .391304  .173913  .434783"
      Print #1, "0  Target layer phases (0=Solid, 1=Gas)"
      Print #1, "1"
      Print #1, "Target Compound Corrections (Bragg)"
      Print #1, "0"
      Print #1, "Target atom displacement energies (eV)"
      Print #1, "       13     14     15"
      Print #1, "Target atom lattice binding energies (eV)"
      Print #1, "       3.3    3.4    3.5"
      Print #1, "Target atom surface binding energies (eV)"
      Print #1, "       3.5    4      4.5"
   ElseIf (MaterialType = 6) Then
      Print #1, "Target Elements:    Z Mass(amu)"
      Print #1, "Atom 1 = Ar  =       18  39.948"
      Print #1, "Atom 2 = C   =        6  12.011"
      Print #1, "Atom 3 = H   =        1   1.008"
      Print #1, "Layer   Layer Name /               Width Density    Ar(18)
C(6)     H(1)"
      Print #1, "Numb.   Description                 (Ang) (g/cm3)    Stoich
Stoich   Stoich"
      Print #1, " 1      " & Chr$(34) & "P-10 gas" & Chr$(34) & "          " &
Str$(Thickness) & "  .00167        .642857  .071429  .285714"
      Print #1, "0  Target layer phases (0=Solid, 1=Gas)"
```

```
      Print #1, "1"
      Print #1, "Target Compound Corrections (Bragg)"
      Print #1, "0"
      Print #1, "Target atom displacement energies (eV)"
      Print #1, "      13     14     15"
      Print #1, "Target atom lattice binding energies (eV)"
      Print #1, "      3.3    3.4    3.5"
      Print #1, "Target atom surface binding energies (eV)"
      Print #1, "      3.5    4      4.5"
   ElseIf (MaterialType = 7) Then
      Print #1, "Target Elements:    Z Mass(amu)"
      Print #1, "Atom 1 = Li  =        3   6.0"
      Print #1, "Atom 2 = F   =        9  18.998"
      Print #1, "Layer   Layer Name /               Width Density    Li(18)
F(6)"
      Print #1, "Numb.   Description                 (Ang) (g/cm3)    Stoich
Stoich"
      Print #1, " 1      " & Chr$(34) & "LiF solid" & Chr$(34) & "         " &
Str$(Thickness) & "  .786          .50      .50"
      Print #1, "0  Target layer phases (0=Solid, 1=Gas)"
      Print #1, "0"
      Print #1, "Target Compound Corrections (Bragg)"
      Print #1, "0"
      Print #1, "Target atom displacement energies (eV)"
      Print #1, "      15     15"
      Print #1, "Target atom lattice binding energies (eV)"
      Print #1, "      3.5    3.5"
      Print #1, "Target atom surface binding energies (eV)"
      Print #1, "      4      4"
   ElseIf (MaterialType = 8) Then
      Print #1, "Target Elements:    Z Mass(amu)"
      Print #1, "Atom 1 = Si  =       14  28.086"
      Print #1, "Layer   Layer Name /               Width Density    Si(14)"
      Print #1, "Numb.   Description                 (Ang) (g/cm3)    Stoich"
      Print #1, " 1      " & Chr$(34) & "Si solid" & Chr$(34) & "          " &
Str$(Thickness) & "  2.321          1.00"
      Print #1, "0  Target layer phases (0=Solid, 1=Gas)"
      Print #1, "0"
      Print #1, "Target Compound Corrections (Bragg)"
      Print #1, "0"
      Print #1, "Target atom displacement energies (eV)"
      Print #1, "      15"
      Print #1, "Target atom lattice binding energies (eV)"
      Print #1, "      3.5"
      Print #1, "Target atom surface binding energies (eV)"
      Print #1, "      4"
   ElseIf (MaterialType = 9) Then
      Print #1, "Target Elements:    Z Mass(amu)"
      Print #1, "Atom 1 = Si  =       14  28.086"
      Print #1, "Atom 2 = O   =        8  15.999"
      Print #1, "Layer   Layer Name /               Width Density    Si(14)
O(8)"
      Print #1, "Numb.   Description                 (Ang) (g/cm3)    Stoich
Stoich"
      Print #1, " 1      " & Chr$(34) & "glass" & Chr$(34) & "          " &
Str$(Thickness) & "  1.724          .333      .667"
      Print #1, "0  Target layer phases (0=Solid, 1=Gas)"
      Print #1, "0"
      Print #1, "Target Compound Corrections (Bragg)"
      Print #1, "0"
      Print #1, "Target atom displacement energies (eV)"
      Print #1, "      15     15"
      Print #1, "Target atom lattice binding energies (eV)"
      Print #1, "      3.5    3.5"
      Print #1, "Target atom surface binding energies (eV)"
      Print #1, "      4      4"
   ElseIf (MaterialType = 10) Then
      Print #1, "Target Elements:    Z Mass(amu)"
      Print #1, "Atom 1 = Si  =       14  28.086"
      Print #1, "Atom 2 = O   =        8  15.999"
      Print #1, "Atom 3 = B   =        5  10.0"
      Print #1, "Layer   Layer Name /               Width Density    Si(14)
O(8)     B(5)"
      Print #1, "Numb.   Description                 (Ang) (g/cm3)    Stoich
Stoich   Stoich"
      Print #1, " 1      " & Chr$(34) & "doped glass solid" & Chr$(34) & "
" & Str$(Thickness) & "  2.017          .1776      .3553      .4671"
```

```
      Print #1, "0  Target layer phases (0=Solid, 1=Gas)"
      Print #1, "0"
      Print #1, "Target Compound Corrections (Bragg)"
      Print #1, "0"
      Print #1, "Target atom displacement energies (eV)"
      Print #1, "       15     15     20"
      Print #1, "Target atom lattice binding energies (eV)"
      Print #1, "       3.5    3.5    3"
      Print #1, "Target atom surface binding energies (eV)"
      Print #1, "       4      4      2"
   ElseIf (MaterialType = 11) Then
      Print #1, "Target Elements:    Z Mass(amu)"
      Print #1, "Atom 1 = Si  =       14  28.086"
      Print #1, "Atom 2 = O   =        8  15.999"
      Print #1, "Atom 4 = Li  =        3   6.0"
      Print #1, "Layer   Layer Name /              Width Density    Si(14)
O(8)     Li(3)"
      Print #1, "Numb.   Description               (Ang) (g/cm3)    Stoich
Stoich   Stoich"
      Print #1, " 1      " & Chr$(34) & "doped glass solid" & Chr$(34) & "
" & Str$(Thickness) & "  1.442          .2544      .5088      .2368"
      Print #1, "0  Target layer phases (0=Solid, 1=Gas)"
      Print #1, "0"
      Print #1, "Target Compound Corrections (Bragg)"
      Print #1, "0"
      Print #1, "Target atom displacement energies (eV)"
      Print #1, "       15     15     20"
      Print #1, "Target atom lattice binding energies (eV)"
      Print #1, "       3.5    3.5    3"
      Print #1, "Target atom surface binding energies (eV)"
      Print #1, "       4      4      2"
   ElseIf (MaterialType = 12) Then
      Print #1, "Target Elements:    Z Mass(amu)"
      Print #1, "Atom 1 = Li  =        3   6.0"
      Print #1, "Layer   Layer Name /              Width Density    Li(3)"
      Print #1, "Numb.   Description               (Ang) (g/cm3)    Stoich"
      Print #1, " 1      " & Chr$(34) & "inert Li solid" & Chr$(34) & "
" & Str$(Thickness) & "  .462           1.0"
      Print #1, "0  Target layer phases (0=Solid, 1=Gas)"
      Print #1, "0"
      Print #1, "Target Compound Corrections (Bragg)"
      Print #1, "0"
      Print #1, "Target atom displacement energies (eV)"
      Print #1, "       15"
      Print #1, "Target atom lattice binding energies (eV)"
      Print #1, "       3.5"
      Print #1, "Target atom surface binding energies (eV)"
      Print #1, "       4"
   ElseIf (MaterialType = 13) Then
      Print #1, "Target Elements:    Z Mass(amu)"
      Print #1, "Atom 1 = B   =        5   10.0"
      Print #1, "Layer   Layer Name /              Width Density     B(5)"
      Print #1, "Numb.   Description               (Ang) (g/cm3)    Stoich"
      Print #1, " 1      " & Chr$(34) & "inert B solid" & Chr$(34) & "
" & Str$(Thickness) & "  2.174          1.0"
      Print #1, "0  Target layer phases (0=Solid, 1=Gas)"
      Print #1, "0"
      Print #1, "Target Compound Corrections (Bragg)"
      Print #1, "0"
      Print #1, "Target atom displacement energies (eV)"
      Print #1, "       20"
      Print #1, "Target atom lattice binding energies (eV)"
      Print #1, "       3"
      Print #1, "Target atom surface binding energies (eV)"
      Print #1, "       2"
   End If
   Print #1, "Stopping Power Version (2=1996, 0=2000, 1=2000 interpolated)"
   Print #1, "0"
   Print #1, Chr$(26) 'Insert an end-of-file character at the end of the file.
This
   '*must* be done.

   Close #1
End Sub

'1 is B, 2 is Li, 3 is Al, 4 is Ar, 5 is 90% Ar/10% isobutane, 6 is P-10 (90%
Ar/10% methane)
```

```
'7 is LiF, 8 is Si, 9 is glass (SiO2), 10 is doped glass - 20 vol% B-10, 11 is
doped
'glass - 20 vol% Li-6, 12 is inert lithium, 13 is inert boron

Public Sub SetupModifiedTrim_Win_In(ByVal z As Long, ByVal M As Single, ByVal
pulselayer As Boolean, ByVal MaterialType, ByVal Thickness As Long, ByVal
RandomSeed As Long)
   'Thickness needs to be in Angstroms (either that or put something in to
convert to Angstroms)
   Open path$ & "trim_win.in" For Output As #1

   Print #1, "==> SRIM-2000.41 <== TRIM.IN is for TRIM(DOS), TRIM(Win) uses
TRIM_WIN.IN ==="
   Print #1, "Ion: Z1 ,  M1,  Energy (keV), Angle,Number,Bragg Corr,AutoSave
Number."
   Print #1, "   " & z & "   " & M & "   20000   0   99999   0   10000"
   Print #1, "Cascades(1=No;2=Full;3=Sputt;4-5=Ions;6-7=Neutrons), Random
Number Seed, Reminders"
   Print #1, "    4   " & RandomSeed & "   0"
   Print #1, "Diskfiles (0=no,1=yes): Ranges, Backscatt, Transmit, Sputtered,
Recoils (2=Alt-C)"
   Print #1, "    1   1   1   1   " & CStr(EnergyStepSize)
   Print #1, "Target material : Number of Elements & Layers"
   '1 is B, 2 is Li, 3 is Al, 4 is Ar, 5 is 90% Ar/10% isobutane, 6 is P-10
(90% Ar/10% methane)
   Select Case MaterialType
   Case 1:
      Print #1, Chr$(34) & "Target into layer" & Chr$(34) & "    1    1"
      Print #1, "Target Energies (eV): Binding, Surface, Individual
Displacement"
      Print #1, "           3     2      20"
   Case 2:
      Print #1, Chr$(34) & "Target into layer" & Chr$(34) & "    1    1"
      Print #1, "Target Energies (eV): Binding, Surface, Individual
Displacement"
      Print #1, "           3     2      20"  'Change to appropriate values.
   Case 3:
      Print #1, Chr$(34) & "Target into layer" & Chr$(34) & "    1    1"
      Print #1, "Target Energies (eV): Binding, Surface, Individual
Displacement"
      Print #1, "           3     4      20"
   Case 4:
      Print #1, Chr$(34) & "Target into layer" & Chr$(34) & "    1    1"
      Print #1, "Target Energies (eV): Binding, Surface, Individual
Displacement"
      Print #1, "          3.3   3.5    13"
   Case 5:
      Print #1, Chr$(34) & "Target into layer" & Chr$(34) & "    3    1"
      Print #1, "Target Energies (eV): Binding, Surface, Individual
Displacement"
      Print #1, "           3.5   4      15     15     15"
   Case 6:
      Print #1, Chr$(34) & "Target into layer" & Chr$(34) & "    3    1"
      Print #1, "Target Energies (eV): Binding, Surface, Individual
Displacement"
      Print #1, "           3.5   4      15     15     15"
   Case 7:
      Print #1, Chr$(34) & "Target into layer" & Chr$(34) & "    2    1"
      Print #1, "Target Energies (eV): Binding, Surface, Individual
Displacement"
      Print #1, "           3.5   4      15     15"
   Case 8:
      Print #1, Chr$(34) & "Target into layer" & Chr$(34) & "    1    1"
      Print #1, "Target Energies (eV): Binding, Surface, Individual
Displacement"
      Print #1, "           3.5   4      15"
   Case 9:
      Print #1, Chr$(34) & "Target into layer" & Chr$(34) & "    2    1"
      Print #1, "Target Energies (eV): Binding, Surface, Individual
Displacement"
      Print #1, "           3.5   4      15"
   Case 10:
      Print #1, Chr$(34) & "Target into layer" & Chr$(34) & "    3    1"
      Print #1, "Target Energies (eV): Binding, Surface, Individual
Displacement"
      Print #1, "           3     3      15"
   Case 11:
```

```
      Print #1, Chr$(34) & "Target into layer" & Chr$(34) & "    3    1"
      Print #1, "Target Energies (eV): Binding, Surface, Individual
Displacement"
      Print #1, "           3     3      15"
   Case 12:
      Print #1, Chr$(34) & "Target into layer" & Chr$(34) & "    1    1"
      Print #1, "Target Energies (eV): Binding, Surface, Individual
Displacement"
      Print #1, "           3.5   4      15"
   Case 13:
      Print #1, Chr$(34) & "Target into layer" & Chr$(34) & "    1    1"
      Print #1, "Target Energies (eV): Binding, Surface, Individual
Displacement"
      Print #1, "           3     2      20"
   End Select

   Print #1, "PlotType (0-5); Plot Depths: Xmin, Xmax(Ang.) [=0 0 for Viewing
Full Target]"
   Print #1, "      5                          0            0"
   If (MaterialType = 1) Then
      Print #1, "Target Elements:    Z Mass(amu)"
      Print #1, "Atom 1 = B  =        5  10.811"
      Print #1, "Layer   Layer Name /               Width Density    B(5)"
      Print #1, "Numb.   Description                 (Ang) (g/cm3)    Stoich"
      Print #1, " 1      " & Chr$(34) & "Boron" & Chr$(34) & "          " &
Str$(Thickness) & "  1.289        1"
      Print #1, "0  Target layer phases (0=Solid, 1=Gas)"
      Print #1, "0"
      Print #1, "Target Compound Corrections (Bragg)"
      Print #1, "0"
      Print #1, "Target atom displacement energies (eV)"
      Print #1, "20"
      Print #1, "Target atom lattice binding energies (eV)"
      Print #1, "3"
      Print #1, "Target atom surface binding energies (eV)"
      Print #1, "2"
   ElseIf (MaterialType = 2) Then
      Print #1, "Target Elements:    Z Mass(amu)"
      Print #1, "Atom 1 = Li =        3   6.941"
      Print #1, "Layer   Layer Name /               Width Density    Li(3)"
      Print #1, "Numb.   Description                 (Ang) (g/cm3)    Stoich"
      Print #1, " 1      " & Chr$(34) & "Lithium" & Chr$(34) & "          " &
Str$(Thickness) & "  0.534        1"
      Print #1, "0  Target layer phases (0=Solid, 1=Gas)"
      Print #1, "0"
      Print #1, "Target Compound Corrections (Bragg)"
      Print #1, "0"
      Print #1, "Target atom displacement energies (eV)"
      Print #1, "20"
      Print #1, "Target atom lattice binding energies (eV)"
      Print #1, "3"
      Print #1, "Target atom surface binding energies (eV)"
      Print #1, "2"
   ElseIf (MaterialType = 3) Then
      Print #1, "Target Elements:    Z Mass(amu)"
      Print #1, "Atom 1 = Al =       13  26.982"
      Print #1, "Layer   Layer Name /               Width Density    Al(13)"
      Print #1, "Numb.   Description                 (Ang) (g/cm3)    Stoich"
      Print #1, " 1      " & Chr$(34) & "Aluminium" & Chr$(34) & "          " &
Str$(Thickness) & "  2.702        1"
      Print #1, "0  Target layer phases (0=Solid, 1=Gas)"
      Print #1, "0"
      Print #1, "Target Compound Corrections (Bragg)"
      Print #1, "0"
      Print #1, "Target atom displacement energies (eV)"
      Print #1, "20"
      Print #1, "Target atom lattice binding energies (eV)"
      Print #1, "3"
      Print #1, "Target atom surface binding energies (eV)"
      Print #1, "4"
   ElseIf (MaterialType = 4) Then
      Print #1, "Target Elements:    Z Mass(amu)"
      Print #1, "Atom 1 = Ar  =       18  39.948"
      Print #1, "Layer   Layer Name /               Width Density    Ar(18)"
      Print #1, "Numb.   Description                 (Ang) (g/cm3)    Stoich"
      Print #1, " 1      " & Chr$(34) & "Argon gas" & Chr$(34) & "          " &
Str$(Thickness) & "  .00166       1.0"
```

```
      Print #1, "0  Target layer phases (0=Solid, 1=Gas)"
      Print #1, "1"
      Print #1, "Target Compound Corrections (Bragg)"
      Print #1, "0"
      Print #1, "Target atom displacement energies (eV)"
      Print #1, "       13"
      Print #1, "Target atom lattice binding energies (eV)"
      Print #1, "       3.3"
      Print #1, "Target atom surface binding energies (eV)"
      Print #1, "       3.5"
   ElseIf (MaterialType = 5) Then
      Print #1, "Target Elements:    Z Mass(amu)"
      Print #1, "Atom 1 = Ar  =       18  39.948"
      Print #1, "Atom 2 = C   =        6  12.011"
      Print #1, "Atom 3 = H   =        1   1.008"
      Print #1, "Layer   Layer Name /               Width Density    Ar(18)
C(6)     H(1)"
      Print #1, "Numb.   Description                 (Ang) (g/cm3)    Stoich
Stoich   Stoich"
      Print #1, " 1      " & Chr$(34) & "Ar/isobutane gas" & Chr$(34) & "
" & Str$(Thickness) & "  " & GasDensity & "        .391304  .173913  .434783"
      Print #1, "0  Target layer phases (0=Solid, 1=Gas)"
      Print #1, "1"
      Print #1, "Target Compound Corrections (Bragg)"
      Print #1, "0"
      Print #1, "Target atom displacement energies (eV)"
      Print #1, "       13     14     15"
      Print #1, "Target atom lattice binding energies (eV)"
      Print #1, "       3.3    3.4    3.5"
      Print #1, "Target atom surface binding energies (eV)"
      Print #1, "       3.5    4      4.5"
   ElseIf (MaterialType = 6) Then
      Print #1, "Target Elements:    Z Mass(amu)"
      Print #1, "Atom 1 = Ar  =       18  39.948"
      Print #1, "Atom 2 = C   =        6  12.011"
      Print #1, "Atom 3 = H   =        1   1.008"
      Print #1, "Layer   Layer Name /               Width Density    Ar(18)
C(6)     H(1)"
      Print #1, "Numb.   Description                 (Ang) (g/cm3)    Stoich
Stoich   Stoich"
      Print #1, " 1      " & Chr$(34) & "P-10 gas" & Chr$(34) & "          " &
Str$(Thickness) & "  .00167        .642857  .071429  .285714"
      Print #1, "0  Target layer phases (0=Solid, 1=Gas)"
      Print #1, "1"
      Print #1, "Target Compound Corrections (Bragg)"
      Print #1, "0"
      Print #1, "Target atom displacement energies (eV)"
      Print #1, "       13     14     15"
      Print #1, "Target atom lattice binding energies (eV)"
      Print #1, "       3.3    3.4    3.5"
      Print #1, "Target atom surface binding energies (eV)"
      Print #1, "       3.5    4      4.5"
   ElseIf (MaterialType = 7) Then
      Print #1, "Target Elements:    Z Mass(amu)"
      Print #1, "Atom 1 = Li  =        3   6.0"
      Print #1, "Atom 2 = F   =        9  18.998"
      Print #1, "Layer   Layer Name /               Width Density    Li(18)
F(6)"
      Print #1, "Numb.   Description                 (Ang) (g/cm3)    Stoich
Stoich"
      Print #1, " 1      " & Chr$(34) & "LiF solid" & Chr$(34) & "         " &
Str$(Thickness) & "  .786          .50      .50"
      Print #1, "0  Target layer phases (0=Solid, 1=Gas)"
      Print #1, "0"
      Print #1, "Target Compound Corrections (Bragg)"
      Print #1, "0"
      Print #1, "Target atom displacement energies (eV)"
      Print #1, "       15     15"
      Print #1, "Target atom lattice binding energies (eV)"
      Print #1, "       3.5    3.5"
      Print #1, "Target atom surface binding energies (eV)"
      Print #1, "       4      4"
   ElseIf (MaterialType = 8) Then
      Print #1, "Target Elements:    Z Mass(amu)"
      Print #1, "Atom 1 = Si  =       14  28.086"
      Print #1, "Layer   Layer Name /               Width Density    Si(14)"
      Print #1, "Numb.   Description                 (Ang) (g/cm3)    Stoich"
```

```
      Print #1, " 1      " & Chr$(34) & "Si solid" & Chr$(34) & "         " &
Str$(Thickness) & "  2.321          1.00"
      Print #1, "0  Target layer phases (0=Solid, 1=Gas)"
      Print #1, "0"
      Print #1, "Target Compound Corrections (Bragg)"
      Print #1, "0"
      Print #1, "Target atom displacement energies (eV)"
      Print #1, "      15"
      Print #1, "Target atom lattice binding energies (eV)"
      Print #1, "      3.5"
      Print #1, "Target atom surface binding energies (eV)"
      Print #1, "      4"
   ElseIf (MaterialType = 9) Then
      Print #1, "Target Elements:    Z Mass(amu)"
      Print #1, "Atom 1 = Si  =       14  28.086"
      Print #1, "Atom 2 = O   =        8  15.999"
      Print #1, "Layer   Layer Name /               Width Density    Si(14)
O(8)"
      Print #1, "Numb.   Description                 (Ang) (g/cm3)    Stoich
Stoich"
      Print #1, " 1      " & Chr$(34) & "glass" & Chr$(34) & "         " &
Str$(Thickness) & "  1.724          .333       .667"
      Print #1, "0  Target layer phases (0=Solid, 1=Gas)"
      Print #1, "0"
      Print #1, "Target Compound Corrections (Bragg)"
      Print #1, "0"
      Print #1, "Target atom displacement energies (eV)"
      Print #1, "      15     15"
      Print #1, "Target atom lattice binding energies (eV)"
      Print #1, "      3.5    3.5"
      Print #1, "Target atom surface binding energies (eV)"
      Print #1, "      4      4"
   ElseIf (MaterialType = 10) Then
      Print #1, "Target Elements:    Z Mass(amu)"
      Print #1, "Atom 1 = Si  =       14  28.086"
      Print #1, "Atom 2 = O   =        8  15.999"
      Print #1, "Atom 3 = B   =        5  10.0"
      Print #1, "Layer   Layer Name /               Width Density    Si(14)
O(8)     B(5)"
      Print #1, "Numb.   Description                 (Ang) (g/cm3)    Stoich
Stoich   Stoich"
      Print #1, " 1      " & Chr$(34) & "doped glass solid" & Chr$(34) & "
" & Str$(Thickness) & "  2.017          .1776      .3553      .4671"
      Print #1, "0  Target layer phases (0=Solid, 1=Gas)"
      Print #1, "0"
      Print #1, "Target Compound Corrections (Bragg)"
      Print #1, "0"
      Print #1, "Target atom displacement energies (eV)"
      Print #1, "      15     15     20"
      Print #1, "Target atom lattice binding energies (eV)"
      Print #1, "      3.5    3.5    3"
      Print #1, "Target atom surface binding energies (eV)"
      Print #1, "      4      4      2"
   ElseIf (MaterialType = 11) Then
      Print #1, "Target Elements:    Z Mass(amu)"
      Print #1, "Atom 1 = Si  =       14  28.086"
      Print #1, "Atom 2 = O   =        8  15.999"
      Print #1, "Atom 4 = Li  =        3   6.0"
      Print #1, "Layer   Layer Name /               Width Density    Si(14)
O(8)     Li(3)"
      Print #1, "Numb.   Description                 (Ang) (g/cm3)    Stoich
Stoich   Stoich"
      Print #1, " 1      " & Chr$(34) & "doped glass solid" & Chr$(34) & "
" & Str$(Thickness) & "  1.442          .2544      .5088      .2368"
      Print #1, "0  Target layer phases (0=Solid, 1=Gas)"
      Print #1, "0"
      Print #1, "Target Compound Corrections (Bragg)"
      Print #1, "0"
      Print #1, "Target atom displacement energies (eV)"
      Print #1, "      15     15     20"
      Print #1, "Target atom lattice binding energies (eV)"
      Print #1, "      3.5    3.5    3"
      Print #1, "Target atom surface binding energies (eV)"
      Print #1, "      4      4      2"
   ElseIf (MaterialType = 12) Then
      Print #1, "Target Elements:    Z Mass(amu)"
      Print #1, "Atom 1 = Li  =        3   6.0"
```

```
      Print #1, "Layer   Layer Name /              Width Density    Li(3)"
      Print #1, "Numb.   Description               (Ang) (g/cm3)    Stoich"
      Print #1, " 1      " & Chr$(34) & "inert Li solid" & Chr$(34) & "
" & Str$(Thickness) & "  .462          1.0"
      Print #1, "0  Target layer phases (0=Solid, 1=Gas)"
      Print #1, "0"
      Print #1, "Target Compound Corrections (Bragg)"
      Print #1, "0"
      Print #1, "Target atom displacement energies (eV)"
      Print #1, "      15"
      Print #1, "Target atom lattice binding energies (eV)"
      Print #1, "      3.5"
      Print #1, "Target atom surface binding energies (eV)"
      Print #1, "      4"
   ElseIf (MaterialType = 13) Then
      Print #1, "Target Elements:    Z Mass(amu)"
      Print #1, "Atom 1 = B   =        5   10.0"
      Print #1, "Layer   Layer Name /              Width Density     B(5)"
      Print #1, "Numb.   Description               (Ang) (g/cm3)    Stoich"
      Print #1, " 1      " & Chr$(34) & "inert B solid" & Chr$(34) & "
" & Str$(Thickness) & "  2.174          1.0"
      Print #1, "0  Target layer phases (0=Solid, 1=Gas)"
      Print #1, "0"
      Print #1, "Target Compound Corrections (Bragg)"
      Print #1, "0"
      Print #1, "Target atom displacement energies (eV)"
      Print #1, "      20"
      Print #1, "Target atom lattice binding energies (eV)"
      Print #1, "      3"
      Print #1, "Target atom surface binding energies (eV)"
      Print #1, "      2"
   End If
   Print #1, "Stopping Power Version (2=1996, 0=2000, 1=2000 interpolated)"
   Print #1, "0"
   Print #1, Chr$(26) 'Insert an end-of-file character at the end of the file.
This
   '*must* be done.

   Close #1
End Sub
```

APPENDIX C:
COMPOSITE NEUTRON SCINTILLATOR SIMULATION CODE LISTING

(a) Introduction

This appendix contains the computer code listings for the composite neutron scintillator simulation in the Visual Basic 6 programming language. Some additional material is included in the code that is not discussed in this dissertation. The additional code is concerned with simulation of light transport in the scintillator and is called by the subroutine RunLightTransportSim(). All of the code is currently in the form of a single file, DopedScintillatorSimulator.frm.

This program is a work in progress in the sense that it is periodically modified in order to improve it or to produce new output data. (Most notably, it is intended to form the basis of a more complex simulation including light transport in the scintillator and readout devices.) As such, it is contains comments regarding how it may be modified to do this or that, many lines are more than 80 characters long (and thus wrap onto the next line in this document), and so forth.

User control is accomplished by direct modification of the code. There are two elements to this. The first involves determining what type of simulation is to be run. For example, in the Form_Load() subroutine, a call may be alternated between RunOneCase() and RunManyCases(). The characteristics of each simulation run are then determined by setting the relevant variables (e.g. ParticleRadiiMicrons, VolumeDopingFraction) to their desired values. There is currently no visual front-end for the simulation, although this is something that may be added in the future.

(b) DopedScintillatorSimulator.frm

```
Option Explicit

'*****************************************************************************
'*                                                                           *
'*                      MONTE CARLO SCINTILLATOR SIMULATOR                   *
'*                                   FOR                                     *
'*            MATRIX/FLUORESCENT DOPANT PARTICLE COMPOSITE MATERIAL          *
'*                                                                           *
'*                                Andrew Stephan                             *
'*                                                                           *
'*****************************************************************************

'IMPORTANT INFORMATION ON AXES:
'TRIM uses x as the depth of penetration of ions.  We will use x as the
thickness for planar geometries.
'For the others it doesn't really matter.

Dim path$

'Physical constants
Const Pi = 3.1415926535898
Const ee = 2.71828182846

'Maximum ranges are in Angstroms for 10 MeV particles in 1 g/cm^3 Si.  Scale as
neeeded.
Const MaxH3Range = 7700000#
Const MaxAlphaRange = 1620000#

'Other important constants and variables
Const InitialVolumeDopingFraction = 0.4 'dopant fractional volume in the
composite material
Dim VolumeDopingFraction As Single
Const InitialLi6VolumeDopingFraction = 0.15 'used when the program generates
its own neutron
Dim Li6VolumeDopingFraction As Single
'absorption event locations, not used when reading them in from MCNP
Const InitialParticleRadiiMicrons = 0.05  'in microns
Dim ParticleRadiiMicrons As Single
Const InitialDopantDensity = 3# 'in g/cm^3
Dim DopantDensity As Single
Const InitialMatrixDensity = 2# 'in g/cm^3
Dim MatrixDensity As Single
Const InitialNeutronEnergy = 0.0000001 'in MeV
Dim NeutronEnergy As Single
Const ScintillatorThickness = 0.5 'in cm
Const eVperPhoton = 20# 'this depends totally on the scintillator
characteristics
'and is an average for the Li-6 reaction products (it varies with LET)
Const FanoFactor = 1# 'scintillator material dependent
Const LightAttenuationLength = 0.2  'give a really big value if it is
transparent
'length to attenuate to 50%, could be changed to 1/e or whatever you want

'Values for controlling where the particle transport data comes from.
'Since it is sometimes desirable or necessary to change some of the properties,
'I am setting it up such that the values are set as initially desired by the
user
'and then changed by the program later if needed.
Const InitialEventsToRun = 50
Dim EventsToRun As Long
Const RolloverNumber = 5000 'must be <= number of events in MCNP and/or TRIM
output if
'drawing from those sources, in any event must not exceed 9999 due to
limitations in TRIM!
Dim RunFromMCNP As Boolean
Const InitialRunFromMCNP = False
Dim UseExistingData As Boolean
Const InitialUseExistingData = True 'true - read from existing EXYZ_1.TXT and
EXYZ_2.TXT, false - run TRIM for new data
'Array for storing Li-6 cross sections.
Dim Li6XSection(2, 498) As Double 'Double
Const EnergyStepSize = 25000# 'for TRIM input
```

```
'Values for controlling the data the simulation produces
Const BinSize = 10000# 'eV, for binning the energy deposited in the
scintillator

'*************************************************************************
'*     Items Relating to Running TRIM as a Shell Process - Must Be First      *
'*************************************************************************

'The next section on running a shell process and then detecting when it is
finished is
'taken from and/or patterned on example Q129796 from the Microsoft Knowledge
Base.

Private Type STARTUPINFO
   cb As Long
   lpReserved As String
   lpDesktop As String
   lpTitle As String
   dwX As Long
   dwY As Long
   dwXSize As Long
   dwYSize As Long
   dwXCountChars As Long
   dwYCountChars As Long
   dwFillAttribute As Long
   dwFlags As Long
   wShowWindow As Integer
   cbReserved2 As Integer
   lpReserved2 As Long
   hStdInput As Long
   hStdOutput As Long
   hStdError As Long
End Type

Private Type PROCESS_INFORMATION
   hProcess As Long
   hThread As Long
   dwProcessID As Long
   dwThreadID As Long
End Type

Private Declare Function WaitForSingleObject Lib "kernel32" (ByVal hHandle As
Long, ByVal dwMilliseconds As Long) As Long
Private Declare Function CreateProcessA Lib "kernel32" (ByVal lpApplicationName
As String, ByVal lpCommandLine As String, _
ByVal lpProcessAttributes As Long, ByVal lpThreadAttributes As Long, ByVal
bInheritHandles As Long, ByVal dwCreationFlags _
As Long, ByVal lpEnvironment As Long, ByVal lpCurrentDirectory As String,
lpStartupInfo As STARTUPINFO, lpProcessInformation _
As PROCESS_INFORMATION) As Long
Private Declare Function CloseHandle Lib "kernel32" (ByVal hObject As Long) As
Long
Private Declare Function GetExitCodeProcess Lib "kernel32" (ByVal hProcess As
Long, lpExitCode As Long) As Long
Private Const NORMAL_PRIORITY_CLASS = &H20&
Private Const INFINITE = -1&

'*************************************************************************
'*                                                                       *
'*               POINT WHERE PROGRAM EXECUTION BEGINS AT STARTUP         *
'*                                                                       *
'*************************************************************************

Private Sub Form_Load()
   Randomize Timer

   'Call RunOneCase
   Call RunManyCases

   End
End Sub

'*************************************************************************
'*                                                                       *
'*                     CODE FOR RUNNING ONE SIMULATION CASE              *
'*                                                                       *
'*************************************************************************
```

```
Public Sub RunOneCase()
   '
   Dim scintillator_energy() As Double 'use for the subset currently being run
   Dim matrix_energy() As Double
   Dim all_scintillator_energy() As Single 'use for storing the entire data set
(all events)
   Dim event_depth() As Single
   Dim RunEvents As Long
   Dim RemainingEvents As Long
   Dim currevent As Long 'current event for copying between matrices
   Dim stopflag As Boolean
   Dim i As Long
   Dim particle_ran() As Boolean
   Dim successfully_run As Long

   Call ReadInXSections

   VolumeDopingFraction = InitialVolumeDopingFraction
   Li6VolumeDopingFraction = InitialVolumeDopingFraction
   ParticleRadiiMicrons = InitialParticleRadiiMicrons
   DopantDensity = InitialDopantDensity
   MatrixDensity = InitialMatrixDensity
   NeutronEnergy = InitialNeutronEnergy
   EventsToRun = InitialEventsToRun
   RemainingEvents = EventsToRun
   RunFromMCNP = InitialRunFromMCNP
   UseExistingData = InitialUseExistingData
   stopflag = False
   currevent = 0
   ReDim particle_ran(1)

   If (EventsToRun <= 0) Then
      MsgBox "Number of events requested by user is zero or negative.", ,
"Fatal Error:"
      End
   End If

   Do While (stopflag = False)
      If (RemainingEvents <= 0) Then
         MsgBox "RemainingEvents<=0 in main sub. Debug code.", , "Fatal Error:"
         End
      ElseIf (RemainingEvents <= RolloverNumber) Then
         RunEvents = RemainingEvents
         'RemainingEvents = 0
         'stopflag = True
      Else
         RunEvents = RolloverNumber
         'RemainingEvents = RemainingEvents - RolloverNumber
      End If

      Call RunParticleTransportSim(scintillator_energy(), matrix_energy(), _
      RunEvents, particle_ran())

      If (UseExistingData = False) Then
         UseExistingData = True
      End If

      If (RunFromMCNP = True) Then
         RunFromMCNP = False
      End If

      successfully_run = 0

      For i = 1 To RunEvents Step 1
         If (particle_ran(i) = True) Then
            successfully_run = successfully_run + 1
            all_scintillator_energy(currevent + successfully_run) =
CSng(scintillator_energy(i))
         End If
      Next i

      currevent = currevent + successfully_run
      RemainingEvents = RemainingEvents - succesfully_run

      If (RemainingEvents = 0) Then
         stopflag = True
```

```
      End If
   Loop

   'Add some event depths.  These aren't the correct ones, but they'll do for
testing purposes.
   For i = 1 To EventsToRun Step 1
      event_depth(i) = Rnd * ScintillatorThickness
   Next i

   Call RunLightTransportSim(all_scintillator_energy(), event_depth(),
EventsToRun)
   Call SaveEnergyDepositionData(all_scintillator_energy())

   End
End Sub

'*****************************************************************************
'*                                                                           *
'*                    CODE FOR RUNNING MANY SIMULATION CASES                 *
'*                                                                           *
'*****************************************************************************

Public Sub RunManyCases()
   'Control the cases through altering the code in this subroutine.
   'A single case is controlled entirely by changing the constants at the start
of
   'the program while multiple cases are controlled by directly altering the
code
   'in this subroutine.
   Dim scintillator_energy() As Double 'use for the subset currently being run
   Dim matrix_energy() As Double
   Dim all_scintillator_energy() As Single 'use for storing the entire data set
(all events)
   Dim event_depth() As Single
   Dim RunEvents As Long
   Dim RemainingEvents As Long
   Dim currevent As Long 'current event for copying between matrices
   Dim stopflag As Boolean
   Dim cases As Long
   Dim i As Long
   'Use ii, iii, and iv for setting up repetitive runs that vary only slightly
from run to run
   Dim ii As Long
   Dim iii As Long
   Dim iv As Long
   Dim particle_ran() As Boolean
   Dim successfully_run As Long

   Call ReadInXSections

   VolumeDopingFraction = InitialVolumeDopingFraction
   Li6VolumeDopingFraction = InitialVolumeDopingFraction
   ParticleRadiiMicrons = InitialParticleRadiiMicrons
   DopantDensity = InitialDopantDensity
   MatrixDensity = InitialMatrixDensity
   NeutronEnergy = InitialNeutronEnergy
   EventsToRun = InitialEventsToRun
   RemainingEvents = EventsToRun
   RunFromMCNP = InitialRunFromMCNP
   UseExistingData = InitialUseExistingData
   ReDim particle_ran(1)
   successfully_run = 0

   'The code can be set up to run many different cases sequentially using
   'for next steps or other types of loops and modifying the value of the
   'variables that control what types of calculations are performed.

   For ii = 1 To 2 Step 1
   'particle sizes - do them in descending order of size
   For iii = 1 To 2 Step 1
   '1 is ZnS:Ag/glass, 2 is organic/organic
   For iv = 1 To 10 Step 1
   'covers different doping concentrations

   Select Case ii
   Case 1:
      ParticleRadiiMicrons = 100#
```

```
   EventsToRun = 20000
   RemainingEvents = EventsToRun
Case 2:
   ParticleRadiiMicrons = 50#
   EventsToRun = 20000
   RemainingEvents = EventsToRun
Case 3:
   ParticleRadiiMicrons = 20#
   EventsToRun = 20000
   RemainingEvents = EventsToRun
Case 4:
   ParticleRadiiMicrons = 10#
   EventsToRun = 20000
   RemainingEvents = EventsToRun
Case 5:
   ParticleRadiiMicrons = 5#
   EventsToRun = 20000
   RemainingEvents = EventsToRun
Case 6:
   ParticleRadiiMicrons = 2#
   EventsToRun = 20000
   RemainingEvents = EventsToRun
Case 7:
   ParticleRadiiMicrons = 1#
   EventsToRun = 20000
   RemainingEvents = EventsToRun
Case 8:
   ParticleRadiiMicrons = 0.5
   EventsToRun = 20000
   RemainingEvents = EventsToRun
Case 9:
   ParticleRadiiMicrons = 0.2
   EventsToRun = 10000
   RemainingEvents = EventsToRun
Case 10:
   ParticleRadiiMicrons = 0.1
   EventsToRun = 10000
   RemainingEvents = EventsToRun
Case 11:
   ParticleRadiiMicrons = 0.05
   EventsToRun = 5000
   RemainingEvents = EventsToRun
End Select

Select Case iii
Case 1:
   DopantDensity = 4.1
   MatrixDensity = 1.409
Case 2:
   DopantDensity = 1#
   MatrixDensity = 1#
End Select

Select Case iv
Case 1:
   VolumeDopingFraction = 0.5
Case 2:
   VolumeDopingFraction = 0.4
Case 3:
   VolumeDopingFraction = 0.3
Case 4:
   VolumeDopingFraction = 0.25
Case 5:
   VolumeDopingFraction = 0.2
Case 6:
   VolumeDopingFraction = 0.15
Case 7:
   VolumeDopingFraction = 0.1
Case 8:
   VolumeDopingFraction = 0.06
Case 9:
   VolumeDopingFraction = 0.04
Case 10:
   VolumeDopingFraction = 0.02
End Select

   If (EventsToRun <= 0) Then
```

```
         MsgBox "Number of events requested by user is zero or negative.", ,
"Fatal Error:"
         End
      End If

      stopflag = False
      currevent = 0
      ReDim all_scintillator_energy(EventsToRun)

      Do While (stopflag = False)
         If (RemainingEvents <= 0) Then
            MsgBox "RemainingEvents<=0 in main sub. Debug code.", , "Fatal
Error:"
            End
         ElseIf (RemainingEvents <= RolloverNumber) Then
            RunEvents = RemainingEvents
         Else
            RunEvents = RolloverNumber
         End If

         Call RunParticleTransportSim(scintillator_energy(), matrix_energy(),
RunEvents, _
         particle_ran())

         If (UseExistingData = False) Then
            UseExistingData = True
         End If

         If (RunFromMCNP = True) Then
            RunFromMCNP = False
         End If

         successfully_run = 0

         For i = 1 To RunEvents Step 1
            If (particle_ran(i) = True) Then
               successfully_run = successfully_run + 1
               all_scintillator_energy(currevent + successfully_run) =
CSng(scintillator_energy(i))
            End If
         Next i

         currevent = currevent + successfully_run
         RemainingEvents = RemainingEvents - successfully_run

         If (RemainingEvents = 0) Then
            stopflag = True
         End If
      Loop

      Call SaveEnergyDepositionData(all_scintillator_energy())

      Select Case iii
      Case 1:
         FileCopy ("Spectrum.TXT"), ("Spectrum_ZnSglass_" &
VolumeDopingFraction & "vol_" & ParticleRadiiMicrons & "um.TXT")
      Case 2:
         FileCopy ("Spectrum.TXT"), ("Spectrum_organic_" & VolumeDopingFraction
& "vol_" & ParticleRadiiMicrons & "um.TXT")
      End Select

   Next iv
   Next iii
   Next ii

   End
End Sub


'*****************************************************************************
'*                                                                           *
'*                      DATA ANALYSIS AND OUTPUT ROUTINES                    *
'*                                                                           *
'*  This section includes routines for processing data from the particle     *
'*  and light creation and transport sections.  Information of interest that *
'*  the user might want to save includes the energy deposited in the dopant  *
'*  scintillating particles, the number of photons reaching each surface, the *
```

```
'*  distribution of the photons into the readout fibers, how many photons     *
'*  succeed in reaching the PMT at the end of each fiber, and various things  *
'*  related to all of the above.  The routines in this section adddress some  *
'*  of those issues.                                                          *
'*                                                                            *
'******************************************************************************

'******************************************************************************
'*     Subroutine for Binning and Saving Energy Deposition in the Dopant      *
'******************************************************************************

Public Sub SaveEnergyDepositionData(all_scintillator_energy() As Single)
   '
   Dim EnergySpectrum() As Long
   Dim numbins As Long
   Dim binnum As Long
   Dim totalenergy As Double
   Dim avgenergy_all As Double
   Dim avgenergy_pulses As Double
   Dim std_dev_all As Double
   Dim std_dev_pulses As Double
   Dim sumenergysqrd As Double
   Dim numpulses As Long
   Dim pulseprobability As Single
   Dim i As Long

   numbins = CLng((4780000# + NeutronEnergy) / BinSize) + 1
   ReDim EnergySpectrum(numbins)
   numpulses = 0
   totalenergy = 0#
   sumenergysqrd = 0#

   'Bin energy depositions
   For i = 1 To EventsToRun Step 1
      binnum = all_scintillator_energy(i) / 10000#
      EnergySpectrum(binnum) = EnergySpectrum(binnum) + 1
      sumenergysqrd = sumenergysqrd + CDbl(all_scintillator_energy(i) ^ 2)

      If (all_scintillator_energy(i) > 0#) Then
         numpulses = numpulses + 1
         totalenergy = totalenergy + CDbl(all_scintillator_energy(i))
      End If
   Next i

   avgenergy_all = totalenergy / CDbl(EventsToRun)
   avgenergy_pulses = totalenergy / CDbl(numpulses)

   pulseprobability = CSng(numpulses) / CSng(EventsToRun)
   std_dev_all = ((CDbl(EventsToRun) * sumenergysqrd - totalenergy ^ 2) /
(CDbl(EventsToRun) * (CDbl(EventsToRun - 1)))) ^ 0.5
   std_dev_pulses = ((CDbl(numpulses) * sumenergysqrd - totalenergy ^ 2) /
(CDbl(numpulses) * (CDbl(numpulses - 1)))) ^ 0.5

   'Output to file
   Open "Spectrum.txt" For Output As #1

   Print #1, "Average energy (all):" & Chr$(9) & avgenergy_all
   Print #1, "Average energy (pulses):" & Chr$(9) & avgenergy_pulses
   Print #1, "Standard deviation (all):" & Chr$(9) & std_dev_all
   Print #1, "Standard deviation (pulses):" & Chr$(9) & std_dev_pulses
   Print #1, "Pulse probability:" & Chr$(9) & pulseprobability
   Print #1, ""

   For i = 1 To numbins Step 1
      Print #1, i & Chr$(9) & EnergySpectrum(i)
   Next i

   Close #1
End Sub

'******************************************************************************
'*                                                                            *
'*             LIGHT TRANSPORT AND COLLECTION SIMULATION ROUTINES             *
'*                                                                            *
'*  Given the amount and location of energy deposited in the scintillator by  *
'*  a neutron capture event, this section figures out how many photons are    *
'*  created, the direction each one goes, where it arrives on the surface of  *
```

```
'*  the scintillator, and what happens to it at that point.  Certain things   *
'*  that could be simulated are not being simulated as they aren't needed at  *
'*  this time.  However, the code will be organized such that additional       *
'*  functions may be easily added later on in appropriate places.              *
'*                                                                             *
'*******************************************************************************

'*******************************************************************************
'*                    Subroutine for Controlling Everything                    *
'*******************************************************************************

Public Sub RunLightTransportSim(scintillator_energy() As Single, EventDepths()
_
 As Single, NumEvents As Long)
   '
   Dim NumPhotons As Long
   Dim i As Long
   Dim ii As Long
   Dim xdir As Single
   Dim ydir As Single
   Dim zdir As Single
   Dim depth As Single
   Dim escaped As Boolean
   Dim ypos As Single
   Dim zpos As Single
   Dim NumPhotonsFront As Long
   Dim NumPhotonsBack As Long

   Open "LightEmission.txt" For Output As #1

   Print #1, "Event number" & Chr$(9) & "Photons reaching front" & Chr$(9) & _
    "Photons reaching back"

   For i = 1 To NumEvents Step 1
      NumPhotons = GetNumPhotons(CSng(scintillator_energy(i)))
      depth = EventDepths(i)
      NumPhotonsFront = 0
      NumPhotonsBack = 0

      For ii = 1 To NumPhotons Step 1
         Call GetRandomDirection(xdir, ydir, zdir)
         Call TrackPhotonToEdge(xdir, ydir, zdir, escaped, depth, ypos, zpos)

         If (escaped = True) Then
            If (xdir > 0#) Then
               NumPhotonsFront = NumPhotonsFront + 1
            Else
               NumPhotonsBack = NumPhotonsBack + 1
            End If
         End If
      Next ii

      Print #1, i & Chr$(9) & NumPhotonsFront & Chr$(9) & NumPhotonsBack
   Next i

   Close #1
End Sub

'*******************************************************************************
'*        Subroutine for Creating Photons and Finding Their Directions         *
'*******************************************************************************

Public Function GetNumPhotons(energydeposited As Double)
   'Returns the number of photons given a total amount of energy deposited in
the
   'scintillating dopant particles.  Statistical variation is included.
   Dim NumPhotons_fractional As Single
   Dim RandomNormal As Single
   Dim std_dev As Single

   NumPhotons_fractional = energydeposited / eVperPhoton
   RandomNormal = GetRandomNormal
   std_dev = (NumPhotons_fractional ^ 0.5) * (FanoFactor ^ 0.5)
   GetNumPhotons = Round(NumPhotons_fractional + RandomNormal * std_dev)

   If (GetNumPhotons < 1) Then
      GetNumPhotons = 0
```

```
   End If

   Exit Function
End Function

'*****************************************************************************
'*            Subroutine for Tracking Photons to Scintillator Edge           *
'*****************************************************************************

Public Sub TrackPhotonToEdge(xdir As Single, ydir As Single, zdir As Single, _
 escaped As Boolean, depth As Single, ypos As Single, zpos As Single)
   'UNTESTED ROUTINE! Note failure possibility: assumptions about coordinates
for
   'depth, scintillator thickness, etc, need to be correct.
   Dim escapedistance As Single
   Dim escapeprobability As Single
   Dim opticallengths As Single

   If (xdir < 0#) Then
      If (xdir > -1E-20) Then
         escaped = False
         Exit Sub
      Else
         escapedistance = -depth / xdir
      End If
   ElseIf (xdir > 0#) Then
      If (xdir < 1E-20) Then
         escaped = False
         Exit Sub
      Else
         escapedistance = (ScintillatorThickness - depth) / xdir
      End If
   Else
      escaped = False
      Exit Sub
   End If

   opticallengths = escapedistance / LightAttenuationLength
   escapeprobability = 0.5 ^ opticallengths

   If (Rnd > escapeprobability) Then
      escaped = False
      Exit Sub
   End If

   ypos = escapedistance * ydir
   zpos = escapedistance * zdir
   escaped = True

   Exit Sub
End Sub

'*****************************************************************************
'*                  Miscellaneous Mathematical Subroutines                   *
'*****************************************************************************

Public Function Round(Num As Single)
   'Round a number to the closest integer.
   Dim difference As Single
   Dim truncated As Long

   truncated = CLng(Num)
   difference = Num - CSng(truncated)

   If (difference >= 0.5) Then
      Round = truncated + 1
   ElseIf (difference <= -0.5) Then
      Round = truncated - 1
   Else
      Round = truncated
   End If
End Function

Public Sub GetRandomDirection(xdir As Single, ydir As Single, zdir As Single)
   'Returns the x, y, and z components of a randomly directed unit vector.
   Dim sumofsquares As Single
   Dim sqrofsumofsquares As Single
```

```
   Dim flag As Boolean

   flag = False

   Do While (flag = False)
      xdir = Rnd * 2 - 1
      ydir = Rnd * 2 - 1
      zdir = Rnd * 2 - 1

      sumofsquares = xdir ^ 2 + ydir ^ 2 + zdir ^ 2
      sqrofsumofsquares = sumofsquares ^ 0.5

      If (sumofsquares <= 1) And (sumofsquares > 0) Then
         'Enter if the point in inside the sphere.
         'Normalize to a unit vector.
         xdir = xdir / sqrofsumofsquares
         ydir = ydir / sqrofsumofsquares
         zdir = zdir / sqrofsumofsquares
         flag = True
      End If
   Loop
End Sub

Public Function GetRandomNormal()
   'Returns a random number selected from a normal distribution.
   Dim fac As Double
   Dim r As Double
   Dim v1 As Double
   Dim v2 As Double
   Dim flag As Boolean

   flag = False
   Do While (flag = False)
      v1 = 2 * Rnd - 1
      v2 = 2 * Rnd - 1
      r = v1 ^ 2 + v2 ^ 2
      If (r < 1) Then
         flag = True
      End If
   Loop
   fac = Sqr(-2 * Log(r) / r)
   GetRandomNormal = v2 * fac
End Function

'*****************************************************************************
'*                                                                           *
'*                    PARTICLE TRANSPORT SIMULATION ROUTINES                 *
'*                                                                           *
'*  Creating the reaction products, running TRIM, using the transport data to *
'*  follow the reactions products throught the matrix/scintillator composite  *
'*  and track their energy deposition, and so forth.                         *
'*                                                                           *
'*****************************************************************************

'*****************************************************************************
'*                    Subroutines for Controlling Everything                 *
'*****************************************************************************

Public Sub RunParticleTransportSim(scintillator_energy() As Double,
matrix_energy() _
 As Double, RunEvents As Long, particle_ran() As Boolean)
   '
   Dim xpos As Double
   Dim ypos As Double
   Dim zpos As Double
   Dim i As Long
   Dim ii As Long
   Dim temp$
   Dim ParticleEndInfo() As Double
   Dim readZ As Long
   Dim dummy As Long
   Dim dummy2 As Long
   Dim x As Double
   Dim y As Double
   Dim z As Double
   Dim particlenum As Long
   Dim previousparticlenum As Long
```

```
   Dim E As Double
   Dim x11 As Double
   Dim y11 As Double
   Dim z11 As Double
   Dim E11 As Double
   Dim x12 As Double
   Dim y12 As Double
   Dim z12 As Double
   Dim E12 As Double
   Dim x21 As Double
   Dim y21 As Double
   Dim z21 As Double
   Dim E21 As Double
   Dim x22 As Double
   Dim y22 As Double
   Dim z22 As Double
   Dim E22 As Double
   Dim elic As Double
   Dim EventNum As Long
   Dim ParticleRadii As Double
   Dim xlength As Double 'length of box in x dimension
   Dim ylength As Double '   "             y
   Dim zlength As Double '   "             z
   Dim xtrim As Double 'where the particle is in Trim coordinates
   Dim ytrim As Double '   "
   Dim ztrim As Double '   "
   Dim xtrimoffset As Double 'subtract these from the Trim coordinates before
processing
   Dim ytrimoffset As Double '   "
   Dim ztrimoffset As Double '   "
   Dim xcenter As Double 'x location of the dopant particle sphere within the
box
   Dim ycenter As Double 'y   "
   Dim zcenter As Double 'z   "
   Dim xreactpos As Double 'x location of the reaction
   Dim yreactpos As Double 'y   "
   Dim zreactpos As Double 'z   "
   Dim deltaE As Double
   Dim xleft As Double 'x, y, z distances left to go in the original 1 g/cm^3
segment travel distance
   Dim yleft As Double
   Dim zleft As Double
   Dim totalleft As Double 'distance left to go in original trim 1 g/cm^3
segment travel distance
   Dim energyleft As Double 'Within the context of the energy drop over an
individual travel segment.
   Dim energydrop As Double
   Dim travelfraction As Double
   Dim indopant As Boolean
   Dim stopflag As Boolean
   Dim stopflag2 As Boolean
   Dim xstart As Double
   Dim ystart As Double
   Dim zstart As Double
   Dim xcint As Double
   Dim ycint As Double
   Dim zcint As Double
   Dim senergydep As Double
   Dim menergydep As Double
   Dim prevparticlenum1 As Long
   Dim prevparticlenum2 As Long
   Dim particlenum1 As Long
   Dim particlenum2 As Long
   Dim currinputfile As Long
   Dim eventxcenter As Double
   Dim eventycenter As Double
   Dim eventzcenter As Double
   Dim errorinparticle As Boolean 'If something goes wrong with a particle
particle,
   'chuck out that particle and keep going instead of crashing the program with
a
   'fatal error.

   ReDim scintillator_energy(RunEvents)
   ReDim matrix_energy(RunEvents)
   ReDim particle_ran(RunEvents)
```

```
   For i = 1 To RunEvents Step 1
      particle_ran(i) = True
   Next i

   EventNum = RunEvents
   ChDir$ ("C:\Program Files\SRIM 2000")
   Call SetVals(ParticleRadii, xlength, ylength, zlength)

   If (UseExistingData = False) Then
      Call PrepareAndRunTrim(EventNum)
   End If

   'Create EXYZ.TXT files for each particle type including ending points.
   ReDim ParticleEndInfo(EventNum, 2, 4)

   For i = 1 To 2 Step 1
      Open path$ & "RANGE_3D" & i & ".TXT" For Input As #1
      For ii = 1 To 22 Step 1
         Line Input #1, temp$
      Next ii

      'The program will think it is hitting a final particle right before it
      'hits the end of file but it will make all the variables be zero and be
      'okay.
      particlenum = 0

      Do While (EOF(1) = False And particlenum < EventNum)
         Input #1, particlenum, xpos, ypos, zpos

         ParticleEndInfo(particlenum, i, 1) = xpos
         ParticleEndInfo(particlenum, i, 2) = ypos
         ParticleEndInfo(particlenum, i, 3) = zpos
         ParticleEndInfo(particlenum, i, 4) = 0#
      Loop
      Close #1
   Next i

   DoEvents

   Open "EXYZ_1.TXT" For Input As #1
   Open "EXYZ_2.TXT" For Input As #2

   For i = 1 To 15 Step 1
      Line Input #1, temp$
      Line Input #2, temp$
   Next i

   stopflag = False
   prevparticlenum1 = 0
   prevparticlenum2 = 0
   currinputfile = 1
   errorinparticle = False

   If (EOF(1) = True Or EOF(2) = True) Then
      MsgBox "EXYZ_x.TXT file empty - no reaction product transport data.", _
      vbExclamation, "Fatal Error:"
      End
   End If

   Input #1, particlenum1, E11, x11, y11, z11, elic
   Input #2, particlenum2, E21, x21, y21, z21, elic

   'Convert energies from keV to eV.
   E11 = E11 * 1000#
   E21 = E21 * 1000#

   i = 0
   xstart = 0#
   ystart = 0#
   zstart = 0#

   Call PickSphereLocation(ParticleRadii, xstart, ystart, zstart, xlength,
ylength, _
   zlength, xcenter, ycenter, zcenter)

   DoEvents
```

```
   Do While (stopflag = False)
      Call PickReactionLocation(xstart, ystart, zstart, xlength, ylength,
zlength, _
      ParticleRadii, xcenter, ycenter, zcenter, xreactpos, yreactpos,
zreactpos)

      If (TestPointPositionSphere(xreactpos, yreactpos, zreactpos, xcenter,
ycenter, _
      zcenter, ParticleRadii) = 0 And TestPointPositionBox(xreactpos,
yreactpos, _
      zreactpos, xstart, ystart, zstart, xlength, ylength, zlength) = 2) Then
         stopflag = True
         indopant = False
      End If
   Loop

   eventxcenter = xcenter
   eventycenter = ycenter
   eventzcenter = zcenter
   xcint = xreactpos
   ycint = yreactpos
   zcint = zreactpos
   indopant = False
   stopflag = False

   Do While (stopflag = False)
      DoEvents
      If (currinputfile = 1) Then
         x12 = x11
         y12 = y11
         z12 = z11
         E12 = E11
         prevparticlenum1 = particlenum1

         If (EOF(1) = False) Then
            Input #1, particlenum1, E11, x11, y11, z11, elic
            E11 = E11 * 1000# 'Convert from keV to eV
            particlenum = particlenum1
            previousparticlenum = prevparticlenum1
         Else
            particlenum1 = particlenum1 + 1
            particlenum = particlenum1
            previousparticlenum = prevparticlenum1
         End If
      Else
         x22 = x21
         y22 = y21
         z22 = z21
         E22 = E21
         prevparticlenum2 = particlenum2

         If (EOF(2) = False) Then
            Input #2, particlenum2, E21, x21, y21, z21, elic
            E21 = E21 * 1000# 'Convert from keV to eV
            particlenum = particlenum2
            previousparticlenum = prevparticlenum2
         Else
            particlenum2 = particlenum2 + 1
            particlenum = particlenum2
            previousparticlenum = prevparticlenum2
         End If

         If (EOF(1) = True And EOF(2) = True) Then
            'There is no equivalent statement included in the currinputfile = 1
case.
            'This is deliberate! (It isn't needed.)  This is needed hear to
handle
            'the final particle in the second input file as EOF(2) doesn't
become
            'true until after the Input #2 statement a few lines back *after*
EOF(2)
            'has been tested and found to be false.  Thus it can't actually
become
            'true until after that statement.  The EOF(2) = True case can
likely be
            'dispensed with.  I haven't tried this as everything works and I'd
rather
```

```
            'leave well enough alone as long as it works correctly.

            particlenum2 = particlenum2 + 1
            particlenum = particlenum2
            previousparticlenum = prevparticlenum2
         End If
      End If

      If (EOF(1) = True And EOF(2) = True) Then
         If (particlenum < RunEvents) Then
            MsgBox "Ran fewer events than requested. RolloverNumber is too
high.", , "Fatal Error:"
            End
         End If

         stopflag = True
      End If

      If (particlenum <> previousparticlenum) Then
         If (previousparticlenum > 0) Then
            'Run the segment between the last point in EXYZ_x.TXT and the final
            'stopping point of the particle.
            If (currinputfile = 1) Then
               xleft = ParticleEndInfo(previousparticlenum, 1, 1) - x12
               yleft = ParticleEndInfo(previousparticlenum, 1, 2) - y12
               zleft = ParticleEndInfo(previousparticlenum, 1, 3) - z12
               energyleft = E12 - ParticleEndInfo(previousparticlenum, 1, 4)
            Else
               xleft = ParticleEndInfo(previousparticlenum, 2, 1) - x22
               yleft = ParticleEndInfo(previousparticlenum, 2, 2) - y22
               zleft = ParticleEndInfo(previousparticlenum, 2, 3) - z22
               energyleft = E22 - ParticleEndInfo(previousparticlenum, 2, 4)
            End If

            If (errorinparticle = False) Then
               Call RunParticleTrackSegment(xcint, ycint, zcint, indopant,
xleft, yleft, _
               zleft, energyleft, senergydep, menergydep, xstart, ystart,
zstart, xlength, _
               ylength, zlength, xcenter, ycenter, zcenter, ParticleRadii,
errorinparticle)
            End If

            If (errorinparticle = True) Then
               particle_ran(previousparticlenum) = False
            End If

            scintillator_energy(previousparticlenum) =
scintillator_energy(previousparticlenum) + senergydep
            matrix_energy(previousparticlenum) =
matrix_energy(previousparticlenum) + menergydep
         Else
            'There was no previous particle.
         End If

         If (particlenum > EventNum And currinputfile = 2) Then
            'We've run as many cases as requested.
            Close #1
            Close #2
            Exit Sub
         End If

         If (particlenum = 0 And currinputfile = 2) Then
            'We've exhausted all of our data in the TRIM output files.
            'WARNING - This possibility is probably not fully supported in the
code.
            'I have not yet encountered it. (I don't think it can happen unless
the
            'two EXYZ files have different numbers of particles in them due to
previous
            'tests and actions in this subroutine.)
            Close #1
            Close #2
            Exit Sub
         End If

         xstart = 0#
```

```
         ystart = 0#
         zstart = 0#

         If (currinputfile = 1) Then
            'Insert something to go to the second particle.
            currinputfile = 2
            'Use the same reaction location as for the first particle.
            xcint = xreactpos
            ycint = yreactpos
            zcint = zreactpos
            indopant = False
            xcenter = eventxcenter
            ycenter = eventycenter
            zcenter = eventzcenter
         Else
            'Insert something to go to the next event.
            currinputfile = 1
            stopflag2 = False

            Call PickSphereLocation(ParticleRadii, xstart, ystart, zstart,
xlength, _
            ylength, zlength, xcenter, ycenter, zcenter)

            Do While (stopflag2 = False)
               Call PickReactionLocation(xstart, ystart, zstart, xlength,
ylength, _
               zlength, ParticleRadii, xcenter, ycenter, zcenter, xreactpos,
yreactpos, _
               zreactpos)

               If (TestPointPositionSphere(xreactpos, yreactpos, zreactpos,
xcenter, _
               ycenter, zcenter, ParticleRadii) = 0 And
TestPointPositionBox(xreactpos, _
               yreactpos, zreactpos, xstart, ystart, zstart, xlength, ylength,
zlength) _
               = 2) Then
                  stopflag2 = True
                  indopant = False
               End If
            Loop

            eventxcenter = xcenter
            eventycenter = ycenter
            eventzcenter = zcenter
            indopant = False
            xcint = xreactpos
            ycint = yreactpos
            zcint = zreactpos
         End If

         errorinparticle = False
      Else
         'It's still the same particle.
         If (currinputfile = 1) Then
            xleft = x12 - x11
            yleft = y12 - y11
            zleft = z12 - z11
            energyleft = E12 - E11
         Else
            xleft = x22 - x21
            yleft = y22 - y21
            zleft = z22 - z21
            energyleft = E22 - E21
         End If

         If (errorinparticle = False) Then
            Call RunParticleTrackSegment(xcint, ycint, zcint, indopant, xleft,
yleft, _
            zleft, energyleft, senergydep, menergydep, xstart, ystart, zstart,
xlength, _
            ylength, zlength, xcenter, ycenter, zcenter, ParticleRadii,
errorinparticle)
         End If

         If (errorinparticle = True) Then
            particle_ran(particlenum) = False
```

```
         Else
            scintillator_energy(particlenum) = scintillator_energy(particlenum)
+ senergydep
            matrix_energy(particlenum) = matrix_energy(particlenum) +
menergydep
         End If
      End If
   Loop

   Close #1
   Close #2

   Exit Sub
End Sub

Public Sub PrepareAndRunTrim(NumberOfRuns As Long)
   'This subroutine is used if new particle transport data is to be generated
by
   'Trim.  Reading in reaction locations from MCNP runs is optional and is also
   'handled in this subroutine.
   'It is assumed that all target material is the same (i.e. no mixing of Li-6
and B-10).
   Dim fullpath$
   Dim atomtype As Long
   Dim xpos As Double
   Dim ypos As Double
   Dim zpos As Double
   Dim xcos As Double
   Dim ycos As Double
   Dim zcos As Double
   Dim energy As Double
   Dim P1M As Double
   Dim P1Z As Long
   Dim E1 As Double
   Dim x1 As Double
   Dim y1 As Double
   Dim z1 As Double
   Dim P2M As Double
   Dim P2Z As Long
   Dim E2 As Double
   Dim x2 As Double
   Dim y2 As Double
   Dim z2 As Double
   Dim i As Long
   Dim temp$
   Dim spacestring$
   Dim xtrimstart As Double
   Dim ytrimstart As Double
   Dim ztrimstart As Double
   Dim xtrimthickness As Double
   Dim ENeutron As Double
   Dim ScintThickness As Double
   Dim opticallength As Double
   Dim EventNum As Long
   Dim currentrnd As Double
   Dim probability As Double
   Dim ratio As Double
   Dim TravelDistance As Double
   Dim RandomSeed As Long

   ENeutron = NeutronEnergy
   ScintThickness = ScintillatorThickness
   EventNum = NumberOfRuns

   If (RunFromMCNP = True) Then
      path$ = "C:\mcnp4c2\"
      fullpath$ = path$ & "iSGPPpositions.txt"
      Open fullpath$ For Input As #1
   End If

   path$ = "C:\Program Files\SRIM 2000\"
   fullpath$ = path$ & "OLDTRIM.DAT"
   Open fullpath$ For Input As #2
   fullpath$ = path$ & "TRIM1.DAT"
   Open fullpath$ For Output As #10
   fullpath$ = path$ & "TRIM2.DAT"
   Open fullpath$ For Output As #11
```

```
   'Insert boilerplate at the start of the file.
   For i = 1 To 10 Step 1
      Line Input #2, temp$
      Print #10, temp$
      Print #11, temp$
   Next i
   Close #2

   EventNum = 0
   'Position values at which particles are to start.
   ytrimstart = 0#
   ztrimstart = 0#
   'Assumes MaxH3Range > MaxAlphaRange.
   xtrimstart = MaxH3Range
   xtrimthickness = 2# * MaxH3Range

   If (RunFromMCNP = True) Then
      'Read in neutron info and reaction locations from the MCNP output file.
      Do While (EOF(1) = False And EventNum < NumberOfRuns)
         Input #1, atomtype, xpos, ypos, zpos, xcos, ycos, zcos, energy
         Call TransformCoordinates(xpos, ypos, zpos, xcos, ycos, zcos)
         If (TestPointPlanar(xpos) = False) Then
            MsgBox "TestPoint indicated the start point is incorrectly
located.", , "Error:"
            MsgBox xpos, , "xpos:"
            End
         End If

         Call CalculateAllNeutronReactionInfo(energy, atomtype, xcos, ycos,
zcos, P1M, _
         P1Z, E1, x1, y1, z1, P2M, P2Z, E2, x2, y2, z2)

         EventNum = EventNum + 1
         spacestring$ = ""
         If (EventNum < 10) Then
            spacestring$ = "       "
         ElseIf (EventNum < 100) Then
         spacestring$ = "      "
         ElseIf (EventNum < 1000) Then
            spacestring$ = "     "
         ElseIf (EventNum < 10000) Then
            spacestring$ = "    "
         ElseIf (EventNum < 100000) Then
            spacestring$ = "   "
         Else
            spacestring$ = "  "
         End If

         'NOTE: TRIM requests directional cosines and MCNP gives direction
vectors.  They are numerically identical.
         Print #10, EventNum & spacestring$ & P1M & " " & Format(E1,
"00000000.000") _
         & " " & xtrimstart & " " & ytrimstart & " " & ztrimstart & " " &
Format(x1, _
         "0.0000") & " " & Format(y1, "0.0000") & " " & Format(z1, "0.0000")
         Print #11, EventNum & spacestring$ & P2M & " " & Format(E2,
"00000000.000") _
         & " " & xtrimstart & " " & ytrimstart & " " & ztrimstart & " " &
Format(x2, _
         "0.0000") & " " & Format(y2, "0.0000") & " " & Format(z2, "0.0000")
      Loop
   Else
      'Generate neutron reaction locations using the neutron absorption cross
sections.
      opticallength = MacroscopicXSection(ENeutron) * ScintThickness

      For i = 1 To NumberOfRuns Step 1
         currentrnd = Rnd
         probability = 1# - ee ^ -opticallength
         ratio = 1# / probability
         TravelDistance = -Log(1# - currentrnd / ratio)

         Call CalculateAllNeutronReactionInfo(ENeutron, 3006, 1#, 0#, 0#, P1M,
_
         P1Z, E1, x1, y1, z1, P2M, P2Z, E2, x2, y2, z2)
```

```
         EventNum = EventNum + 1
         spacestring$ = ""
         If (EventNum < 10) Then
            spacestring$ = "       "
         ElseIf (EventNum < 100) Then
            spacestring$ = "      "
         ElseIf (EventNum < 1000) Then
            spacestring$ = "     "
         ElseIf (EventNum < 10000) Then
            spacestring$ = "    "
         ElseIf (EventNum < 100000) Then
            spacestring$ = "   "
         Else
            spacestring$ = "  "
         End If

         Print #10, EventNum & spacestring$ & P1M & " " & Format(E1,
"00000000.000") _
         & " " & xtrimstart & " " & ytrimstart & " " & ztrimstart & " " &
Format(x1, _
         "0.0000") & " " & Format(y1, "0.0000") & " " & Format(z1, "0.0000")
         Print #11, EventNum & spacestring$ & P2M & " " & Format(E2,
"00000000.000") _
         & " " & xtrimstart & " " & ytrimstart & " " & ztrimstart & " " &
Format(x2, _
         "0.0000") & " " & Format(y2, "0.0000") & " " & Format(z2, "0.0000")
      Next i
   End If

   'Insert an end-of-file character at the end of each file (this *must* be
done)
   'and then close it.
   Print #10, Chr$(26)
   Print #11, Chr$(26)
   Close #10
   Close #11

   Close #1
   DoEvents

   'Now create the appropriate TRIM_WIN.IN files.
   RandomSeed = Int(Rnd * 999999#)
   Call SetupModifiedTrim_Win_In(P1Z, P1M, xtrimthickness, RandomSeed)
   path$ = "C:\Program Files\SRIM 2000\"
   FileCopy path$ & "TRIM_WIN.IN", path$ & "TRIM_WIN1.IN"

   RandomSeed = Int(Rnd * 999999#)
   Call SetupModifiedTrim_Win_In(P2Z, P2M, xtrimthickness, RandomSeed)
   path$ = "C:\Program Files\SRIM 2000\"
   FileCopy path$ & "TRIM_WIN.IN", path$ & "TRIM_WIN2.IN"

   DoEvents

   'Now run the Trim input files.
   '(All info should be in EXYZ and RANGE_3D files - no particles should be
able to escape.)
   FileCopy path$ & "TRIM1.DAT", path$ & "TRIM.DAT"
   FileCopy path$ & "TRIM_WIN1.IN", path$ & "TRIM_WIN.IN"
   ChDir path$
   Call Run_Modified_TRIM
   FileCopy path$ & "EXYZ.TXT", path$ & "EXYZ_1.TXT"
   FileCopy path$ & "RANGE_3D.TXT", path$ & "RANGE_3D1.TXT"

   DoEvents

   FileCopy path$ & "TRIM2.DAT", path$ & "TRIM.DAT"
   FileCopy path$ & "TRIM_WIN2.IN", path$ & "TRIM_WIN.IN"
   ChDir path$
   Call Run_Modified_TRIM
   FileCopy path$ & "EXYZ.TXT", path$ & "EXYZ_2.TXT"
   FileCopy path$ & "RANGE_3D.TXT", path$ & "RANGE_3D2.TXT"

   Exit Sub
End Sub

'*************************************************************************
'*          Subroutine to Set and Check Some Variables at Startup        *
```

```
'*************************************************************************************

Public Sub SetVals(ParticleRadii As Double, xlength As Double, ylength As
Double, zlength As Double)
   Dim length As Double
   'Set and check critical variable values at startup.
   If (VolumeDopingFraction > 0.5) Then
      MsgBox "Volume doping fraction cannot exceed 0.5.", vbExclamation, "Fatal
Error:"
      End
   End If
   ParticleRadii = ParticleRadiiMicrons * 10000# 'Convert microns to angstroms.

   length = (((4# / 3#) * Pi * ParticleRadii ^ 3) / VolumeDopingFraction) ^ (1#
/ 3#)
   'In this case we're using a box of equal side lengths.
   xlength = length
   ylength = length
   zlength = length
End Sub

'*************************************************************************************
'*            Subroutine for Running Individual Track Segments                       *
'*************************************************************************************

Public Sub RunParticleTrackSegment(xcint As Double, ycint As Double, zcint As
Double, _
 indopant As Boolean, xleft As Double, yleft As Double, zleft As Double,
energyleft As _
 Double, senergydep As Double, menergydep As Double, xstart As Double, ystart
As Double, _
 zstart As Double, xlength As Double, ylength As Double, zlength As Double,
xcenter As _
 Double, ycenter As Double, zcenter As Double, ParticleRadii As Double,
errorinparticle _
 As Boolean)
   '
   Dim xcross1 As Double
   Dim ycross1 As Double
   Dim zcross1 As Double
   Dim cross1 As Boolean
   Dim xcross2 As Double
   Dim ycross2 As Double
   Dim zcross2 As Double
   Dim cross2 As Boolean
   Dim xtest As Double
   Dim ytest As Double
   Dim ztest As Double
   Dim fraction As Double
   Dim energydrop As Double
   Dim totalleft As Double
   Dim posval As Long
   Dim i As Long

   menergydep = 0#
   senergydep = 0#
   totalleft = (xleft ^ 2 + yleft ^ 2 + zleft ^ 2) ^ 0.5
   i = 0

   Do While (energyleft > 0#)
      DoEvents

      i = i + 1
      totalleft = (xleft ^ 2 + yleft ^ 2 + zleft ^ 2) ^ 0.5

      'Adding this to make sure the indopant value is set correctly.  I think
the
      'indopant value occasionally gets messed up when the particle crosses
surfaces
      'due to rounding and precision errors.
      posval = TestPointPositionSphere(xcint, ycint, zcint, xcenter, ycenter,
zcenter, ParticleRadii)
      If (posval = 0) Then
         indopant = False
      ElseIf (posval = 2) Then
         indopant = True
      End If
```

```
      If (indopant = False) Then
         xtest = xcint + xleft / MatrixDensity
         ytest = ycint + yleft / MatrixDensity
         ztest = zcint + zleft / MatrixDensity
      Else
         xtest = xcint + xleft / DopantDensity
         ytest = ycint + yleft / DopantDensity
         ztest = zcint + zleft / DopantDensity
      End If

      'Test to see if it crosses the dopant particle boundary.
      Call FindCrossingPointSpherical(indopant, xcint, ycint, zcint, xtest,
ytest, _
      ztest, xcross1, ycross1, zcross1, cross1, xcross2, ycross2, zcross2,
cross2, _
      xcenter, ycenter, zcenter, ParticleRadii, errorinparticle)

      If (errorinparticle = True) Then
         Exit Sub
      End If

      If (cross1 = True) Then
         If (indopant = True) Then
            fraction = Distance(xcint, ycint, zcint, xcross1, ycross1, zcross1)
* _
            DopantDensity / totalleft
         Else
            fraction = Distance(xcint, ycint, zcint, xcross1, ycross1, zcross1)
* _
            MatrixDensity / totalleft
         End If

         If (fraction > 1.000001) Then
            MsgBox "Travel fraction too large in particle transport section.
Debug.", _
            vbExclamation, "Fatal Error:"
            End
         End If
         If (fraction > 1#) Then
            fraction = 1#
         End If
         If (fraction < 0#) Then
            MsgBox "Travel fraction negative in particle transport section.
Debug.", _
            vbExclamation, "Fatal Error:"
            End
         End If

         xleft = (1# - fraction) * xleft
         yleft = (1# - fraction) * yleft
         zleft = (1# - fraction) * zleft
         energydrop = fraction * energyleft
         energyleft = (1# - fraction) * energyleft

         'Add the lost energy to the current event bin.
         If (indopant = True) Then
            senergydep = senergydep + energydrop
         Else
            menergydep = menergydep + energydrop
         End If

         'Set the current location values.
         xcint = xcross1
         ycint = ycross1
         zcint = zcross1

         'Change the indopant boolean value if needed
         If (indopant = False) Then
            indopant = True
         Else
            indopant = False
         End If
      ElseIf (cross1 = False) Then
         If (indopant = True) Then
            'We have already established that it does not exit the dopant
sphere.
```

```
            'Thus all the remaining energy gets deposited in the dopant
particle.
            xcint = xcint + xleft / DopantDensity
            ycint = ycint + yleft / DopantDensity
            zcint = zcint + zleft / DopantDensity
            xleft = 0#
            yleft = 0#
            zleft = 0#
            energydrop = energyleft
            energyleft = 0#
            senergydep = senergydep + energydrop
         Else
            'Do some error checking first. (Mainly for me while I program
this.)
            If (TestPointPositionSphere(xcint, ycint, zcint, xcenter, ycenter,
zcenter, _
             ParticleRadii) = 2) Then
               MsgBox "Location indicators are contradictory in
RunParticleTrackSegment(). Debug.", _
               vbExclamation, "Debug:"
               End
            End If

            'Test to see if it exits the box region.
            Call FindCrossingPointBox(xcint, ycint, zcint, xtest, ytest, ztest,
xcross1, _
            ycross1, zcross1, cross1, xcross2, ycross2, zcross2, cross2,
xstart, ystart, _
            zstart, xlength, ylength, zlength, errorinparticle)

            If (errorinparticle = True) Then
               Exit Sub
            End If

            If (cross1 = False) Then
               'It stayed inside the box.  All energy is deposited in the
matrix.
               xcint = xcint + xleft / MatrixDensity
               ycint = ycint + yleft / MatrixDensity
               zcint = zcint + zleft / MatrixDensity
               xleft = 0#
               yleft = 0#
               zleft = 0#
               energydrop = energyleft
               energyleft = 0#
               menergydep = menergydep + energydrop
            Else
               'It exited the box.  Move the particle up to the crossing point.
               fraction = Distance(xcint, ycint, zcint, xcross1, ycross1,
zcross1) * _
               MatrixDensity / totalleft

               xleft = (1# - fraction) * xleft
               yleft = (1# - fraction) * yleft
               zleft = (1# - fraction) * zleft
               energydrop = fraction * energyleft
               energyleft = (1# - fraction) * energyleft
               menergydep = menergydep + energydrop

               'Set the current location values.
               xcint = xcross1
               ycint = ycross1
               zcint = zcross1

               'Reset the box position.
               'Find which plane the particle exits and go from there.
               If (xcint = xstart) Then
                  xstart = xstart - xlength
                  'The two lines below give a random offset in the two
directions where the
                  'crossing did not happen.  This should make the simulation
more accurate.
                  'They can be removed if desired; only the change in xstart is
necessary.
                  ystart = (ycint - 2# * Tolerance - ylength) + Rnd * (ylength
- 4# * Tolerance)
```

```
                  zstart = (zcint - 2# * Tolerance - zlength) + Rnd * (zlength
- 4# * Tolerance)
               ElseIf (xcint = xstart + xlength) Then
                  xstart = xstart + xlength
                  ystart = (ycint - 2# * Tolerance - ylength) + Rnd * (ylength
- 4# * Tolerance)
                  zstart = (zcint - 2# * Tolerance - zlength) + Rnd * (zlength
- 4# * Tolerance)
               ElseIf (ycint = ystart) Then
                  ystart = ystart - ylength
                  xstart = (xcint - 2# * Tolerance - xlength) + Rnd * (xlength
- 4# * Tolerance)
                  zstart = (zcint - 2# * Tolerance - zlength) + Rnd * (zlength
- 4# * Tolerance)
               ElseIf (ycint = ystart + ylength) Then
                  ystart = ystart + ylength
                  xstart = (xcint - 2# * Tolerance - xlength) + Rnd * (xlength
- 4# * Tolerance)
                  zstart = (zcint - 2# * Tolerance - zlength) + Rnd * (zlength
- 4# * Tolerance)
               ElseIf (zcint = zstart) Then
                  zstart = zstart - zlength
                  xstart = (xcint - 2# * Tolerance - xlength) + Rnd * (xlength
- 4# * Tolerance)
                  ystart = (ycint - 2# * Tolerance - ylength) + Rnd * (ylength
- 4# * Tolerance)
               ElseIf (zcint = zstart + zlength) Then
                  zstart = zstart + zlength
                  xstart = (xcint - 2# * Tolerance - xlength) + Rnd * (xlength
- 4# * Tolerance)
                  ystart = (ycint - 2# * Tolerance - ylength) + Rnd * (ylength
- 4# * Tolerance)
               End If

               Call PickSphereLocation(ParticleRadii, xstart, ystart, zstart,
xlength, _
               ylength, zlength, xcenter, ycenter, zcenter)
            End If
         End If
      End If
   Loop

   DoEvents

   Exit Sub
End Sub

'*************************************************************************
'*          Subroutines for Setting up and Controlling TRIM Runs         *
'*************************************************************************

Public Sub CalculateAllNeutronReactionInfo(ByVal ENeutron As Double, ByVal
TargetAtom _
 As Long, nxcos As Double, nycos As Double, nzcos As Double, P1M As Double, P1Z
As _
 Long, E1 As Double, x1 As Double, y1 As Double, z1 As Double, P2M As Double,
P2Z As _
 Long, E2 As Double, x2 As Double, y2 As Double, z2 As Double)
   'This sub is intended to cover all aspects of the reaction product creation.
   'ENeutron is in MeV.
   Dim sumofsquares As Double
   Dim sqrofsumofsquares As Double
   Dim flag As Boolean
   Dim xdir As Double
   Dim ydir As Double
   Dim zdir As Double
   Dim Va As Double 'velocity of neutron (particle "a" in reaction)
   Dim Vax As Double
   Dim Vay As Double
   Dim Vaz As Double
   Dim Vb As Double
   Dim Vbx As Double
   Dim Vby As Double
   Dim Vbz As Double
   Dim Vclosing As Double
   Dim Q As Double 'in MeV
   Dim Vcom As Double
```

```
Dim Vcomx As Double
Dim Vcomy As Double
Dim Vcomz As Double
Dim TM As Double 'target mass in amu
Dim Vc As Double
Dim Vcx As Double
Dim Vcy As Double
Dim Vcz As Double
Dim Vd As Double
Dim Vdx As Double
Dim Vdy As Double
Dim Vdz As Double
Dim Ecom As Double
Dim Elab As Double
Dim nxdir As Double
Dim nydir As Double
Dim nzdir As Double
Dim copyx1 As Double
Dim copyy1 As Double
Dim copyz1 As Double
Dim copyx2 As Double
Dim copyy2 As Double
Dim copyz2 As Double

'Transform directional cosines to direction vector
If (nxcos = 1#) Then
   nxdir = 1#
ElseIf (nxcos = -1#) Then
   nxdir = -1#
Else
   nxdir = Atn(-nxcos / Sqr(-nxcos * nxcos + 1)) + 2 * Atn(1)
End If
If (nycos = 1#) Then
   nydir = 1#
ElseIf (nycos = -1#) Then
   nydir = -1#
Else
   nydir = Atn(-nycos / Sqr(-nycos * nycos + 1)) + 2 * Atn(1)
End If
If (nzcos = 1#) Then
   nzdir = 1#
ElseIf (nzcos = -1#) Then
   nzdir = -1#
Else
   nzdir = Atn(-nzcos / Sqr(-nzcos * nzcos + 1)) + 2 * Atn(1)
End If

'Get the product types.
If (TargetAtom = 5010) Then
   P1M = 4.002
   P1Z = 2
   P2M = 7.014
   P2Z = 3
ElseIf (TargetAtom = 3006) Then
   P1M = 3.016
   P1Z = 1
   P2M = 4.002
   P2Z = 2
Else
   MsgBox "Unrecognized neutron target type.", , "Error:"
   End
End If

'Get the product energies (before adding the neutron energy) and Q values.
If (TargetAtom = 5010) Then
   If (Rnd < 0.92) Then
      E1 = 1.47
      E2 = 0.84
      Q = 2.31
   Else
      E1 = 1.777
      E2 = 1.015
      Q = 2.792
   End If
   TM = 10.007
ElseIf (TargetAtom = 3006) Then
   E1 = 2.73
```

```
      E2 = 2.05
      Q = 4.78
      TM = 6.009
   Else
      MsgBox "Unrecognized neutron target type.", , "Error:"
      End
   End If

   flag = False

   'Returns the x, y, and z components of a randomly directed unit vector.
   Do While (flag = False)
      xdir = Rnd * 2 - 1
      ydir = Rnd * 2 - 1
      zdir = Rnd * 2 - 1

      sumofsquares = xdir ^ 2 + ydir ^ 2 + zdir ^ 2
      sqrofsumofsquares = sumofsquares ^ 0.5

      If (sumofsquares <= 1) And (sumofsquares > 0) Then
         'Enter if the point is inside the sphere.
         'Normalize to a unit vector.
         xdir = xdir / sqrofsumofsquares
         ydir = ydir / sqrofsumofsquares
         zdir = zdir / sqrofsumofsquares
         flag = True
      End If
   Loop

   'Randomly select which one gets which set of directions.  This is done as a
   'precaution to help keep the directions totally random.
   If (Rnd < 0.5) Then
      x1 = xdir
      y1 = ydir
      z1 = zdir
      x2 = -xdir
      y2 = -ydir
      z2 = -zdir
   Else
      x1 = -xdir
      y1 = -ydir
      z1 = -zdir
      x2 = xdir
      y2 = ydir
      z2 = zdir
   End If

   'Determine the initial speeds of the incident particle and target and the
closing
   'speed in the lab frame as a fraction of c, the speed of light.
   Va = (2# * ENeutron / (1.008664 * 931.5)) ^ 0.5
   Vax = Va * nxdir
   Vay = Va * nydir
   Vaz = Va * nzdir

   Vb = 0#
   Vbx = 0#
   Vby = 0#
   Vbz = 0#

   Vcom = (1.008664 * Va) / (1.008664 + TM)
   Vcomx = Vcom * nxdir
   Vcomy = Vcom * nydir
   Vcomz = Vcom * nzdir

   Vax = Vax - Vcomx
   Vay = Vay - Vcomy
   Vaz = Vaz - Vcomz
   Va = (Vax ^ 2# + Vay ^ 2# + Vaz ^ 2#) ^ 0.5

   Vbx = Vbx - Vcomx
   Vby = Vby - Vcomy
   Vbz = Vbz - Vcomz
   Vb = (Vbx ^ 2# + Vby ^ 2# + Vbz ^ 2#) ^ 0.5

   'Find the total kinetic energy in the COM system and add the Q value.
```

```
   Ecom = (0.5 * 1.008664 * 931.5 * (Va ^ 2)) + (0.5 * TM * 931.5 * (Vb ^ 2)) +
Q

   'Find the velocities of the two reaction products in the COM system.  (The
directions
   'don't matter at this point.)
   Vd = ((2# * Ecom * P1M) / (931.5 * (P2M ^ 2 + P1M * P2M))) ^ 0.5
   Vc = (P2M / P1M) * Vd

   E1 = (0.5 * P1M * 931.5 * (Vc ^ 2))
   E2 = (0.5 * P2M * 931.5 * (Vd ^ 2))

   'Find the individual directional components and add the COM speed.
   Vcx = Vc * x1
   Vcy = Vc * y1
   Vcz = Vc * z1
   Vdx = Vd * x2
   Vdy = Vd * y2
   Vdz = Vd * z2

   Vcx = Vcx + Vcomx
   Vcy = Vcy + Vcomy
   Vcz = Vcz + Vcomz
   Vdx = Vdx + Vcomx
   Vdy = Vdy + Vcomy
   Vdz = Vdz + Vcomz

   'Calculate the total velocities.
   Vc = (Vcx ^ 2 + Vcy ^ 2 + Vcz ^ 2) ^ 0.5
   Vd = (Vdx ^ 2 + Vdy ^ 2 + Vdz ^ 2) ^ 0.5

   'Calculate the new direction unit vectors.
   x1 = Vcx / Vc
   y1 = Vcy / Vc
   z1 = Vcz / Vc
   x2 = Vdx / Vd
   y2 = Vdy / Vd
   z2 = Vdz / Vd

   'Calculate the new energies.
   E1 = (0.5 * P1M * 931.5 * (Vc ^ 2))
   E2 = (0.5 * P2M * 931.5 * (Vd ^ 2))

   'Convert the energies from MeV to eV.
   E1 = E1 * 1000000#
   E2 = E2 * 1000000#

   Exit Sub
End Sub

Public Sub SetupModifiedTrim_Win_In(ByVal PZ As Long, ByVal PM As Double, ByVal
Thickness As Long, ByVal RandomSeed As Long)
   'Thickness needs to be in Angstroms (either that or put something in to
convert to Angstroms)
   path$ = "C:\Program Files\SRIM 2000\"
   Open path$ & "trim_win.in" For Output As #1

   Print #1, "==> SRIM-2000.41 <== TRIM.IN is for TRIM(DOS), TRIM(Win) uses
TRIM_WIN.IN ==="
   Print #1, "Ion: Z1 ,  M1,  Energy (keV), Angle,Number,Bragg Corr,AutoSave
Number."
   Print #1, "   " & PZ & "   " & PM & "   20000   0   99999   0   10000"
   Print #1, "Cascades(1=No;2=Full;3=Sputt;4-5=Ions;6-7=Neutrons), Random
Number Seed, Reminders"
   Print #1, "    4   " & RandomSeed & "   0"
   Print #1, "Diskfiles (0=no,1=yes): Ranges, Backscatt, Transmit, Sputtered,
Recoils (2=Alt-C)"
   Print #1, "    1   1   1   1   " & CStr(EnergyStepSize)
   Print #1, "Target material : Number of Elements & Layers"
   Print #1, Chr$(34) & "Target into layer" & Chr$(34) & "    6    1"
   Print #1, "Target Energies (eV): Binding, Surface, Individual Displacement"
   Print #1, "              3.5   4      15     15     15     15     15     15"
   Print #1, "PlotType (0-5); Plot Depths: Xmin, Xmax(Ang.) [=0 0 for Viewing
Full Target]"
   Print #1, "       5                              0            0"
   'Data for lithiated sol-gel glass + ZnS:Ag scintillator
   Print #1, "Target Elements:    Z Mass(amu)"
```

```
   Print #1, "Atom 1 = Li  =        3    6.065"
   Print #1, "Atom 2 = Si  =       14   28.086"
   Print #1, "Atom 3 = O   =        8   15.999"
   Print #1, "Atom 4 = Zn  =       30   65.39"
   Print #1, "Atom 5 = S   =       16   32.066"
   Print #1, "Atom 6 = Ag  =       47  107.87"
   Print #1, "Layer   Layer Name /                Width Density     Li(3)
Si(14)    O(8)    Zn(30)    S(16)    Ag(47)"
   Print #1, "Numb.   Description                 (Ang) (g/cm3)    Stoich
Stoich   Stoich   Stoich   Stoich    Stoich"
   Print #1, " 1      " & Chr$(34) & "doped glass solid" & Chr$(34) & "
" & Str$(Thickness) & "  2.203          .1384      .1737      .3473      .1698
.1698      .0010"
   Print #1, "0  Target layer phases (0=Solid, 1=Gas)"
   Print #1, "0"
   Print #1, "Target Compound Corrections (Bragg)"
   Print #1, "0"
   Print #1, "Target atom displacement energies (eV)"
   Print #1, "       15     15     15     15     15     15"
   Print #1, "Target atom lattice binding energies (eV)"
   Print #1, "       3.5    3.5    3.5    3.5    3.5    3.5"
   Print #1, "Target atom surface binding energies (eV)"
   Print #1, "       4      4      4      4      4      4"
   Print #1, "Stopping Power Version (2=1996, 0=2000, 1=2000 interpolated)"
   Print #1, "0"
   Print #1, Chr$(26) 'Insert an end-of-file character at the end of the file.
This
   '*must* be done.

   Close #1

   Exit Sub
End Sub

Public Function ExecCmd(cmdline$)
   Dim proc As PROCESS_INFORMATION
   Dim start As STARTUPINFO
   Dim ret&

   ' Initialize the STARTUPINFO structure:
   start.cb = Len(start)

   ' Start the shelled application:
   ret& = CreateProcessA(vbNullString, cmdline$, 0&, 0&, 1&,
NORMAL_PRIORITY_CLASS, 0&, vbNullString, start, proc)

   ' Wait for the shelled application to finish:
   ret& = WaitForSingleObject(proc.hProcess, INFINITE)
   Call GetExitCodeProcess(proc.hProcess, ret&)
   Call CloseHandle(proc.hThread)
   Call CloseHandle(proc.hProcess)
   ExecCmd = ret&
End Function

Public Sub Run_Modified_TRIM()
   Dim retval As Long
   retval = ExecCmd(path$ & "trim_win_modified")
End Sub

'*****************************************************************************
'*   Subroutine Helps Convert MCNP Output to TRIM Input - Handles Geometry    *
'*****************************************************************************

Public Sub TransformCoordinates(xpos As Double, ypos As Double, zpos As Double,
xcos As Double, ycos As Double, _
   zcos As Double)
   'This subroutine translates the MCNP coordinates into appropriate
coordinates for Trim.  This is necessary for
   'several reasons: (1) the x, y, and z coordinates systems may not be the
same in the two programs (and usually
   'aren't); (2) the measurement systems aren't the same (e.g. angstroms vs.
cm); and (3) there are typically
   'offsets from zero that need to be subtracted out.

   'To change the conversion between MCNP and the TRIM input, modify this
subroutine.
   Dim copyx As Double
```

```
   Dim copyy As Double
   Dim copyz As Double
   Dim copyxcos As Double
   Dim copyycos As Double
   Dim copyzcos As Double

   'Remove offsets in MCNP.
   'xpos = xpos - xstart
   'ypos = ypos - ystart
   'zpos = zpos - zstart

   'Transform between x, y, and z if needed.
   copyx = xpos
   copyy = ypos
   copyz = zpos
   copyxcos = xcos
   copyycos = ycos
   copyzcos = zcos
   xpos = copyz
   ypos = 0# 'If the geometry is an unbounded plane, the y and z positions
don't matter.
   zpos = 0#
   xcos = copyzcos
   ycos = copyycos
   zcos = copyxcos

   'Transform from cm to angstroms. (Right now we aren't including y and z
location,
   'only x which is the depth in the scintillator in our current setup.)
   xpos = xpos * 100000000
   'ypos = ypos * 100000000
   'zpos = zpos * 100000000

   'Now add in any offsets required within the Trim system.
   'xpos = xpos
   'ypos = ypos
   'xpos = zpos
End Sub

'*************************************************************************
'*        Mathematical Particle Transport Subroutines for Spheres        *
'*************************************************************************

Public Function TestPointPositionSphere(x As Double, y As Double, z As Double,
xcenter _
 As Double, ycenter As Double, zcenter As Double, radius As Double)
   '0 - outside the sphere, 1 - on its surface, 2 - inside the sphere
   Dim radialdistance As Double

   radialdistance = ((xcenter - x) ^ 2 + (ycenter - y) ^ 2 + (zcenter - z) ^ 2)
^ 0.5

   If (radialdistance > (radius + Tolerance)) Then
      TestPointPositionSphere = 0
   ElseIf (radialdistance <= (radius + Tolerance) And radialdistance >= (radius
- Tolerance)) Then
      TestPointPositionSphere = 1
   Else
      TestPointPositionSphere = 2
   End If
End Function

Public Sub FindCrossingPointSpherical(indopant As Boolean, x1 As Double, y1 As
Double, _
 z1 As Double, x2 As Double, y2 As Double, z2 As Double, xcross1 As Double,
ycross1 _
 As Double, zcross1 As Double, cross1 As Boolean, xcross2 As Double, ycross2 As
Double, _
 zcross2 As Double, cross2 As Boolean, xcenter As Double, ycenter As Double,
zcenter _
 As Double, radius As Double, errorinparticle As Boolean)
   'Given two points, either or both of which may or may not be inside a
sphere,
   'along with the pertinent information for the sphere (location and size),
this
   'subroutine finds the location(s) at which the line between the two points
crosses
```

```
   'the surface of the sphere, if they exist.
   'Possible failure modes: The first point is practically on the sphere edge
and
   'TestPointPositionSphere() shows it as being so, *but*
RunParticleTrackSegment()
   'didn't know it was at the edge as it happened to fall just short of it.  In
   'this case, the second crossing point will be treated as if it were the
first,
   'and if there is only a first crossing point, it won't show up.
   Dim a As Double
   Dim b As Double
   Dim c As Double
   Dim xdir As Double
   Dim ydir As Double
   Dim zdir As Double
   Dim fraction As Double
   Dim p1insphere As Boolean
   Dim p2insphere As Boolean
   Dim p1position As Double
   Dim p2position As Double
   Dim copyx As Double
   Dim copyy As Double
   Dim copyz As Double

   p1position = TestPointPositionSphere(x1, y1, z1, xcenter, ycenter, zcenter,
radius)
   p2position = TestPointPositionSphere(x2, y2, z2, xcenter, ycenter, zcenter,
radius)

   xdir = x2 - x1
   ydir = y2 - y1
   zdir = z2 - z1

   If (p1position = 0 And p2position = 0) Then
      'There are either two or zero crossing points.
      a = (x2 - x1) ^ 2 + (y2 - y1) ^ 2 + (z2 - z1) ^ 2
      b = 2# * ((x2 - x1) * (x1 - xcenter) + (y2 - y1) * (y1 - ycenter) + (z2 -
z1) * (z1 - zcenter))
      c = xcenter ^ 2 + ycenter ^ 2 + zcenter ^ 2 + x1 ^ 2 + y1 ^ 2 + z1 ^ 2 -
2# * (xcenter * x1 + ycenter * y1 + zcenter * z1) - radius ^ 2

      'See if there is a solution to the quadratic equation.  If not, exit the
sub
      'with no crossing points returned.
      If ((b ^ 2 - 4# * a * c) < 0#) Then
         cross1 = False
         cross2 = False
         Exit Sub
      End If

      'Find the two solutions to the quadratic equation.
      fraction = (-b - (b ^ 2 - 4# * a * c) ^ 0.5) / (2# * a)
      cross1 = True
      xcross1 = x1 + fraction * xdir
      ycross1 = y1 + fraction * ydir
      zcross1 = z1 + fraction * zdir

      fraction = (-b + (b ^ 2 - 4# * a * c) ^ 0.5) / (2# * a)
      cross2 = True
      xcross2 = x1 + fraction * xdir
      ycross2 = y1 + fraction * ydir
      zcross2 = z1 + fraction * zdir

      'Make sure the crossing points aren't the same as the end points.
      If (Distance(x1, y1, z1, xcross1, ycross1, zcross1) < Tolerance) Then
         cross1 = False
      End If
      If (Distance(x1, y1, z1, xcross2, ycross2, zcross2) < Tolerance) Then
         cross2 = False
      End If
      If (Distance(x2, y2, z2, xcross1, ycross1, zcross1) < Tolerance) Then
         cross1 = False
      End If
      If (Distance(x2, y2, z2, xcross2, ycross2, zcross2) < Tolerance) Then
         cross2 = False
      End If
```

```
      'See if the crossing points are between the start and finish points.
They
      'should be based on earlier checking but we'll do this to be sure.
      If (xdir <> 0#) Then
         If (x1 > x2) Then
            If (xcross1 >= x1 Or xcross1 <= x2) Then
               cross1 = False
            End If
            If (xcross2 >= x1 Or xcross2 <= x2) Then
               cross2 = False
            End If
         Else
            If (xcross1 <= x1 Or xcross1 >= x2) Then
               cross1 = False
            End If
            If (xcross2 <= x1 Or xcross2 >= x2) Then
               cross2 = False
            End If
         End If
      ElseIf (ydir <> 0#) Then
         If (y1 > y2) Then
            If (ycross1 >= y1 Or ycross1 <= y2) Then
               cross1 = False
            End If
            If (ycross2 >= y1 Or ycross2 <= y2) Then
               cross2 = False
            End If
         Else
            If (ycross1 <= y1 Or ycross1 >= y2) Then
               cross1 = False
            End If
            If (ycross2 <= y1 Or ycross2 >= y2) Then
               cross2 = False
            End If
         End If
      ElseIf (zdir <> 0#) Then
         If (z1 > z2) Then
            If (zcross1 >= z1 Or zcross1 <= z2) Then
               cross1 = False
            End If
            If (zcross2 >= z1 Or zcross2 <= z2) Then
               cross2 = False
            End If
         Else
            If (zcross1 <= z1 Or zcross1 >= z2) Then
               cross1 = False
            End If
            If (zcross2 <= z1 Or zcross2 >= z2) Then
               cross2 = False
            End If
         End If
      Else
         errorinparticle = True
         Exit Sub
      End If

      If ((cross1 = False And cross2 = True) Or (cross1 = True And cross2 =
False)) Then
         errorinparticle = True
         Exit Sub
      End If
   ElseIf ((p1position = 2 And p2position = 2) Or (p1position = 1 And
p2position = 1) _
    Or (p1position = 1 And p2position = 2) Or (p2position = 2 And p1position =
1)) Then
      'There are no crossing points.
      cross1 = False
      cross2 = False
   ElseIf ((p1position = 0 And p2position = 2) Or (p1position = 2 And
p2position = 0)) Then
      'There is one crossing point.

      'There is a single crossing point somewhere between the two points.
      a = (x2 - x1) ^ 2 + (y2 - y1) ^ 2 + (z2 - z1) ^ 2
      b = 2# * ((x2 - x1) * (x1 - xcenter) + (y2 - y1) * (y1 - ycenter) + (z2 -
z1) * (z1 - zcenter))
```

```
      c = xcenter ^ 2 + ycenter ^ 2 + zcenter ^ 2 + x1 ^ 2 + y1 ^ 2 + z1 ^ 2 -
2# * (xcenter * x1 + ycenter * y1 + zcenter * z1) - radius ^ 2

      'Of the two solutions to the quadratic equation, find the correct one.
      fraction = (-b - (b ^ 2 - 4# * a * c) ^ 0.5) / (2# * a)
      If (fraction <= 0# Or fraction >= 1#) Then
         cross1 = False
      Else
         cross1 = True
         xcross1 = x1 + fraction * xdir
         ycross1 = y1 + fraction * ydir
         zcross1 = z1 + fraction * zdir
      End If

      fraction = (-b + (b ^ 2 - 4# * a * c) ^ 0.5) / (2# * a)
      If (fraction <= 0# Or fraction >= 1#) Then
         cross2 = False
      Else
         cross2 = True
         xcross2 = x1 + fraction * xdir
         ycross2 = y1 + fraction * ydir
         zcross2 = z1 + fraction * zdir
      End If

      'See if the crossing points are between the start and finish points.
They
      'should be based on earlier checking but we'll do this to be sure.
      If (xdir <> 0#) Then
         If (x1 > x2) Then
            If (xcross1 >= x1 Or xcross1 <= x2) Then
               cross1 = False
            End If
            If (xcross2 >= x1 Or xcross2 <= x2) Then
               cross2 = False
            End If
         Else
            If (xcross1 <= x1 Or xcross1 >= x2) Then
               cross1 = False
            End If
            If (xcross2 <= x1 Or xcross2 >= x2) Then
               cross2 = False
            End If
         End If
      ElseIf (ydir <> 0#) Then
         If (y1 > y2) Then
            If (ycross1 >= y1 Or ycross1 <= y2) Then
               cross1 = False
            End If
            If (ycross2 >= y1 Or ycross2 <= y2) Then
               cross2 = False
            End If
         Else
            If (ycross1 <= y1 Or ycross1 >= y2) Then
               cross1 = False
            End If
            If (ycross2 <= y1 Or ycross2 >= y2) Then
               cross2 = False
            End If
         End If
      ElseIf (zdir <> 0#) Then
         If (z1 > z2) Then
            If (zcross1 >= z1 Or zcross1 <= z2) Then
               cross1 = False
            End If
            If (zcross2 >= z1 Or zcross2 <= z2) Then
               cross2 = False
            End If
         Else
            If (zcross1 <= z1 Or zcross1 >= z2) Then
               cross1 = False
            End If
            If (zcross2 <= z1 Or zcross2 >= z2) Then
               cross2 = False
            End If
         End If
      Else
         errorinparticle = True
```

```
         Exit Sub
      End If

      If (cross1 = False And cross2 = True) Then
         xcross1 = xcross2
         ycross1 = ycross2
         zcross1 = zcross2
         cross1 = True
         cross2 = False
      ElseIf ((cross1 = False And cross2 = False) Or (cross1 = True And cross2
= True)) Then
         errorinparticle = True
         Exit Sub
      End If
   ElseIf ((p1position = 0 And p2position = 1) Or (p1position = 1 And
p2position = 0)) Then
      'There is either one crossing point or no crossing point.  If there is
one, find it.
      a = (x2 - x1) ^ 2 + (y2 - y1) ^ 2 + (z2 - z1) ^ 2
      b = 2# * ((x2 - x1) * (x1 - xcenter) + (y2 - y1) * (y1 - ycenter) + (z2 -
z1) * (z1 - zcenter))
      c = xcenter ^ 2 + ycenter ^ 2 + zcenter ^ 2 + x1 ^ 2 + y1 ^ 2 + z1 ^ 2 -
2# * (xcenter * x1 + ycenter * y1 + zcenter * z1) - radius ^ 2

      'Get the solutions to the quadratic equation and then test them.
      fraction = (-b - (b ^ 2 - 4# * a * c) ^ 0.5) / (2# * a)
      If (fraction <= 0# Or fraction >= 1#) Then
         cross1 = False
      Else
         cross1 = True
         xcross1 = x1 + fraction * xdir
         ycross1 = y1 + fraction * ydir
         zcross1 = z1 + fraction * zdir
      End If

      fraction = (-b + (b ^ 2 - 4# * a * c) ^ 0.5) / (2# * a)
      If (fraction <= 0# Or fraction >= 1#) Then
         cross2 = False
      Else
         cross2 = True
         xcross2 = x1 + fraction * xdir
         ycross2 = y1 + fraction * ydir
         zcross2 = z1 + fraction * zdir
      End If

      'See if the crossing points are between the start and finish points.
They
      'should be based on earlier checking but we'll do this to be sure.
      If (xdir <> 0#) Then
         If (x1 > x2) Then
            If (xcross1 >= x1 Or xcross1 <= x2) Then
               cross1 = False
            End If
            If (xcross2 >= x1 Or xcross2 <= x2) Then
               cross2 = False
            End If
         Else
            If (xcross1 <= x1 Or xcross1 >= x2) Then
               cross1 = False
            End If
            If (xcross2 <= x1 Or xcross2 >= x2) Then
               cross2 = False
            End If
         End If
      ElseIf (ydir <> 0#) Then
         If (y1 > y2) Then
            If (ycross1 >= y1 Or ycross1 <= y2) Then
               cross1 = False
            End If
            If (ycross2 >= y1 Or ycross2 <= y2) Then
               cross2 = False
            End If
         Else
            If (ycross1 <= y1 Or ycross1 >= y2) Then
               cross1 = False
            End If
            If (ycross2 <= y1 Or ycross2 >= y2) Then
```

```
            cross2 = False
         End If
      End If
   ElseIf (zdir <> 0#) Then
      If (z1 > z2) Then
         If (zcross1 >= z1 Or zcross1 <= z2) Then
            cross1 = False
         End If
         If (zcross2 >= z1 Or zcross2 <= z2) Then
            cross2 = False
         End If
      Else
         If (zcross1 <= z1 Or zcross1 >= z2) Then
            cross1 = False
         End If
         If (zcross2 <= z1 Or zcross2 >= z2) Then
            cross2 = False
         End If
      End If
   Else
      errorinparticle = True
      Exit Sub
   End If

   'Reject any points that are basically the same as the existing crossing
point
   'that was given.
   If (p1position = 1) Then
      If (Distance(x1, y1, z1, xcross1, ycross1, zcross1) < (Tolerance *
10#)) Then
         cross1 = False
      End If
      If (Distance(x1, y1, z1, xcross2, ycross2, zcross2) < (Tolerance *
10#)) Then
         cross2 = False
      End If
   Else
      If (Distance(x2, y2, z2, xcross1, ycross1, zcross1) < (Tolerance *
10#)) Then
         cross1 = False
      End If
      If (Distance(x2, y2, z2, xcross2, ycross2, zcross2) < (Tolerance *
10#)) Then
         cross2 = False
      End If
   End If

   'Swap the two points found if needsbe.
   If (cross1 = False And cross2 = True) Then
      xcross1 = xcross2
      ycross1 = ycross2
      zcross1 = zcross2
      cross1 = True
      cross2 = False
   End If

   'If two crossing points were found, something is seriously wrong.
   If (cross1 = True And cross2 = True) Then
      'First, attempt to recover from the error.  The error is normally
caused by
      'the line of travel being almost tangent to the surface of the dopant
particle,
      'allowing significant variations in what is counted and/or calculated
as being
      'on the surface of the dopant particle.  If we can ascertain that one
of the
      'crossing points is the same as the segment start point, we can toss
it out
      'and then use the remaining crossing point.  This problem could be
avoided by
      'using a larger tolerance value *but* this would then induce other
problems
      'elsewhere.

      'Put them in order if they aren't already.
      If (Distance(x1, y1, z1, xcross2, ycross2, zcross2) < Distance(x1, y1,
z1, xcross1, ycross1, zcross1)) Then
```

```
            copyx = xcross1
            copyy = ycross1
            copyz = zcross1
            xcross1 = xcross2
            ycross1 = ycross2
            zcross1 = zcross2
            xcross2 = xcross1
            ycross2 = ycross1
            zcross2 = zcross1
         End If

         If (Distance(x1, y1, z1, xcross1, ycross1, zcross1) < Distance(x1, y1,
z1, xcross2, ycross2, zcross2)) Then
            If (Distance(x1, y1, z1, xcross1, ycross1, zcross1) < 0.01) Then
               xcross1 = xcross2
               ycross1 = ycross2
               zcross1 = zcross2
               cross2 = False
            End If
         Else
            errorinparticle = True
            Exit Sub
         End If
      End If
   Else
      MsgBox "Can't match positions with scenario in
FindCrossingPointSpherical(). Debug.", _
      , "Fatal Error:"
      End
   End If
End Sub

'*****************************************************************************
'*         Mathematical Particle Transport Subroutines for Boxes             *
'*****************************************************************************

Public Function TestPointPositionBox(x As Double, y As Double, z As Double,
xstart _
 As Double, ystart As Double, zstart As Double, xlength As Double, ylength As
Double, _
 zlength As Double)
   '0 - outside the box, 1 - on its surface, 2 - inside the box

   'Test the two x surface possibilities: on the lowest x side or on the
highest x side.
   If _
    (x > (xstart - Tolerance) And x < (xstart + Tolerance) And y > (ystart _
    - Tolerance) And y < (ystart + ylength + Tolerance) And z > (zstart - _
    Tolerance) And z < (zstart + zlength + Tolerance)) _
   Or _
    (x > (xstart + xlength - Tolerance) And x < (xstart + xlength + Tolerance)
_
    And y > (ystart - Tolerance) And y < (ystart + ylength + Tolerance) And _
    z > (zstart - Tolerance) And z < (zstart + zlength + Tolerance)) _
   Then
      TestPointPositionBox = 1
      Exit Function
   End If

   'Test the two y surface possibilities: on the lowest y side or on the
highest y side.
   If _
    (x > (xstart - Tolerance) And x < (xstart + xlength + Tolerance) And y > _
    (ystart - Tolerance) And y < (ystart + Tolerance) And z > (zstart - _
    Tolerance) And z < (zstart + zlength + Tolerance)) _
   Or _
    (x > (xstart - Tolerance) And x < (xstart + xlength + Tolerance) And y > _
    (ystart + ylength - Tolerance) And y < (ystart + ylength + Tolerance) And _
    z > (zstart - Tolerance) And z < (zstart + zlength + Tolerance)) _
   Then
      TestPointPositionBox = 1
      Exit Function
   End If

   'Test the two z surface possibilities: on the lowest z side or on the
highest z side.
   If _
```

```
    (x > (xstart - Tolerance) And x < (xstart + xlength + Tolerance) And y > _
    (ystart - Tolerance) And y < (ystart + ylength + Tolerance) And z > _
    (zstart - Tolerance) And z < (zstart + Tolerance)) _
   Or _
    (x > (xstart - Tolerance) And x < (xstart + xlength + Tolerance) And y > _
    (ystart - Tolerance) And y < (ystart + ylength + Tolerance) And z > _
    (zstart + zlength - Tolerance) And z < (zstart + zlength + Tolerance)) _
   Then
      TestPointPositionBox = 1
      Exit Function
   End If

   'If it isn't on the surface, test to see if it is outside of the box.  If it
   'isn't outside, it must lie inside the box.
   If ((x < xstart Or x > (xstart + xlength)) Or (y < ystart Or y > (ystart +
ylength)) _
   Or (z < zstart Or z > (zstart + zlength))) Then
      TestPointPositionBox = 0
   Else
      TestPointPositionBox = 2
   End If

   Exit Function
End Function

Public Sub FindCrossingPointBox(x1 As Double, y1 As Double, z1 As Double, x2 As
Double, _
 y2 As Double, z2 As Double, xcross1 As Double, ycross1 As Double, zcross1 As
Double, _
 cross1 As Boolean, xcross2 As Double, ycross2 As Double, zcross2 As Double,
cross2 As _
 Boolean, xstart As Double, ystart As Double, zstart As Double, xlength As
Double, _
 ylength As Double, zlength As Double, errorinparticle As Boolean)
   'Assumptions are made in this subroutine.  For example, it is assumed that a
   'set of points describing a line starting at the edge of the box and then
moving
   'from the box won't be passed to this subroutine.
   Dim xdir As Double
   Dim ydir As Double
   Dim zdir As Double
   Dim fraction As Double
   Dim xfraction As Double
   Dim yfraction As Double
   Dim zfraction As Double
   Dim p1position As Long
   Dim p2position As Long
   Dim minfraction As Double
   Dim minmin As Double
   Dim xend As Double
   Dim yend As Double
   Dim zend As Double
   Dim crossing() As Boolean
   Dim crossingpoints() As Double
   Dim copyx As Double
   Dim copyy As Double
   Dim copyz As Double
   Dim distance1 As Double
   Dim distance2 As Double
   Dim numcrossings As Long
   Dim i As Long
   Dim ii As Long

   xdir = x2 - x1
   ydir = y2 - y1
   zdir = z2 - z1
   xend = xstart + xlength
   yend = ystart + ylength
   zend = zstart + zlength
   p1position = TestPointPositionBox(x1, y1, z1, xstart, ystart, zstart,
xlength, ylength, zlength)
   p2position = TestPointPositionBox(x2, y2, z2, xstart, ystart, zstart,
xlength, ylength, zlength)

   If (p1position = 2 And p2position = 2) Then
      'The line is entirely enclosed in the box.
      cross1 = False
```

```
      cross2 = False
      Exit Sub
   ElseIf ((p1position = 2 And p2position = 0) Or (p1position = 0 And
p2position = 2)) Then
      'There is one crossing point.  Find it.
      cross1 = True
      cross2 = False
      'The crossing point is given by the largest fractional distance along
each
      'directional component prior to crossing its plane that is less than or
equal
      'to one. (If it is one then the second point is on the surface.)
      If ((x1 > xend And x2 < xend) Or (x1 < xend And x2 > xend)) Then
         xfraction = (xend - x1) / (x2 - x1)
      ElseIf ((x1 > xstart And x2 < xstart) Or (x1 < xstart And x2 > xstart))
Then
         xfraction = (xstart - x1) / (x2 - x1)
      Else
         xfraction = 0#
      End If

      If ((y1 > yend And y2 < yend) Or (y1 < yend And y2 > yend)) Then
         yfraction = (yend - y1) / (y2 - y1)
      ElseIf ((y1 > ystart And y2 < ystart) Or (y1 < ystart And y2 > ystart))
Then
         yfraction = (ystart - y1) / (y2 - y1)
      Else
         yfraction = 0#
      End If

      If ((z1 > zend And z2 < zend) Or (z1 < zend And z2 > zend)) Then
         zfraction = (zend - z1) / (z2 - z1)
      ElseIf ((z1 > zstart And z2 < zstart) Or (z1 < zstart And z2 > zstart))
Then
         zfraction = (zstart - z1) / (z2 - z1)
      Else
         zfraction = 0#
      End If

      'Find the minimum fraction provided it is less than one and greater than
zero.
      minfraction = 1#

      If (xfraction < minfraction And xfraction > 0#) Then
         minfraction = xfraction
      End If
      If (yfraction < minfraction And yfraction > 0#) Then
         minfraction = yfraction
      End If
      If (zfraction < minfraction And zfraction > 0#) Then
         minfraction = zfraction
      End If

      'Find the direction it is associated with and set the crossing point
using the
      'exact boundary location so as to avoid rounding errors, etc.
      If (xfraction = minfraction) Then
         If ((x1 > xend And x2 < xend) Or (x1 < xend And x2 > xend)) Then
            xcross1 = xend
         Else
            xcross1 = xstart
         End If
         ycross1 = y1 + minfraction * ydir
         zcross1 = z1 + minfraction * zdir
      ElseIf (yfraction = minfraction) Then
         If ((y1 > yend And y2 < yend) Or (y1 < yend And y2 > yend)) Then
            ycross1 = yend
         Else
            ycross1 = ystart
         End If
         xcross1 = x1 + minfraction * xdir
         zcross1 = z1 + minfraction * zdir
      ElseIf (zfraction = minfraction) Then
         If ((z1 > zend And z2 < zend) Or (z1 < zend And z2 > zend)) Then
            zcross1 = zend
         Else
            zcross1 = zstart
```

```
         End If
         xcross1 = x1 + minfraction * xdir
         ycross1 = y1 + minfraction * ydir
      Else
         errorinparticle = True
         Exit Sub
      End If

      If (Distance(x1, y1, z1, xcross1, ycross1, zcross1) < (Tolerance * 2#))
Then
         'The first point returned is basically the same point as the starting
point.
         'Go to the next lowest distance fraction to get the crossing point.
         minmin = minfraction
         minfraction = 1#

         If (xfraction < minfraction And xfraction > minmin) Then
            minfraction = xfraction
         End If
         If (yfraction < minfraction And yfraction > minmin) Then
            minfraction = yfraction
         End If
         If (zfraction < minfraction And zfraction > minmin) Then
            minfraction = zfraction
         End If

         If (xfraction = minfraction) Then
            If ((x1 > xend And x2 < xend) Or (x1 < xend And x2 > xend)) Then
               xcross1 = xend
            Else
               xcross1 = xstart
            End If
            ycross1 = y1 + minfraction * ydir
            zcross1 = z1 + minfraction * zdir
         ElseIf (yfraction = minfraction) Then
            If ((y1 > yend And y2 < yend) Or (y1 < yend And y2 > yend)) Then
               ycross1 = yend
            Else
               ycross1 = ystart
            End If
            xcross1 = x1 + minfraction * xdir
            zcross1 = z1 + minfraction * zdir
         ElseIf (zfraction = minfraction) Then
            If ((z1 > zend And z2 < zend) Or (z1 < zend And z2 > zend)) Then
               zcross1 = zend
            Else
               zcross1 = zstart
            End If
            xcross1 = x1 + minfraction * xdir
            ycross1 = y1 + minfraction * ydir
         Else
            errorinparticle = True
            Exit Sub
         End If
      End If

      If (cross1 = False) Then
         MsgBox "Unanticipated conditions in 2,0 or 0,2 section of
FindCrossingPointBox(). Debug.", , "Fatal Error:"
         End
      End If
      If (cross2 = True) Then
         MsgBox "Unanticipated conditions in 2,0 or 0,2 section of
FindCrossingPointBox(). Debug.", , "Fatal Error:"
         End
      End If

      Exit Sub
   ElseIf ((p1position = 2 And p2position = 1) Or (p1position = 1 And
p2position = 2)) Then
      'The line lies entirely within the box.
      cross1 = False
      cross2 = False
   ElseIf (p1position = 1 And p2position = 0) Then
      'If the particle is headed into the box, there is a crossing point.  If
it is
```

```
      'headed out, there isn't one.  Search for up to two crossing points and
then
      'test any that get found to see if they are actually on the edge but not
at
      'the same location as the start point.

      'Prepare to find if the line crosses each plane and where it crosses, if
so.
      ReDim crossing(6)
      ReDim crossingpoints(6, 3)

      If (xdir <> 0#) Then
         'Check the first x plane
         If ((x1 < xstart And x2 > xstart) Or (x1 > xstart And x2 < xstart))
Then
            fraction = (xstart - x1) / xdir
            crossingpoints(1, 1) = xstart
            crossingpoints(1, 2) = y1 + fraction * ydir
            crossingpoints(1, 3) = z1 + fraction * zdir
            'Now see if the points meet the y and z requirements to be on the
box surface.
            If ((crossingpoints(1, 2) > ystart And crossingpoints(1, 2) < yend)
And _
            (crossingpoints(1, 3) > zstart And crossingpoints(1, 3) < zend))
Then
               crossing(1) = True
            Else
               crossing(1) = False
            End If
         Else
            crossing(1) = False
         End If

         'Check the second x plane
         If ((x1 < xend And x2 > xend) Or (x1 > xend And x2 < xend)) Then
            fraction = (xend - x1) / xdir
            crossingpoints(2, 1) = xend
            crossingpoints(2, 2) = y1 + fraction * ydir
            crossingpoints(2, 3) = z1 + fraction * zdir
            'Now see if the points meet the y and z requirements to be on the
box surface.
            If ((crossingpoints(2, 2) > ystart And crossingpoints(2, 2) < yend)
And _
            (crossingpoints(2, 3) > zstart And crossingpoints(2, 3) < zend))
Then
               crossing(2) = True
            Else
               crossing(2) = False
            End If
         Else
            crossing(2) = False
         End If
      Else
         crossing(1) = False
         crossing(2) = False
      End If

      If (ydir <> 0#) Then
         'Check the first y plane
         If ((y1 < ystart And y2 > ystart) Or (y1 > ystart And y2 < ystart))
Then
            fraction = (ystart - y1) / ydir
            crossingpoints(3, 1) = x1 + fraction * xdir
            crossingpoints(3, 2) = ystart
            crossingpoints(3, 3) = z1 + fraction * zdir
            'Now see if the points meet the x and z requirements to be on the
box surface.
            If ((crossingpoints(3, 1) > xstart And crossingpoints(3, 1) < xend)
And _
            (crossingpoints(3, 3) > zstart And crossingpoints(3, 3) < zend))
Then
               crossing(3) = True
            Else
               crossing(3) = False
            End If
         Else
            crossing(3) = False
```

```
         End If

         'Check the second y plane
         If ((y1 < yend And y2 > yend) Or (y1 > yend And y2 < yend)) Then
            fraction = (yend - y1) / ydir
            crossingpoints(4, 1) = x1 + fraction * xdir
            crossingpoints(4, 2) = yend
            crossingpoints(4, 3) = z1 + fraction * zdir
            'Now see if the points meet the y and z requirements to be on the
box surface.
            If ((crossingpoints(4, 1) > xstart And crossingpoints(4, 1) < xend)
And _
            (crossingpoints(4, 3) > zstart And crossingpoints(4, 3) < zend))
Then
               crossing(4) = True
            Else
               crossing(4) = False
            End If
         Else
            crossing(4) = False
         End If
      Else
         crossing(3) = False
         crossing(4) = False
      End If

      If (zdir <> 0#) Then
         'Check the first z plane
         If ((z1 < zstart And z2 > zstart) Or (z1 > zstart And z2 < zstart))
Then
            fraction = (zstart - z1) / zdir
            crossingpoints(5, 1) = x1 + fraction * xdir
            crossingpoints(5, 2) = y1 + fraction * ydir
            crossingpoints(5, 3) = zstart
            'Now see if the points meet the x and y requirements to be on the
box surface.
            If ((crossingpoints(5, 1) > xstart And crossingpoints(5, 1) < xend)
And _
            (crossingpoints(5, 2) > ystart And crossingpoints(5, 2) < yend))
Then
               crossing(5) = True
            Else
               crossing(5) = False
            End If
         Else
            crossing(5) = False
         End If

         'Check the second z plane
         If ((z1 < zend And z2 > zend) Or (z1 > zend And z2 < zend)) Then
            fraction = (zend - z1) / zdir
            crossingpoints(6, 1) = x1 + fraction * xdir
            crossingpoints(6, 2) = y1 + fraction * ydir
            crossingpoints(6, 3) = zend
            'Now see if the points meet the x and y requirements to be on the
box surface.
            If ((crossingpoints(6, 1) > xstart And crossingpoints(6, 1) < xend)
And _
            (crossingpoints(6, 2) > ystart And crossingpoints(6, 2) < yend))
Then
               crossing(6) = True
            Else
               crossing(6) = False
            End If
         Else
            crossing(6) = False
         End If
      Else
         crossing(5) = False
         crossing(6) = False
      End If

      'Test any crossing points to see if they are different from the start
point.
      For i = 1 To 6 Step 1
         If (crossing(i) = True) Then
```

```
            If (Distance(x1, y1, z1, crossingpoints(i, 1), crossingpoints(i,
2), _
            crossingpoints(i, 3)) < (Tolerance * 2#)) Then
               crossing(i) = False
            End If
         End If
      Next i

      'Do some error checking and get ready to pass back the crossing point
location.
      numcrossings = 0
      For i = 1 To 6 Step 1
         If (crossing(i) = True) Then
            numcrossings = numcrossings + 1
         End If
      Next i

      If (numcrossings = 0) Then
         cross1 = False
         cross2 = False
      ElseIf (numcrossings = 1) Then
         For i = 1 To 6 Step 1
            If (crossing(i) = True) Then
               xcross1 = crossingpoints(i, 1)
               ycross1 = crossingpoints(i, 2)
               zcross1 = crossingpoints(i, 3)
            End If
         Next i
         cross1 = True
      Else
         MsgBox "Found " & numcrossings & " crossings in
FindCrossingPointBox(). Debug.", _
          , "Fatal Error:"
         End
      End If
   ElseIf (p1position = 0 And p2position = 1) Then
      MsgBox "Unanticipated conditions in FindCrossingPointBox(). Code not
written.", _
       , "Fatal Error:"
      End
   Else
      'Neither point is in the box but the line between the points could still
go
      'through the box.  If it does, there will be two crossing points.
      If ((x1 < xstart And x2 < xstart) Or (x1 > xend And x2 > xend) Or (y1 <
ystart And _
       y2 < ystart) Or (y1 > yend And y2 > yend) Or (z1 < zstart And z2 <
zstart) Or (z1 > _
       zend And z2 > zend)) Then
         'The line can't cross the box.
         cross1 = False
         cross2 = False
         Exit Sub
      Else
         'Prepare to find if the line crosses each plane and where it crosses,
if so.
         ReDim crossing(6)
         ReDim crossingpoints(6, 3)

         If (xdir <> 0#) Then
            'Check the first x plane
            If ((x1 < xstart And x2 > xstart) Or (x1 > xstart And x2 < xstart))
Then
               fraction = (xstart - x1) / xdir
               crossingpoints(1, 1) = xstart
               crossingpoints(1, 2) = y1 + fraction * ydir
               crossingpoints(1, 3) = z1 + fraction * zdir
               'Now see if the points meet the y and z requirements to be on
the box surface.
               If ((crossingpoints(1, 2) > ystart And crossingpoints(1, 2) <
yend) And _
               (crossingpoints(1, 3) > zstart And crossingpoints(1, 3) < zend))
Then
                  crossing(1) = True
               Else
                  crossing(1) = False
               End If
```

```
            Else
               crossing(1) = False
            End If

            'Check the second x plane
            If ((x1 < xend And x2 > xend) Or (x1 > xend And x2 < xend)) Then
               fraction = (xend - x1) / xdir
               crossingpoints(2, 1) = xend
               crossingpoints(2, 2) = y1 + fraction * ydir
               crossingpoints(2, 3) = z1 + fraction * zdir
               'Now see if the points meet the y and z requirements to be on
the box surface.
               If ((crossingpoints(2, 2) > ystart And crossingpoints(2, 2) <
yend) And _
               (crossingpoints(2, 3) > zstart And crossingpoints(2, 3) < zend))
Then
                  crossing(2) = True
               Else
                  crossing(2) = False
               End If
            Else
               crossing(2) = False
            End If
         Else
            crossing(1) = False
            crossing(2) = False
         End If

         If (ydir <> 0#) Then
            'Check the first y plane
            If ((y1 < ystart And y2 > ystart) Or (y1 > ystart And y2 < ystart))
Then
               fraction = (ystart - y1) / ydir
               crossingpoints(3, 1) = x1 + fraction * xdir
               crossingpoints(3, 2) = ystart
               crossingpoints(3, 3) = z1 + fraction * zdir
               'Now see if the points meet the x and z requirements to be on
the box surface.
               If ((crossingpoints(3, 1) > xstart And crossingpoints(3, 1) <
xend) And _
               (crossingpoints(3, 3) > zstart And crossingpoints(3, 3) < zend))
Then
                  crossing(3) = True
               Else
                  crossing(3) = False
               End If
            Else
               crossing(3) = False
            End If

            'Check the second y plane
            If ((y1 < yend And y2 > yend) Or (y1 > yend And y2 < yend)) Then
               fraction = (yend - y1) / ydir
               crossingpoints(4, 1) = x1 + fraction * xdir
               crossingpoints(4, 2) = yend
               crossingpoints(4, 3) = z1 + fraction * zdir
               'Now see if the points meet the x and z requirements to be on
the box surface.
               If ((crossingpoints(4, 1) > xstart And crossingpoints(4, 1) <
xend) And _
               (crossingpoints(4, 3) > zstart And crossingpoints(4, 3) < zend))
Then
                  crossing(4) = True
               Else
                  crossing(4) = False
               End If
            Else
               crossing(4) = False
            End If
         Else
            crossing(3) = False
            crossing(4) = False
         End If

         If (zdir <> 0#) Then
            'Check the first z plane
```

```
            If ((z1 < zstart And z2 > zstart) Or (z1 > zstart And z2 < zstart))
Then
               fraction = (zstart - z1) / zdir
               crossingpoints(5, 1) = x1 + fraction * xdir
               crossingpoints(5, 2) = y1 + fraction * ydir
               crossingpoints(5, 3) = zstart
               'Now see if the points meet the x and y requirements to be on
the box surface.
               If ((crossingpoints(5, 1) > xstart And crossingpoints(5, 2) <
xend) And _
               (crossingpoints(5, 2) > xstart And crossingpoints(5, 3) < xend))
Then
                  crossing(5) = True
               Else
                  crossing(5) = False
               End If
            Else
               crossing(5) = False
            End If

            'Check the second z plane
            If ((z1 < zend And z2 > zend) Or (z1 > zend And z2 < zend)) Then
               fraction = (zend - z1) / zdir
               crossingpoints(6, 1) = x1 + fraction * xdir
               crossingpoints(6, 2) = y1 + fraction * ydir
               crossingpoints(6, 3) = zend
               'Now see if the points meet the x and y requirements to be on
the box surface.
               If ((crossingpoints(6, 1) > xstart And crossingpoints(6, 1) <
xend) And _
               (crossingpoints(6, 2) > ystart And crossingpoints(6, 2) < yend))
Then
                  crossing(6) = True
               Else
                  crossing(6) = False
               End If
            Else
               crossing(6) = False
            End If
         Else
            crossing(5) = False
            crossing(6) = False
         End If

         'First, do error checking and make sure exactly two crossing points
were found.
         numcrossings = 0
         For i = 1 To 6 Step 1
            If (crossing(i) = True) Then
               numcrossings = numcrossings + 1
            End If
         Next i

         If (numcrossings = 0) Then
            cross1 = False
            cross2 = False
            Exit Sub
         ElseIf (numcrossings <> 2) Then
            MsgBox "Wrong number of surface crossing points found for a box
geometry.  Debug code." _
            , vbExclamation, "Fatal Error:"
            End
         End If

         'Now put the two crossing points in the correct order.
         cross1 = True
         cross2 = True
         ii = 0
         For i = 1 To 6 Step 1
            If (crossing(i) = True) Then
               ii = ii + 1
               If (ii = 1) Then
                  xcross1 = crossingpoints(i, 1)
                  ycross1 = crossingpoints(i, 2)
                  zcross1 = crossingpoints(i, 3)
               ElseIf (ii = 2) Then
                  xcross2 = crossingpoints(i, 1)
```

```
                ycross2 = crossingpoints(i, 2)
                zcross2 = crossingpoints(i, 3)
             Else
                MsgBox "More than two crossing points in
FindCrossingPointBox(). Debug code.", _
                vbExclamation, "Fatal Error:"
             End If
          End If
       Next i

       distance1 = Distance(x1, y1, z1, xcross1, ycross1, zcross1)
       distance2 = Distance(x1, y1, z1, xcross2, ycross2, zcross2)

       If (distance1 > distance2) Then
          copyx = xcross1
          copyy = ycross1
          copyz = zcross1
          xcross1 = xcross2
          ycross1 = ycross2
          zcross1 = zcross2
          xcross2 = copyx
          ycross2 = copyy
          zcross2 = copyz
       End If

       Exit Sub
    End If
 End If
End Sub

'*************************************************************************
'* Subroutines for Planar Geometry Mathematics *
'*************************************************************************

Public Function TestPointPlanar(x As Double)
   'Determines whether or not the x location falls within the scintillator
plane.
   If (x >= 0# And x <= (ScintillatorThickness * 100000000#)) Then
      TestPointPlanar = True
   Else
      TestPointPlanar = False
   End If
End Function

'*************************************************************************
'*         Subroutines for Picking Random (But Bounded) Locations        *
'*************************************************************************

Public Sub PickSphereLocation(radius As Double, xstart As Double, ystart As
Double, _
 zstart As Double, xlength As Double, ylength As Double, zlength As Double,
xcenter _
 As Double, ycenter As Double, zcenter As Double)
   'Pick a random location for the center of the dopant particle sphere within
the available space.
   'The box in which the dopant particle sphere is placed can be of different
dimensions in different
   'directions.
   Dim xspread As Double
   Dim yspread As Double
   Dim zspread As Double

   xspread = xlength - 2# * radius
   yspread = ylength - 2# * radius
   zspread = zlength - 2# * radius
   xcenter = xstart + radius + Rnd * xspread
   ycenter = ystart + radius + Rnd * yspread
   zcenter = zstart + radius + Rnd * zspread
End Sub

Public Sub PickReactionLocation(xstart As Double, ystart As Double, zstart As
Double, _
 xlength As Double, ylength As Double, zlength As Double, radius As Double,
xcenter As _
 Double, ycenter As Double, zcenter As Double, x As Double, y As Double, z As
Double)
```

```
   'A reaction location is picked that is outside the sphere and inside the
box.
   '(x,y,z) is the randomly selected reaction location.
   Dim flag As Boolean

   flag = False
   Do While (flag = False)
      x = xstart + Rnd * xlength
      y = ystart + Rnd * ylength
      z = zstart + Rnd * zlength

      If ((TestPointPositionSphere(x, y, z, xcenter, ycenter, zcenter, radius)
= 0) And _
      (TestPointPositionBox(x, y, z, xstart, ystart, zstart, xlength, ylength,
zlength) = 2)) Then
         flag = True
      End If
   Loop
End Sub

'*************************************************************************
'*                  Miscellaneous Mathematical Subroutines               *
'*************************************************************************

Public Function Distance(x1 As Double, y1 As Double, z1 As Double, x2 As
Double, y2 As Double, z2 As Double)
   'Find the distance between two points (Cartesian coordinates)
   Distance = ((x2 - x1) ^ 2 + (y2 - y1) ^ 2 + (z2 - z1) ^ 2) ^ 0.5
End Function

Public Function Tolerance()
   'Fudge in distance calculations for finding surface crosssing points.
   'An alternative way to do this is to use a tolerance value that varies with
   'some associated distance value.  If that is done, pass the distance value
   'to this subroutine and then select a tolerance value based on it.
   Tolerance = 0.00001
End Function

'*************************************************************************
'*              Subroutines for Handling Neutron Interactions            *
'*************************************************************************

Public Function MacroscopicXSection(ENeutron As Double)
   'Spits back the macroscopic cross section for a neutron of a given energy.
   'The values used to calculate the cross section are material specific, so be
   'sure to enter the correct material information (see the constants at the
start
   'of the program) in order to get accurate output.
   Dim TargetAtomsPerCC As Double
   Dim MicroscopicXSection As Double
   Dim currenergy As Double
   Dim prevenergy As Double
   Dim currxsection As Double
   Dim prevxsection As Double
   Dim i As Long

   TargetAtomsPerCC = Li6VolumeDopingFraction * 4.633E+22

   'Cross section arrays give the energy in MeV and the microscopic cross
section in barns.

   'Target atom type is Li-6
   i = 1
   currenergy = Li6XSection(1, i)
   currxsection = Li6XSection(2, i)

   If (ENeutron < currenergy) Then
      'Note: we could cause it to use the cross section of the minimum energy
supported even
      'if its energy is below this.  This would be an extremely simple change
to make.
      MsgBox "Neutron energy below supported range.", , "Error:"
      End
   End If

   Do While (currenergy < ENeutron And i < 498)
      i = i + 1
```

```
      prevenergy = currenergy
      prevxsection = currxsection
      currenergy = Li6XSection(1, i)
      currxsection = Li6XSection(2, i)
   Loop

   If (ENeutron > currenergy) Then
      'Note: we could cause it to use the cross section of the maximum energy
supported even
      'if its energy is above this.  This would be an extremely simple change
to make.
      MsgBox "Neutron energy above supported range.", , "Error:"
      End
   End If

   MicroscopicXSection = prevxsection + (currxsection - prevxsection) *
((ENeutron _
   - prevenergy) / (currenergy - prevenergy))
   MacroscopicXSection = MicroscopicXSection * TargetAtomsPerCC

   Exit Function
End Function

Public Sub ReadInXSections()
   'Reads in microscopic cross sections from the microscopic cross section
files.
   Dim energy As Double
   Dim XSection As Double
   Dim i As Long

   Open "C:\Andrew\MicromegasSim\Li-6xsections.txt" For Input As #1
   For i = 1 To 498 Step 1
      Input #1, energy, XSection
      Li6XSection(1, i) = energy
      Li6XSection(2, i) = XSection * 1E-24
   Next i
   Close #1

   Exit Sub
End Sub
```

VITA

Andrew Curtis Stephan was born in Richmond, Virginia on May 23, 1975. After living in Richmond, Dayton, Ohio, and on a farm near Oregonia, Ohio, his family moved overseas in 1983. He spent the greater part of his childhood years living in Kuwait City, Kuwait and Wellington, New Zealand before returning to the United States in December 1991. He attended high school in Wilmington, North Carolina and graduated from Oak Ridge High School in Oak Ridge, Tennessee in June 1993. He received two Bachelor of Science degrees in May 1997 from Presbyterian College in Clinton, South Carolina with majors in Physics, Mathematics, and Economics and a minor in History. Following this, he entered the Nuclear Engineering program at The University of Tennessee in Knoxville, receiving a Master of Science degree in December 1999, a Graduate Certificate in Nuclear Criticality Safety in the Spring of 2003, and a Doctor of Philosophy degree in December 2003. While pursuing his graduate degrees, he has been involved in a variety of professional activities including founding a company to develop radiation detection instrumentation, holding a post-master's research position at the Spallation Neutron Source (SNS), Oak Ridge National Laboratory, consulting with a local high-tech startup, and successfully pursuing funding for nuclear-related R&D projects. He enjoys a variety of activities from dancing and hiking to church involvement and reading.

He is currently working for a company he co-founded located in the Knoxville, Tennessee area.