Stand-alone High Resolution Neutron Spectrometer

Alec Raymond

A senior thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Bachelor of Science

John Ellsworth, Advisor

Department of Physics and Astronomy

Brigham Young University

April 2016

ABSTRACT

Stand-alone High Resolution Neutron Spectrometer

Alec Raymond
Department of Physics and Astronomy, BYU
Bachelor of Science

We are interested in improving neutron spectrometric capabilities in the energy range of 0.1 to 3 MeV. Current neutron spectrometry technologies do not have good energy resolution in this energy range without the use of either a complicated or a multiple-detector setup. We have developed a stand-alone detector with improved features in this energy range. The detector uses $Li_6Gd(BO_3)_3:Ce$ crystal in a thin slab of polyvinyl toluene scintillator for capture-gated neutron detection and a second thin slab of solid plastic scintillator for proton recoil detection. Comparisons of proton recoil pulse area to measured time-of-flight data and theoretical neutron energy spectra indicate a useful correlation with neutron energy. The spectrometer we have developed functions as an initial prototype.

Keywords: neutron detection, neutron spectrometry, proton recoil, capture-gated, LGB, stand-alone

ACKNOWLEDGMENTS

I would like to thank John E. Ellsworth for his patience and willingness to teach me his ways. I am also grateful to Lawrence B. Rees for his feedback and advice in group meetings and and J. Bart Czirr for answering countless questions, both asked and unasked. I would also like to thank all of my fellow undergraduate researchers for teaching me some of the things they have learned along the way, especially KaeCee Terry. I would like to thank the Department of Physics and Astronomy at Brigham Young University for funding my research.

Contents

Chapter 1

Introduction

1.1 Purpose of Neutron Detection and Spectrometry

The ability to detect neutrons and measure their energy is important for both research and radiation safety applications. Nuclear reactions produce characteristic particles at distinct energies. Studying the particles and their energies provides insight into the nuclear processes that occur during the reaction. Good detection and spectrometric technologies improve our ability to identify and characterize unknown or low-rate reactions. This is useful for studying anomalous results in nuclear research as well as for detecting and inventorying nuclear material. Reliable and accurate detectors are also critical for monitoring radiation in work environments such as reactor or accelerator facilities.

1.2 Laboratory Nuclear Astrophysics Research

The Laboratory Nuclear Astrophysics Research (LNAR) group at Brigham Young University is interested in studying stellar-like nuclear reactions, typically using d-d fusion. Typical nuclear astrophysics research is done using reactants with energies much greater than those actually found in stars. This is done to push the resulting signals above background levels. As the energy of the reactants is increased, the probability of fusion also increases as it becomes easier and easier to tunnel through the Coulomb barrier caused by electrostatic repulsion between positively charged nuclei. Using higher energy reactants creates a fundamental difference between nuclear reactions in stars and nuclear reactions in the laboratory. As detector capabilities have improved in recent decades, studies have been done using reactant energies closer to those actually found in stars. Many of these studies have yielded unexpected results. They have found higher-than-expected rates of fusion in condensed matter environments such as deuterium-loaded metal foils. This effect has in the literature been attributed to electron screening potentials which effectively lower the energy barrier to fusion. The effect varies from material to material and is especially strong in some metals. For a summary of screening potentials measured in several studies, see Table A.1 in Appendix A. Improving neutron spectrometry capabilities enables us to more reliably measure the rate of these reactions in the laboratory as we study potential explanations for the unexpected phenomenon.

1.3 Thesis Overview

My purpose in writing this is both to describe the development of a specific detector and its capabilities and also to provide a brief introduction to many of the concepts and methods used in the LNAR group's neutron detection research. I will provide an overview of common neutron detection and spectrometric techniques and a few existing technologies utilizing those techniques. I will discuss the development of our neutron spectrometer and several important features of the detector. I will also discuss how we acquire and analyze data from the spectrometer. Finally, I will compare the measured energy spectrum from our spectrometer to time-of-flight data and to the test energy spectrum and discuss the detector's usefulness and applications.

Chapter 2

Methods

2.1 Neutron Detection

Detectors do not yet exist which are capable of detecting neutrons directly. Instead, it is necessary to observe the effects of neutron interactions, typically through derived charged particles which are directly detectable by current technology (Knoll 2010). Several techniques have been developed over the decades to detect the charged particles produced by neutron interactions. Detection methods differ depending on the energy of the neutrons to be detected. These techniques can be divided into categories by the energy range (also called temperature) of the neutrons they are capable of detecting. There are several commonly used neutron temperature ranges but in this thesis I will address only two of them: fast neutrons and slow neutrons.

2.1.1 Fast Neutrons

Fast neutrons have sufficient energy (>1 MeV) that they do not have large probabilities of interaction, expressible in terms of cross section, with other particles via nuclear reactions. (Cross section is measured in barns, as in the broad side of a barn. Though it must be a rather small barn as 1 barn is equal to 10^{-24} cm^2.) For example, ^{10}B has a very low cross section for neutrons with energy above about 1 MeV (see Fig. 2.1).

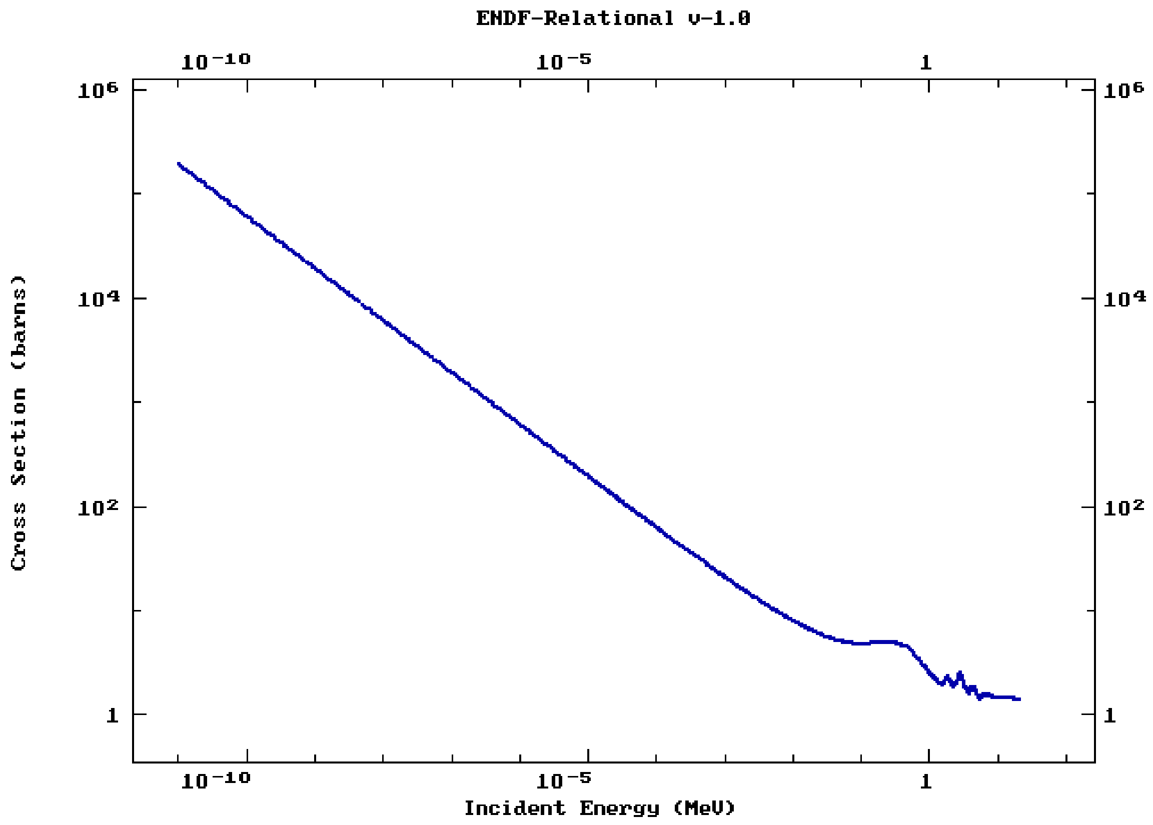


Figure 2.1 The total neutron cross section of ^{10}B versus energy (ENDF 2016).

Instead, fast neutrons interact with other particles via elastic collision. When a neutron collides with a heavy (compared to the mass of a neutron) particle or nucleus the neutron will bounce off, imparting little of its energy to the larger particle (think ping-pong ball hitting a bowling ball). However, when a neutron hits a similarly sized particle, such as a proton, it transfers a large portion of its energy to the proton, up to the total energy of the impinging neutron (think ping-pong ball hitting a ping-pong ball). The newly energized proton then moves through scintillating material creating a light output detectable using photomultiplier tubes (PMTs). The PMT then converts the light into an electrical pulse. Since the energy of the neutron was largely transferred to the proton it is possible to infer energy information about the original neutron from the detected proton recoil pulse. As the neutron transfers energy to protons, it is moderated to a sufficiently slow speed that the probability of interaction with other particles via a nuclear reaction becomes significant.

2.1.2 Slow Neutrons

Slow neutrons do not have sufficient energy to transfer to protons for the recoiled protons to produce detectable light in a scintillating material. Instead, they are detected by the production of charged particles via nuclear reactions such as neutron-capture:

$$^{6}\mathrm{Li}(\mathrm{n},\alpha)^{3}\mathrm{H},$$

$$^{157}\mathrm{Gd}(\mathrm{n},\gamma)^{158}\mathrm{Gd},$$

$$^{10}\mathrm{B}(\mathrm{n},\alpha)^{7}\mathrm{Li},$$

$$^{3}\mathrm{He}(\mathrm{n},\mathrm{p})^{3}\mathrm{H}.$$

$$^{113}\mathrm{Cd}(\mathrm{n},\gamma)^{114}\mathrm{Cd}.$$

(The standard notation $a(b,c)d$ indicates a particle b hitting a target a, producing a particle c and a product d.) Many successful neutron detectors exist utilizing these reactions in various configurations including ^{3}He tubes, ^{6}Li glass, $Li_6Gd(BO_3)_3$ crystal (LGB), and ^{113}Cd sheets.

The light output from LGB (the scintillator we use in our detector) neutron capture reactions typically occurs over a longer period of time (hundreds of nanoseconds) than recoil reactions (tens of nanoseconds) resulting in wider pulses. The distinctly wider pulses are useful for capture-gated neutron detectors such as our spectrometer. However, unlike proton recoil pulses, capture pulses contain energy information from the reaction products, rather than the original neutron. Neutron energy information was lost in the moderation process necessary for a capture reaction to occur. A combination of recoil and capture pulses is useful for retaining energy information while simultaneously identifying true neutron detection events.

2.2 Neutron Spectrometry

In addition to being able to detect neutrons, it is important to be able to measure their energy. This can be difficult as the energy information of the neutron is often lost in the detection process. However, several technologies exist that are capable of neutron energy measurements. Each method has its own strengths and weaknesses regarding resolution, efficiency, portability, and useful energy range.

2.2.1 Time-of-flight

A common technique used to measure neutron energy is time-of-flight. The reactions that produce neutrons also produce charged particles such as gammas, alphas, or fission fragments. The time-of-flight technique uses these other products to create a start signal when the neutron-producing reaction has occurred. The most accurate time-of-flight spectrometers use charged particle spark chambers to detect the charged fission fragments to produce a start signal.

The time-of-flight technique requires the use of more than one detector: one near the neutron source to detect the charged particle products and a second at a known distance (typically on the order of meters) to detect the arrival of the neutron, producing a stop signal (see Fig. 2.2).

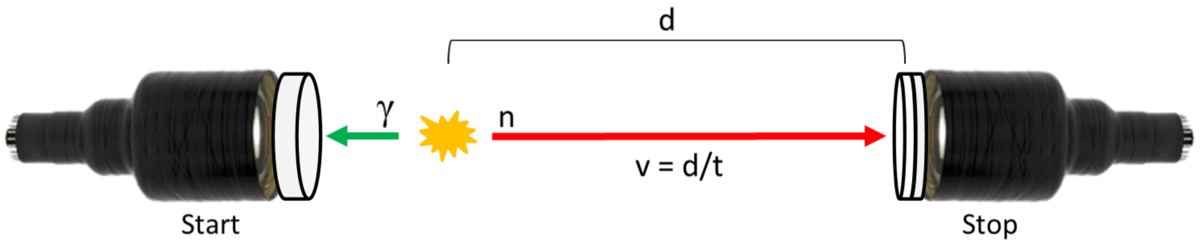


Figure 2.2 A diagram showing the important features of a time-of-flight spectrometer.

By measuring the difference in time between the start signal and the stop signal you can calculate the kinetic energy of the neutron using

$$E = m_n c^2 \left(\frac{1}{\sqrt{1 - \frac{(\frac{d}{t})^2}{c^2}}} - 1 \right), \tag{2.1}$$

where m_n is the mass of a neutron, c is the speed of light, d is the distance from the source to the stop detector, and t is the difference in time between the start pulse and the stop pulse. Neutrons do not move at the speed of light, but they do travel at sufficiently high speed that it can be necessary to use relativistic mechanics to calculate their kinetic energy (neutrons with energy greater than around 939 MeV are considered relativistic, so the effect for our energy region of interest, 0.1 to 3 MeV, is negligible) (Knoll 2010).

Fission fragments are most commonly used to produce a start signal because they are always produced alongside neutrons, do not attenuate significantly through air, and are easy to detect promptly after they are produced. Additionally, the detection of a fission fragment is a better indicator of a neutron-producing reaction. For example, ^{252}Cf decays 97% of the time via alpha decay, which does not produce any neutrons. Only 3% of the decays are spontaneous fission, producing neutrons and fission fragments. Gating the start detector on fission fragment detection events immediately removes 97% of the noise that would otherwise be present if gammas were used for the start signal.

Care must be taken that the start detector is not so close to the source that the recovery time of the detector (set by the type of scintillator and photomultiplier tube used) is longer than the time between pulses, effectively blinding the detector. Increasing the distance from the stop detector to the source will increase the resolution of the spectrometer as the longer path length will spread out the distribution of discernible flight times. However, for sources that emit neutrons in every direction equally (isotropic), increasing the path length also decreases solid angle and thus the neutron flux through the detector, requiring longer run times.

Time-of-flight systems are especially useful in permanent installations like reactor or accelerator facilities. Such facilities typically have the necessary space and resources to develop and maintain a time-of-flight spectrometer as well as the need for a high resolution spectrometric setup. Time-of-flight spectrometers provide very high resolution measurements. Additionally, they can use a variety of detection methods since the two detectors need only be capable of detecting the start particle and the neutrons rather than being able to independently measure their energy. However, time-of-flight is not an acceptable solution where portability is a necessary requirement or where it is not feasible to have a detector at the source.

2.2.2 Bonner Sphere

The Bonner sphere spectrometer is more portable than a time-of-flight setup. This spectrometer uses several spheres of moderating material of various diameters (see Fig. 2.3).

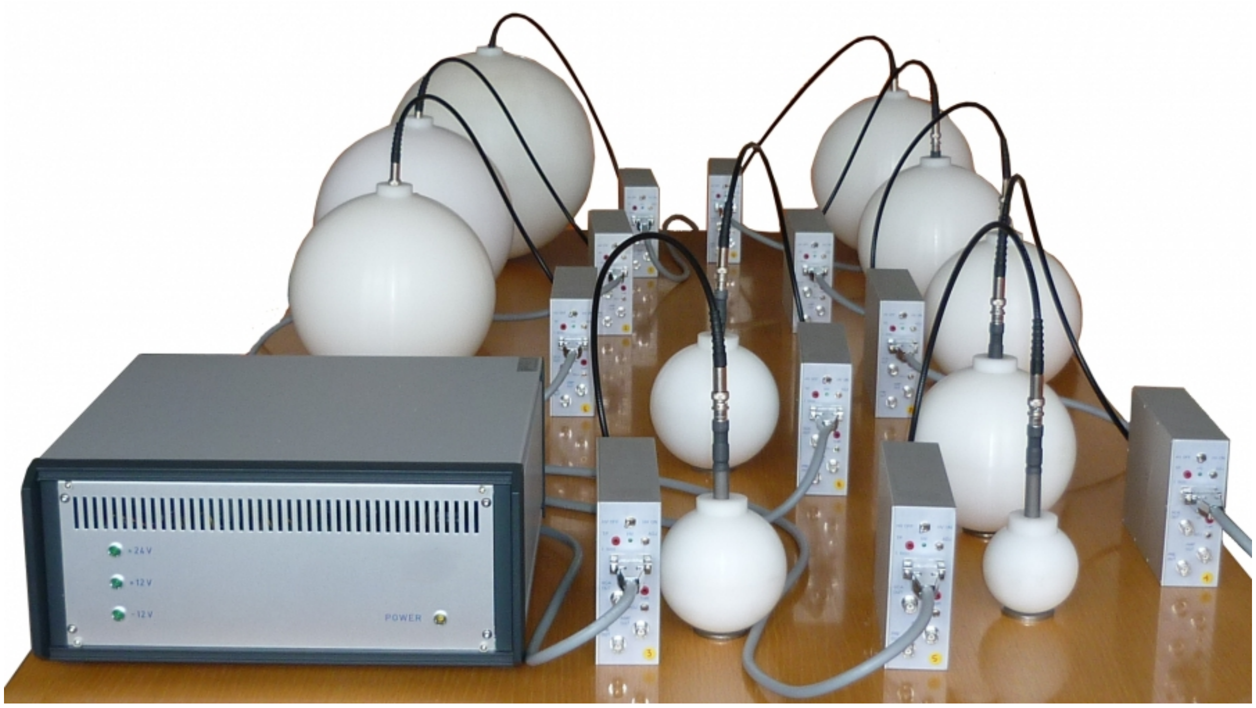


Figure 2.3 A Bonner sphere neutron spectrometer (ElseNuclear 2016).

When a neutron enters each sphere, it may be moderated to a sufficiently low energy that a detector in the center of the sphere can capture and detect the neutron (typically a ^{3}He detector).

By using a variety of sphere diameters, neutrons that fall within different energy windows can be detected by each sphere. For example, a neutron entering a small sphere would be sufficiently moderated and subsequently detected only if it does not have so much energy that it passes completely through the detector before it is sufficiently moderated. Increasing the diameter of the moderating material allows higher and higher energy neutrons to be detected. However, as the diameter of the sphere increases, the probability of the neutrons successfully making it to the detector in the center decreases, necessitating the use of both small and large spheres.

Each sphere will produce an output spectrum which can then be combined with the spectra from the other spheres into a single energy spectrum across the total energy range of the spheres used. (see Fig. 2.4).

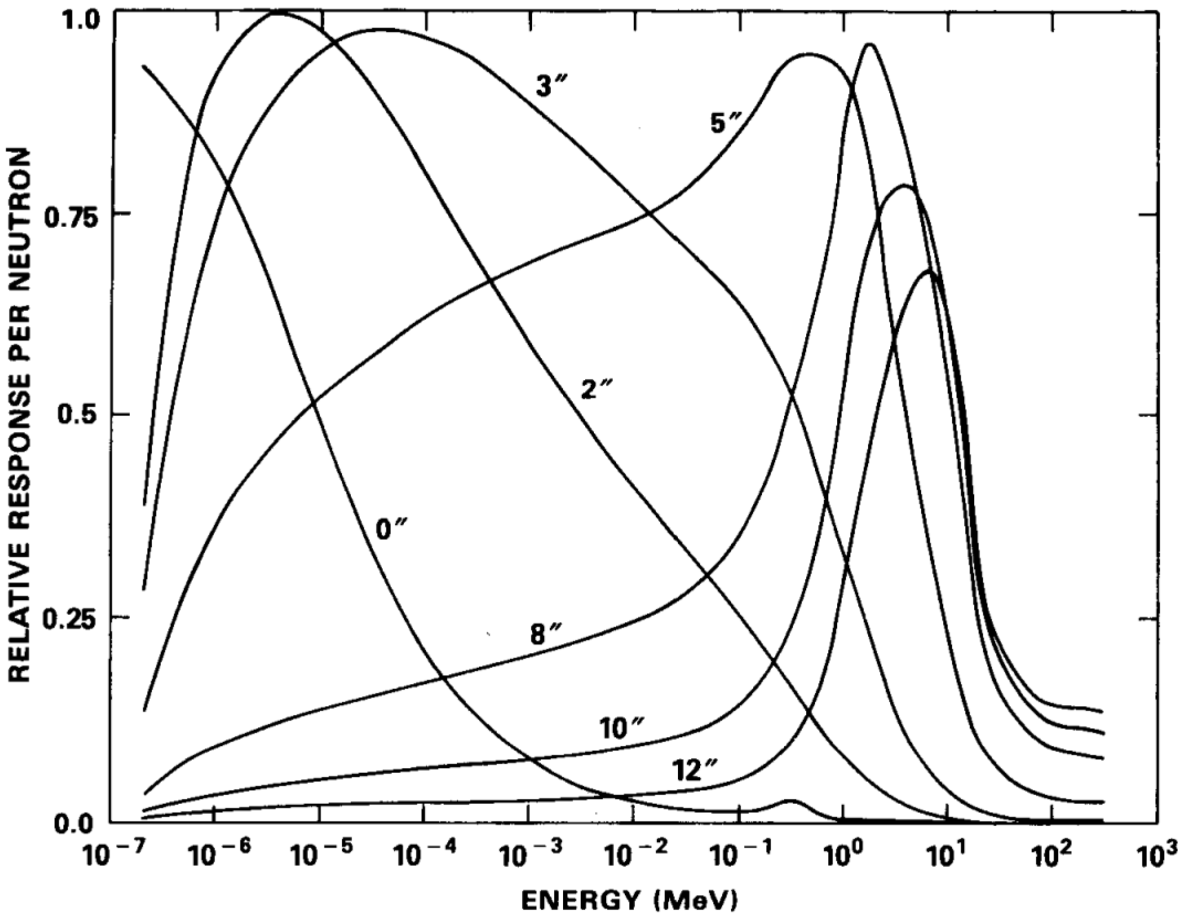


Figure 2.4 Energy spectra from a Bonner sphere spectrometer. Each curve represents the energy spectrum from a different diameter sphere (Knoll 2010).

This process is known as unfolding the spectra. The resolution of the spectrometer is limited by the number of different sizes of spheres used and the capability of the unfolding code to accurately assemble the total neutron energy spectrum. While more portable than a time-of-flight spectrometer, the Bonner sphere spectrometer requires many detectors, each with power supplies, signal processing electronics, and their own opportunities for component failure. A variation of the sphere design has been developed that uses nested cylinders rather than several spheres (Dubeau et al. 2012) (see Fig. 2.5).

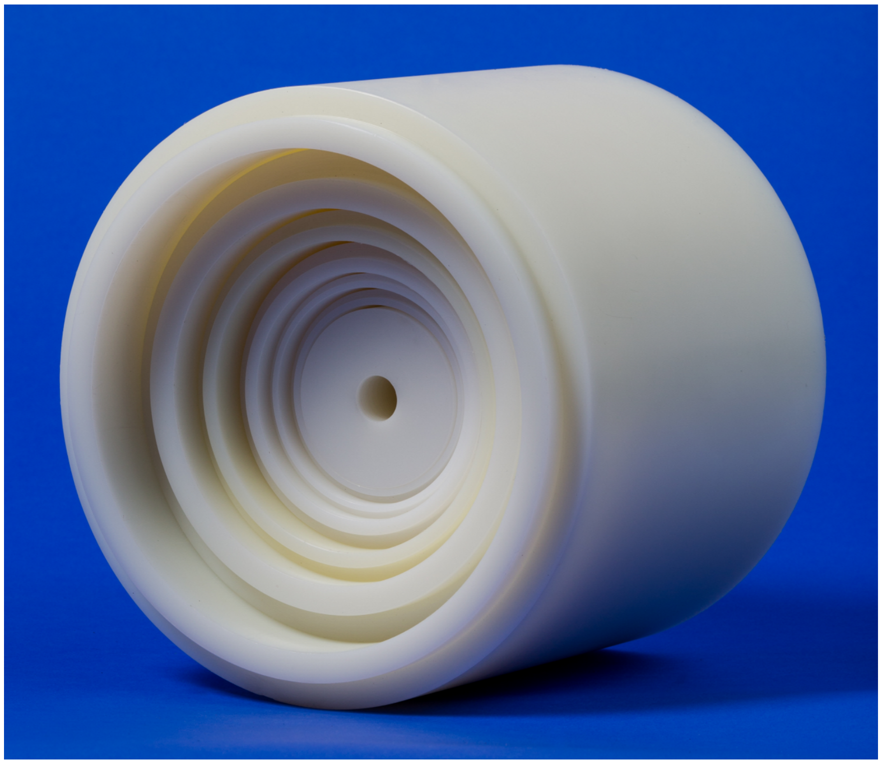


Figure 2.5 Moderating material from a nested neutron spectrometer (Nicholishiell 2013). Image used under the Creative Commons Attribution-Share Alike 3.0 Unported license.

This reduces the number of detectors and increases the portability substantially, but does not improve the resolution capabilities of the spectrometer.

2.2.3 Proton Recoil Telescope

Proton recoil telescopes are capable of performing much higher resolution neutron spectrometry without the need of a second detector, at the cost of efficiency. A collimated beam of neutrons is directed into a very thin hydrogenous radiator. When a neutron collides with a proton, the proton is expelled from the radiator at an angle related to the portion of the energy of the impinging neutron that was transferred to the proton (see Fig. 2.6).

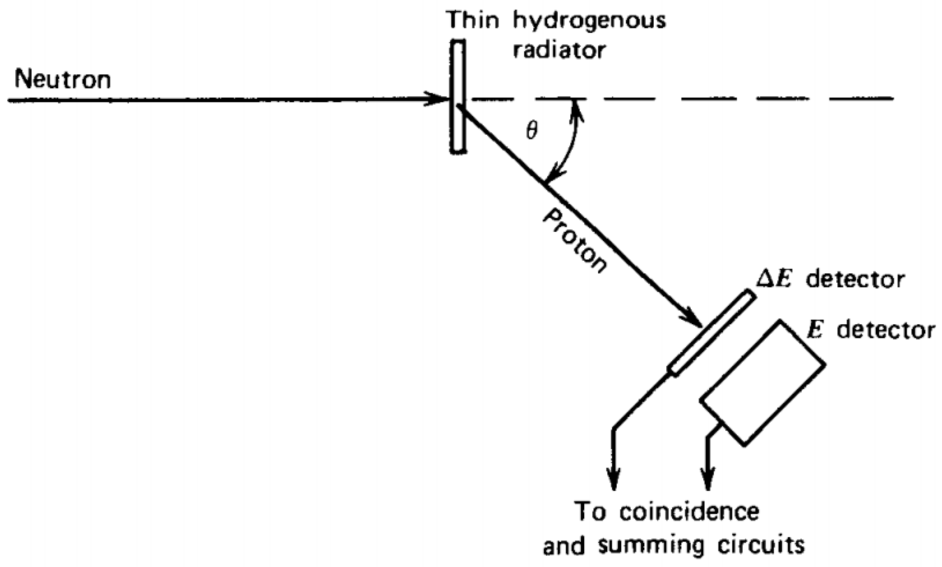

Thin hydrogenous radiator
Neutron
θ
Proton
ΔE detector
E detector
To coincidence and summing circuits

Figure 2.6 A diagram showing the important features of a proton recoil telescope (Knoll 2010).

By measuring the energy of the recoiled proton, the energy of the original neutron can be calculated using

$$E_p = E_n \cos\theta .$$

The resolution of the detector is only limited by the resolution of the proton spectrometer used to detect the recoiled proton and the width of the detector. Using a narrow detector reduces the possible values of θ but also reduces the efficiency of the detector. Typical proton recoil telescopes have very poor efficiency, on the order of 1 in 10^5 (Knoll 2010). In addition to poor efficiency, this spectrometer requires that the travel path from the radiator to the proton detector be evacuated, creating additional engineering difficulties. Our detector is a sort of proton recoil telescope with greatly improved efficiency and an enforced trigger on events with an angle near zero for protons ejected from the radiator. We also couple the detector directly to the hydrogenous radiator, eliminating the need for an evacuated travel path. More details are given in the following sections.

2.3 Detector Development

2.3.1 Scintillators

Scintillators are materials that emit light when charged particles move through them, depositing energy and exciting molecules (see Fig. 2.7).

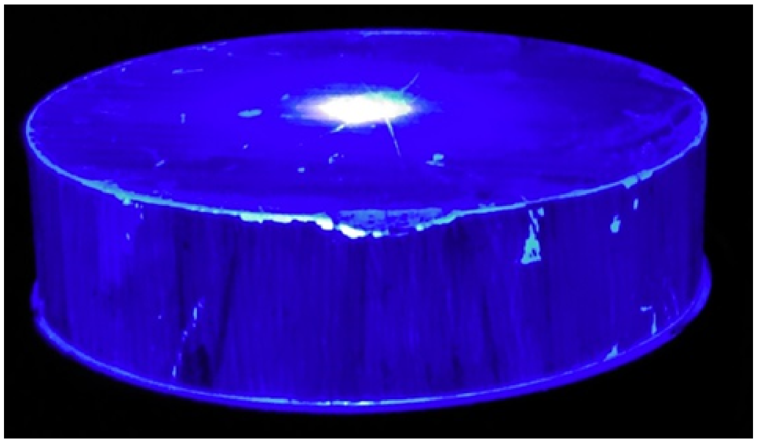


Figure 2.7 EJ-200 plastic scintillator being excited with an ultraviolet laser.

There are two major types of scintillators: organic and inorganic. We use both types in our spectrometer.

Organic scintillators are composed of hydrocarbons with some kind of scintillating dopant such as anthracene. They may be liquid or solid. Liquid organic scintillators have historically been better suited for pulse shape discrimination (PSD) (Zaitseva et al. 2012). The organic scintillator we use in our spectrometer is EJ-200 solid plastic scintillator (see EJ-200 Plastic Scintillator in Appendix B). This particular plastic scintillator is very fast and thus well-suited for high resolution timing measurements. While liquid organic scintillators are better suited for PSD, solid plastic scintillator is much more practical for use in our spectrometer as it can be fairly easily machined into thin cylindrical shapes. (Care must be taken when machining and cleaning plastic scintillator that the optical surfaces are not scratched or marred in any way as this reduces light output and consistency. Solvents (including isopropyl alcohol) should never be used to clean plastic scintillator.) Organic scintillators emit light when excited by charged particles or high energy photons moving through them, depositing energy. When the excited scintillator de-excites it emits photons of a wavelength characteristic of the scintillator type.

The plastic scintillator in our spectrometer serves two primary purposes. First, the hydrogen in the organic scintillator acts as moderating material to slow impinging neutrons sufficiently that they can be captured in the inorganic crystals in the detector. Second, as the neutrons are slowed through collisions with protons, the newly-energized protons move through the scintillator depositing energy and producing light pulses that can be detected by the photomultiplier tube. These pulses serve as a stop signal for a time-of-flight setup and also retain much of the energy information of the original neutron. This preserved energy information is central to our spectrometer design and will be discussed in more detail later. Although the timing of the plastic scintillator is very fast, pulses resulting from gammas and neutrons are too similarly shaped for reliable pulse shape discrimination (see Fig. 2.8).

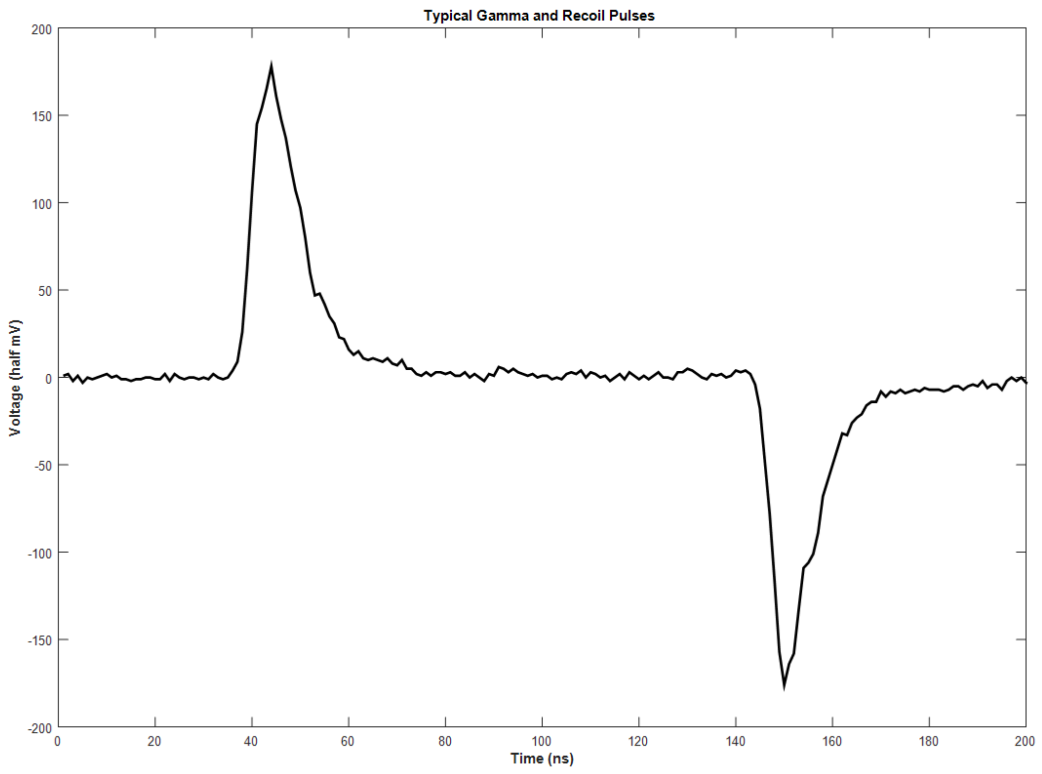


Figure 2.8 Typical gamma and recoil pulses from EJ-200 scintillator. The positive pulse is from a gamma and the negative pulse is from a recoil proton (probably: identified by proximity in time to a capture pulse outside the plotted window).

Thus, another method is required to ensure triggering on true neutron detection events.

The second major subtype of scintillators is inorganic scintillators. Inorganic scintillators are typically materials that emit light when a neutron-capture reaction occurs such as in ^{6}Li glass. The inorganic scintillator used in our detector is $Li_6Gd(BO_3)_3 : Ce$ (LGB) in crystalline form where the LGB captures the neutron and the cerium dopant acts as the scintillating material (see Fig. 2.9).

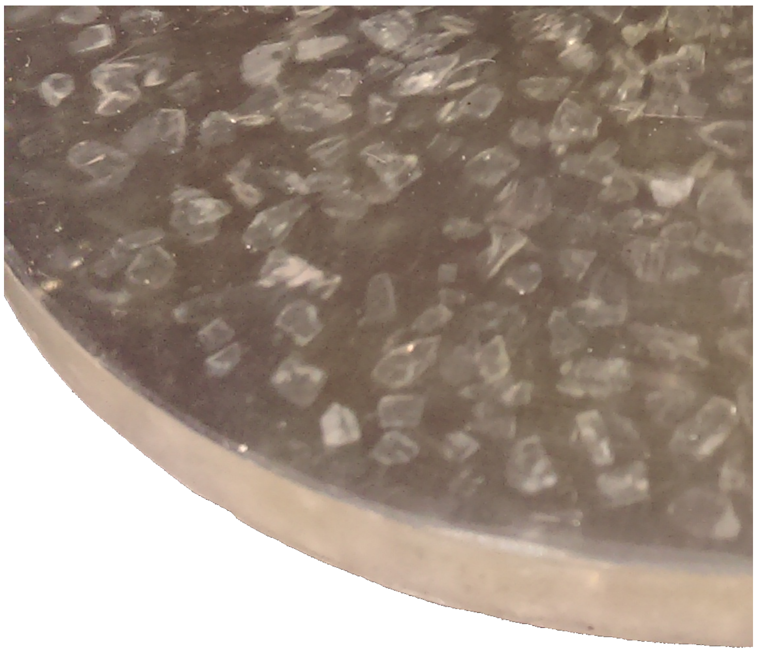


Figure 2.9 Crystalline LGB inorganic scintillator suspended in EJ-200 plastic organic scintillator.

As neutrons collide with protons in moderating material, such as organic scintillator, their energy decreases and their probability of interaction with inorganic scintillators increases (see Fig. 2.10).

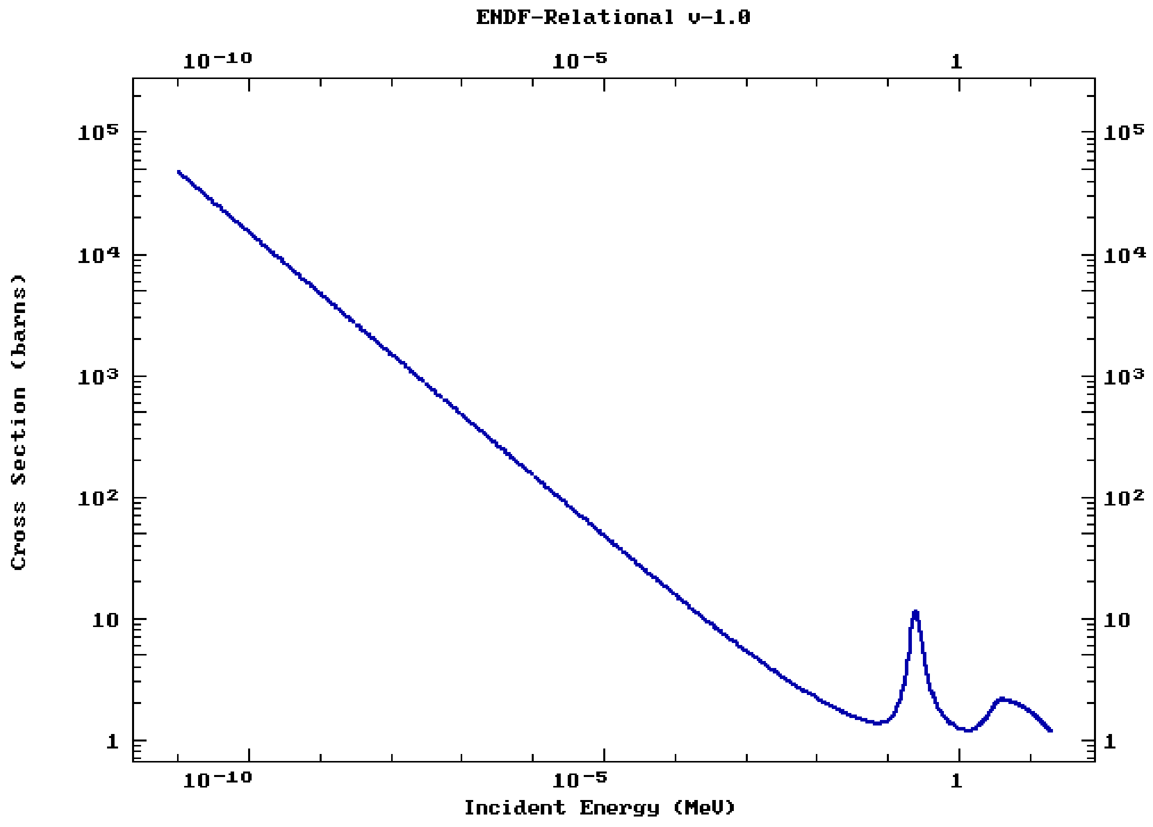


Figure 2.10 The total neutron cross section of ^{6}Li versus energy (ENDF 2016).

The neutron-capture reactions in ^{6}Li and ^{10}B in LGB produce very wide and distinct light output pulses (see Fig. 2.11) (^{6}Li in glass or plastic produces narrow pulses that are difficult to distinguish from gammas).

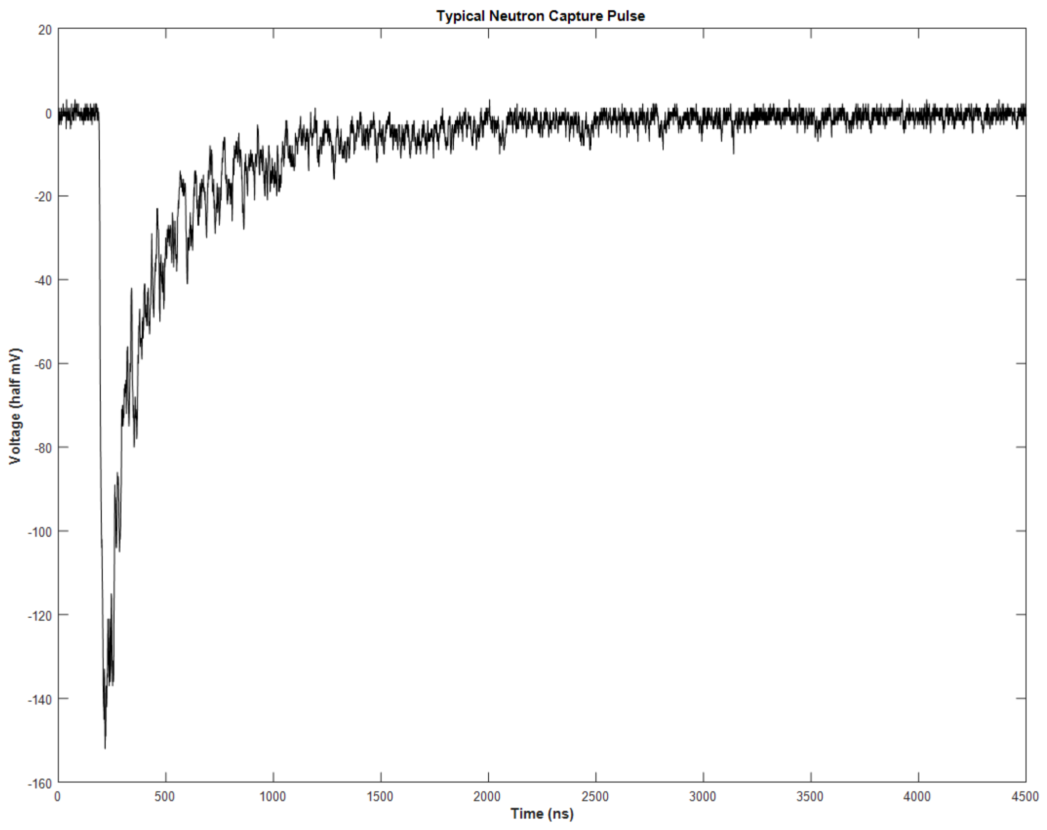


Figure 2.11 A typical neutron capture pulse in LGB.

We gate our spectrometer on these wide pulses in order to discriminate against gammas.

Our detector produces signature double pulses when a neutron is detected: one from proton recoil in the organic scintillator and one from neutron-capture in the inorganic scintillator (see Fig. 2.12).

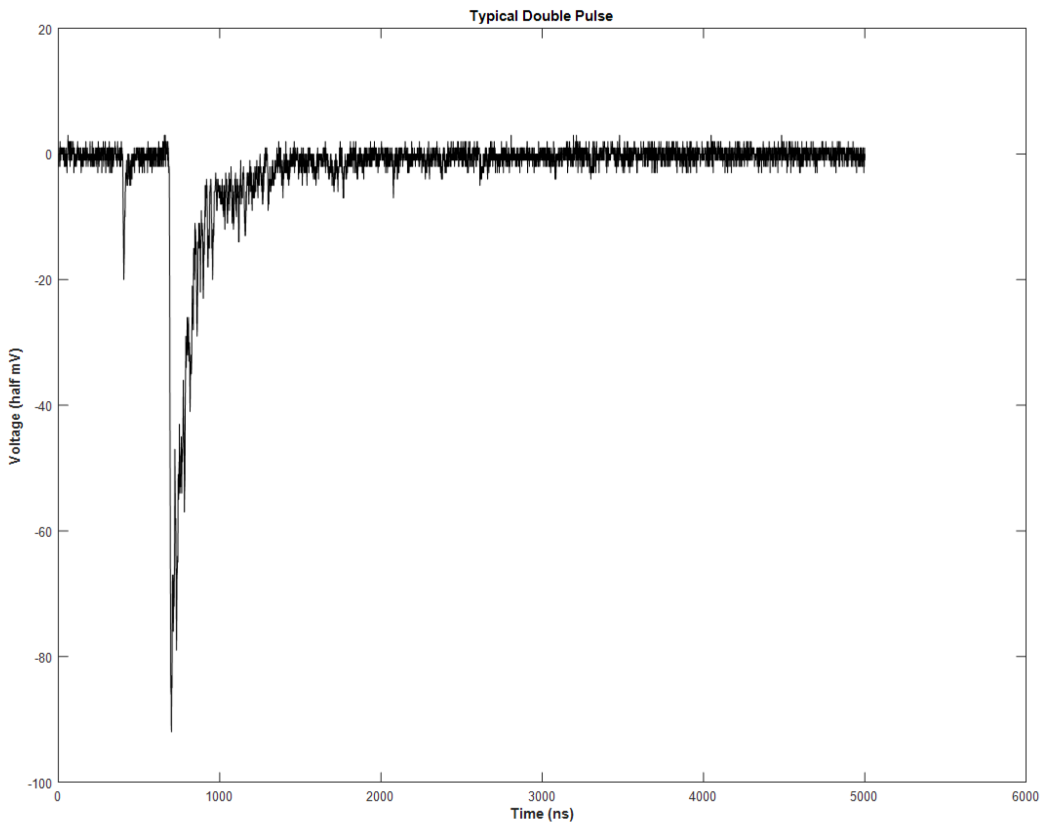


Figure 2.12 A typical double pulse seen by our detector. The first narrow pulse is from the proton recoil. The second wide pulse is from capture in ^{6}Li or ^{10}B.

This ensures that we mostly observe valid neutron events and, only occasionally, other coincidental events such as gammas. This significantly reduces the need for pulse shape discrimination. However, because the plastic scintillator is also sensitive to gammas, it is fairly common for pulses from gammas to be misidentified as proton recoil pulses. This can be minimized by enforcing a timing window between the supposed proton recoil and the neutron capture pulses.

In addition to gating our detector on neutron events, we enforce mostly one-to-one energy transfers in the initial proton recoil. On average, neutrons do not impart all of their energy to a proton in a single collision. We enforce one-to-one energy transfers using a combination of capture-gating and very thin scintillators. When a proton recoils off a neutron, the interaction follows standard elastic scattering physics (see Fig. 2.13).

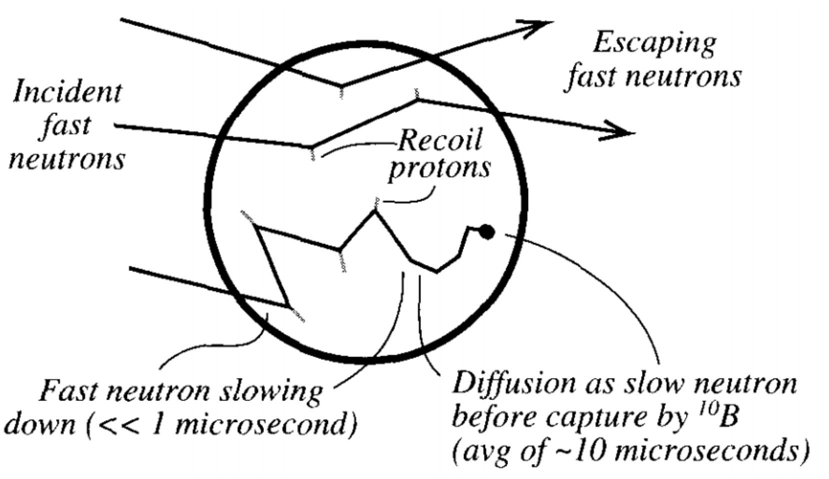

Incident fast neutrons
Escaping fast neutrons
Recoil protons
Fast neutron slowing down (<< 1 microsecond)
Diffusion as slow neutron before capture by ^{10}B (avg of ~10 microseconds)

Figure 2.13 Possible results of proton recoil interactions (Knoll 2010). Slowed neutrons in our detector can also capture in ^{6}Li in addition to the ^{10}B shown in the figure. In our detector, the diffusion time is closer to an average of a few hundred nanoseconds.

The closer to a one-to-one energy transfer, the closer the neutron will recoil ninety degrees from its original path. When using very thin scintillators, if the neutron does not recoil ninety degrees it will pass out of the scintillator before it can either recoil again or be captured in the LGB. Using thin scintillators also has the advantage of limiting the light loss due to absorption in the material. Once the light has been produced in the scintillator, it must be detected and converted to an electrical signal for analysis.

2.3.2 Photomultiplier Tubes

Photomultiplier tubes (PMT) use the photoelectric effect to convert photons into electrons and then multiply the electrons into larger and larger cascades that can then be detected as an electrical signal (see Fig. 2.14).

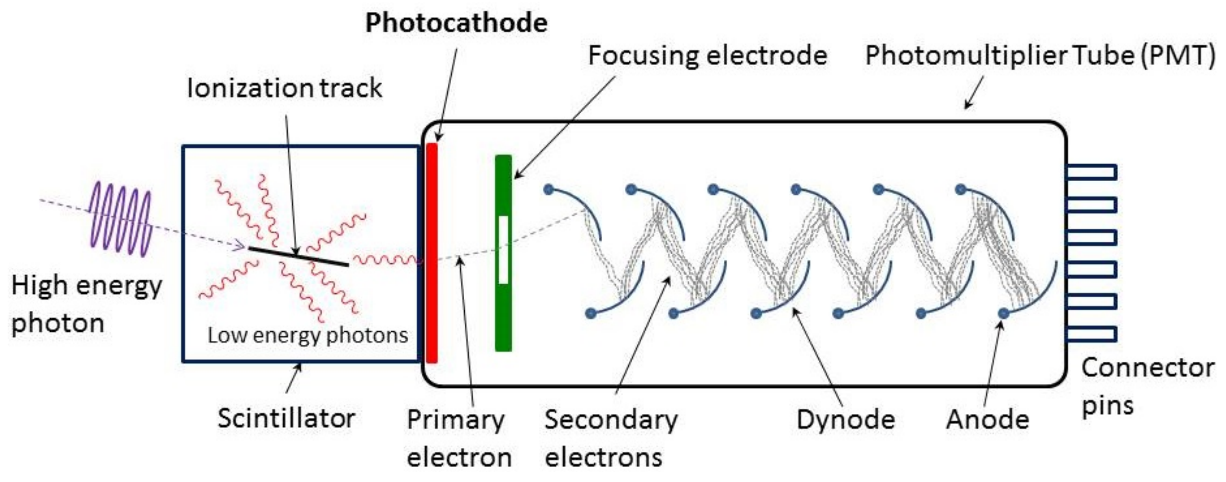


Figure 2.14 Interactions in a photomultiplier tube (Qwerty123uiop 2013). Image used under the Creative Commons Attribution-Share Alike 3.0 Unported license.

For good photon to electron conversion, the work function of the photoelectric material on the inside face of the PMT must be matched to the output wavelength spectrum of the scintillator used.

It is important to ensure that power is never applied to the PMT when it is exposed to ambient light as this will cause severe damage. Thus, any detectors using a PMT must be contained in some sort of light tight container. It is useful to do initial tests in a light-tight box, providing easy access for quick changes to the detector. After initial tests are complete it is often more convenient to install the detector into a more portable light-tight can.

We tested two different types of PMTs with our spectrometer. The first was a 10-dynode Adit B133D01 (see B133D01 Photomultiplier Tube in Appendix B). Initial tests showed that this PMT would be good for preserving the pulse shapes resulting from light output from the scintillator. However, additional tests have shown that this is not likely to be true. Additionally, the Adit PMT does not have good timing capabilities (see Fig. 2.15).

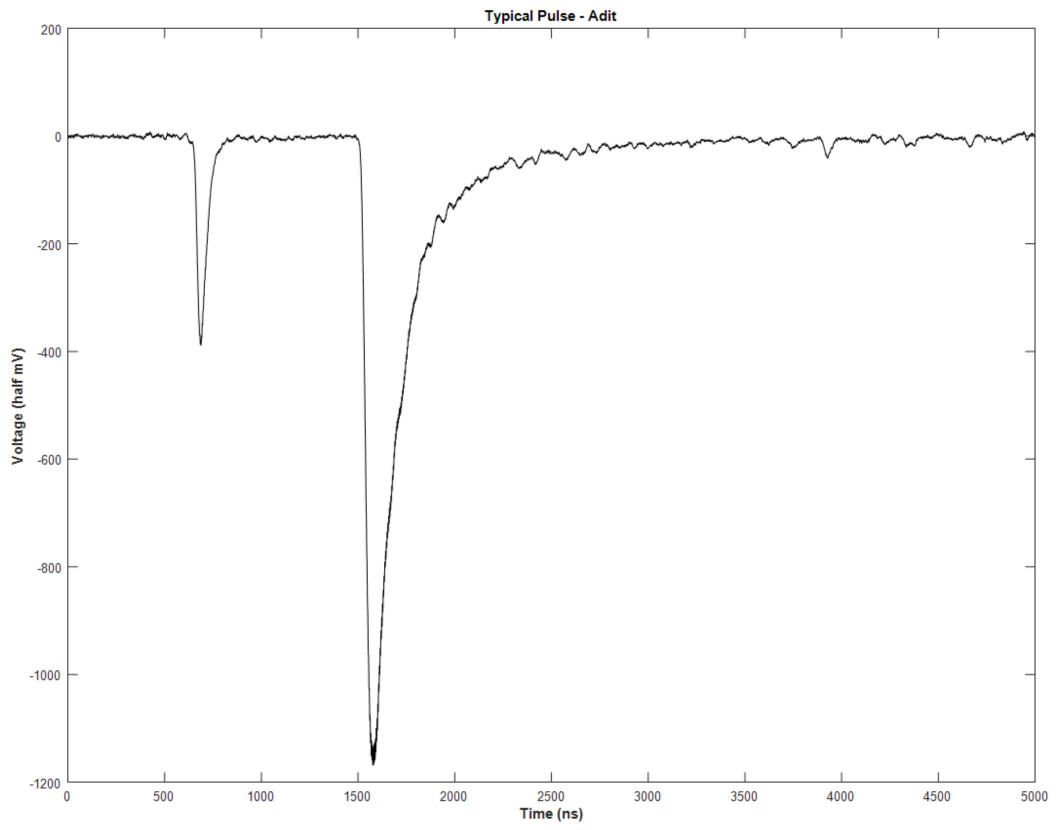


Figure 2.15 A typical waveform from a neutron detection event using an Adit B133D01 PMT and LGB scintillator. The first pulse is the proton recoil pulse and the second, wider pulse is the neutron capture pulse.

The second kind of PMT we used is a Hamamatsu R1250 (see Hamamatsu Photomultiplier Tube R1250 in Appendix B). This 14-dynode PMT has substantially better timing capabilities than the Adit (see Fig. 2.16).

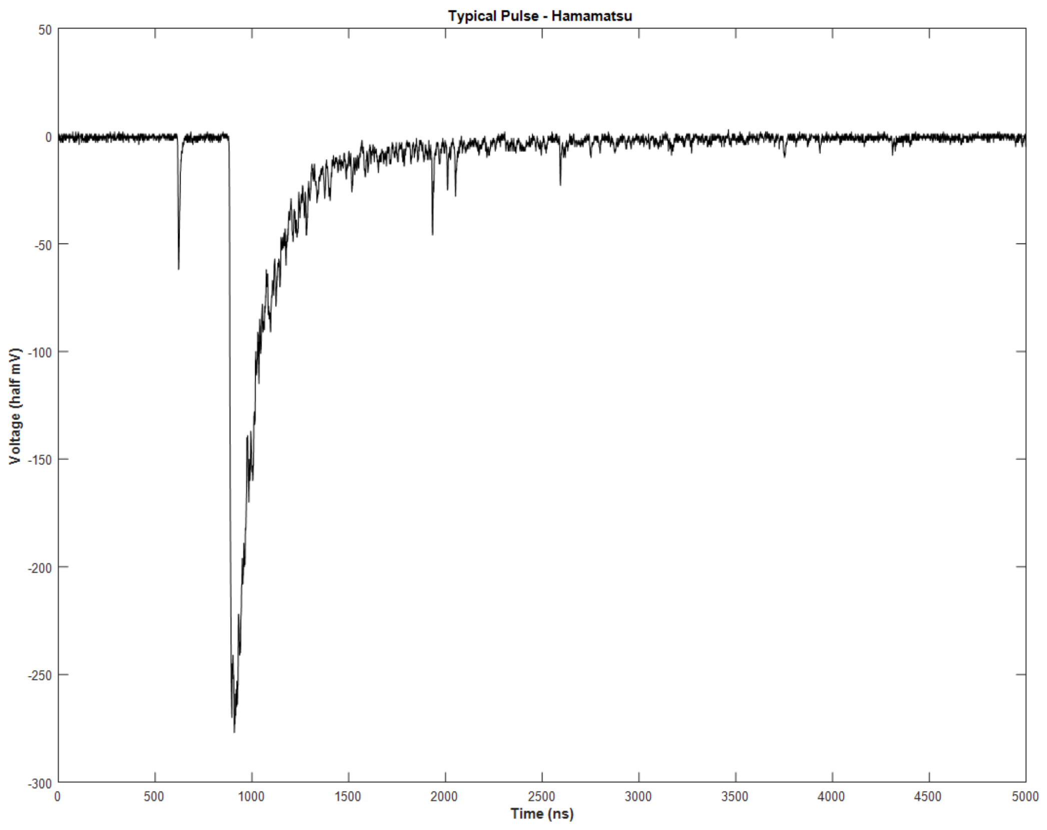


Figure 2.16 A typical waveform from a neutron detection event using a Hamamatsu R1250 PMT and LGB scintillator. The first pulse is the proton recoil pulse and the second, wider pulse is the neutron capture pulse.

The increased timing resolution in the Hamamatsu tube makes it simpler to separate recoil and capture pulses for small drift times. The significantly narrower recoil pulses also make pulse shape discrimination using pulse area more reliable (this will be discussed in more detail later on). It also reveals more information about the structure of the LGB response to a neutron detection event. The Hamamatsu waveform suggests that some of the light output occurs after the initial peak, rather than decaying from the initial output as the Adit waveform suggests.

Although the Hamamatsu provides better temporal resolution, there is a huge variation (nearly

50%!) in signal output across the face of the PMT, making it great for time-of-flight but not good for pulse shape analysis (see Fig. 2.17).

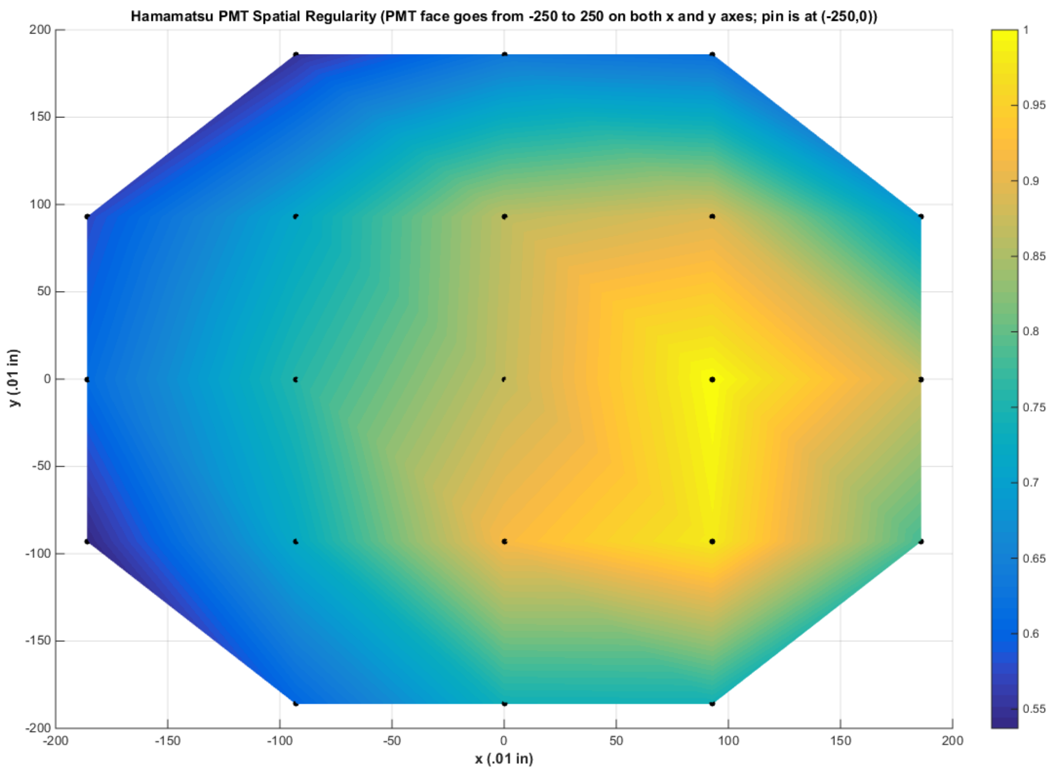


Figure 2.17 Spatial inconsistency across the face of a Hamamatsu R1250 photomultiplier tube. A small piece of ^{6}Li glass was placed at each black dot and a pulse height histogram was taken. The color value indicates the relative peak value of each histogram.

We used the Hamamatsu PMT both to get high resolution timing data for comparing our spectrometer to time-of-flight data and also to get high resolution information about the pulse shapes produced by the detector. The final spectrometer was planned to use an Adit PMT to maximize energy information preservation since we would no longer need good timing information for comparison to time-of-flight. Though the slower PMT makes it more difficult to distinguish between occasionally very close start-recoil and recoil-capture pulse pairs, we initially thought the gain in pulse area consistency across the PMT would be a more significant effect. Additional spatial consistency studies of the Adit PMT indicate that it is even worse than a Hamamatsu for both spatial and timing consistency. 25 randomly selected locations on the face of the PMT were tested using a very small piece of ^{6}Li glass. The variation across the tube is at least 60% (see Fig. 2.18).

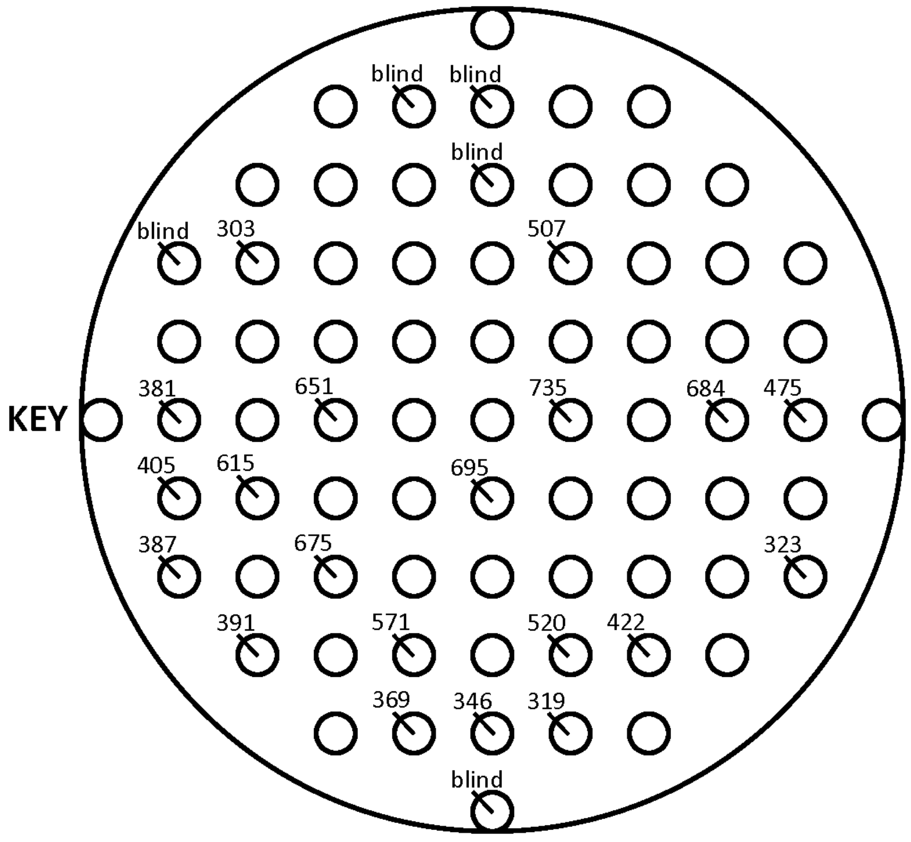


Figure 2.18 Spatial inconsistency looking down at the face of an Adit B133D01 photomultiplier tube. A small piece of ^{6}Li glass was placed in each labeled circle (chosen randomly) and a pulse height histogram was taken. The label indicates the peak value of each histogram. The locations marked "blind" did not produce any triggers at all.

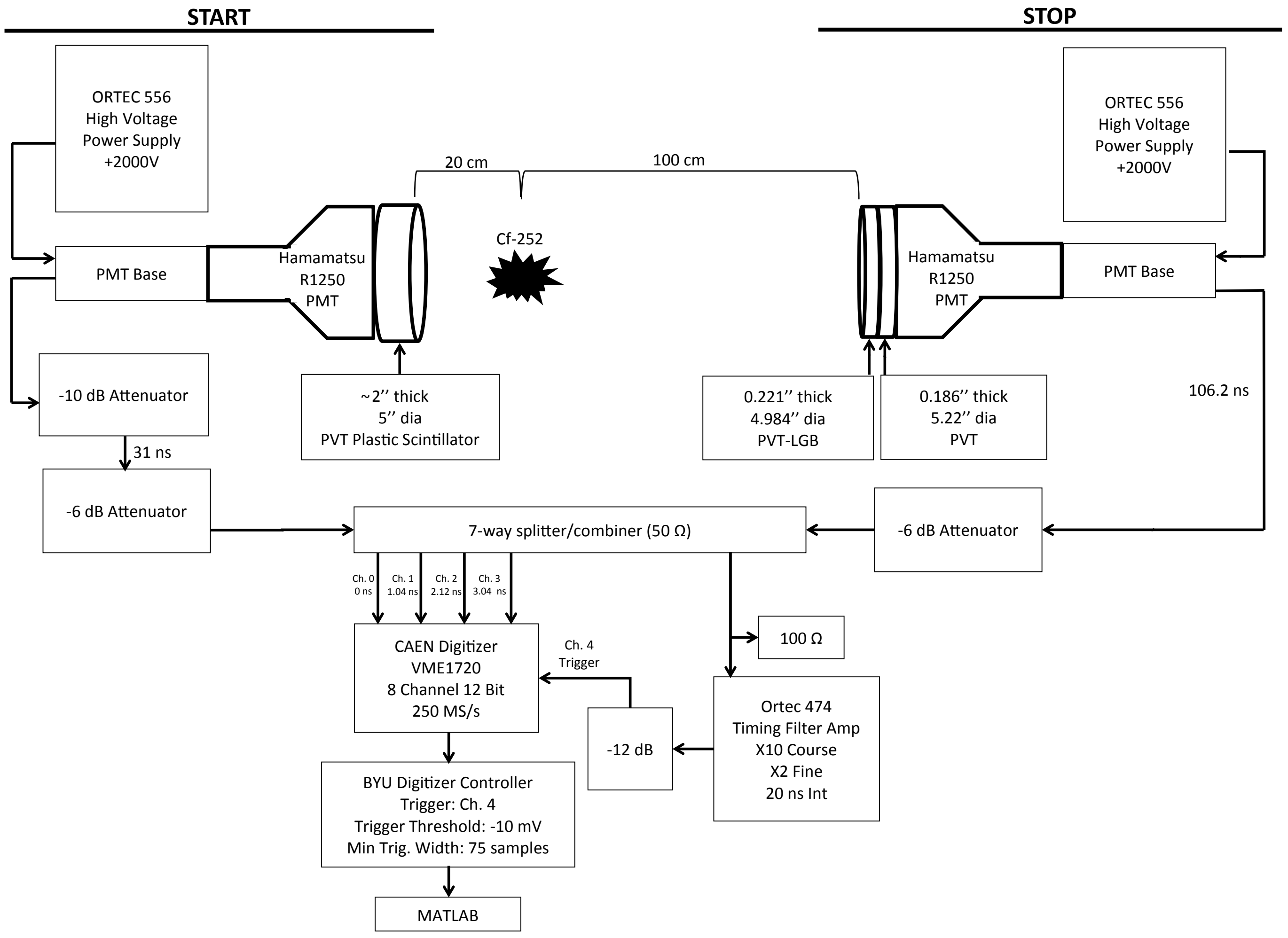

START
STOP
ORTEC 556 High Voltage Power Supply +2000V
ORTEC 556 High Voltage Power Supply +2000V
PMT Base
PMT Base
Hamamatsu R1250 PMT
Hamamatsu R1250 PMT
20 cm
100 cm
Cf-252
-10 dB Attenuator
31 ns
~2'' thick 5'' dia PVT Plastic Scintillator
0.221'' thick 4.984'' dia PVT-LGB
0.186'' thick 5.22'' dia PVT
106.2 ns
-6 dB Attenuator
7-way splitter/combiner (50 Ω)
-6 dB Attenuator
Ch. 0 0 ns
Ch. 1 1.04 ns
Ch. 2 2.12 ns
Ch. 3 3.04 ns
100 Ω
CAEN Digitizer VME1720 8 Channel 12 Bit 250 MS/s
Ch. 4 Trigger
-12 dB
Ortec 474 Timing Filter Amp X10 Course X2 Fine 20 ns Int
BYU Digitizer Controller Trigger: Ch. 4 Trigger Threshold: -10 mV Min Trig. Width: 75 samples
MATLAB

Figure 2.19 The developmental setup used to test and characterize the spectrometer. The detector is installed in a time-of-flight setup.

2.3.3 Developmental Setup

In order to determine the spectrometric capabilities of our detector we installed it as the stop detector in a time-of-flight setup (see Fig. 2.19). The ideal waveform produced by a neutron detection event from this setup would contain four (and only four) distinct pulses (see Fig. 2.20).

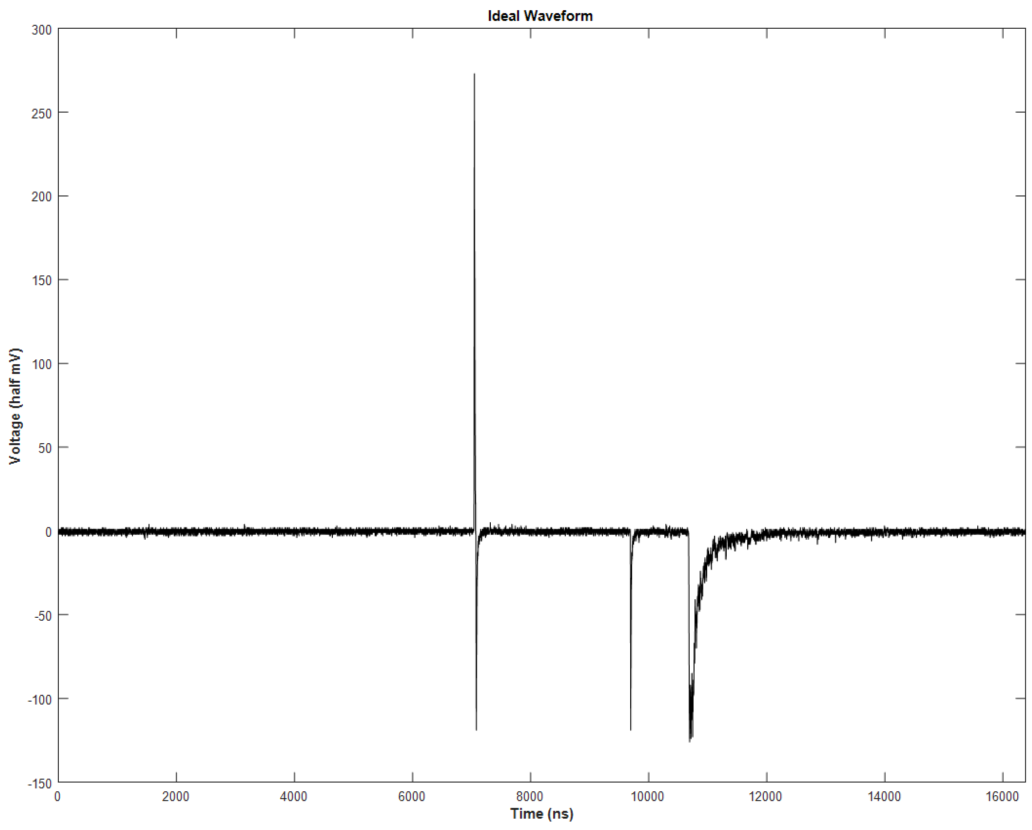


Figure 2.20 An ideal neutron detection event from the developmental setup.

The first pulse (positive) results from a gamma detection event in the start detector. The second pulse (negative) results from a gamma detection event in the stop detector. This would consistently occur within a narrow window of time since gammas all travel at the speed of light. The third pulse (negative) is from a proton recoil detection event and the fourth pulse (negative) from a neutron capture event. In practice, it is very rare that a non-coincidental gamma is detected in the stop detector since the solid angle is so low a meter from the source. Additionally, there are almost always other coincidental pulses scattered throughout the event, complicating the analysis process.

This section will address many of the considerations necessary for getting good time-of-flight data. First, in order for time-of-flight to be accurately measured it is critical to know the timing characteristics of the system. Electrical signals do not travel instantly from point to point through cables and other components. A good estimate for choosing cable lengths is around 1 ns per foot of RG-58 coaxial cable. The lengths can then be measured by splitting the signal from a pulse generator along two path lengths and measuring the difference in time between the arrival of the signal along the two paths using an oscilloscope. It is useful to start with two paths of equal length and then add the cable to be measured to one path.

It is not necessarily the best choice for the cable lengths from the start and stop detectors to be the same. By delaying one or the other (typically the stop path) you can increase separation between closely correlated pulses making it much easier to resolve the separate pulses. For example, if your setup has a path length of 1 meter you will detect a gamma in the start detector and then a gamma in the stop detector 3.3 ns later, effectively giving you an idea of where 0 is in your time-of-flight plots. If your cable lengths from each detector are the same and you are using a digitizer with a rate of 250 million samples per second (which we do) it will be impossible to resolve the two separate pulses. If the path lengths are different, two separate pulses will be resolved and the timing difference between them can then be accounted for in analysis.

Once all the cable lengths have been measured and the timing characteristics of the setup are known, it is useful to measure the overall timing difference between the two paths, including all cables and electronics. This is done by placing the detectors face to face with only enough space between them to fit a gamma emitter, such as ^{60}Co. The time-of-flight between the two detectors should now be zero, so any measured variation from zero is a characteristic of the setup and needs to be accounted for in analysis.

In addition to timing considerations, it is critical that signal, cable, and component electrical impedances are matched everywhere in the system. Most of the cabling and nuclear instrumentation modules (NIM) we use have an impedance of 50 Ω. If any components are added to the system with a different impedance, it will cause characteristic echos in the signal as a result of reflection off the impedance mismatch. It will also attenuate the original signal, damaging the pulse shape information critical to the use of the detector as a spectrometer. One common component in our time-of-flight setup that does not have 50 Ω input impedance is the Ortec 474 timing filter and amplifier. Its 100 Ω impedance requires a 100 Ω terminator in parallel with the input to match the signal impedance. When matching impedance it may be necessary to review resistor summing rules:

$$\begin{aligned} \text{series:} \quad & R = \sum R_i \\ \text{parallel:} \quad & \frac{1}{R} = \sum \frac{1}{R_i} \end{aligned}$$

The developmental setup uses a CAEN V1720 digitizer with 12 bit voltage resolution over ± 1 volt and 250 million samples per second per channel (one sample every 4 ns). Having more channels available than needed, we split the signal into four channels, each delayed one nanosecond from the previous. This effectively quadrupled our sample rate to one sample every 1 ns, greatly improving the timing resolution. Rather than do this separately for the start and stop detectors, we summed the two with different polarities; the start waveform with a positive polarity and the stop waveform with a negative polarity (see Fig. 2.20). This provides better resolution for time-of-flight measurements and greater information about the pulse structure from the detector. In combining, splitting, and delaying the two detector signals it was necessary to match impedances at every step along the way. This was done using a 50 Ω impedance-matched star-configuration 7-way splitter/combiner (one input for each detector, four outputs to the digitizer, and one output to the timing filter to provide an integrated trigger for the digitizer). This trick gives us a voltage resolution of 0.5 mV and a timing resolution of 1 ns.

Another important consideration when building the developmental setup is the voltage requirements and limitations of each component in the system. The digitizer and the timing filter amplifiers have an input limit of only ± 1 V while the PMTs output up to 30 V from a cosmic ray detection event. Attenuators are necessary to protect the equipment. On the other hand, it is best to have as much signal over background as possible so we kept the signal voltages near the upper limits of each component in the setup.

The developmental setup differs from the final stand-alone operation of the spectrometer in several key ways. First, all of the timing considerations are no longer important since the detector does not need to be in a time-of-flight configuration. This greatly simplifies the setup. Additionally, it is not necessary to have such high timing resolution, eliminating the need for a the splitter/combiner and all of the delay cabling. The developmental setup uses a Hamamatsu PMT to increase the resolution of the time-of-flight data. This produces capture pulses that contain many peaks over a long period of time rather than a single wide capture pulse. This pulse shape (see Fig. 2.16) is difficult to trigger on, so we used an integrated waveform (see Fig. 2.19) to trigger the digitizer. The final setup will use a slower Adit PMT, eliminating the need for the integration feature of the Ortec 474 timing filter amp for the integrated trigger. However, the 10-dynode Adit PMT has significantly less gain than the 14-dynode Hamamatsu, so the amplification feature of the 474 may be necessary. Other components necessary for operation of the spectrometer in its intended standalone environment include a power supply for the PMT and a digitizer (see Fig. 2.21).

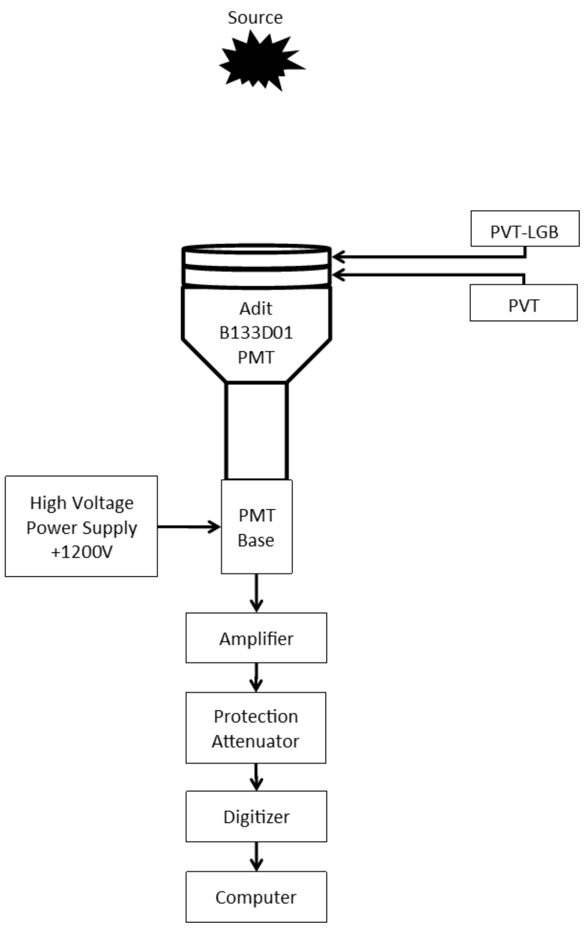


Figure 2.21 The necessary components for the spectrometer to operate in its final intended standalone mode.

2.3.4 Detector Characterization

The spectrometer consists of three main components: a first scintillating slab containing inorganic LGB crystals in organic polyvinyl toluene (PVT), a second scintillating slab of PVT, and a photomultiplier tube (see Fig. 2.22).

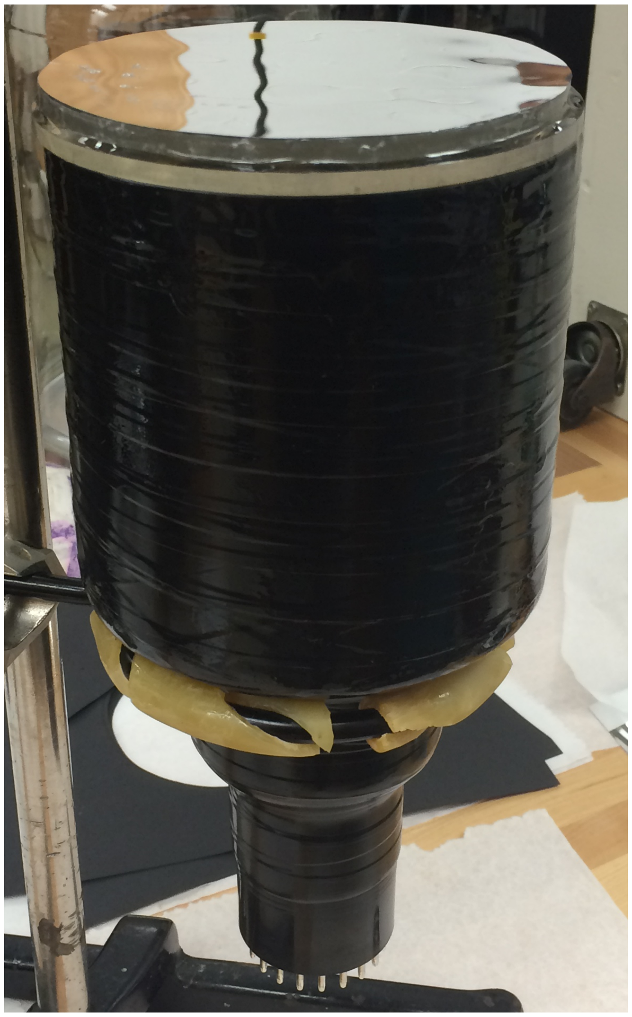


Figure 2.22 Our neutron spectrometer. The reflective layer on top is aluminized mylar. The first thin slab beneath the mylar is LGB in PVT scintillator. Beneath the LGB is PVT scintillator. Beneath the PVT scintillator is a Hamamatsu R1250 photomultiplier tube.

The LGB slab measures 0.221 inches thick with a diameter of 4.984 inches. The PVT slab measures 0.186 inches thick with a diameter of 5.22 inches. The scintillators are optically coupled to each other and to the face of the PMT with BC-630 silicone grease (see BC-630 Silicone Optical Grease in Appendix B). When run in a time-of-flight setup, the PMT used is a Hamamatsu R1250, chosen for its excellent timing characteristics. When in standalone mode, the detector was planned to use an Adit B133D01 PMT, chosen for its supposedly better energy information preservation.

Additional study has shown that an entirely different photon detector may be necessary. This is discussed in more detail later on.

The detector works in a manner similar to a proton recoil telescope. A neutron collides with a proton in the thin hydrogenous first slab of scintillator, transferring its energy. If all the energy is transferred the neutron will drift ninety degrees to its original path until it either captures in the LGB crystal or exits the detector. The detector is gated on the capture pulse, enforcing neutron detection. If the neutron does not transfer all of its energy to the first collision proton it will move off at an angle of less than ninety degrees and will quickly leave the crystal-filled detection plane. Thus, triggering on captures enforces not only neutron detection events, but also one to one energy transfers. The recoiled proton subsequently excites the scintillator in the second slab of PVT emitting photons that are detected by the PMT (see Fig. 2.23).

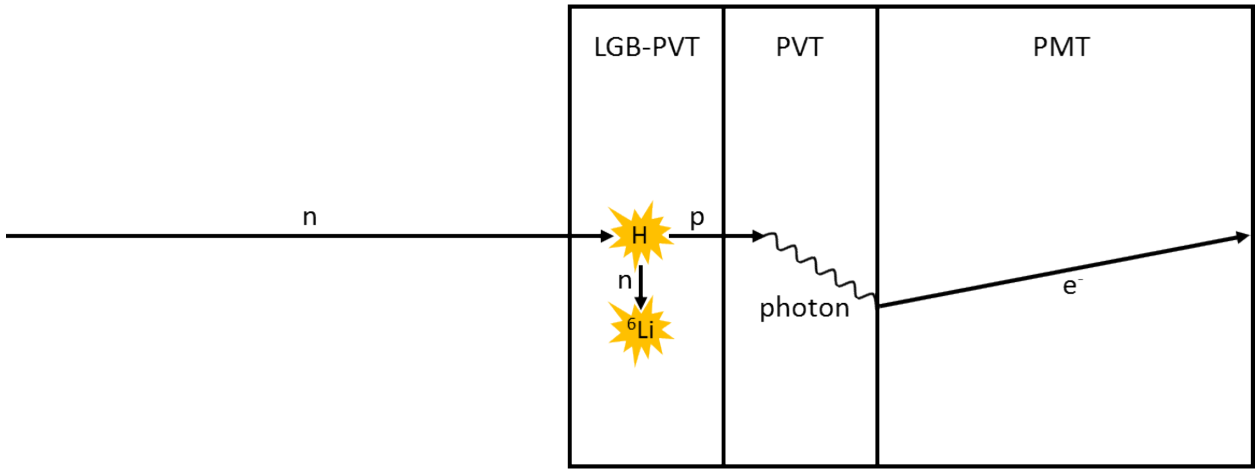


Figure 2.23 A typical neutron detection interaction in the spectrometer. Note that the diagram is not to scale. The scintillators are actually much thinner so that the neutron will escape the detector if the collision does not result in a one-to-one energy transfer.

The number of neutrons successfully detected compared to the total number of neutrons that pass through the detector undetected is referred to as the efficiency of the detector. For example, if 100 neutrons pass through the detector and seven of them are detected, the detector is said to have an efficiency of 7%. In order to calculate efficiency, it is necessary to know how many neutrons pass through the detector over a given period of time. This can be calculated from the rate of the neutron source and the solid angle of the detector. After the efficiency of the detector has been calculated, it is possible to go in the other direction and calculate unknown source rates from the number of neutrons detected in a given period of time. We tested our detector using a sealed ^{252}Cf source. ^{252}Cf undergoes radioactive decay via alpha decay 96.9% of the time:

$$^{252}\mathrm{Cf} \rightarrow {}^{4}\mathrm{He} + {}^{248}\mathrm{Cm}$$

and by spontaneous fission the other 3.09%, producing various fission fragments and neutrons. Our source had a rate of 88.96 μCi on 1 August 2012. To calculate the current rate you use

$$A = A_0 2^{-t/T_{1/2}}$$

where A is the current rate, A_0 is the original rate, t is the time since the original rate was measured, and $T_{1/2}$ is the half-life of the material. The half-life of ^{252}Cf is 2.645 years, giving a current rate of 34.28 μCi at the time of calculation (with such a short half-life this rate will not stay meaningful for long). A more convenient unit for the next few calculations is the becquerel (Bq), which is one decay per second. 1 curie is 3.7 x 10^{10} becquerels giving us a current source rate of 1.27 x 10^{6} Bq. Since only 3.09% of decays produce neutrons via spontaneous fission and each fission produces on average 3.7675 neutrons that gives us a neutron rate of approximately 147,000 neutrons per second. These neutrons are emitted isotropically (equally in every direction) from the source. The solid angle of the detector is approximately the ratio of the surface area of the face of the detector to the surface area of a sphere with a radius of the distance from the source to the detector (one meter in our setup). This ratio more closely approximates the actual solid angle with increasing distances from the source. Multiplying the number of neutrons emitted from the source each second by this ratio yields the number of neutrons passing through the detector face each second, 148.6 neutrons per second. Over a ten day period, the rate of the spectrometer (triggered on neutron capture pulses) was 1.3 counts per second, yielding an efficiency of 0.87%. Alternatively, assuming there is a linear relationship between detector efficiency and scintillator thickness, the efficiency of the spectrometer could be calculated from the known efficiency of a similar detector. In our case, one made with four inches of the same type and diameter of scintillator made by Photogenics. Using this method, the efficiency of our spectrometer ought to be around 0.6%. This estimate is in relative agreement with the previous calculation.

2.4 Data Acquisition

Data was acquired using a CAEN V1720 digitizer (see V1720 / Digitizers | CAEN in Appendix B). The digitizer was controlled by a C program lovingly referred to as Ugly Controller adapted from digitizer controller software provided by CAEN called CAEN WaveDump. The primary adaptation enabled the digitizer to trigger on pulse width in addition to the standard threshold trigger. Each channel used for data acquisition needs to be DC offset so that the noise level is centered around zero volts in order for predictable trigger behavior and maximum use of the digitizer's 2 V_{pp} range. Each event consisted of 4096 samples (16384 ns) per channel. We used five channels of the available eight: four for signal input and one for the integrated trigger input. The trigger was set to a threshold of 10 mV below the center of the noise level and a width of 75 channels (300 ns). The data was transmitted to the computer via optical link. See Appendix C for a typical configuration file used to configure the digitizer for data acquisition.

2.5 Data Analysis

The data was analyzed using software called ToFSpec developed in Matlab. The software reads the binary data saved by the digitizer and processes it in several steps, culminating in several plots of interest. The first step iterates through the data and removes events that do not contain any pulses from the start detector. The second step combines the four sequentially delayed waveforms from each channel into a single waveform. The third step looks through each waveform and picks out each pulse in the waveform, assigning them labels according to which detector they were produced by. Finally, the fourth step analyzes each individual pulse, measuring various pulse metrics such as height, width, and area. These features are discussed in greater detail in the next section. After analyzing the individual pulses, the software builds several plots displaying the various features compared with each other (for pulse shape discrimination), time-of-flight, and energy to look for useful correlations. The analysis process also aids in calibrating the detector for stand-alone spectrometric measurement.

2.5.1 Pulse Metrics

The software characterizes each pulse by recording several pulse metrics: start sample number, peak height, area, early area, time-of-flight (as applicable), and energy (calculated from time-of-flight).

The start sample number is used in time-of-flight calculations and for enforcing timing windows between start and recoil pulses and also between recoil and capture pulses. The windows were chosen to exclude a majority of coincidental events that made it through the event processing stages. For example, neutrons below approximately 0.1 MeV do not have sufficient energy to produce a proton recoil pulse. Therefore, if we see a proton recoil pulse with a time-of-flight longer than what a 0.1 MeV neutron would have, it is likely a coincidental gamma event misidentified as a proton recoil event.

The peak height is related to the energy of the recoil proton that caused the light output in the scintillator by

$$H = kE^{3/2}$$

where H is the pulse height, k is a proportionality constant determined by the scintillating material, and E is the energy of the recoil proton (Knoll 2010). It is assumed that the relationship is the same for pulse area. Pulse area seems to most accurately preserve the energy information among all the pulse metrics. We use the pulse area values to construct an energy spectrum of the original neutrons. The early area is defined as the area of the pulse that occurs before some sample in the pulse and is used for pulse shape discrimination (see Fig. 2.24).

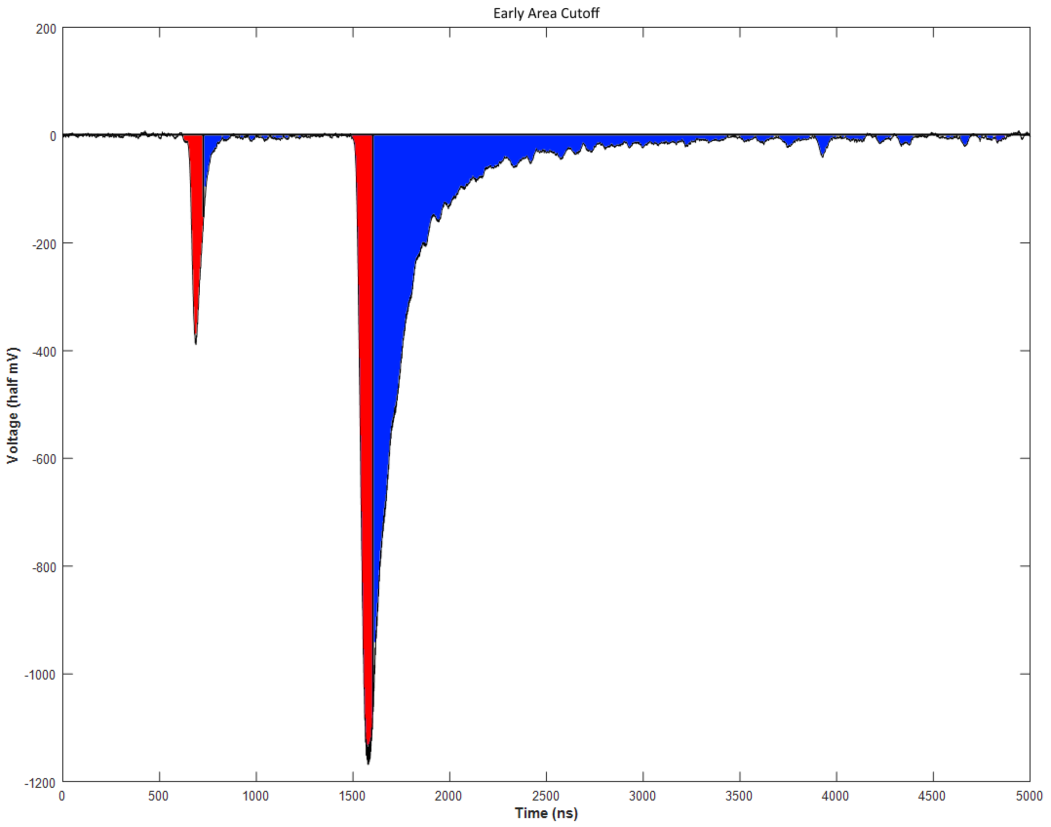


Figure 2.24 A waveform with an arbitrary early area cutoff marked. The area of the pulse in red would be considered early. The area in red and blue would be considered the total area.

For the data collected with a Hamamatsu PMT, the early area cutoff was set at 15 samples (15 ns) after the start of the pulse. For an Adit PMT, the cutoff should be chosen further into the pulse. The exact number can be chosen by inspecting the pulses in the data.

Time-of-flight data is calculated for all pulses except the first pulse in the waveform. Ideally, this would be actual time-of-flight (start gamma to stop recoil) or drift time (stop recoil to stop capture). However, it could also be the time between coincidental pulses, which is meaningless. Although this would be among the first places to improve the analysis process, time-of-flight histograms indicate that this method works sufficiently well (see Fig. 2.25).

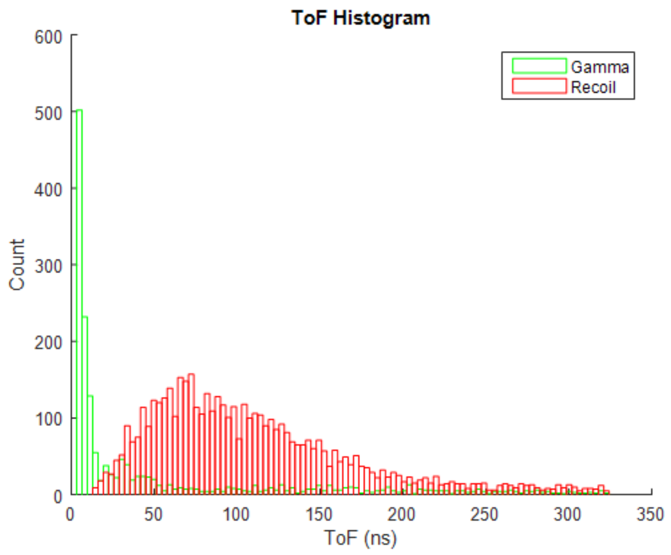


Figure 2.25 A histogram of time-of-flight values of all pulses identified as either gammas or recoils.

The time-of-flight information is used to do some initial pulse type identification. Gammas only take 3.3 ns to travel 1 m so any pulse with a time-of-flight in this region is either a gamma or a coincidental. The energy is then calculated from the time-of-flight. Time-of-flight and energy are each plotted against pulse area in an effort to find the proportionality constant for our scintillator. This calibrates the spectrometer for taking standalone energy measurements.

2.5.2 Pulse Shape Discrimination

In order to determine the proportionality constant *k* it is necessary to find the relationship between the area of pulses caused by recoil events and time-of-flight (see Fig. 2.26).

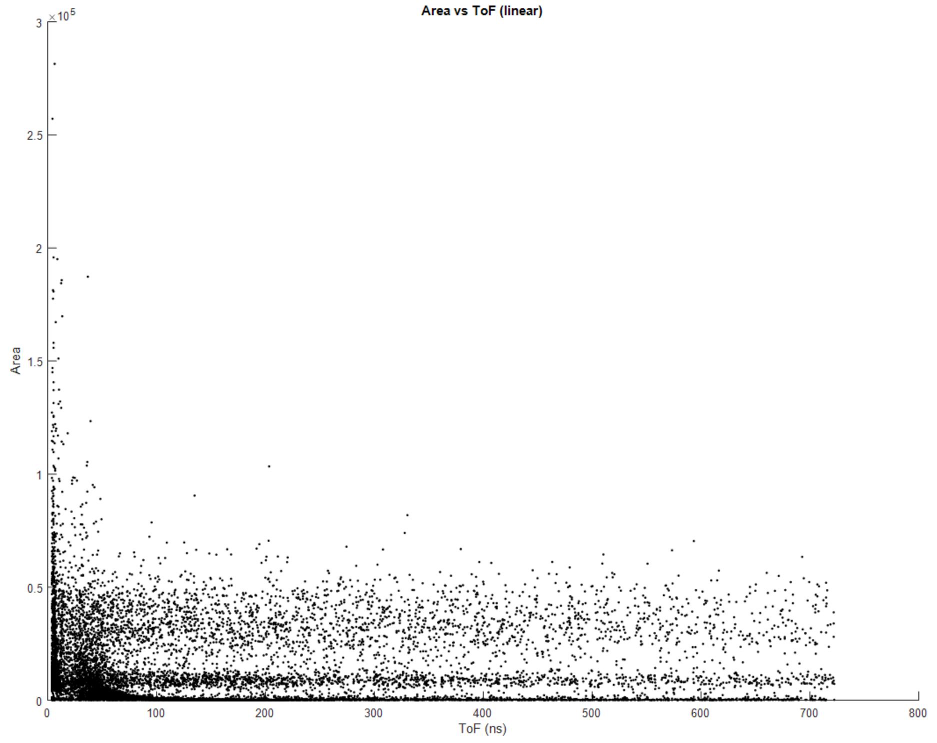


Figure 2.26 A plot of the area pulse metric versus time-of-flight. This plot includes all pulses from the stop detector except the first pulse from each event (since the first pulse in each event does not have a time-of-flight value).

Though some regions seem to appear in the area versus time-of-flight plot, it is difficult to recognize any useful correlations. This is because the plot contains all pulses except the first pulse in each waveform (since the first pulse serves as the start pulse for the first pair of pulses in each waveform). It is necessary to find some way to sort out the pulses caused by recoil events for analysis. This is done using the ratio of early area to total area compared to the total area of each pulse (see Fig. 2.27).

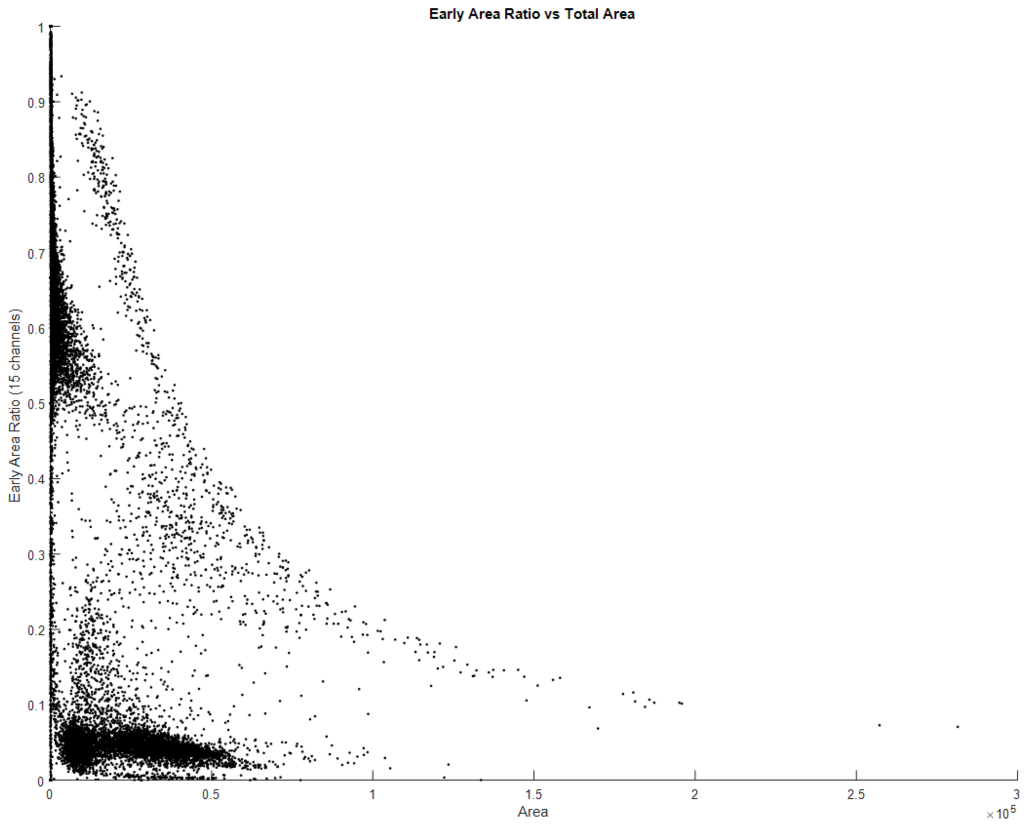


Figure 2.27 A plot of the ratio of early area to total area versus total area. The regions in this plot are useful for pulse shape discrimination.

The regions in this plot are much better defined and depend on the shape of the individual pulses. Capture pulses are very wide and have very low early area to total area ratios with a variety of total areas. There are two regions that fit these parameters. It is likely that one region contains capture in ^{6}Li and the other in ^{10}B. Recoils and gammas are very narrow and have much higher early area to total area ratios. Using these features it is possible to determine which types of interactions collect into each region. (see Fig. 2.28).

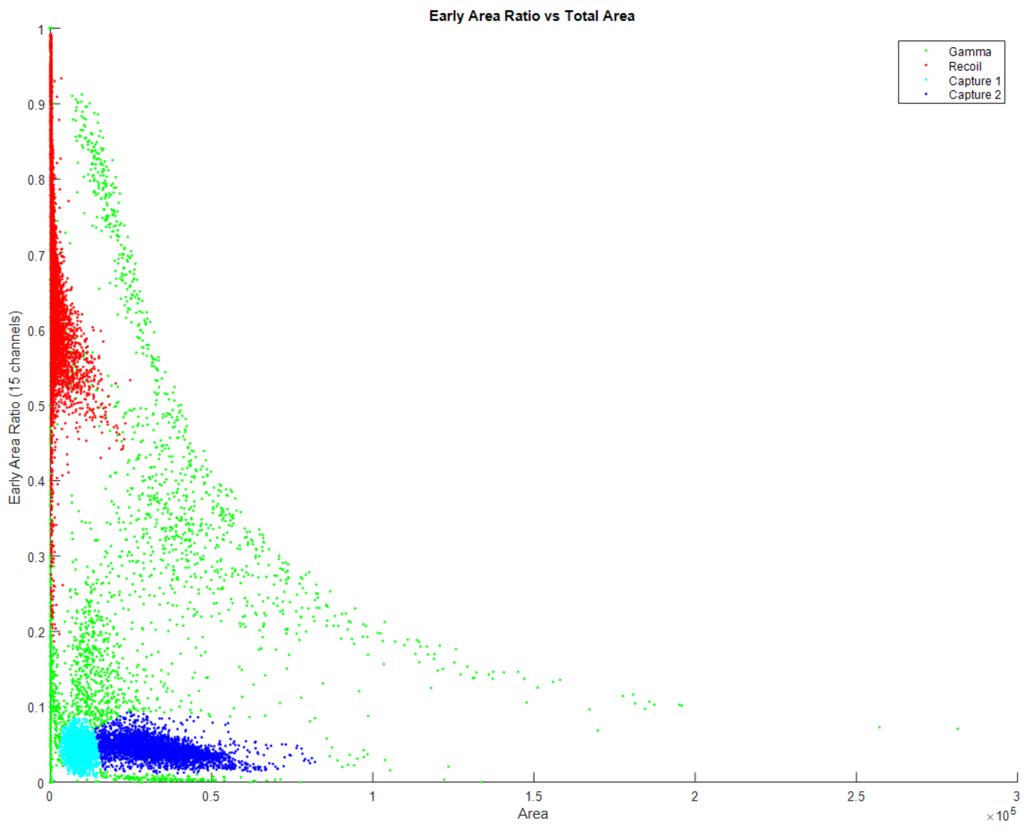


Figure 2.28 A plot of the ratio of early area to total area versus total area with regions selected. Events marked in red are labeled recoil events, blue are capture in ^{6}Li, and cyan are capture in ^{10}B (determined by energy considerations). The green is mostly gammas, but also includes other unidentified events such as muon interactions.

It is useful to plot each of the pulse metrics by themselves (in a histogram) after sorting in order to ensure the selected regions make physical sense. Once the pulses have been sorted using pulse shape discrimination, it is possible to sort the area versus time-of-flight plots by pulse type (see Fig. 2.29).

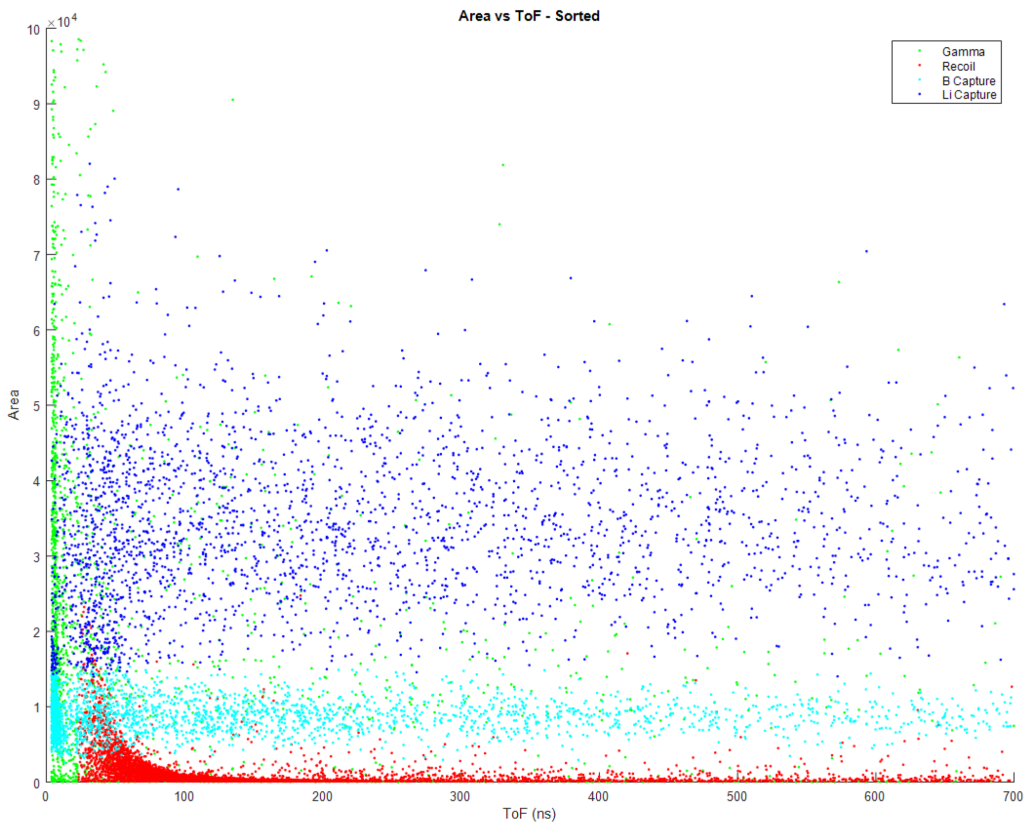


Figure 2.29 A plot of the area pulse metric versus time-of-flight sorted based on the pulse shape discrimination from the early area ratio plot regions.

Now the originally obscured features of Figure 2.26 become much more apparent. The pulses identified as gammas (green) are clustered together with very low time-of-flight with coincidental gammas scattered across the x-axis. The two capture regions (blue, ^{6}Li; and cyan, ^{10}B) with fairly narrow area distributions along the y-axis are spread across a wide range of time-of-flight, as expected. Finally, the recoils (red) begin to appear under all the other data. They can be easily extracted (see Fig. 2.30).

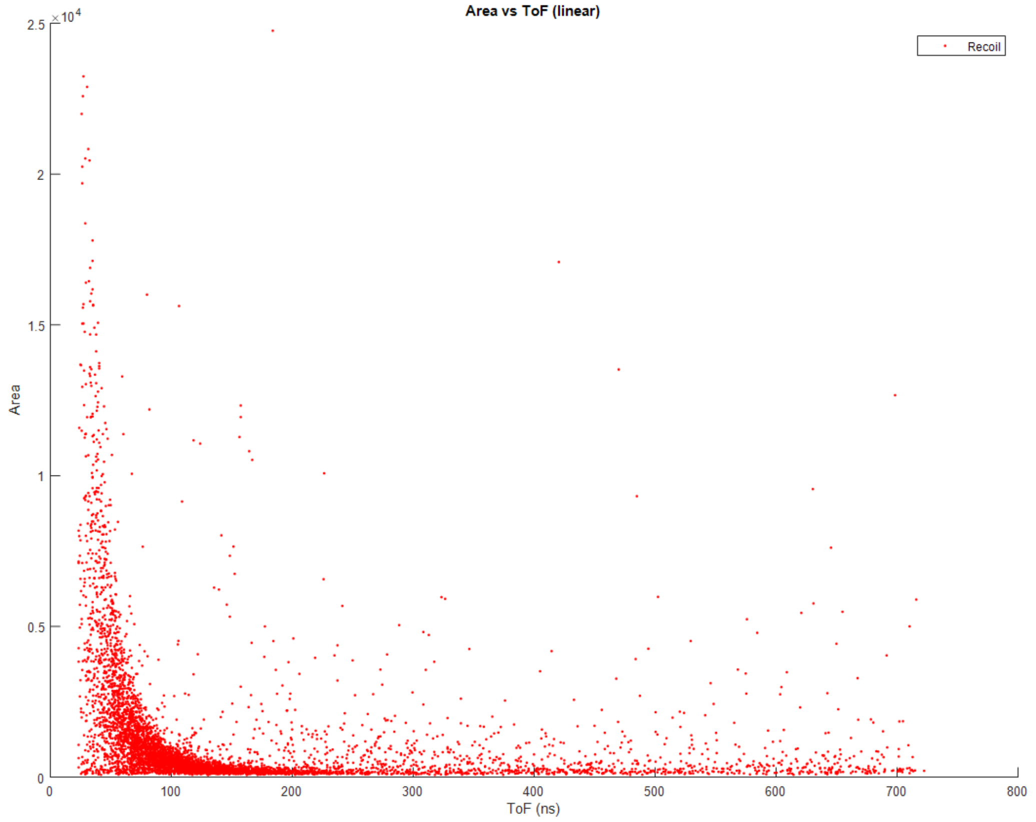


Figure 2.30 A plot of the area pulse metric versus time-of-flight containing only pulses labeled as recoil events based on the pulse shape discrimination from the early area ratio plot regions.

Here it becomes very obvious that there is a correlation between recoil pulse area and time-of-flight. Ideally, there should be only a thin, strong line following the top of the curve. We instead see a curve with a large amount of "rain" beneath it and a few scattered pulses above it. The pulses above the curve are likely explained by room-return neutrons.

We believe the rain to be a result of the non-linear response of the plastic scintillator at low energies and the spatial inconsistency of the Hamamatsu PMT. For a given energy there would be a distribution of pulse areas up to a maximum value depending on where the photon entered the face of the PMT. Several tests involving masking off parts of the PMT face did not seem to improve the rain effect at all. It may also be caused by edge effects if the neutrons are being detected around the outside edge of the PMT. We have not yet found a solution to this problem.

2.5.3 Room Return

We also made attempts to verify whether the random events were in fact caused by room return. Any neutrons that are not captured, either in a detector or otherwise, will bounce in a random direction, sometimes toward the detector. A neutron that is detected after traveling an indirect path to the detector will have a long time-of-flight that depends on the path it took, rather than its original energy. Additionally, some of its energy is deposited in whatever particle it recoiled off of so that the pulse area when it is finally detected is no longer correlated with its original energy. In order to test whether the randomly distributed points on the plot are caused by room return we took some data with a shadowbar between the source and the detector. The shadow bar is made of a hydrogenous moderator so that any neutrons traveling a direct path to the detector are absorbed before they reach the detector. Any events detected can be attributed to room return. (Note that the shadow bar itself would increase the rate of room return, but not by much.) The shadow bar data supports the explanation of the data above the curve (see Fig. 2.31).

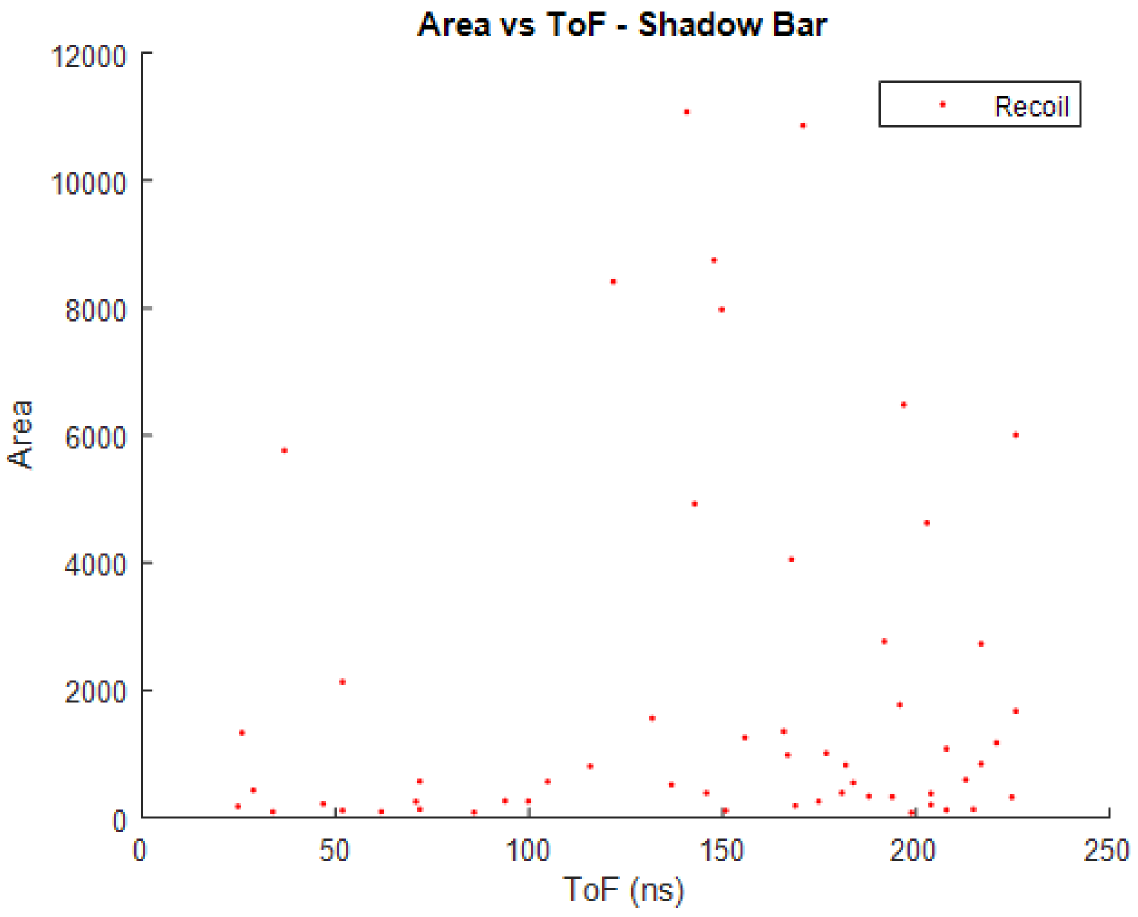


Figure 2.31 A plot of the area pulse metric versus time-of-flight. This data was taken with a shadow bar in place in an effort to measure only room return neutrons.

2.5.4 Fitting the Data

With the recoils now identified, it is possible to find the proportionality constants. A least squares regression is used to fit the area to energy relation to the data (see Fig. 2.32).

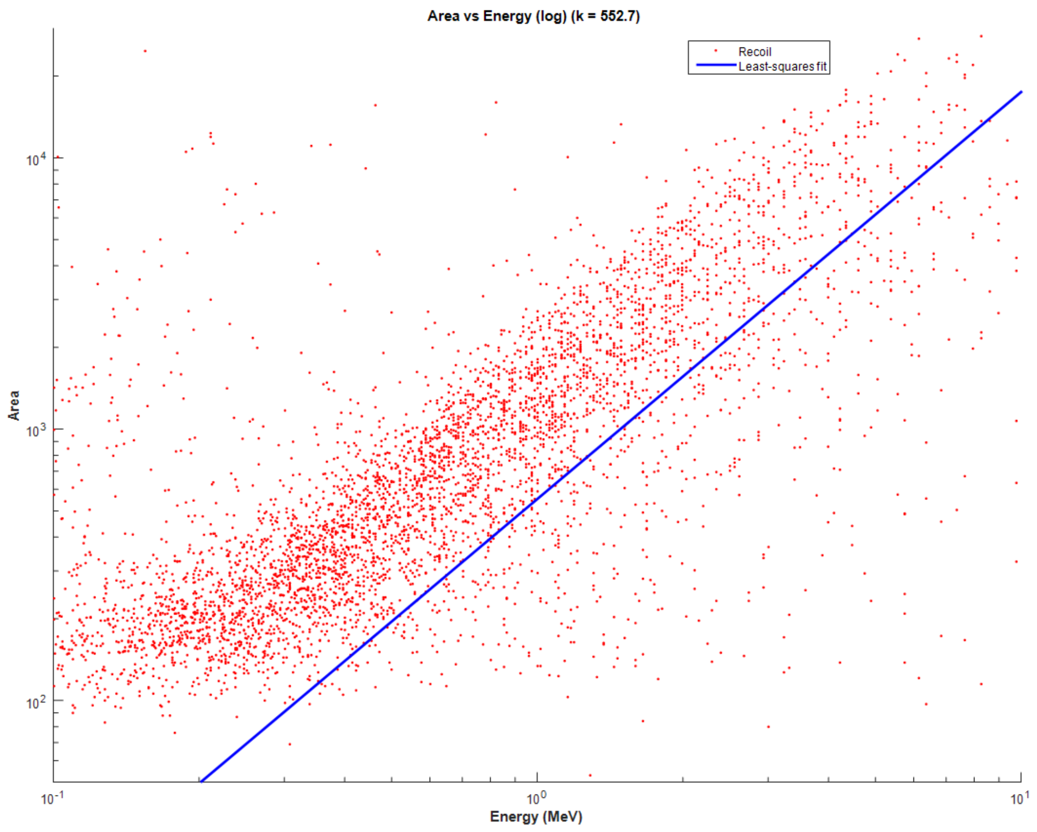


Figure 2.32 Log plot of area versus energy of proton recoil events. A least squares regression is used to find an initial value of k.

This fitting method gives a good initial guess for the value of k which is then tweaked by hand to more closely approximate the data (the least squares regression is strongly influenced by outliers) (see Fig. 2.33).

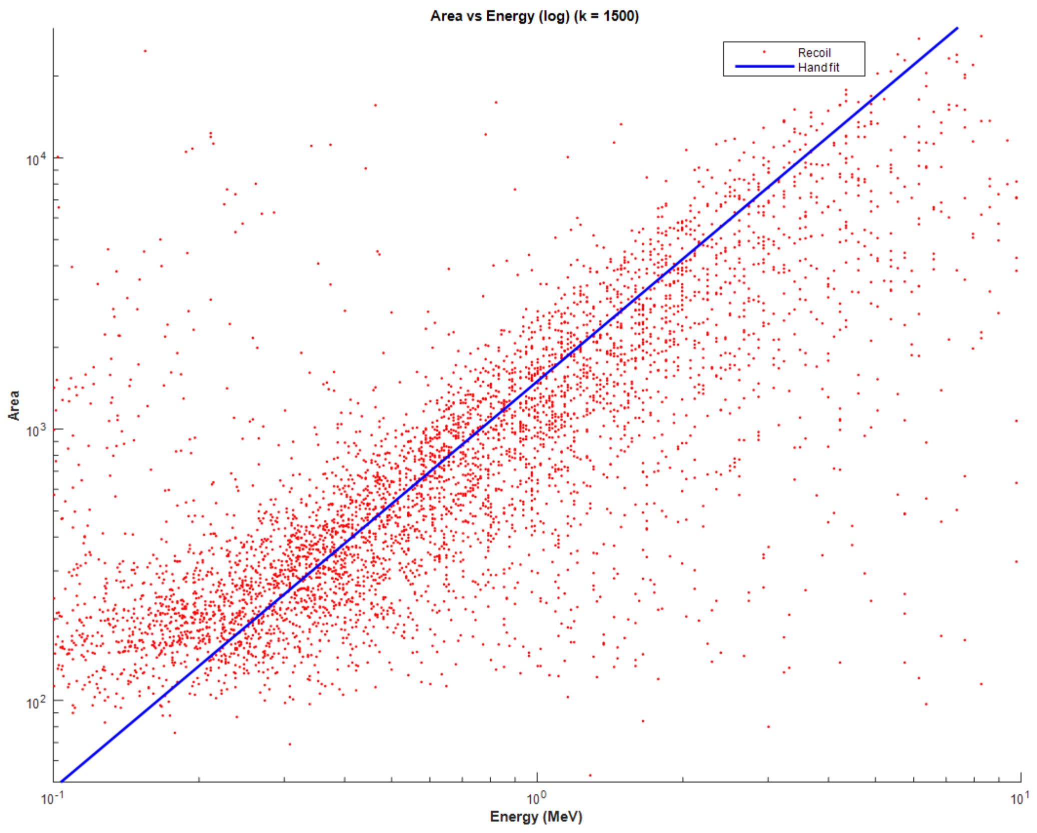


Figure 2.33 Log plot of area versus energy of proton recoil events. The value of *k* has been tweaked by hand to better fit the data.

Neutron energies can then be calculated from pulse areas values.

Chapter 3

Results

3.1 Energy Spectrum

The neutron energy spectrum calculated from pulse area approximates the spectrum for ^{252}Cf in the range of around 0.1 MeV to 3 MeV. For comparison, the actual neutron energy spectrum for ^{252}Cf is very closely approximated by a Watt distribution

$$f(E) = C \exp(-E/a) \sinh((bE)^{1/2})$$

where C is a constant of proportionality, E is neutron energy, a and b are constants specified by the isotope. For ^{252}Cf, $a = 1.025$ and $b = 2.926$ (X-5 Monte Carlo Team 2003) (see Fig. 3.1).

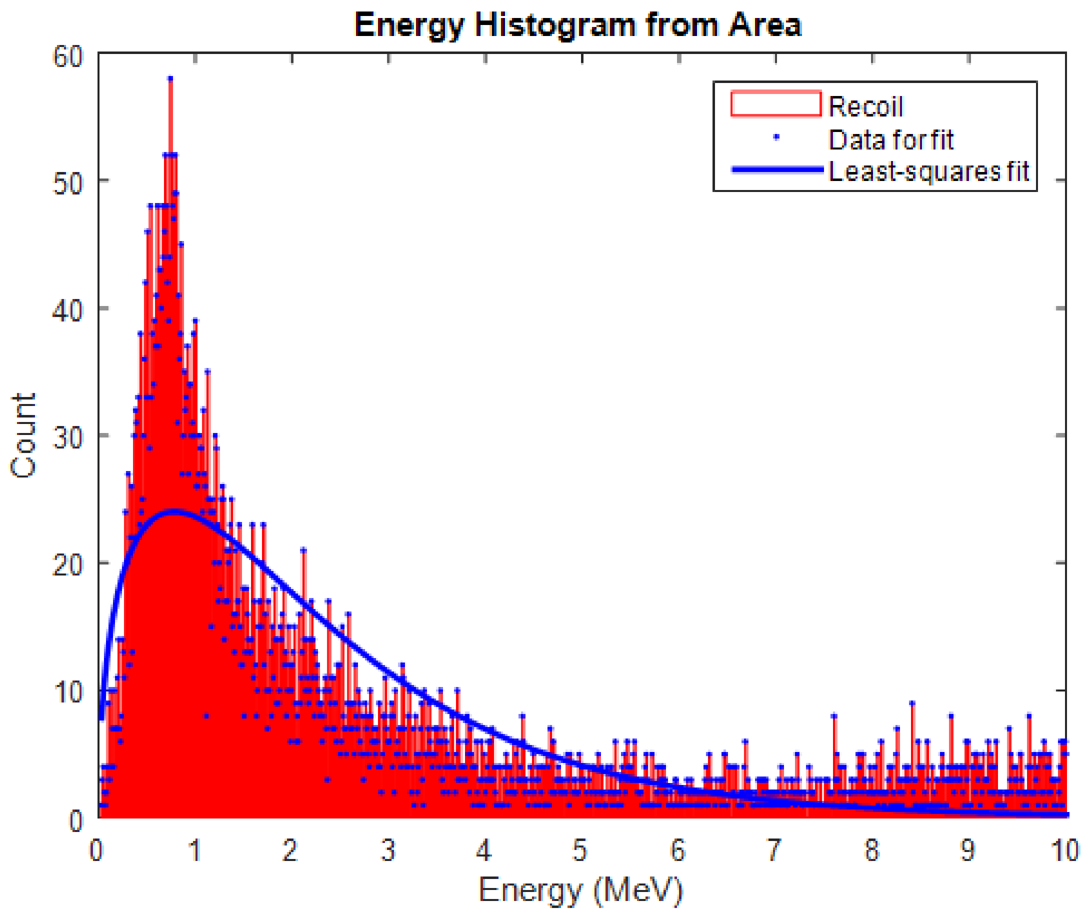


Figure 3.1 The neutron energy spectrum of ^{252}Cf calculated from proton recoil pulse areas. The Watt distribution approximating the neutron spectrum has been fit to the histogram using a non-linear least squares fitting method.

The energy distribution measured by the spectrometer has the right general shape and a peak near the most common neutron energy in the energy spectrum, but it does not fit the Watt distribution perfectly, especially for energies higher than 1 or 2 MeV. We believe this is due to limitations in the scintillator and photomultiplier tubes used. Our attempts to explain and correct these effects will be discussed in the next section. Since the detector was installed in a time-of-flight system, we can also compare the energy spectrum calculated from pulse area to the spectrum calculated from time-of-flight (see Fig. 3.2).

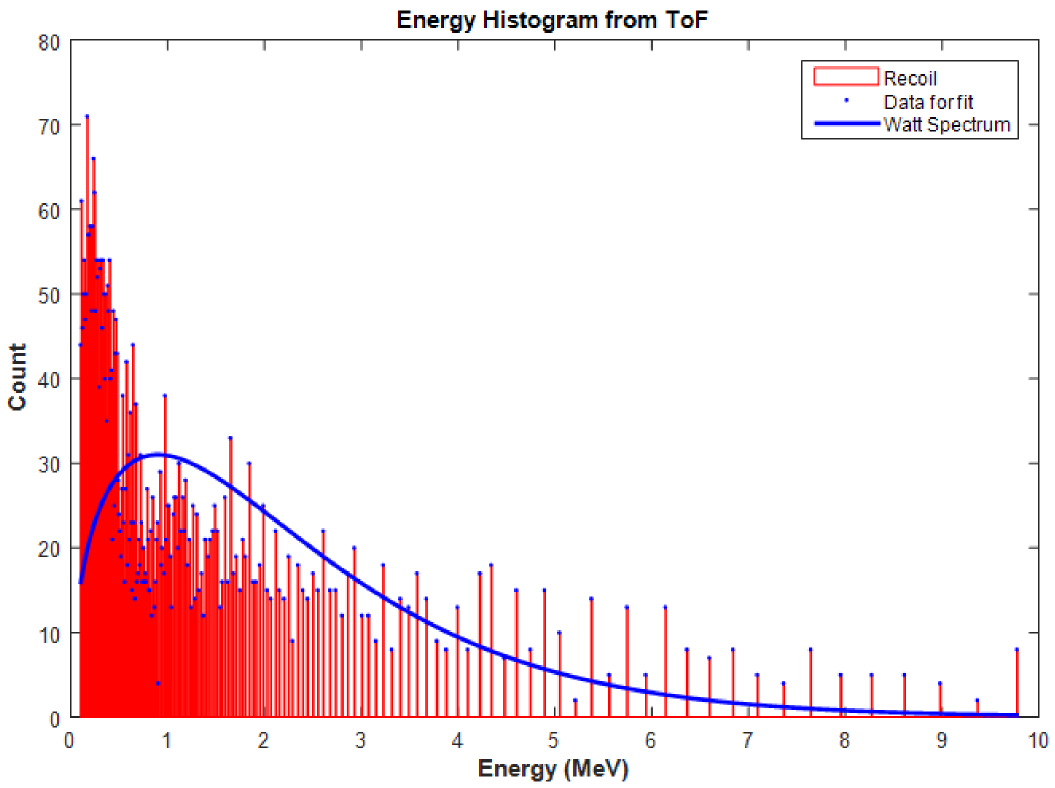


Figure 3.2 The neutron energy spectrum of ^{252}Cf calculated from time-of-flight. The Watt distribution approximating the neutron spectrum has been fit to the histogram using a non-linear least squares fitting method.

The energy histogram calculated from area actually seems to fit the predicted spectrum better than the spectrum calculated from time-of-flight. This is almost certainly due to poorly enforced timing windows for the time-of-flight calculations rather than any actual superiority of the spectrometer to time-of-flight data.

3.1.1 Corrections to the Energy Spectrum

There are at least two important effects that may explain the discrepancy between the calculated energy spectrum and the Watt distribution. First, the plastic scintillator we used does not have a linear energy to light output relationship. Light output is disproportionately higher for higher energy protons. However, if the non-linearity can be characterized, a correction can be applied to the energy spectrum. Second, the cross section for elastic scattering in hydrogen changes dramatically over the energy range of interest (see Fig. 3.3).

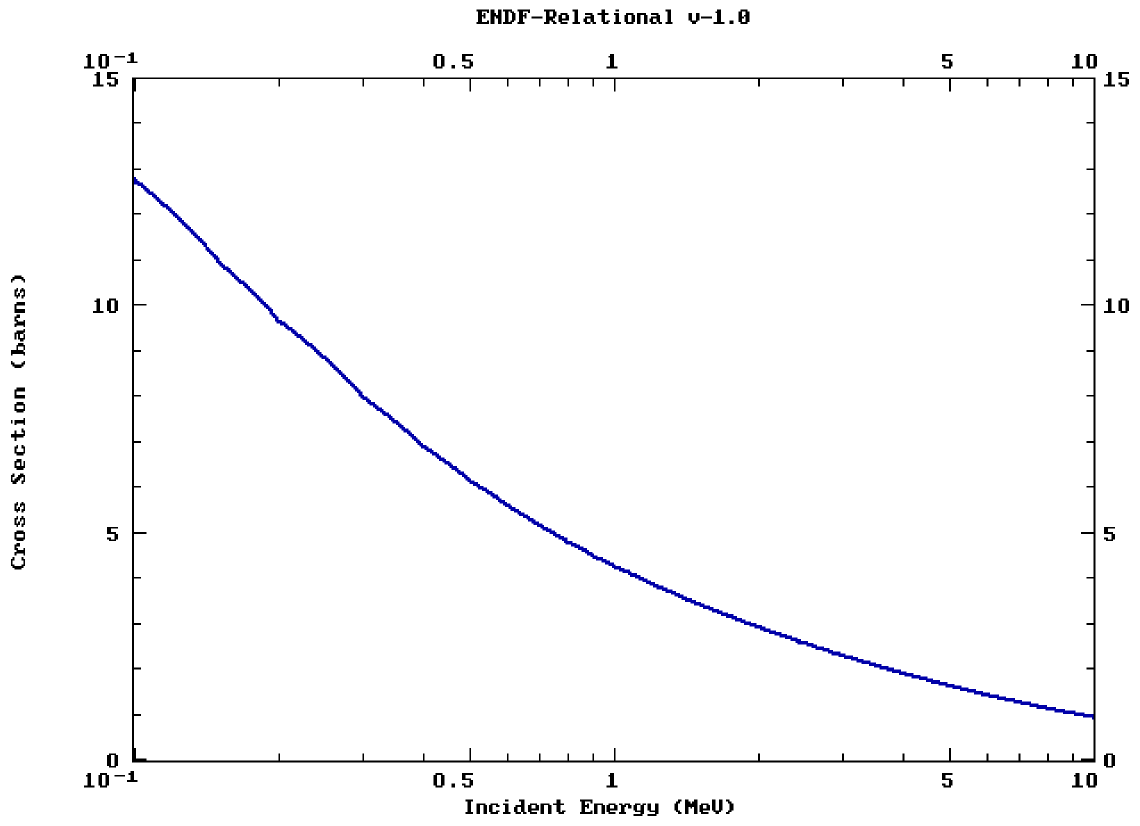


Figure 3.3 The total neutron cross section of ^{1}H versus energy (ENDF 2016).

As before, if the cross section information is known, we can correct the measured energy spectrum accordingly (see the analyzePulses function in Appendix D for a rough method of implementing these corrections). These corrections, along with corrections to flatten the PMT response, will hopefully improve the resolution of the spectrometer.

3.2 Applications of the Spectrometer

The detector shows some promise as a spectrometer in the energy range of around 0.1 MeV to 3 MeV, but requires additional work to become practically useful. This energy range includes the neutron energy produced by d-d fusion (2.45 MeV) and is thus useful for exploring the increased rates of d-d fusion in condensed matter. However, as stated before, the spread of measured area for a given neutron energy is currently too large for the detector to be a useful spectrometer (see Fig. 3.4).

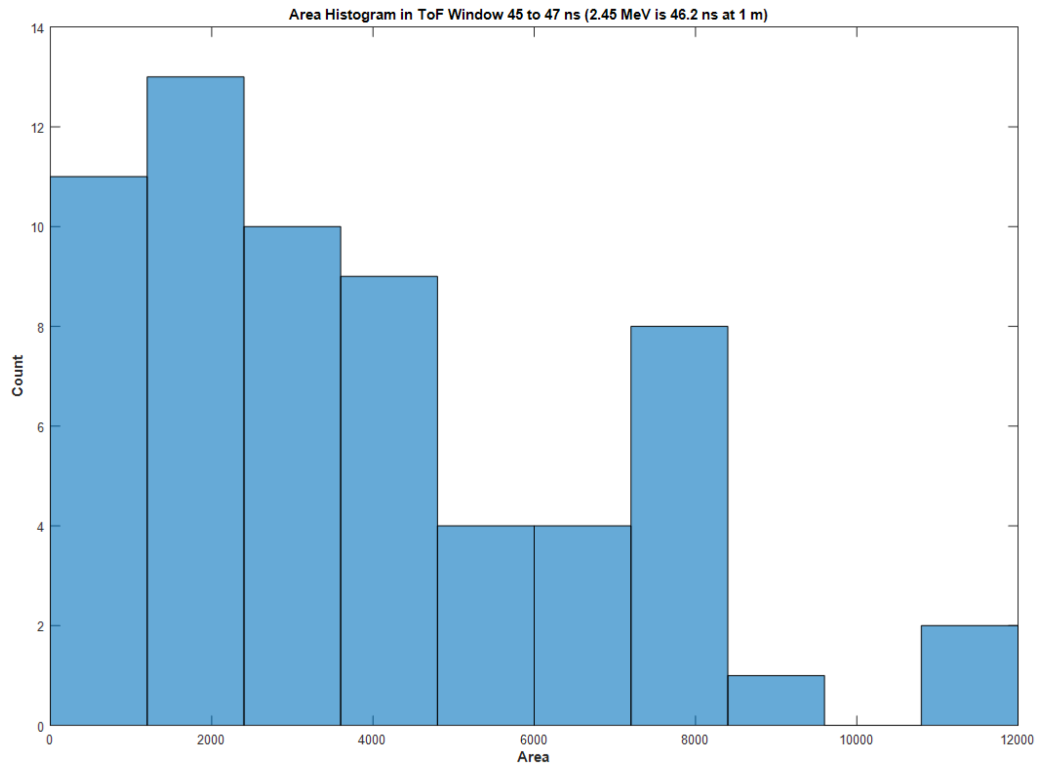


Figure 3.4 A histogram of measured pulse area for a narrow time-of-flight window, 45-47 ns. 2.45 MeV corresponds to 46.2 ns at 1 m.

Ideally, the histogram would have a narrow peak near the bin corresponding with the largest pulse area. Other effects, such as inconsistency in the PMT and non-linear light output in the plastic scintillator spread the peak down and need to be corrected for.

If such corrections can be made, the detector would perform a similar role for neutrons as the NaI detector for gammas. It would be very portable and easy to use at the expense of excellent resolving power. More portable than a time-of-flight system with higher resolution than a Bonner sphere system, this spectrometer would help to improve our neutron spectrometric capabilities.

3.3 Future Work

Some additional work is necessary to further improve the resolution and overall utility of the detector. Current photomultiplier tubes do not provide consistent photon to electron conversion across the face of the tube. This effect is significant: as much as a 55% variation for the Hamamatsu R1250 and greater than 60% for the Adit B133D01. If the spatial inconsistency is measured for the tube to be used with the detector, it may be possible to correct the inconsistency with a neutral density filter. It may be necessary to seek an alternative photon detection method.

Another opportunity for future work is improving the pulse shape discrimination capabilities of the detector and the software. Improvements in both scintillator technology and software methods will help to reduce misidentification of gammas as proton recoils and vice versa. The capabilities of this neutron spectrometer will continually improve with future developments in scintillator and photomultiplier tube technology and in software pulse shape discrimination methods.

Appendix A

Screening Potential Table

Table A.1 Screening potentials of d-d fusion in deuterated foils of several metals.

Target	U_e (eV)	Source	Year
$SrD_{1.0}$	350-800	Huke, Phys. Rev. C 78, 015803	2008
Pd	800	Bonomo, Nucl. Phys. A 719, 37	2003
Pd	800	Raoila, Eur. Phys. J. A 19, 283	2004
PtD_x	730	Bonomo, Nucl. Phys. A 719, 37	2003
Sb	720	Raoila, Eur. Phys. J. A 19, 283	2004
Pt	680	Rolfs, Prog. Theor. Phys. Supplement 154, 373	2004
Pt	675	Raiola, Eur. Phys. J. A 27, 79	2006
Pt	670	Raoila, Eur. Phys. J. A 19, 283	2004
Co	640	Bonomo, Nucl. Phys. A 719, 37	2003
Co	640	Raiola, Eur. Phys. J. A 27, 79	2006
Co	640	Raoila, Eur. Phys. J. A 19, 283	2004
PdO	600	Kasagi, J. Phys. Soc. Jpn. 71, 2881	2002
PdO	600	Kasagi, Surf. Coat. Tech. 201, 8574	2007

Table A.1 cont.

Target	U_e (eV)	Source	Year
Tl	550	Bonomo, Nucl. Phys. A 719, 37	2003
Tl	550	Raoila, Eur. Phys. J. A 19, 283	2004
Li_l	543	Toriyabe, Phys. Rev. C 85, 054620	2012
Bi	540	Raoila, Eur. Phys. J. A 19, 283	2004
Pt	530	Raiola, Eur. Phys. J. A 27, 79	2006
Pt	530	Raiola, Eur. Phys. J. A 27, 79	2006
Al	520	Bonomo, Nucl. Phys. A 719, 37	2003
Al	520	Raoila, Eur. Phys. J. A 19, 283	2004
In	520	Raoila, Eur. Phys. J. A 19, 283	2004
Ba	490	Raoila, Eur. Phys. J. A 19, 283	2004
Co	480	Raiola, Eur. Phys. J. A 27, 79	2006
Pb	480	Raoila, Eur. Phys. J. A 19, 283	2004
Pt	480	Raiola, Eur. Phys. J. A 27, 79	2006
V	480	Raoila, Eur. Phys. J. A 19, 283	2004
Zn	480	Bonomo, Nucl. Phys. A 719, 37	2003
Zn	480	Raoila, Eur. Phys. J. A 19, 283	2004
Cu	470	Raoila, Eur. Phys. J. A 19, 283	2004
LiF	470	Engstler, Z. Phys. A 342, 471	1992
Nb	470	Raoila, Eur. Phys. J. A 19, 283	2004
Pt	465	Raiola, Eur. Phys. J. A 27, 79	2006
Fe	460	Raoila, Eur. Phys. J. A 19, 283	2004
Fe	450	Bonomo, Nucl. Phys. A 719, 37	2003
Ni	450	Bonomo, Nucl. Phys. A 719, 37	2003

Table A.1 cont.

Target	U_e (eV)	Source	Year
H_2/D_{2g}	440	Engstler, Z. Phys. A 342, 471	1992
Mg	440	Bonomo, Nucl. Phys. A 719, 37	2003
Mg	440	Raoila, Eur. Phys. J. A 19, 283	2004
Pb	440	Bonomo, Nucl. Phys. A 719, 37	2003
Pt	440	Bonomo, Nucl. Phys. A 719, 37	2003
Mo	420	Raoila, Eur. Phys. J. A 19, 283	2004
Re	420	Bonomo, Nucl. Phys. A 719, 37	2003
Nb	400	Bonomo, Nucl. Phys. A 719, 37	2003
Cd	390	Bonomo, Nucl. Phys. A 719, 37	2003
Mn	390	Raoila, Eur. Phys. J. A 19, 283	2004
LiF	380	Engstler, Z. Phys. A 342, 471	1992
Ni	380	Raoila, Eur. Phys. J. A 19, 283	2004
Hf	370	Raiola, Eur. Phys. J. A 27, 79	2006
Cd	360	Raoila, Eur. Phys. J. A 19, 283	2004
Er	360	Raiola, Eur. Phys. J. A 27, 79	2006
Mn	350	Bonomo, Nucl. Phys. A 719, 37	2003
V	350	Bonomo, Nucl. Phys. A 719, 37	2003
Dy	340	Raiola, Eur. Phys. J. A 27, 79	2006
Gd	340	Raiola, Eur. Phys. J. A 27, 79	2006
Ta	340	Bonomo, Nucl. Phys. A 719, 37	2003
$TaD_{0.13}$	340	Bonomo, Nucl. Phys. A 719, 37	2003
Tb	340	Raiola, Eur. Phys. J. A 27, 79	2006
Ag	330	Raoila, Eur. Phys. J. A 19, 283	2004

Table A.1 cont.

Target	U_e (eV)	Source	Year
H_2/D_{2g}	330	Engstler, Z. Phys. A 342, 471	1992
TaD	322	Czerski, Nucl. Instrum. Methods Phys. Res. B 193, 183	2002
$TaD_{0.9}$	322	Huke, Phys. Rev. C 78, 015803	2008
Cr	320	Raoila, Eur. Phys. J. A 19, 283	2004
Sc	320	Raiola, Eur. Phys. J. A 27, 79	2006
Y	320	Bonomo, Nucl. Phys. A 719, 37	2003
ZrD_2	319	Czerski, J. Phys. G: Nucl. Part. Phys. 35 014012	2008
Sm	314	Raiola, Eur. Phys. J. A 27, 79	2006
$Pd_{0.3}$	313	Huke, Phys. Rev. C 78, 015803	2008
$TaD_{0.5}$	313	Bystritsky, Nuc. Phys. A 889, 93-104	2012
LiF	310	Wang, J. Phys. G: Nucl. Part. Phys. 39, 015201	2012
Pd	310	Kasagi, J. Phys. Soc. Jpn. 71, 2881	2002
Pd	310	Kasagi, Surf. Coat. Tech. 201, 8574	2007
$TaD_{0.13}$	309	Raiola, Eur. Phys. J. A 13, 377	2002
TaD	302	Czerski, Europhys. Lett. 68, 363	2004
H_2/D_{2g}	300	Engstler, Z. Phys. A 342, 471	1992
LiF	300	Engstler, Z. Phys. A 342, 471	1992
ZrD_2	297	Czerski, Eur. Phys. J. A 27, 83	2006
ZrD_2	297	Czerski, Nucl. Instrum. Methods Phys. Res. B 193, 183	2002
$ZrD_{2.1}$	297	Huke, Phys. Rev. C 78, 015803	2008
$PdD_{0.2}$	296	Czerski, Europhys. Lett. 68, 363	2004
Ti	295	Raiola, Eur. Phys. J. A 27, 79	2006
$TiD_{0.23}$	295	Raiola, Eur. Phys. J. A 27, 79 (and references therein)	2006

Table A.1 cont.

Target	U_e (eV)	Source	Year
ZrD_2	295	Czerski, Europhys. Lett. 68, 363	2004
Ti	290	Raiola, Eur. Phys. J. A 27, 79	2006
Au	280	Raoila, Eur. Phys. J. A 19, 283	2004
Ta	270	Raoila, Eur. Phys. J. A 19, 283	2004
$TaD_{0.13}$	270	Raiola, Eur. Phys. J. A 19, 283	2004
Y	270	Raiola, Eur. Phys. J. A 27, 79	2006
Lu	265	Raiola, Eur. Phys. J. A 27, 79	2006
Tm	260	Raiola, Eur. Phys. J. A 27, 79	2006
Ti	250	Raiola, Eur. Phys. J. A 27, 79	2006
$TiD_{0.26}$	250	Raiola, Eur. Phys. J. A 27, 79 (and references therein)	2006
W	250	Raoila, Eur. Phys. J. A 19, 283	2004
La	245	Raiola, Eur. Phys. J. A 27, 79	2006
^6Li_l	235	Fang, J. Phys. Soc. Jpn. 80, 084201	2011
Re	230	Raoila, Eur. Phys. J. A 19, 283	2004
Rh	230	Bonomo, Nucl. Phys. A 719, 37	2003
Rh	230	Raoila, Eur. Phys. J. A 19, 283	2004
Cr	220	Bonomo, Nucl. Phys. A 719, 37	2003
Mo	220	Bonomo, Nucl. Phys. A 719, 37	2003
Ru	220	Bonomo, Nucl. Phys. A 719, 37	2003
W	220	Bonomo, Nucl. Phys. A 719, 37	2003
H_2/D_{2g}	218	Wang, J. Phys. G: Nucl. Part. Phys. 39, 015201	2012
Ru	215	Raoila, Eur. Phys. J. A 19, 283	2004
Sr	210	Raoila, Eur. Phys. J. A 19, 283	2004

Table A.1 cont.

Target	U_e (eV)	Source	Year
Zr	205	Raiola, Eur. Phys. J. A 27, 79	2006
$ZrD_{0.13}$	205	Raiola, Eur. Phys. J. A 27, 79	2006
ZrD_2	205	Bystritsky, Nucl. Phys. A 889 93-104	2012
Ce	200	Raiola, Eur. Phys. J. A 27, 79	2006
Fe	200	Kasagi, J. Phys. Soc. Jpn. 71, 2881	2002
Fe	200	Kasagi, Surf. Coat. Tech. 201, 8574	2007
Ir	200	Bonomo, Nucl. Phys. A 719, 37	2003
Ir	200	Raoila, Eur. Phys. J. A 19, 283	2004
Re	200	Kasagi, Surf. Coat. Tech. 201, 8574	2007
Sn	200	Bonomo, Nucl. Phys. A 719, 37	2003
AlD	191	Czerski, Europhys. Lett. 68, 363	2004
AlD	190	Czerski, Nucl. Instrum. Methods Phys. Res. B 193, 183	2002
$AlD_{0.8}$	190	Huke, Phys. Rev. C 78, 015803	2008
Nd	190	Raiola, Eur. Phys. J. A 27, 79	2006
Be	180	Bonomo, Nucl. Phys. A 719, 37	2003
Be	180	Raoila, Eur. Phys. J. A 19, 283	2004
Ho	165	Raiola, Eur. Phys. J. A 27, 79	2006
Li	150	Huke, Phys. Rev. C 78, 015803	2008
^7Li_l	140	Fang, J. Phys. Soc. Jpn. 80, 084201	2011
TaD	136	Czerski, Eur. Phys. J. A 27, 83	2006
Sn	130	Raoila, Eur. Phys. J. A 19, 283	2004
TiD_2	125	Bystritsky, Nuc. Phys. A 889, 93-104	2012
Cu	120	Kasagi, Surf. Coat. Tech. 201, 8574	2007

Table A.1 cont.

Target	U_e (eV)	Source	Year
Eu	120	Raiola, Eur. Phys. J. A 27, 79	2006
ZrD_2	112	Czerski, Eur. Phys. J. A 27, 83	2006
Yb	110	Raiola, Eur. Phys. J. A 27, 79	2006
$TiD_{1.3}$	100	Czerski, Eur. Phys. J. A 27, 83	2006
HfD_x	87	Bonomo, Nucl. Phys. A 719, 37	2003
ZrD_x	83	Bonomo, Nucl. Phys. A 719, 37	2003
Yb	81	Yuki, J. Phys. G: Nucl. Part. Phys. 23, 1459	1997
Ge	80	Raoila, Eur. Phys. J. A 19, 283	2004
Ni	80	Kasagi, Surf. Coat. Tech. 201, 8574	2007
Yb	80	Kasagi, Surf. Coat. Tech. 201, 8574	2007
PrD_x	78	Bonomo, Nucl. Phys. A 719, 37	2003
Au	70	Kasagi, J. Phys. Soc. Jpn. 71, 2881	2002
Au	70	Kasagi, Surf. Coat. Tech. 201, 8574	2007
Ho	70	Raoila, Eur. Phys. J. A 19, 283	2004
Pr	70	Raoila, Eur. Phys. J. A 19, 283	2004
Tm	70	Raoila, Eur. Phys. J. A 19, 283	2004
Y	70	Raoila, Eur. Phys. J. A 19, 283	2004
$TiD_{3.76}$	66	Kasagi, J. Phys. Soc. Jpn. 71, 2881	2002
Ti	65	Kasagi, J. Phys. Soc. Jpn. 71, 2881	2002
Ti	65	Kasagi, Surf. Coat. Tech. 201, 8574	2007
Au	61	Bonomo, Nucl. Phys. A 719, 37	2003
C	60	Raoila, Eur. Phys. J. A 19, 283	2004
Ge	60	Bonomo, Nucl. Phys. A 719, 37	2003

Table A.1 cont.

Target	U_e (eV)	Source	Year
La	60	Raoila, Eur. Phys. J. A 19, 283	2004
Si	60	Raoila, Eur. Phys. J. A 19, 283	2004
C	52	Bonomo, Nucl. Phys. A 719, 37	2003
C	50	Raiola, Eur. Phys. J. A 27, 79	2006
CaO_2	50	Raoila, Eur. Phys. J. A 19, 283	2004
DyD_x	50	Bonomo, Nucl. Phys. A 719, 37	2003
Er	50	Raoila, Eur. Phys. J. A 19, 283	2004
ErD_x	50	Bonomo, Nucl. Phys. A 719, 37	2003
Eu	50	Raoila, Eur. Phys. J. A 19, 283	2004
Gd	50	Raoila, Eur. Phys. J. A 19, 283	2004
Ti	50	Raiola, Eur. Phys. J. A 27, 79	2006
$TiD_{1.1}$	50	Raiola, Eur. Phys. J. A 27, 79 (and references therein)	2006
Si	45	Bonomo, Nucl. Phys. A 719, 37	2003
Cu	43	Bonomo, Nucl. Phys. A 719, 37	2003
Lu	40	Raoila, Eur. Phys. J. A 19, 283	2004
Yb	40	Raoila, Eur. Phys. J. A 19, 283	2004
YbD_x	40	Bonomo, Nucl. Phys. A 719, 37	2003
Zr	40	Raoila, Eur. Phys. J. A 19, 283	2004
$ZrD_{1.1}$	40	Raiola, Eur. Phys. J. A 19, 283	2004
Al_2O_3	30	Bonomo, Nucl. Phys. A 719, 37	2003
Al_2O_3	30	Raoila, Eur. Phys. J. A 19, 283	2004
B	30	Bonomo, Nucl. Phys. A 719, 37	2003
B	30	Raoila, Eur. Phys. J. A 19, 283	2004

Table A.1 cont.

Target	U_e (eV)	Source	Year
BeO	30	Bonomo, Nucl. Phys. A 719, 37	2003
BeO	30	Raoila, Eur. Phys. J. A 19, 283	2004
Ce	30	Raoila, Eur. Phys. J. A 19, 283	2004
Dy	30	Raoila, Eur. Phys. J. A 19, 283	2004
Hf	30	Raoila, Eur. Phys. J. A 19, 283	2004
Nd	30	Raoila, Eur. Phys. J. A 19, 283	2004
Sc	30	Raoila, Eur. Phys. J. A 19, 283	2004
ScD_x	30	Bonomo, Nucl. Phys. A 719, 37	2003
Sm	30	Raoila, Eur. Phys. J. A 19, 283	2004
SmD_x	30	Bonomo, Nucl. Phys. A 719, 37	2003
Tb	30	Raoila, Eur. Phys. J. A 19, 283	2004
Ti	30	Raiola, Eur. Phys. J. A 27, 79	2006
Ti	30	Raoila, Eur. Phys. J. A 19, 283	2004
$TiD_{1.3}$	30	Raiola, Eur. Phys. J. A 19, 283	2004
TiD_x	30	Bonomo, Nucl. Phys. A 719, 37	2003
Ag	23	Bonomo, Nucl. Phys. A 719, 37	2003
Ti	19	Yuki, J. Phys. G: Nucl. Part. Phys. 23, 1459	1997
C	0	Huke, Phys. Rev. C 78, 015803	2008
CD	-20	Czerski, Nucl. Instrum. Methods Phys. Res. B 193, 183	2002

End of Table A.1

Appendix B

Datasheets

EJ-200 PLASTIC SCINTILLATOR

This plastic scintillator combines the two important properties of long optical attenuation length and fast timing and is therefore particularly useful for time-of-flight systems using scintillators greater than one meter long. Typical measurements of 4 meter optical attenuation length are achieved in strips of cast sheet in which a representative size is 2 cm x 20 cm x 300 cm.

The combination of long attenuation length, high light output and an emission spectrum well matched to the common photomultipliers recommends EJ-200 as the detector of choice for many industrial applications such as gauging and environmental protection where high sensitivity of signal uniformity are critical operating requirements.

Physical and Scintillation Constants:

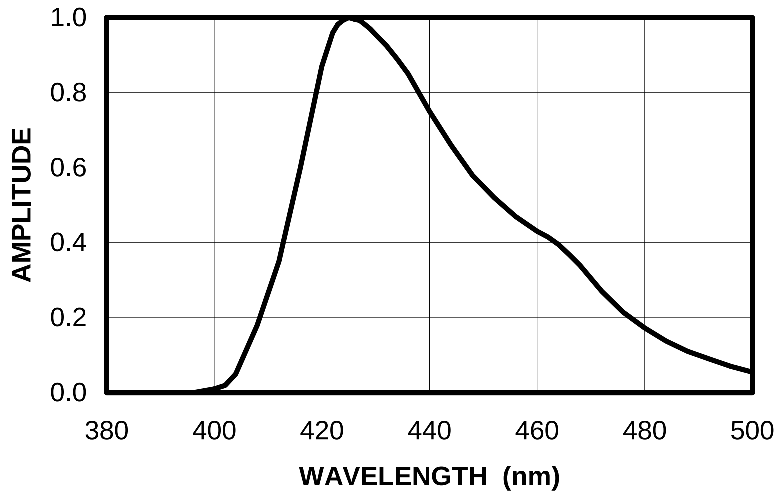
Light Output, % Anthracene	64
Scintillation Efficiency, photons/1 MeV e^-	10,000
Wavelength of Max. Emission, nm	425
Rise Time, ns	0.9
Decay Time, ns	2.1
Pulse Width, FWHM, ns	~2.5
No. of H Atoms per cm^3, x 10^{22}	5.17
No. of C Atoms per cm^3, x 10^{22}	4.69
No. of Electrons per cm^3, x 10^{23}	3.33
Density, g/cc:	1.023

Polymer Base: …………. Polyvinyltoluene
Refractive Index: ……….1.58
Vapor Pressure: ……….. Is vacuum-compatible
Coefficient of Linear Expansion: ……………… 7.8 x 10^{-5} below +67°C

Light Output vs. Temperature:
At +60°C, L.O. = 95% of that at +20°C
No change from +20°C to -60°C

Chemical Compatibility: Is attacked by aromatic solvents, chlorinated solvents, ketones, solvent bonding cements, etc. It is stable in water, dilute acids and alkalis, lower alcohols and silicone greases. It is safe to use most epoxies and "super glues" with EJ-200.

EJ-200 EMISSION SPECTRUM



ELJEN TECHNOLOGY
1300 W. Broadway
Sweetwater TX 79556 USA

Tel: (325) 235-4276 or (888) 800-8771
Fax: (325) 235-0701
Website: www.eljentechnology.com

B133D01 Photomultiplier Tube

The B133D01 is a 5" diameter 10-stage end-on photomultiplier with extended sensitivity in the blue, green and red. Designed for scintillation counting and other applications where high quantum efficiency, low dark current, good collection efficiency, and gain stability are of paramount importance.

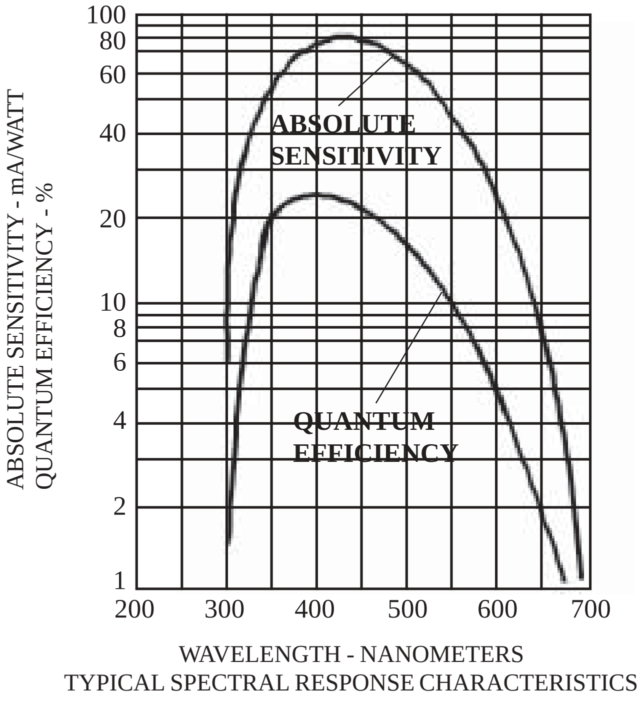
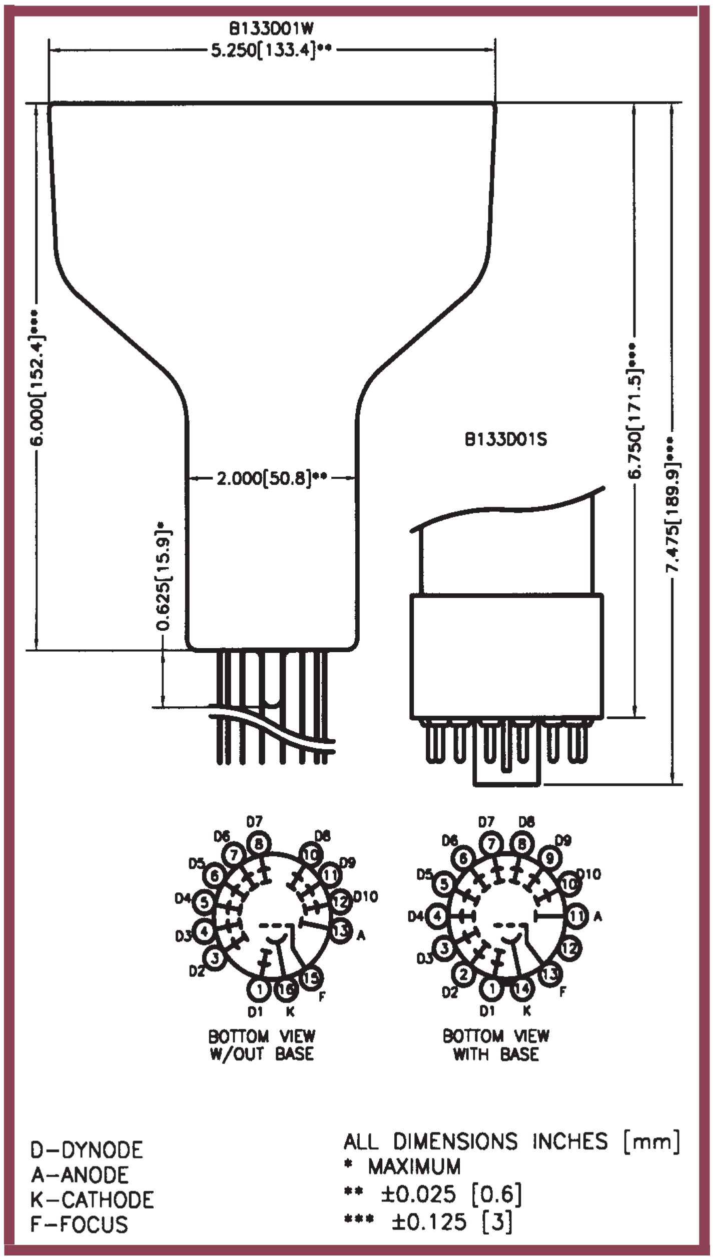


FIGURE 1



Photocathode: Semitransparent Extended Bialkali	
Spectral Response	See Figure 1
Wavelength of maximum response	470 ± 50 nm
Minimum diameter	119.38 mm
Window shape	plano-plano, circular
Window index of refraction @ 436 nm	1.523
Dynodes	BeCu, Box & Grid
Capacitance (anode to all electrodes)	9.5 pF
Operating position	Any
Weight	655 grams

Rev. 01/04

B133D01 Photomultiplier Tube

ELECTRICAL OPERATING RATINGS

	MINIMUM	TYPICAL	MAXIMUM[5]	UNITS
Cathode to dynode No. 1 voltage	40	150	300	VDC
Cathode to anode voltage		1100	1500	VDC
Voltage between consecutive dynodes			100	VDC
Ambient storage temperature		23	60	°C
Anode current, average over 30 sec.			1.0	µA
Cathode current		1	5	µA
Cathode luminous sensitivity:[1]				
With 2854° K tungsten source	80	120	180	µA/lm
With blue light source[2]	5	12	15	µA/lm(B)
With red light source[3]	5	10	15	µA/lm(R)
Quantum efficiency @ 420 nm		25		%
Cathode radiant sensitivity @ 420 nm		97		mA/W
@ 540 nm		45		mA/W
@ 600 nm		25		mA/W
@ 680 nm		4		mA/W
Anode luminous sensitivity1100 VDC: With 2854° K tungsten source of 1 x 10^{-3} lm	3	20	50	A/lm
Current amplification @1100 VDC		1 X 10^6		
Anode dark current [4] @ 22° C	1	10	20	nA

(1) With 150 VDC between cathode and all other elements connected as anode.

(2) This measurement is made with a blue filter (Corning CS-5-58, 1/2 stock thickness) interposed between a calibrated 2854° K tungsten light source and the photocathode. The (B) appearing in the units signifies that the measurement is made with the blue filter in place.

(3) This measurement is made with a red filter (Corning CS-2-62) interposed between a calibrated 2854° K tungsten light source and the photocathode. The (R) appearing in the units signifies that the measurement is made with the red filter in place.

(4) Measured at the supply voltage which gives an anode sensitivity of 20 A/lm

(5) Recommended operating maximums.

NOTE: When ordering one of the following basing options must be added, i.e. B133D01S

BASING OPTIONS: L - Long Base S - Short Base W - Wire Leads (No Base)

Voltage dividers available made to customer specifications.

ADIT

P.O. Box 870 • 300 Crane • Sweetwater, Texas 79556
800-399-4557 • 915-235-1418 • Fax: 915-235-2872 • E-mail: adit@aditpmt.com • Website: www.aditpmt.com

HAMAMATSU
PHOTON IS OUR BUSINESS

PHOTOMULTIPLIER TUBE R1250

For High Energy Physics, Fast Time Response, High Pulse Linearity
127 mm (5 Inch) Diameter, Bialkali Photocathode, 14-Stage, Head-on Type

GENERAL

Parameter		Description	Unit
Spectral Response		300 to 650	nm
Wavelength of Maximum Response		420	nm
Photocathode	Material	Bialkali	—
	Minimum Effective Area	ϕ120	mm
Window Material		Borosilicate glass	—
Dynode	Structure	Linear focused	—
	Number of Stages	14	—
Operating Ambient Temperature		-30 to +50	°C
Storage Temperature		-30 to +50	°C
Base		20-pin base	—
Suitable Socket		E678-20B (supplied)	—

MAXIMUM RATINGS (Absolute Maximum Values)

Parameter		Value	Unit
Supply Voltage	Between Anode and Cathode	3000	V
	Between Anode and Last Dynode	500	V
Average Anode Current		0.2	mA

CHARACTERISTICS (at 25 °C)

Parameter		Min.	Typ.	Max.	Unit
Cathode Sensitivity	Luminous (2856 K)	55	70	—	µA/lm
	Blue Sensitivity Index (CS 5-58)	7.0	9.0	—	—
	Quantum Efficiency at 390 nm	—	22	—	%
Anode Sensitivity	Luminous (2856 K)	300	1000	—	A/lm
	Blue Sensitivity Index (CS 5-58)	—	130	—	—
Gain		—	1.4×10^7	—	—
Anode Dark Current (after 30 min storage in darkness)		—	50	300	nA
Time Response	Anode Pulse Rise Time	—	2.5	—	ns
	Electron Transit Time	—	54	—	ns
	Transit Time Spread	—	1.2	—	ns
Pulse Height Resolution with ^{137}Cs		—	8.3	—	%
Gain Deviation	Long Term	—	1.0	—	%
	Short Term	—	1.0	—	%
Pulse Linearity *	2 % Deviation	—	160	—	mA
	5 % Deviation	—	250	—	mA

NOTE: Measured with special voltage distribution ratios shown in the Table 2.

Table 1: VOLTAGE DISTRIBUTION RATIO AND SUPPLY VOLTAGE

Electrode	K		G1		G2		Dy1		Dy2		Dy3		Dy4		Dy5		Dy6		Dy7		Dy8		Dy9		Dy10		Dy11		Dy12		Dy13		Dy14		P
Ratio		2.5		7.5		0		1.2		1.8		1		1		1		1		1		1		1		1		1.5		1.5		3		2.5	

Supply Voltage: 2000 Vdc, K: Cathode, Dy: Dynode, P: Anode, G: Grid

Table 2: SPECIAL VOLTAGE DISTRIBUTION RATIO AND SUPPLY VOLTAGE FOR PULSE LINEARITY MEASUREMENT

Electrode	K		G1		G2		Dy1		Dy2		Dy3		Dy4		Dy5		Dy6		Dy7		Dy8		Dy9		Dy10		Dy11		Dy12		Dy13		Dy14		P
Ratio		2.5		7.5		0		1.2		1.8		1		1		1		1		1.2		1.5		2		2.8		4		5.7		8		5	
Capacitors in µF																						0.01		0.01		0.02		0.02		0.02		0.04		0.06	

Supply Voltage: 2500 Vdc, K: Cathode, Dy: Dynode, P: Anode, G: Grid

PHOTOMULTIPLIER TUBE R1250

Figure 1: Typical Spectral Response

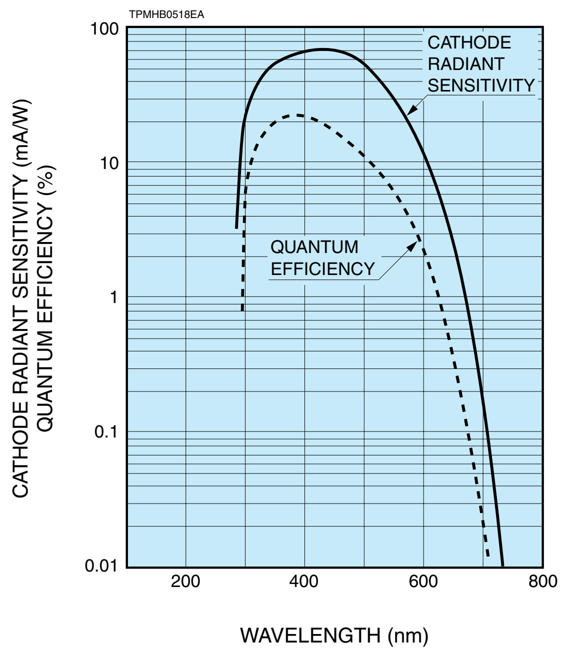


Figure 2: Typical Gain Characteristics

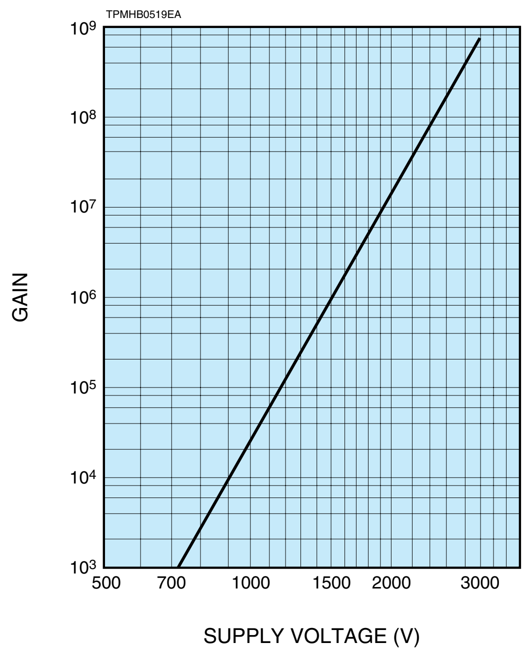
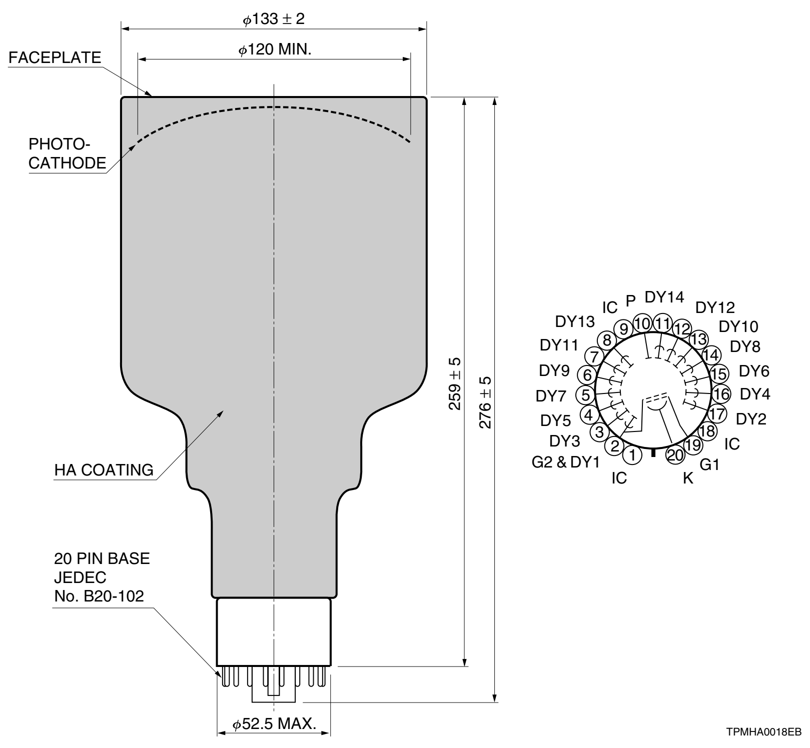
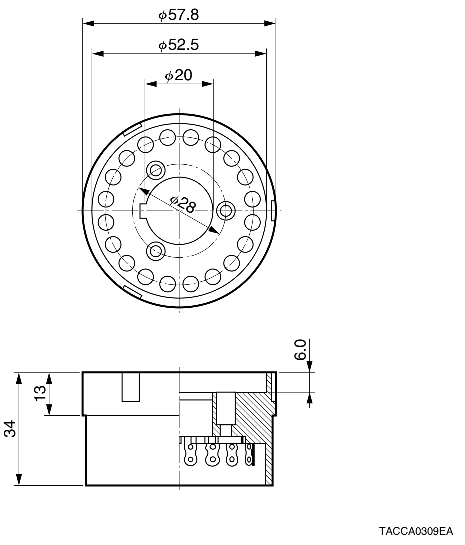


Figure 3: Dimensional Outline and Basing Diagram (Unit: mm)



Socket E678-20B (Supplied)



HAMAMATSU PHOTONICS K.K. **www.hamamatsu.com**

HAMAMATSU PHOTONICS K.K., Electron Tube Division
314-5, Shimokanzo, Iwata City, Shizuoka Pref., 438-0193, Japan, Telephone: (81)539/62-5248, Fax: (81)539/62-2205
U.S.A.: Hamamatsu Corporation: 360 Foothill Road, P. O. Box 6910, Bridgewater. N.J. 08807-0910, U.S.A., Telephone: (1)908-231-0960, Fax: (1)908-231-1218 E-mail: usa@hamamatsu.com
Germany: Hamamatsu Photonics Deutschland GmbH: Arzbergerstr. 10, D-82211 Herrsching am Ammersee, Germany, Telephone: (49)8152-375-0, Fax: (49)8152-2658 E-mail: info@hamamatsu.de
France: Hamamatsu Photonics France S.A.R.L.: 19, Rue du Saule Trapu, Parc du Moulin de Massy, 91882 Massy Cedex, France, Telephone: (33)1 69 53 71 00, Fax: (33)1 69 53 71 10 E-mail: infos@hamamatsu.fr
United Kingdom: Hamamatsu Photonics UK Limited: 2 Howard Court, 10 Tewin Road Welwyn Garden City Hertfordshire AL7 1BW, United Kingdom, Telephone: 44-(0)1707-294888, Fax: 44(0)1707-325777 E-mail: info@hamamatsu.co.uk
North Europe: Hamamatsu Photonics Norden AB: Smidesvägen 12, SE-171-41 SOLNA, Sweden, Telephone: (46)8-509-031-00, Fax: (46)8-509-031-01 E-mail: info@hamamatsu.se
Italy: Hamamatsu Photonics Italia: S.R.L.: Strada della Moia, 1/E, 20020 Arese, (Milano), Italy, Telephone: (39)02-935 81 733, Fax: (39)02-935 81 741 E-mail: info@hamamatsu.it

TPMH1213E03
DEC. 2010 IP

Detector Assembly Materials

Saint-Gobain Crystals can provide you with various detector assembly materials. For more detailed specifications on BC-600, BC-620 and BC-622A, individual data sheets are available.

BC-600 Optical Cement –

BC-600 optical cement is a clear epoxy, which sets at room temperature and has a refractive index close to that of SGC plastic scintillators. It is therefore ideal for optically cementing plastic scintillators to light pipes or optical windows. It is not recommended for coupling scintillators to photomultiplier tubes. For that application, we recommend BC-634A or BC-630.

BC-620 Reflector Paint for Plastic Scintillators –

BC-620 is a highly efficient reflector employing a special grade of titanium dioxide in a water soluble binder. It is applied directly onto plastic scintilllators, acrylic light guides, glass and metals. It is not intended for direct contact with liquid scintillators. It is a diffuse reflector and, therefore, should not be applied to sheets of scintillator or light guide material where the length is much longer than the thickness.

BC-620 can be removed with warm water.

BC-620 Consists of	%
Anatase Titanium Dioxide	40
Acrylic Emulsion Resin	24
Water	32
Glycol Coalescent	2.8
Surfactants & Thickeners	1.2

BC-622A Reflector Paint for Liquid Scintillator Tanks –

BC-622A reflector paint is intended for use with liquid scintillators. It is particularly useful in large steel or aluminum tanks, which require application of the paint at the research site. It is a diffuse reflector and, therefore, should not be used on the major surfaces of long, narrow tanks (total internal reflector and employed in these).

Can be removed from metal by submersing in Methol alcohol.

BC-630 Silicone Optical Grease –

BC-630 is a clear, colorless, silicone, optical coupling compound that features excellent light transmission and low evaporation and bleed at 25°C. It has a specific gravity of 1.06, an Index of Refraction of 1.465 and has a very flat transmission of approximately 95% for wavelengths between 280nm and 700nm. There is a sharp fall off below 280nm. Transmission at 270 and below is about zero. We supply this single component formulation in 60ml and 500ml jars.

Assembly Materials Available –

Optical Cement

Reflector Paint for Plastic Scintillators

Reflector Paint for Liquid Scintillator Tanks

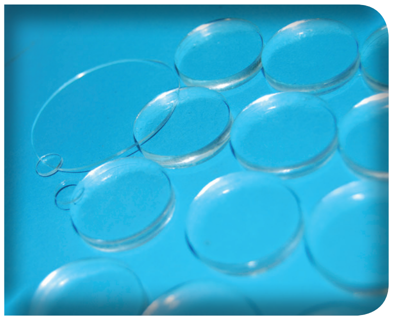
Silicone Optical Grease

Optical Interface Pads

Black Wrapping Tape

Plastic Masking Paper

PTFE Reflector Tape


Pictured are a variety of BC-634A sizes

SAINT-GOBAIN

Detector Assembly Materials

BC-634A Optical Interface Pad –

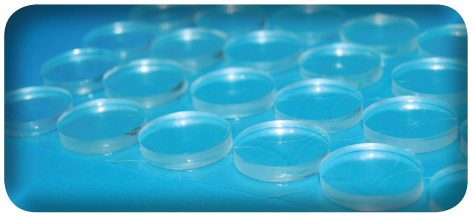
BC-634A is a self-wetting, flexible pad just hard enough to resist tearing while handling.

It is formulated for use within the temperature range of -10 to +60°C, has an index of refraction of 1.42 and an internal transmission >98% around 400nm.

If you cannot maintain sufficient interface pressure, apply a thin film of coupling grease to both sides of the interface pad.

BC-637 High Temperature Optical Interface Pad –

BC-637 interface pads are placed between the plastic and photomultiplier tube. BC-637 is rated at 200°C.



BC-638 Black Wrapping Tape –

BC-638 is a black adhesive tape 2" (50.8mm) wide by .008" (0.2mm) thick. Wrapping a plastic scintillator in one layer will give you a light-tight seal. We provide BC-638 in 36 yard (32.9m) rolls.

BC-640 Plastic Masking Paper –

This material is an adhesive-backed masking paper routinely used for protecting the surfaces of plastic scintillator during handling or storage. We supply BC-640 in rolls 12” (30.5cm) wide by 300' (91.4m) long.

BC-642 PTFE Reflector Tape –

BC-642 is a .003" (0.08 mm) thick (normal) Teflon tape and is frequently used as a reflecting material for non-hygroscopic scintillators. Three layers give you optimum reflectivity. It comes in rolls 2" (50.8mm) wide by 540" (13.7m) long.

USA

Saint-Gobain Crystals
17900 Great Lakes Parkway
Hiram, OH 44234
Tel: (440) 834-5600
Fax: (440) 834-7680

Europe

Saint-Gobain Crystals
104 Route de Larchant
BP 521
77794 Nemours Cedex, France
Tel: 33 (1) 64 45 10 10
Fax: 33 (1) 64 45 10 01

P.O. Box 3093
3760 DB Soest
The Netherlands
Tel: 31 35 60 29 700
Fax: 31 35 60 29 214

Japan

Saint-Gobain KK, Crystals Division
3-7, Kojimachi, Chiyoda-ku,
Tokyo 102-0083 Japan
Tel: 81 (0) 3 3263 0559
Fax: 81 (0) 3 5212 2196

China

Saint-Gobain (China) Investment Co, Ltd
15-01 CITIC Building
19 Jianguomenwai Ave.
Beijing 100004 China
Tel: 86 (0) 10 6513 0311
Fax: 86 (0) 10 6512 9843

India

Saint-Gobain Crystals and Detectors
Sy. No. 171/2, Maruthi Industrial Estate
Hoody Rajapalya, Whitefield Main Road
Bangalore 560048 India
Tel: 91 80 42468989
Fax: 91 80 28416501

www.crystals.saint-gobain.com

CAEN Tools for Discovery Electronic Instrumentation

V1720

8 Channel 12bit 250 MS/s Digitizer

- 12 bit 250 MS/s ADC
- FPGA for real time Digital Pulse Processing:
  - Charge Integration (DPP-CI)
  - Pulse Shape Discrimination (DPP-PSD)
  - Zero Suppression (Standard Firmware)
- 8 channels
- 2 Vpp input range (single ended or differential)
- 16-bit programmable DC offset adjustment: ±1 V
- Trigger Time stamps
- Memory buffer: 1.25 or 10 MS/ch, up to 1024 events
- Programmable event size and pre-post trigger adjustment
- Analog Sum/Majority and digital over/under threshold flags for Global Trigger logic
- Front panel clock In/Out available for multiboard synchronisation (direct feed through or PLL based synthesis)
- 16 programmable LVDS I/Os
- Optical Link interface (CAEN proprietary protocol)
- VME64X compliant interface
- Firmware upgradeable via VME/Optical Link
- A2818(PCI) / A3818 (PCIe) Controller available for handling up to 8/32 modules Daisy chained via Optical Link
- Libraries, Demos (C and LabView) and Software tools for Windows and Linux

The **V1720** is a 1-unit wide VME 6U module housing a **8** Channel **12** bit **250 MS/s** Flash ADC Waveform Digitizer and featuring 2 Vpp single ended input dynamics. Versions with 2 Vpp differential input full scale range are also available.
The DC offset adjustment (range ±1 V) by programmable 16bit DACs (one for each channel) on single ended input versions allows a right sampling of a bipolar (Vin = ±1 V) up to a full positive (Vin = 0 ÷ +2 V) or negative (Vin = 0 ÷ -2 V) analog input swing without losing dynamic resolution.

FPGA for real time Digital Pulse Processing (*)
The model is available in 7 versions with different FPGA densities: **V1720/V1720B/V1720C/V1720D** equipped with Cyclone EP1C4 (4.000 LEs) and **V1720E/V1720F/V1720G** equipped with Cyclone EP1C20 (20.000 LEs).

The module features front panel Clock Input and Output as well as a PLL for clock synthesis from internal/external references. The data stream is continuously written in a circular memory buffer. When the trigger occurs, the FPGA writes further N samples for the post trigger and freezes the buffer that can be read either by VMEbus or Optical Link. The acquisition can continue without dead time in a new buffer.

Each channel has a **SRAM Multi-Event Buffer** divisible into **1 ÷ 1024** buffers of programmable size. Two sizes of the channel digital memory are available by ordering options: **1.25 MS/ch (mod. V1720/V1720C/V1720E/V1720F)** and **10 MS/ch (mod. V1720B/V1720D/V1720G)**. 'Zero suppression' and 'data reduction' algorithms allow substantial savings in data amount readout and processing, rejecting samples smaller than programmable threshold. V1720 supports multi-board synchronization allowing all ADCs to be synchronized to a common clock source and ensuring Trigger time stamp alignment. Once synchronized, all data will be aligned and coherent across multiple V1720 boards.

The trigger signal can be provided externally via the front panel Trigger Input as well as via the software, but it can also be generated internally thanks to threshold self-trigger capability. The trigger from one board can be propagated to the other boards through the front panel Trigger Output.

An Analog Output is available with four operating modes supported:
- **Waveform Generator:** 1 Vpp ramp generator
- **Majority:** output signal is proportional to the number of ch. under/over threshold (1 step = 125 mV)
- **Buffer Occupancy:** output signal is proportional to the Multi Event Buffer Occupancy: 1 buffer ~ 1 mV
- **Voltage level:** output signal is a programmable voltage level

V1720 houses **VME** (VME64X compliant) and **Optical Link** interfaces. The VME interface allows data transfers of **60 MB/s** (MBLT64), **100 MB/s** (2eVME), **160 MB/s** (2eSST). The Optical Link supports transfer rate of **80 MB/s** and offers Daisy chain capability. Therefore, it is possible to connect up to 8/32 ADC modules to a single Optical Link controller (Mod. A2818/A3818).

Software available (Windows and Linux):
CAEN provides drivers for all the different types of physical communication channels, a set of C and LabView libraries (**CAENComm** and **CAENDigitizer**), demo applications and utilities:
- **CAENSCOPE**: fully graphical program that implements a simple oscilloscope.
- **CAENUpgrader**: tool that allows the user to update the firmware of the digitizers, change the PLL settings, load, when requested, the license for the pay firmware and other utilities.
- **CAEN WaveDump**: software console application that can be used to configure and readout event data from any model of the CAEN digitizer family and save the data into a memory buffer allocated for this purpose.

CAEN provides also for this model two **Digital Pulse Processing firmware** for Physics Applications. This feature allows to perform on-line processing on detector signal directly digitized:

- **DPP-CI Digital Pulse Processing for the Charge Integration**
  x720 digitizer running DPP-CI firmware is well suited for data acquisition and processing of signals from scintillators/photomultipliers or SiPM detectors, implementing a digital version of the traditional QDC (Charge-to-Digital Converter).
- **DPP-PSD Digital Pulse Processing for Pulse Shape Discrimination**
  x720(*) and x751 digitizers running DPP-PSD firmware accept signals directly from the detector and implement a digital replacement of dual gate QDC, discriminator and gate generator.

(*) *The DPP-PSD firmware runs only on* V1720E/V1720F/V1720G

Package	1-unit wide VME 6U module
Analog Input	8 channels (MCX 50 Ohm) Single ended or differential Input range: 2 Vpp Bandwidth: 125 MHz Programmable DAC for Offset Adjustment x channel (single ended only): ±1 V
Digital Conversion	Resolution: 12 bit Sampling rate: 31.25 to 250 MS/s simultaneously on each channel
System Performance	ENOB: 10.14 (64 kS Buffer) SFDR: 82.0 dB SINAD: 62.85 dB SIGMA: 0.95 LSB rms (64 kS Buffer, open input) THD: 74.1 dB
ADC Sampling Clock generation	Three operating modes: - PLL mode: internal reference (50 MHz loc. oscillator) - PLL mode: external reference on CLK_IN - PLL Bypass mode: ext. clock on CLK_IN drives directly ADC clocks (Freq.: 31.25 ÷ 250 MHz)
Digital I/O	CLK_IN (AMP Modu II): - AC coupled differential input clock LVDS, ECL, PECL, LVPECL, CML (single ended NIM/TTL available by custom cable) - Jitter<100ppm CLK_OUT (AMP Modu II): - DC coupled differential LVDS clock output locked at ADC sampling clock (Freq.:31.25 - 250 MHz) TRG_IN (NIM/TTL, Zin = 50 Ohm): external trigger input TRG_OUT (NIM/TTL, Rt = 50 Ohm): local trigger output S_IN (NIM/TTL, Zin = 50 Ohm): SYNC/SAMPLE/START front panel input
Memory Buffer	1.25 MS/ch or 10 MS/ch Multi Event Buffer Programmable event size and pre-post trigger Divisible into 1 ÷ 1024 buffers
Trigger	Common Trigger - External (signal on TRG_IN) - Software (by VMEbus or Optical Link) - Self trigger (internal threshold self-trigger)
Trigger Time Stamp	31-bit counter - 16 ns resolution - 17 s range
Multi Modules Synchronization	Clock propagation: by Daisy chain or Fan Out Trigger propagation: by Daisy chain or Fan Out Time stamp synchronization
ADC and Memory controller FPGA	One Altera Cyclone EP1C4 or EP1C20 per channel
Analog Monitor	12 bit/100 MHz DAC FPGA controlled output with four operating modes: - Test Waveform: 1 Vpp test ramp generator - Majority: MON/Σ output signal is proportional to the number of channels (enabled) under/over threshold (1 step = 125 mV) - Buffer Occupancy: MON/Σ output signal is proportional to the Multi Event Buffer Occupancy - Voltage level: MON/Σ output signal is a programmable voltage level
LVDS I/O	16 general purpose LVDS I/Os controlled by FPGA Busy, Data Ready, Memory full, Individual Trig-Out and other functions can be programmed An Input Pattern from the LVDS I/Os can be associated to each trigger as an event marker
VME interface	VME64X compliant Data modes: D32, BLT32, MBLT64, CBLT32/64, 2eVME, 2eSST, Multi Cast Cycles Transfer rate: 60 MB/s (MBLT64), 100 MB/s (2eVME), 160 MB/s (2eSST) Sequential and random access to the data of the Multi Event Buffer The Chained readout allows to read one event from all the boards in a VME crate with a BLT access
Optical Link	CAEN proprietary protocol, up to 80 MB/s transfer rate Daisy chainable: it is possible to connect up to 8/32 ADC modules to a single Optical Link Controller (Mod. A2818/A3818)
Upgrade	Firmware can be upgraded via VMEbus or Optical Link
Software	General purpose C and LabView Libraries Demo and Software Tools for Windows and Linux

Appendix C

Data Acquisition Code - Ugly Controller

A sample config file used to set up the digitizer for data acquisition. This particular config file was one of the actual ones used to take the data presented in this thesis.

```
% BYU Pulse Waveform Recorder Configuration File
% ----------------------------------------------------------------
% Settings common to all channels
% ----------------------------------------------------------------
[COMMON]

% OPEN: open the digitizer
% options: USB 0 0      Desktop/NIM digitizer through USB
%          USB 0 BA     VME digitizer through USB-V1718 (BA = BaseAddress of
    the VME board, 32 bit hex)
%          PCI 0 0 0    Desktop/NIM/VME through CONET (optical link)
%          PCI 0 0 BA   VME digitizer through V2718 (BA = BaseAddress of the
    VME board, 32 bit hex)
%OPEN USB 0 0
%OPEN USB 0 32100000
%OPEN PCI 0 0 0
OPEN PCI 0 0 32100000

% DAC_SLOPE: scales the dc offset for the dac
% Range 0.5 to 2.0
% DAC_SLOPE 1.0 % DT5720 s\n 31
DAC_SLOPE   1.19    % DT5720 s\n 79


% DAC_INTERCEPT: dac offset for calibrating the dc offset
% Range -1000 to 1000
% DAC_INTERCEPT 0   % DT5720 s\n 31
```

```
DAC_INTERCEPT   610 % DT5720 s\n 79

% DIRECTORY: directory to write data files to (full path, no slash at the end)
    . 100 characters max length
DIRECTORY D:\Data\PVT-LGBPan5\2016-02-10

% EXTENSION: data file extension (include the dot). 10 characters max length
EXTENSION .dat

% TOTAL_SECONDS: number of seconds to stop digitizing after. Choose 0 for
    continuous run
TOTAL_SECONDS 0

% TOTAL_EVENTS: number of collected events to stop digitizing after. Choose 0
    for continuous run
TOTAL_EVENTS 0

% EVENTS_PER_FILE: number of events to be written to file before creating a
    new file to write to
EVENTS_PER_FILE 10000

% POST_TRIGGER: post trigger size in percent of the whole acquisition window
% options: 0 to 100
% On models 742 there is a delay of about 35nsec on signal Fast Trigger TR;
    the post trigger is added to this delay
POST_TRIGGER  30

% RECORD_LENGTH = number of samples in the acquisition window
RECORD_LENGTH  4096

% EXTERNAL_TRIGGER: external trigger input settings. When enabled, the ext.
    trg. can be either
% propagated (ACQUISITION_AND_TRGOUT) or not (ACQUISITION_ONLY) through the
    TRGOUT
% options: DISABLED, ACQUISITION_ONLY, ACQUISITION_AND_TRGOUT
EXTERNAL_TRIGGER DISABLED

% ENABLE_ZLE: enable zero length encoding (set individual channel thresholds
    below)
% options: YES, NO
ENABLE_ZLE NO

% MAX_NUM_EVENTS_BLT: maximum number of events to read out in one Block
    Transfer. High values corresponds to
% options: 1 to 1023
MAX_NUM_EVENTS_BLT 50

% USE_INTERRUPT: number of events that must be ready for the readout when the
    IRQ is asserted.
% Zero means that the interrupts are not used (readout runs continuously)
USE_INTERRUPT 1

% MEM_BUFFERS: number by which CAEN internal memory is to be divided.
% Options: 0 to 10, where 0x00 is 1, 0x01 is 2, 0x02 is 4, 0x03 is 8, ... 0x0A
    is 1024.
MEM_BUFFERS 0

% FPIO_LEVEL: type of the front panel I/O LEMO connectors
% options: NIM, TTL
FPIO_LEVEL  NIM

% TEST_PATTERN: if enabled, data from ADC are replaced by test pattern (
   triangular wave)
% options: YES, NO
TEST_PATTERN   NO

% ----------------------------------------------------------------
% Individual Settings
% ----------------------------------------------------------------

% The following settings are applied specific to each channel.
%
% ENABLE_INPUT: enable/disable one channel
% Options: YES, NO
%
% DC_OFFSET: DC offset is in increments of approximately 2/65536 volts %
% Options: -2047.00 to 2047.00.
%
% TRIGGER_EDGE: rising or falling edge for trigger
% Options: RISING or FALLING
%
% TRIGGER_THRESHOLD: threshold for the channel auto trigger (ADC counts)
% are expressed in increments of 2/4096 volt. To specify zero volts use
% 2047.  For 1.0 volt use 4095 and -1.0 volt use 0.
% Options: -2047 to 2047.
%
% CHANNEL_TRIGGER: sets the channel trigger to one of three modes when enabled
   :
% options: DISABLED, ACQUISITION_ONLY, ACQUISITION_AND_TRGOUT
%
% ZLE_THRESHOLD: zero length encoding voltage threshold. All samples not
   exceeding
% threshold voltage are discarded (except Pretrigger and Posttrigger samples).
% options -2047 to 2047.
%
% ZLE_LOGIC: POSITIVE keeps samples over ZLE_THRESHOLD and NEGATIVE keeps
   samples
% under ZLE_THRESHOLD.
% Options: POSITIVE or NEGATIVE.
%
% ZLE_PRE_THRESHOLD: number of samples to keep before ZLE_THRESHOLD is true.
% Options: 0 to 65535.
%
% ZLE_POST_THRESHOLD: number of samples to keep after ZLE_THRESHOLD goes
   false.
% Options: 0 to 65535.

```

```
[0]
ENABLE_INPUT        YES
CHANNEL_TRIGGER     DISABLED
DC_OFFSET       134
TRIGGER_EDGE        FALLING
TRIGGER_THRESHOLD   -200
MIN_TRIG_WIDTH      10
ZLE_THRESHOLD       -100
ZLE_POLARITY        NEGATIVE
ZLE_PRE_THRESHOLD   10
ZLE_POST_THRESHOLD  10

[1]
ENABLE_INPUT        YES
CHANNEL_TRIGGER     DISABLED
DC_OFFSET       157
TRIGGER_EDGE        FALLING
TRIGGER_THRESHOLD   -250
MIN_TRIG_WIDTH      4
ZLE_THRESHOLD       2040
ZLE_POLARITY        NEGATIVE
ZLE_PRE_THRESHOLD   10
ZLE_POST_THRESHOLD  10

[2]
ENABLE_INPUT        YES
CHANNEL_TRIGGER     DISABLED
DC_OFFSET       106
TRIGGER_EDGE        FALLING
TRIGGER_THRESHOLD   250
MIN_TRIG_WIDTH      4
ZLE_THRESHOLD       2040
ZLE_POLARITY        NEGATIVE
ZLE_PRE_THRESHOLD   10
ZLE_POST_THRESHOLD  10

[3]
ENABLE_INPUT        YES
CHANNEL_TRIGGER     DISABLED
DC_OFFSET       154
TRIGGER_EDGE        FALLING
TRIGGER_THRESHOLD   0
MIN_TRIG_WIDTH      4
ZLE_THRESHOLD       2000
ZLE_POLARITY        POSITIVE
ZLE_PRE_THRESHOLD   10
ZLE_POST_THRESHOLD  10

[4]
ENABLE_INPUT            YES
CHANNEL_TRIGGER         ACQUISITION_ONLY
DC_OFFSET               139
TRIGGER_EDGE        FALLING
TRIGGER_THRESHOLD       -10
```

```
MIN_TRIG_WIDTH      75
ZLE_THRESHOLD       2000
ZLE_POLARITY        POSITIVE
ZLE_PRE_THRESHOLD   10
ZLE_POST_THRESHOLD  10

[5]
ENABLE_INPUT            NO
CHANNEL_TRIGGER         DISABLED
DC_OFFSET               0
TRIGGER_EDGE        FALLING
TRIGGER_THRESHOLD       0
MIN_TRIG_WIDTH      4
ZLE_THRESHOLD       2000
ZLE_POLARITY        POSITIVE
ZLE_PRE_THRESHOLD   10
ZLE_POST_THRESHOLD  10

[6]
ENABLE_INPUT            NO
CHANNEL_TRIGGER         DISABLED
DC_OFFSET               33
TRIGGER_EDGE        FALLING
TRIGGER_THRESHOLD       0
MIN_TRIG_WIDTH      4
ZLE_THRESHOLD       2000
ZLE_POLARITY        POSITIVE
ZLE_PRE_THRESHOLD   10
ZLE_POST_THRESHOLD  10

[7]
ENABLE_INPUT            NO
CHANNEL_TRIGGER         DISABLED
DC_OFFSET               45
TRIGGER_EDGE        FALLING
TRIGGER_THRESHOLD       0
MIN_TRIG_WIDTH      4
ZLE_THRESHOLD       2000
ZLE_POLARITY        POSITIVE
ZLE_PRE_THRESHOLD   10
ZLE_POST_THRESHOLD  10
```

Appendix D

Data Analysis Code - ToFSpec

The following is the data analysis code, ToFSpec. It consists of a single main script and several functions that each perform specific common tasks. I started trying to adapt existing software (Anspec) to suit our analysis needs but quickly abandoned the attempt, opting instead to do a total rewrite. I have used the same option menu styles as Anspec in places in an effort to ease the transition between the two when it is necessary to use both regularly.

```
% Main File
%
% Use to view and analyze events from .dat files produced by Best
% Controller and a CAEN DT5720 digitizer.
% Also works with .dat files from CAEN V1720 and byuCAENcorder4 controller.
%
% Portions of code used from several sources (JEE, Anspec)
%
% Created 2 Oct 2015
% By Alec Raymond (alec.raymond@yahoo.com)
% Last Updated 30 Mar 2016
% By Alec Raymond (alec.raymond@yahoo.com)

% To-Do List
% add an option to view/change metadata
% implement metadata generally/eliminate all magic numbers - nearly complete
% investigate loading method (much faster via matlab open than "load"
%   function in a script)
%   partial loading - matfile function
% generalize references to reduced/combined data - currently have to
%   "combine" data even if you only have a single channel
% do not call getEvent after final event of incomplete files - may need
%   fine tuning to work when headerEnabled = 0
% make interpolation in combineWaveforms more procedural - unnecessary for
%   summed waveforms
% find better way to calculate time-of-flight (enforce start to recoil
% only)

clear;

% metadata defaults
metadata.headersEnabled = 1; % 1 = true; 0 = false
metadata.channelsEnabled = 5; % number of channels enabled
metadata.samplesPerEventPerChannel = 4096; % in samples (4 ns per sample)
metadata.maxEventsPerFile = 10000; % set using digitizer controller
metadata.noiseThreshold = 15; % for checking for valid pulses
metadata.captureThreshold = 15; % for checking for valid captures
metadata.numEventsPerDataReducedFile = 100000; % set size of dataReduced
    output files
metadata.numEventsPerPulsesFoundFile = 100000; % set size of pulsesFound
    output files
metadata.smoothSpan = 1; % span for moving average for smoothing waveform
metadata.highThreshold = 10; % threshold for pulse finder
metadata.lowThreshold = 8; % threshold for pulse finder
metadata.widthHighThreshold = 5; % how long a pulse must stay above threshold
metadata.widthLowThreshold = 20; % how long a tail must stay below threshold
metadata.triggerLocation = 2867; % approx channel of trigger
metadata.triggerWidth = 75; % width of trigger in channels
metadata.triggerPorch = 100; % for finding approx capture start location -
    choose by plotting raw events
metadata.zeroDelayChannel = 1; % the channel containing the zero-delay
    waveform (start counting from 1 rather than 0)
metadata.longestDelayChannel = 4; % the channel containing the longest-delay
    waveform (all other delays must be contiguously increasing between these
    two channels)
metadata.pulseFrontPorch = 3; % number of channels to save in front of pulse
    start channel
metadata.pulseBackPorch = 3; % number of channels to save after pulse end
    channel
metadata.stopCableDelayLength = 75.2; % in channels (this is difference
    between stop and start cable lengths, not total stop cable length)
metadata.captureStartChannel = 10700; % determine using stopPulseStart
    histogram
metadata.earlyAreaCutoff = 15; % channels to include in early area after start
     of pulse
metadata.bins = 1000; % number of bins for histograms
metadata.distance = 1; % distance from source to stop detector in m
metadata.minEnergy = .1; % min energy of interest in MeV
metadata.maxEnergy = 10; % max energy of interest in MeV

choice=' ';
while choice==' '
    fprintf(1,'\nWhat do you want to do?\n');
    fprintf(1,'  0: Quit\n');
    fprintf(1,'  1: Plot events from .dat file\n');
    fprintf(1,'  2: Remove invalid events\n');
    fprintf(1,'  3: Plot valid events\n');
    fprintf(1,'  4: Combine waveforms\n');
    fprintf(1,'  5: Plot combined events\n');
    fprintf(1,'  6: Find pulses\n');
    fprintf(1,'  6a: Find pulses TEST\n');
    fprintf(1,'  7: Analyze found pulses\n');
    choice=input('Select Item> ','s');
    switch choice
%___________________________________________________________________________
        case '0' % Quit
            break;
%___________________________________________________________________________
        case '1' % Event-by-event inspection of .dat file
            [dir, files] = getFiles();
            plotRawEvents(dir, files, metadata);
            choice=' ';
%___________________________________________________________________________
        case '2' % Remove invalid events - saves out .mat file
            [dir, files] = getFiles();
            saveDir = getSaveDirectory();
            reduceData(dir, files, saveDir, metadata);
            choice=' ';
%___________________________________________________________________________
        case '3' % Plot valid events from .mat file
            [dir, files] = getFiles();
            plotValidEvents(dir, files, metadata);
            choice=' ';
%___________________________________________________________________________
        case '4' % Combine delayed waveforms from .mat file
            [dir, files] = getFiles();
            saveDir = getSaveDirectory();
            combineWaveforms(dir, files, saveDir, metadata);
            choice=' ';
%___________________________________________________________________________
        case '5' % Plot combined events from .mat file
            [dir, files] = getFiles();
            plotCombinedEvents(dir, files);
            choice=' ';
%___________________________________________________________________________
        case '6' % Find all pulses in all waveforms from .mat file
            [dir, files] = getFiles();
            saveDir = getSaveDirectory();
            findPulses(dir, files, saveDir, metadata);
            choice=' ';
%___________________________________________________________________________
        case '6a' % Find all pulses in all waveforms from .mat file
            [dir, files] = getFiles();
            saveDir = getSaveDirectory();
            findPulsesTest(dir, files, saveDir, metadata);
            choice=' ';
%___________________________________________________________________________
        case '7' % Analyze found pulses from .mat file
```

```
            [dir, files] = getFiles();
            analyzePulses(dir, files, metadata);
            choice=' ';
%______________________________________________________________________________
        otherwise %Invalid input
            choice=' ';
    end
end
```

This function is used to plot unprocessed events directly from the .dat files from the digitizer.

```
function plotRawEvents(dir, files, metadata)

% Use to plot raw waveforms in .dat files starting at a user defined event

channelsEnabled = metadata.channelsEnabled;
maxEventsPerFile = metadata.maxEventsPerFile;

fileCount = length(files);

% get starting point from user
jstart = input('\nWhich file do you want to start with? ');
istart = input('Which event? ');
fprintf('Any key to continue\nCntl-c to quit\n');

% loop through the files
for j = jstart : fileCount

    dataFileName = [dir, '/', char(files(j))];
    fid = fopen(dataFileName,'r');

    % loop through the events
    for i = istart : maxEventsPerFile

        % skip to selected event
        if j == jstart && i == istart
            for k = 1 : istart - 1
                getEvent(fid, metadata);
            end
        end

        waveforms = getEvent(fid, metadata);

        close all
        % plot event
        for k = 1:channelsEnabled
            subplot(channelsEnabled,1,k)
%             hold on
            plot(waveforms.waveforms(:,k))
%             legend('0','1','2','3','4','5','6','7')
%             xlim([2000 2030])
```

```
            ylabelStr = ['Waveform ' num2str(k)];
            ylabel(ylabelStr)
            if k == 1
                titleStr = ['Event ' num2str(i)];
                title(titleStr)
            end
%             hold off
        end
        pause
    end
    fclose(fid);
end
```

This function is used to select the .dat files to be processed.

```
function [dir, files]=getFiles()

% Use to navigate to and select files for processing.

dir = uigetdir('','Select folder that contains raw data files');
files = cellstr(uigetfile([dir,'\*.*'],'Select file', 'MultiSelect', 'on'));
```

The function is used to choose where to save output files to.

```
function dir=getSaveDirectory()

% Use to choose where to save output files.

dir = uigetdir('','Select folder to save output files to');
```

This function reads events from .dat files.

```
function [waveforms, header]=getEvent(fid, metadata)

% Returns waveforms and header (if enabled)

headersEnabled = metadata.headersEnabled;
channelsEnabled = metadata.channelsEnabled;
samplesPerEventPerChannel = metadata.samplesPerEventPerChannel;

% read headers if enabled
if headersEnabled == 1
    header = fread(fid, 4, 'ubit32');
end

% read waveforms
```

```matlab
for i = 1:channelsEnabled
    % find end of file and set waveforms to empty so reduceData stops
    % looking for events
    if feof(fid)
        waveforms = [];
        break
    end
    waveforms.waveforms(:,i) = fread(fid, samplesPerEventPerChannel, 'uint16')
        ;
end
```

This function sets the baseline of a waveform to zero.

```matlab
function waveform = adjustWaveform(waveform)

% identify baseline
waveformBase = mode(waveform);

% set baseline to zero
waveform = waveform - waveformBase;
```

This function checks for valid pulses from both detectors and throws away events that do not qualify.

```matlab
function reduceData(dir, files, saveDir, metadata)

% Use to remove invalid events from the data set.
%
% An invalid event is an event that does not have any peaks above the noise
% level in one or more of its waveforms.
%
% The .mat file that this script saves out contains all information from
% the original data set except the following:
%
%   headers
%   events with channels that contain no waveform
%   ability to easily reprocess into a different format
%
% After checking output files to ensure the reduction process was set up
% correctly, the original data can be deleted with no loss of information
% except as noted above
%
% The .mat file uses ~80% less memory than the original .dat files
% (depending greatly on proportion of invalid events - typically 1/3 are valid
  )
%
% Threw an error using "save" 3 Nov 2015 on 58th .dat file
% trying 50 .dat file batches
% shouldn't have run out of memory anywhere, still unresolved;
% error has not occurred since then


% set constants
noiseThreshold = metadata.noiseThreshold; % channels (CAEN spreads 2 V over
    4096 channels on y axis)
captureThreshold = metadata.captureThreshold;
maxEventsPerFile = metadata.maxEventsPerFile; % set using Best Controller
channelsEnabled = metadata.channelsEnabled; % number of channels enabled
numEventsPerDataReducedFile = metadata.numEventsPerDataReducedFile;
triggerLocation = metadata.triggerLocation; % approx channel of trigger
triggerWidth = metadata.triggerWidth; % width of trigger in channels
triggerPorch = metadata.triggerPorch;


% initialize variables
eventValidCount = 0;
totalEventValidCount = 0;
numDataReduced = 0;

fileCount = length(files);
% loop through the files
for j = 1 : fileCount

    filename = [dir, '\', char(files(j))];
    fid = fopen(filename,'r');
    fprintf('\nNumber of .dat files to reduce: %u\nCurrent file: %u - %s\
        nSearching...\n', fileCount, j, filename);

    % loop through the events
    for i = 1 : maxEventsPerFile

        waveforms = getEvent(fid, metadata);
        % stop getting events after last event in file
        if isempty(waveforms)
            break
        end

%         % use if start and stop waveforms are not added
%         eventInvalid = 0;
%
%         % check for peak above noise level in each waveform -
%         for k = 1:channelsEnabled
%             waveform = waveforms.waveforms(:,k);
%             waveform = adjustWaveform(waveform);
%             if max(abs(waveform)) <= noiseThreshold
%                 eventInvalid = 1;
%                 break
%             end
%         end
%
%         % continue to next event if invalid
%         if eventInvalid == 1
%               continue
%           end

        % use if start and stop waveforms are summed into a single waveform
        waveform = waveforms.waveforms(:,1);
        waveform = adjustWaveform(waveform);
        % check for start pulse peak above noise level in first waveform
        if max(waveform(1:triggerLocation-triggerWidth-triggerPorch)) <=
            noiseThreshold
            continue
        end
        % check for capture pulse peak below noise level in first waveform
        % (sometimes the trigger occurs unexpectedly)
        if min(waveform) >= -captureThreshold
            continue
        end
        % check for recoil pulse peak below noise level in first waveform
        if min(waveform(1:triggerLocation-triggerWidth-triggerPorch)) >= -
            noiseThreshold
            continue
        end

        % save event if valid
        eventValidCount = eventValidCount + 1;
        totalEventValidCount = totalEventValidCount + 1;

        for k = 1:channelsEnabled
            waveform = waveforms.waveforms(:,k);
            waveform = adjustWaveform(waveform);
            dataReduced(eventValidCount).Waveforms(:,k) = waveform;
        end

        % save out dataReduced and reset it (for smaller dataReduced file
        % sizes)
        if length(dataReduced) == numEventsPerDataReducedFile
            saveStr = strcat(saveDir,'\B_Reduced',int2str(numDataReduced),
                datestr(datetime('now'),'_dd-mmm-yyyy_HH-MM-SS'));
            save(saveStr, 'dataReduced')
            numDataReduced = numDataReduced + 1;
            eventValidCount = 0;
            clear dataReduced
        end
    end
    fclose(fid);
    fprintf('Valid events found: %u', totalEventValidCount);
end

% check to see whether any valid events were found; otherwise, dataReduced
% will not even be created

if eventValidCount > 0
    fprintf('\nSaving...\n');
    saveStr = strcat(saveDir,'\B_Reduced',int2str(numDataReduced),datestr(
        datetime('now'),'_dd-mmm-yyyy_HH-MM-SS'));
    save(saveStr, 'dataReduced')
    fprintf('\nComplete!\nNumber of files reduced: %u\nTotal valid events
       found: %u\n', fileCount, totalEventValidCount)
else
    fprintf('\nComplete!\nNumber of files reduced: %u\nNo valid events found\n
       ', fileCount)
end
```

This function plots validated events for your viewing pleasure (and to make sure it's working correctly...).

```
function plotValidEvents(dir, files, metadata)

% Use to plot waveforms in .mat files.

channelsEnabled = metadata.channelsEnabled; % number of channels enabled

fileCount = length(files);

% loop through the files
for j = 1 : fileCount

    matFileName = [dir, '\', char(files(j))];
    fprintf('\nLoading .mat file...\n');
    load(matFileName);
    fprintf('  Any key: continue\n  cntl-c: quit\n');

    % loop through the events
    for i = 1 : length(dataReduced)

        % plot event
        for k = 1:channelsEnabled
            subplot(channelsEnabled,1,k)
            plot(dataReduced(i).Waveforms(:,k))
            ylabelStr = ['Waveform ' num2str(k)];
            ylabel(ylabelStr)
            if k == 1
                titleStr = ['File ' num2str(j) ' - Event ' num2str(i)];
                title(titleStr)
            end
        end
        pause
    end
    fclose(fid);
end
```

This function combines sequentially delayed waveforms into a single waveform. If necessary, it is capable of interpolating any additional waveforms (if for example, you had one channel for a start detector and three channels for a stop detector, it would triple the length of the start waveform and interpolate the new values while combining the three channels for the stop detector).

```
function combineWaveforms(dir, files, saveDir, metadata)

% Use to combine delayed waveforms into a single waveform.
%
% Assumes the zero delay channel is in the zeroDelayChannel-th channel in
% each event from the Reduced .mat file and the all subsequent delayed
% channels are contiguous in regularly increasing delay length
%
% After checking output files to ensure the combining process was set up
% correctly, the reduced data can be deleted with no loss of information

samplesPerEventPerChannel = metadata.samplesPerEventPerChannel; % in samples
    (4 ns per sample)
zeroDelayChannel = metadata.zeroDelayChannel; % start counting channels from 1
     (different from CAEN, which starts from 0)
longestDelayChannel = metadata.longestDelayChannel; % delayed channels should
    be contiguous and in order of increasing delay
numChannelsToCombine = longestDelayChannel - zeroDelayChannel + 1; % number of
     channels to combine

numDataCombined = 0;

fileCount = length(files);

% loop through the files
for j = 1 : fileCount

    matFileName = [dir, '\', char(files(j))];
    fprintf('\nLoading .mat file...\n');
    load(matFileName);
    fprintf('\nCombining waveforms...\n');

    waitbarstr = ['File ' int2str(j) ' of ' int2str(fileCount) ' - Combining
       waveforms...'];
    h = waitbar(0, waitbarstr);

    % loop through the events
    numEvents = length(dataReduced);
    for i = 1 : numEvents
        waitbar(i/numEvents)
        % combine waveforms
        waveformIndex = 1;
        for n = 1:numChannelsToCombine:samplesPerEventPerChannel*
            numChannelsToCombine - numChannelsToCombine + 1
             for m = 1:numChannelsToCombine
%                combinedWaveform(n+numChannelsToCombine-m) =
%                dataReduced(i).Waveforms(waveformIndex,m+1); % use if the
%                first channel is not to be combined
```

```
                combinedWaveform(n+numChannelsToCombine-m) = dataReduced(i)
                    .Waveforms(waveformIndex,m);
            end
            waveformIndex = waveformIndex + 1;
        end
%           % interpolate uncombined waveforms to match size
%           x = 1:numChannelsToCombine:samplesPerEventPerChannel*
    numChannelsToCombine;
%           v = dataReduced(i).Waveforms(:,3); % this must be set manually
%           xq = 1:1:samplesPerEventPerChannel*numChannelsToCombine;
%           interpolatedWaveform = interp1(x,v,xq);
%           % fill in end of interpolated waveform
%           for k = 0:numChannelsToCombine - 2
%               interpolatedWaveform(end - k) = interpolatedWaveform(end -
    numChannelsToCombine + 1);
%           end
        % save interpolated and combined waveforms
%           dataCombined(i).Waveforms(:,1) = interpolatedWaveform;
%           dataCombined(i).Waveforms(:,2) = combinedWaveform;
        % save combined waveforms
        dataCombined(i).Waveforms(:,1) = combinedWaveform;
%           dataCombined(i).Waveforms(:,2) = interpolatedWaveform;
    end
    close(h)

    % save out dataCombined
    saveStr = strcat(saveDir,'\C_Combined',int2str(numDataCombined),datestr(
        datetime('now'),'_dd-mmm-yyyy_HH-MM-SS'));
    save(saveStr, 'dataCombined')
    numDataCombined = numDataCombined + 1;
end
```

This function plots combined waveforms for additional viewing pleasure.

```
function plotCombinedEvents(dir, files)

% Use to plot waveforms in .mat files.

fileCount = length(files);

% loop through the files
for j = 1 : fileCount

    matFileName = [dir, '\', char(files(j))];
    fprintf('\nLoading .mat file...\n');
    load(matFileName);
    fprintf('  Any key: continue\n  cntl-c: quit\n');

    % loop through the events
%     for i =
    [149,240,364,504,529,732,906,928,936,975,987,1114,1275,1531,1634,1685,1844,1847,1883,19
    for i = 1 : length(dataCombined)

        % plot event
        numWaveforms = size(dataCombined(1).Waveforms,2);
        for k = 1:numWaveforms
            subplot(numWaveforms,1,k)
            plot(dataCombined(i).Waveforms(:,k))
            ylabelStr = ['Waveform ' num2str(k)];
            ylabel(ylabelStr)
            if k == 1
                titleStr = ['File ' num2str(j) ' - Event ' num2str(i)];
                title(titleStr)
            end
        end
        pause
    end
    fclose(fid);
end
```

The function identifies all pulses in every waveforms, saving their start locations and the values of each index in the pulse.

```
function findPulses(dir, files, saveDir, metadata)

% Use to pick out all pulses from all waveforms. Saves pulse start
% locations and values
%
% Currently only works on combined data
%
% Not yet sufficiently satisfying to justify deleting the combined data
% after finding pulses
%
% Very generous in determining what constitutes a pulse - rather have too
% much than leave good pulses out
%
% Saves out all pulses with the following structure:
%   event
%       pulse
%           pulseStart
%           pulseValues
%       pulse
%           ...
%       ...
%   event
%       ...
%   ...

highThreshold = metadata.highThreshold;
lowThreshold = metadata.lowThreshold;
```

```
numEventsPerPulsesFoundFile = metadata.numEventsPerPulsesFoundFile;
widthHighThreshold = metadata.widthHighThreshold;
widthLowThreshold = metadata.widthLowThreshold;
pulseFrontPorch = metadata.pulseFrontPorch;
pulseBackPorch = metadata.pulseBackPorch;
channelsEnabled = metadata.channelsEnabled;
triggerLocation = metadata.triggerLocation;
triggerWidth = metadata.triggerWidth;
triggerPorch = metadata.triggerPorch;

numFile = 0;
eventProcessed = 1;
fileCount = length(files);

% loop through the files
for j = 1 : fileCount
    matFileName = [dir, '\', char(files(j))];
    fprintf('\nLoading .mat file...\n');
    load(matFileName);
    fprintf('\nFinding pulses...\n');
    waitbarstr = ['File ' int2str(j) ' of ' int2str(fileCount) ' - Finding pulses...'];
    h = waitbar(0, waitbarstr);
    % loop through the events
    numEvents = length(dataCombined);
    for i = 1 : numEvents
        waitbar(i/numEvents)
        eventToProcess = dataCombined(:,i);
        waveform = eventToProcess.Waveforms(:,1);
%         smoothWaveform = eventToProcess.Waveforms(:,2);
        pulseNum = 0;
%         % DEBUG PLOTS
% %         subplot(2,1,1)
%         plot(waveform,'b')
% %         subplot(2,1,2)
% %         plot(smoothWaveform,'b')
%         pause
%         % END DEBUG PLOTS
        state = 'init';
        n = 1;
        while n <= length(waveform)
%         for n = 1 : length(waveform)
            switch state;
                case 'init' % in case waveform starts mid pulse
                    if abs(waveform(n)) < highThreshold
                        state = 'findStart';
                    end
                case 'findStart'
                    if abs(waveform(n)) >= highThreshold
                        % do not search after trigger location (pulses in capture tail)
                        if n >= (triggerLocation - triggerWidth)*channelsEnabled - triggerPorch
                            break
                        end
                        pulseStart = n;
                        duration = 0;
                        state = 'findWidth';
                    end
                case 'findWidth'
                    if abs(waveform(n)) < highThreshold
                        state = 'findStart';
                    else
                        duration = duration + 1;
                        if duration >= widthHighThreshold
                            state = 'findEnd';
                        end
                    end
                case 'findEnd'
                    if waveform(pulseStart) > 0 && waveform(n) < lowThreshold
                        pulseEnd = n;
                        duration = 0;
                        state = 'staysLow';
                    elseif waveform(pulseStart) < 0 && waveform(n) > -lowThreshold
                        pulseEnd = n;
                        duration = 0;
                        state = 'staysLow';
                    end
                case 'staysLow'
                    if waveform(pulseStart) > 0 && waveform(n) < lowThreshold
                        duration = duration + 1;
                    elseif waveform(pulseStart) < 0 && waveform(n) > -lowThreshold
                        duration = duration + 1;
                    else
                        state = 'findEnd';
                    end
                    if duration >=  widthLowThreshold
                        state = 'savePulse';
                    end
                case 'savePulse'
                    pulseNum = pulseNum + 1;
                    pulseStart = pulseStart - pulseFrontPorch;
                    % if porch pushes pulse start index beyond 1
                    if pulseStart < 1
                        pulseStart = 1;
                    end
                    pulseEnd = pulseEnd + pulseBackPorch;
                    % if porch pushes pulse end index beyond end of event
                    if pulseEnd > length(waveform)
                        pulseEnd = length(waveform);
                    end
                    event(eventProcessed).pulses(pulseNum).pulseStart = pulseStart;
                    event(eventProcessed).pulses(pulseNum).pulseValues = waveform(pulseStart:pulseEnd);
                    n = pulseEnd;
```

```
                    state = 'findStart';
%                     % DEBUG PLOTS
%                     subplot(2,1,1)
%                     plot(1:length(waveform),waveform,'b',pulseStart,waveform(pulseStart),'gx',pulseEnd,waveform(pulseEnd),'rx')
%                     title(['Event ' num2str(i)])
%                     subplot(2,1,2)
%                     plot(pulseStart:pulseEnd,waveform(pulseStart:pulseEnd))
% %                     subplot(4,1,3)
% %                     plot(1:length(smoothWaveform),smoothWaveform,'b',pulseStart,smoothWaveform(pulseStart),'gx',pulseEnd,smoothWaveform(pulseEnd),'rx')
% %                     subplot(4,1,4)
% %                     plot(pulseStart:pulseEnd,smoothWaveform(pulseStart:pulseEnd))
%                     pause
%                     % END DEBUG PLOTS
            end
            n = n + 1;
        end
        % save out pulses and clear event
        if length(event) == numEventsPerPulsesFoundFile
            saveStr = strcat(saveDir,'\D_PulsesFound',int2str(numFile),datestr(datetime('now'),'_dd-mmm-yyyy_HH-MM-SS'));
            save(saveStr, 'event')
            numFile = numFile + 1;
            eventProcessed = 0;
            clear event
        end
        eventProcessed = eventProcessed + 1;
    end
    close(h)
end
% save out pulses
fprintf('\nSaving...\n');
saveStr = strcat(saveDir,'\D_PulsesFound',int2str(numFile),datestr(datetime('now'),'_dd-mmm-yyyy_HH-MM-SS'));
save(saveStr, 'event')
```

This function calculates several pulse metrics and plots them in many ways. It is also responsible for area to energy conversion.

```
function analyzePulses(dir, files, metadata)

% Use to analyze pulses found, event by event
%
% Some trash eliminated using parameters determined by viewing plots:
% THESE PARAMETERS MUST BE CHANGED WITH ANY CHANGE IN HARDWARE SETUP

captureStartChannel = metadata.captureStartChannel;
```

```
earlyAreaCutoff = metadata.earlyAreaCutoff;
stopCableDelayLength = metadata.stopCableDelayLength;
d = metadata.distance;
bins = metadata.bins;
minEnergy = metadata.minEnergy;
maxEnergy = metadata.maxEnergy;

timeFactor = 10^-9; % to convert nanoseconds to seconds
m = 1.674927*10^-27; % mass of a neutron in kg
c = 2.9979*10^8; % speed of light in m/s
energyFactor = 1.6022*10^-13; % to convert Joules to MeV


choiceLocal=' ';
while choiceLocal==' '
    fprintf(1,'\n Finished analyzing, what do you want to do?\n');
    fprintf(1,'  0: Return to main menu\n');
    fprintf(1,'  1: Calculate pulse characteristics (resets pulse types)\n');
    fprintf(1,'  2: Close all open figures\n');
    fprintf(1,'  3: Change number of bins (plots must be remade)\n');
    fprintf(1,'  4: Make vectors for plots\n');
    fprintf(1,'  5: Choose regions and change pulse types\n');
    fprintf(1,'  6: Change pulse types (hard-coded parameters)\n');
    fprintf(1,'  7: Make sorted plots\n');
    choiceLocal=input('Select Item> ','s');

    switch choiceLocal
%___________________________________________________________________________
        case '0' % Quit
            break;
%___________________________________________________________________________
        case '1' % Calculate pulse characteristics
            fileCount = length(files);

            pulseNum = 0;

            % loop through the files
            for i = 1 : fileCount
                matFileName = [dir, '\', char(files(i))];
                fprintf(['\nLoading ' char(files(i)) '...\n']);
                load(matFileName);
                stopCableDelayLength = input('Set stopCableDelayLength: '); %
                    use when combining multiple runs with varying delay
                    lengths
                fprintf('\nAnalyzing pulses...\n');
                % loop through the events
                for j = 1 : size(event, 2)
                    % loop through the pulses
                    clear validStartChannel % used for calculating time-of-
                        flight
                    for k = 1 : size(event(j).pulses,2)
                        pulseValues = event(j).pulses(k).pulseValues;
                        pulseStart = event(j).pulses(k).pulseStart;
                        % check - pulses occurring after capture start
```

```
if pulseStart > captureStartChannel
    continue
end
pulseNum = pulseNum + 1;
% identify and assign pulse detector
[~, peakIndex] = max(abs(pulseValues));
if pulseValues(peakIndex) > 0 % from start detector
    pulse(pulseNum).type = 'start';
else % from start detector
    pulse(pulseNum).type = 'stop';
    pulseValues = pulseValues * -1; % invert negative pulse
end
% calculate pulse characteristics
pulse(pulseNum).start = pulseStart;
pulse(pulseNum).pulseValues = pulseValues;
pulse(pulseNum).peak = max(pulseValues);
pulse(pulseNum).area = sum(pulseValues);
if earlyAreaCutoff > length(pulseValues)
    pulse(pulseNum).earlyAreaRatio = 1;
else
    pulse(pulseNum).earlyAreaRatio = sum(pulseValues(1:earlyAreaCutoff))/pulse(pulseNum).area;
end
pulse(pulseNum).width = size(pulseValues,1);
% detector specific operations
if strcmp(pulse(pulseNum).type,'start') == 1 % start detector
    % check for and skip trash pulses
    if pulse(pulseNum).area <= 0 % includes noise, leading edge dips, and very occasional start in capture tail pileups
        pulseNum = pulseNum - 1; % will overwrite trashed pulse info
        continue
    end
    if pulse(pulseNum).earlyAreaRatio < 0 || pulse(pulseNum).earlyAreaRatio > 1 % includes noise and very occasional pileup pulses
        pulseNum = pulseNum - 1;
        continue
    end
    if pulse(pulseNum).peak < 300 && pulse(pulseNum).area > 5000
        pulseNum = pulseNum - 1;
        continue
    end
else % stop detector
    % check for and skip trash pulses
    if pulse(pulseNum).area <= 0 % includes both noise (-30000 < area < 0) and double gamma pulses (area <= -30000)
        pulseNum = pulseNum - 1;
        continue
```

```
                            end
                            if pulse(pulseNum).earlyAreaRatio < 0 || pulse(pulseNum).earlyAreaRatio > 1 % includes noise and very occasional pileup pulses
                                pulseNum = pulseNum - 1;
                                continue
                            end
                        end
                        pulse(pulseNum).ToF = NaN; % to match vector size
                        pulse(pulseNum).energy = NaN; % to match vector size
                        % calculate time of flight and energy
                        if exist('validStartChannel','var')
                            pulse(pulseNum).ToF = pulseStart - validStartChannel - stopCableDelayLength;
                            % check for pulses that occur between start
                            % channel and end of stop cable delay length
                            if pulse(pulseNum).ToF < 0
                                pulseNum = pulseNum - 1;
                                continue
                            end
                            t = timeFactor*pulse(pulseNum).ToF;
                            v = d/t;
                            % check for coincidental pulses (real ones
                            % don't move faster than c...)
                            if v > c
                                pulseNum = pulseNum - 1;
                                continue
                            end
                            gamma = real(1/sqrt(1-(v/c)^2));
                            pulse(pulseNum).energy = m*c^2*(gamma-1)/energyFactor;
                        end
                        % enforce energy ROI
                        if pulse(pulseNum).energy < minEnergy
                            pulseNum = pulseNum - 1;
                            continue
                        end
                        if strcmp(pulse(pulseNum).type,'stop') == 1
                            if pulse(pulseNum).energy > maxEnergy
                                pulse(pulseNum).type = 'stopgamma';
                            elseif isnan(pulse(pulseNum).energy)
                                pulse(pulseNum).type = 'stopgamma';
                            end
                        end
                        validStartChannel = pulse(pulseNum).start; % assigned after using start channel from previous valid pulse
                    end
                end
            end
            choiceLocal=' ';
%__________________________________________________________________________
        case '2' % Close all open figures
            close all
            choiceLocal=' ';
%_____________________________________________________________________________
        case '3' % Change number of bins
            answer = inputdlg('Enter number of bins:');
            bins = str2double(answer{1});
            choiceLocal=' ';
%_____________________________________________________________________________
        case '4' % Make vectors for plots
            startPulseNum = 0;
            stopPulseNum = 0;
            stopGammaPulseNum = 0;
            stopRecoilPulseNum = 0;
            stopCapture1PulseNum = 0;
            stopCapture2PulseNum = 0;
            for m = 1:pulseNum
                if strcmp(pulse(m).type,'start') == 1
                    startPulseNum = startPulseNum + 1;
                    startStart(startPulseNum) = pulse(m).start;
                    startPeak(startPulseNum) = pulse(m).peak;
                    startArea(startPulseNum) = pulse(m).area;
                    startEarlyAreaRatio(startPulseNum) = pulse(m)
                        .earlyAreaRatio;
                    startWidth(startPulseNum) = pulse(m).width;
                    startToF(startPulseNum) = pulse(m).ToF;
                    startEnergy(startPulseNum) = pulse(m).energy;
                elseif strcmp(pulse(m).type,'stopgamma') == 1
                    stopGammaPulseNum = stopGammaPulseNum + 1;
                    stopGammaStart(stopGammaPulseNum) = pulse(m).start;
                    stopGammaPeak(stopGammaPulseNum) = pulse(m).peak;
                    stopGammaArea(stopGammaPulseNum) = pulse(m).area;
                    stopGammaEarlyAreaRatio(stopGammaPulseNum) = pulse(m)
                        .earlyAreaRatio;
                    stopGammaWidth(stopGammaPulseNum) = pulse(m).width;
                    stopGammaToF(stopGammaPulseNum) = pulse(m).ToF;
                    stopGammaEnergy(stopGammaPulseNum) = pulse(m).energy;
                elseif strcmp(pulse(m).type,'stoprecoil') == 1
                    stopRecoilPulseNum = stopRecoilPulseNum + 1;
                    stopRecoilStart(stopRecoilPulseNum) = pulse(m).start;
                    stopRecoilPeak(stopRecoilPulseNum) = pulse(m).peak;
                    stopRecoilArea(stopRecoilPulseNum) = pulse(m).area;
                    stopRecoilEarlyAreaRatio(stopRecoilPulseNum) = pulse(m)
                        .earlyAreaRatio;
                    stopRecoilWidth(stopRecoilPulseNum) = pulse(m).width;
                    stopRecoilToF(stopRecoilPulseNum) = pulse(m).ToF;
                    stopRecoilEnergy(stopRecoilPulseNum) = pulse(m).energy;
                elseif strcmp(pulse(m).type,'stopcapture1') == 1
                    stopCapture1PulseNum = stopCapture1PulseNum + 1;
                    stopCapture1Start(stopCapture1PulseNum) = pulse(m).start;
                    stopCapture1Peak(stopCapture1PulseNum) = pulse(m).peak;
                    stopCapture1Area(stopCapture1PulseNum) = pulse(m).area;
                    stopCapture1EarlyAreaRatio(stopCapture1PulseNum) = pulse(m
                        ).earlyAreaRatio;
                    stopCapture1Width(stopCapture1PulseNum) = pulse(m).width;
                    stopCapture1ToF(stopCapture1PulseNum) = pulse(m).ToF;
                    stopCapture1Energy(stopCapture1PulseNum) = pulse(m).energy
                        ;
                elseif strcmp(pulse(m).type,'stopcapture2') == 1
                    stopCapture2PulseNum = stopCapture2PulseNum + 1;
                    stopCapture2Start(stopCapture2PulseNum) = pulse(m).start;
                    stopCapture2Peak(stopCapture2PulseNum) = pulse(m).peak;
                    stopCapture2Area(stopCapture2PulseNum) = pulse(m).area;
                    stopCapture2EarlyAreaRatio(stopCapture2PulseNum) = pulse(m
                        ).earlyAreaRatio;
                    stopCapture2Width(stopCapture2PulseNum) = pulse(m).width;
                    stopCapture2ToF(stopCapture2PulseNum) = pulse(m).ToF;
                    stopCapture2Energy(stopCapture2PulseNum) = pulse(m).energy
                        ;
                else
                    stopPulseNum = stopPulseNum + 1;
                    stopStart(stopPulseNum) = pulse(m).start;
                    stopPeak(stopPulseNum) = pulse(m).peak;
                    stopArea(stopPulseNum) = pulse(m).area;
                    stopEarlyAreaRatio(stopPulseNum) = pulse(m).earlyAreaRatio
                        ;
                    stopWidth(stopPulseNum) = pulse(m).width;
                    stopToF(stopPulseNum) = pulse(m).ToF;
                    stopEnergy(stopPulseNum) = pulse(m).energy;
                end
            end
            choiceLocal=' ';
%_____________________________________________________________________________
        case '5' % Choose regions and reassign pulse type to region name

            % peak vs area plot - recoil selection
            clear h vertices xvertices yvertices
            figure
            scatter(stopArea,stopEarlyAreaRatio,'k.')
            title('Early Area Ratio vs Area - Select Recoil')
            h = impoly;
            vertices = getPosition(h);
            xvertices = vertices(:,1);
            yvertices = vertices(:,2);
            for m = 1:pulseNum
                if strcmp(pulse(m).type,'stop') == 1 % unsorted pulse from
                   stop detector (preserves gamma type assignment from
                   earlier energy check)
%                 if strcmp(pulse(m).type,'start') == 0 % not start detector (
   will overwrite previous type assignment)
                    if inpolygon(pulse(m).area,pulse(m).earlyAreaRatio,
                        xvertices,yvertices) == 1
                        pulse(m).type = 'stoprecoil';
                    else
                        pulse(m).type = 'stopgamma';
                    end
                end
            end
            close
            % early area ratio vs area plot - capture 1 selection
            clear h vertices xvertices yvertices
            figure
            scatter(stopArea,stopEarlyAreaRatio,'k.')
            title('Early Area Ratio vs Area - Select Capture1')
            h = impoly;
            vertices = getPosition(h);
            xvertices = vertices(:,1);
            yvertices = vertices(:,2);
            for m = 1:pulseNum
%                 if strcmp(pulse(m).type,'stop') == 1 % unsorted pulse from stop detector
                if strcmp(pulse(m).type,'start') == 0 % not start detector
                    if inpolygon(pulse(m).area,pulse(m).earlyAreaRatio, xvertices,yvertices) == 1
                        pulse(m).type = 'stopcapture1';
                    end
                end
            end
            close

            % early area ratio vs area plot - capture 2 selection
            clear h vertices xvertices yvertices
            figure
            scatter(stopArea,stopEarlyAreaRatio,'k.')
            title('Early Area Ratio vs Area - Select Capture2')
            h = impoly;
            vertices = getPosition(h);
            xvertices = vertices(:,1);
            yvertices = vertices(:,2);
            for m = 1:pulseNum
%                 if strcmp(pulse(m).type,'stop') == 1 % unsorted pulse from stop detector
                if strcmp(pulse(m).type,'start') == 0 % not start detector
                    if inpolygon(pulse(m).area,pulse(m).earlyAreaRatio, xvertices,yvertices) == 1
                        pulse(m).type = 'stopcapture2';
                    end
                end
            end
            close

            choiceLocal=' ';
%___________________________________________________________________________
        case '6' % Change pulse types based on region selection (hard-coded)
            for m = 1:pulseNum
                if strcmp(pulse(m).type,'stop') == 1 % stop detector
                    % initial separation of stop pulses into gammas, recoils, and captures
                    % by area (use plots to determine parameters)
                    if pulse(m).earlyAreaRatio > .04
                        if pulse(m).area > 10000
                            pulse(m).type = 'stopgamma';
                        else
                            pulse(m).type = 'stoprecoil';
                        end
                    else
                        if pulse(m).area < 15000
                            pulse(m).type = 'stopcapture1';
                        else
                            pulse(m).type = 'stopcapture2';
                        end
                    end
                end
            end
            choiceLocal=' '; % remake vectors
%__________________________________________________________________________
        case '7' % Remake plots

%             figure
%             hold on
%             scatter(stopGammaArea,stopGammaPeak,'k.')
%             scatter(stopRecoilArea,stopRecoilPeak,'k.')
%             scatter(stopCapture1Area,stopCapture1Peak,'k.')
%             scatter(stopCapture2Area,stopCapture2Peak,'k.')
%             hold off
%             title('Peak vs Area')
%             xlabel('Area')
%             ylabel('Peak')
%
%             figure
%             hold on
%             scatter(stopGammaArea,stopGammaPeak,'g.')
%             scatter(stopRecoilArea,stopRecoilPeak,'r.')
%             scatter(stopCapture1Area,stopCapture1Peak,'c.')
%             scatter(stopCapture2Area,stopCapture2Peak,'b.')
%             hold off
%             title('Peak vs Area')
%             xlabel('Area')
%             ylabel('Peak')
%             legend('Gamma','Recoil','Capture 1','Capture 2')

            figure
            hold on
            scatter(stopGammaArea,stopGammaEarlyAreaRatio,'k.')
            scatter(stopRecoilArea,stopRecoilEarlyAreaRatio,'k.')
            scatter(stopCapture1Area,stopCapture1EarlyAreaRatio,'k.')
            scatter(stopCapture2Area,stopCapture2EarlyAreaRatio,'k.')
            hold off
            title('Early Area Ratio vs Total Area')
            xlabel('Area')
            ylabel(['Early Area Ratio (' num2str(earlyAreaCutoff) ' channels)'
                ])

            figure
            hold on
            scatter(stopGammaArea,stopGammaEarlyAreaRatio,'g.')
            scatter(stopRecoilArea,stopRecoilEarlyAreaRatio,'r.')
            scatter(stopCapture1Area,stopCapture1EarlyAreaRatio,'c.')
            scatter(stopCapture2Area,stopCapture2EarlyAreaRatio,'b.')
            hold off
            title('Early Area Ratio vs Total Area')
            xlabel('Area')
            ylabel(['Early Area Ratio (' num2str(earlyAreaCutoff) ' channels)'])
            legend('Gamma','Recoil','Capture 1','Capture 2')

            figure
            h = histogram([stopGammaToF stopRecoilToF],bins,'EdgeColor','k','FaceColor','none');
            title('ToF Histogram')
            xlabel('ToF (ns)')
            ylabel('Count');
%             xlim([0 500])

            figure
            hold on
            histogram(stopGammaToF,bins,'EdgeColor','g','FaceColor','none','BinWidth',h.BinWidth)
            histogram(stopRecoilToF,bins,'EdgeColor','r','FaceColor','none','BinWidth',h.BinWidth)
            hold off
            title('ToF Histogram')
            xlabel('ToF (ns)')
            ylabel('Count');
            legend('Gamma','Recoil')
%             xlim([0 500])

%             figure
%             hold on
%             scatter(stopGammaToF,stopGammaPeak,'k.')
%             scatter(stopRecoilToF,stopRecoilPeak,'k.')
%             scatter(stopCapture1ToF,stopCapture1Peak,'k.')
%             scatter(stopCapture2ToF,stopCapture2Peak,'k.')
%             hold off
%             title('Peak vs ToF (linear)')
%             xlabel('ToF (ns)')
%             ylabel('Peak');
% %             xlim([0 500])
%
%             figure
%             hold on
%             scatter(stopGammaToF,stopGammaPeak,'g.')
%             scatter(stopRecoilToF,stopRecoilPeak,'r.')
%             scatter(stopCapture1ToF,stopCapture1Peak,'c.')
%             scatter(stopCapture2ToF,stopCapture2Peak,'b.')
%             hold off
%             title('Peak vs ToF (linear)')
%             xlabel('ToF (ns)')
%             ylabel('Peak');
%             legend('Gamma','Recoil','Capture 1','Capture 2')
% %             xlim([0 500])
%
```

```
%               figure
%               scatter(stopRecoilToF,stopRecoilPeak,'r.')
%               title('Peak vs ToF (linear)')
%               xlabel('ToF (ns)')
%               ylabel('Peak');
%               legend('Recoil')
% %               xlim([0 500])
%
%               figure
%               hold on
%               scatter(stopGammaToF,stopGammaPeak,'g.')
%               scatter(stopRecoilToF,stopRecoilPeak,'r.')
%               scatter(stopCapture1ToF,stopCapture1Peak,'c.')
%               scatter(stopCapture2ToF,stopCapture2Peak,'b.')
%               set(gca,'xscale','log','yscale','log')
%               hold off
%               title('Peak vs ToF (log)')
%               xlabel('ToF (ns)')
%               ylabel('Peak');
%               legend('Gamma','Recoil','Capture 1','Capture 2')
% %               xlim([1 10^4])
%
%               figure
%               scatter(stopRecoilToF,stopRecoilPeak,'r.')
%               set(gca,'xscale','log','yscale','log')
%               title('Peak vs ToF (log)')
%               xlabel('ToF (ns)')
%               ylabel('Peak');
%               legend('Recoil')
% %               xlim([1 10^4])
%
%               figure
%               scatter(stopRecoilEnergy,stopRecoilPeak,'r.')
%               title('Peak vs Energy (linear)')
%               xlabel('Energy (MeV)')
%               ylabel('Peak');
%               legend('Recoil')
% %               xlim([minEnergy maxEnergy])
%
%               figure
%               scatter(stopRecoilEnergy,stopRecoilPeak,'r.')
%               set(gca,'xscale','log','yscale','log')
%               title('Peak vs Energy (log)')
%               xlabel('Energy (MeV)')
%               ylabel('Peak');
%               legend('Recoil')
%
%               figure
%               hold on
%               scatter(stopGammaToF,stopGammaArea,'k.')
%               scatter(stopRecoilToF,stopRecoilArea,'k.')
%               scatter(stopCapture1ToF,stopCapture1Area,'k.')
%               scatter(stopCapture2ToF,stopCapture2Area,'k.')
%               hold off
%               title('Area vs ToF (linear)')
%               xlabel('ToF (ns)')
%               ylabel('Area');
% %               xlim([0 500])
%
%               figure
%               hold on
%               scatter(stopGammaToF,stopGammaArea,'g.')
%               scatter(stopRecoilToF,stopRecoilArea,'r.')
%               scatter(stopCapture1ToF,stopCapture1Area,'c.')
%               scatter(stopCapture2ToF,stopCapture2Area,'b.')
%               hold off
%               title('Area vs ToF (linear)')
%               xlabel('ToF (ns)')
%               ylabel('Area');
%               legend('Gamma','Recoil','Capture 1','Capture 2')
% %               xlim([0 500])
%
%               figure
%               scatter(stopRecoilToF,stopRecoilArea,'r.')
%               title('Area vs ToF (linear)')
%               xlabel('ToF (ns)')
%               ylabel('Area');
%               legend('Recoil')
% %               xlim([0 500])
%
%               figure
%               hold on
%               scatter(stopGammaToF,stopGammaArea,'g.')
%               scatter(stopRecoilToF,stopRecoilArea,'r.')
%               scatter(stopCapture1ToF,stopCapture1Area,'c.')
%               scatter(stopCapture2ToF,stopCapture2Area,'b.')
%               set(gca,'xscale','log','yscale','log')
%               hold off
%               title('Area vs ToF (log)')
%               xlabel('ToF (ns)')
%               ylabel('Area');
%               legend('Gamma','Recoil','Capture 1','Capture 2')
% %               xlim([1 10^4])
%
%               figure
%               scatter(stopRecoilToF,stopRecoilArea,'r.')
%               set(gca,'xscale','log','yscale','log')
%               title('Area vs ToF (log)')
%               xlabel('ToF (ns)')
%               ylabel('Area');
%               legend('Recoil')
% %               xlim([1 10^4])
%
            figure
            h = histogram(stopRecoilEnergy,bins,'EdgeColor','r','FaceColor','
                none');
            hold on
            % fit energy function to histogram
            xvalues = h.BinEdges(1:end-1);
            yvalues = h.Values;
            [xData, yData] = prepareCurveData( xvalues, yvalues );
            % find and remove empty bins
            emptyBins = find(yData == 0);
            xData(emptyBins) = [];
            yData(emptyBins) = [];
%             ft = fittype( 'a*x.^(1/2).*exp(-x/1.42)', 'independent', 'x', 'dependent', 'y' ); % MCNP manual - Maxwellian Spectrum
            ft = fittype( 'a*exp(-x/1.025).*sinh((2.926*x).^(1/2))', 'independent', 'x', 'dependent', 'y' ); % MCNP manual - Watt Spectrum
            opts = fitoptions( 'Method', 'NonlinearLeastSquares' );
            opts.Display = 'Off';
            opts.StartPoint = 10;
            [fitresultEnergyHist, ~] = fit( xData, yData, ft, opts );
            h = plot(fitresultEnergyHist, 'b-', xData, yData, 'b.');
            set(h,'LineWidth',2)
            title('Energy Histogram (linear)')
            xlabel('Energy (MeV)')
            ylabel('Count');
            legend('Recoil','Data for fit','Watt Spectrum')
%             xlim([minEnergy maxEnergy])
            hold off

            figure
            histogram(stopRecoilEnergy,bins,'EdgeColor','r','FaceColor','none');
            hold on
            h = plot(fitresultEnergyHist, 'b-', xData, yData, 'b.');
            set(h,'LineWidth',2)
            set(gca,'xscale','log','yscale','log')
            title('Energy Histogram (log)')
            xlabel('Energy (MeV)')
            ylabel('Count');
            legend('Recoil','Data for fit','Watt Spectrum')
            xlim([minEnergy maxEnergy])
            hold off

            figure
            scatter(stopRecoilEnergy,stopRecoilArea,'r.')
            hold on
            % initial fit - area to energy calibration function
            xvalues = stopRecoilEnergy;
            yvalues = stopRecoilArea;
            [xData, yData] = prepareCurveData( xvalues, yvalues );
            ft = fittype( 'a*x.^(3/2)', 'independent', 'x', 'dependent', 'y' ); % Knoll 2010, p577
            opts = fitoptions( 'Method', 'NonlinearLeastSquares' );
            opts.Display = 'Off';
            opts.StartPoint = 10;
            [fitresultAreavsEnergy, ~] = fit( xData, yData, ft, opts );
            h = plot(fitresultAreavsEnergy, 'b-');
            set(h,'LineWidth',2)
            title('Area vs Energy (linear)')
            xlabel('Energy (MeV)')
            ylabel('Area');
            ylim([0 max(stopRecoilArea)])
            legend('Recoil','Least-squares fit')
%             xlim([minEnergy maxEnergy])
            hold off

            figure
            scatter(stopRecoilEnergy,stopRecoilArea,'r.')
            hold on
            h = plot(fitresultAreavsEnergy, 'b-');
            set(h,'LineWidth',2)
            set(gca,'xscale','log','yscale','log')
            xlim([minEnergy maxEnergy])
            ylim([0 max(stopRecoilArea)])
            title('Area vs Energy (log)')
            xlabel('Energy (MeV)')
            ylabel('Area');
            legend('Recoil')
            hold off

            % hand tweak initial value for proportionality constant
            handtweak = 1;
            while handtweak ~=0
                fprintf(1,'\nChoose new proportionality constant?\n');
                fprintf(1,'    Current value is %f\n',fitresultAreavsEnergy.a);
                handtweak = input('    Enter new value (0 to keep current value
                   ): ');
                if handtweak ~= 0
                    fitresultAreavsEnergy.a = handtweak;
                else
                    break
                end


                figure
                scatter(stopRecoilEnergy,stopRecoilArea,'r.')
                hold on
                h = plot(fitresultAreavsEnergy, 'b-');
                set(h,'LineWidth',2)
                title('Area vs Energy (linear)')
                xlabel('Energy (MeV)')
                ylabel('Area');
                ylim([0 max(stopRecoilArea)])
                legend('Recoil','Least-squares fit')
                hold off

                figure
                scatter(stopRecoilEnergy,stopRecoilArea,'r.')
                hold on
                h = plot(fitresultAreavsEnergy, 'b-');
                set(h,'LineWidth',2)
                set(gca,'xscale','log','yscale','log')
```

```
        xlim([minEnergy maxEnergy])
        ylim([0 max(stopRecoilArea)])
        title('Area vs Energy (log)')
        xlabel('Energy (MeV)')
        ylabel('Area');
        legend('Recoil','Least-squares fit')
        hold off
      end

      % calculate energy histogram from area histogram
      k = fitresultAreavsEnergy.a;
      energyFromArea = (stopRecoilArea./k).^(2/3);
      figure
      henergyFromArea = histogram(energyFromArea,bins,'EdgeColor','r','FaceColor','none');
      hold on
      % fit energy function to histogram
      xvalues = henergyFromArea.BinEdges(1:end-1);
      yvalues = henergyFromArea.Values;
      [xData, yData] = prepareCurveData( xvalues, yvalues );
      % find and remove empty bins
      emptyBins = find(yData == 0);
      xData(emptyBins) = [];
      yData(emptyBins) = [];
%       ft = fittype( 'a*x.^(1/2).*exp(-x/1.42)', 'independent', 'x', 'dependent', 'y' ); % MCNP manual - Maxwellian Spectrum
      ft = fittype( 'a*exp(-x/1.025).*sinh((2.926*x).^(1/2))', 'independent', 'x', 'dependent', 'y' ); % MCNP manual - Watt Spectrum
      opts = fitoptions( 'Method', 'NonlinearLeastSquares' );
      opts.Display = 'Off';
      opts.StartPoint = 10;
      [fitresultEnergyHist, ~] = fit( xData, yData, ft, opts );
      h = plot(fitresultEnergyHist, 'b-', xData, yData, 'b.');
      set(h,'LineWidth',2)
      legend('Recoil','Data for fit','Watt Spectrum')
%       xlim([minEnergy maxEnergy])
      title('Energy Histogram from Area')
      xlabel('Energy (MeV)')
      ylabel('Count');
      hold off

      % correct for Hydrogen cross section - data from https://www-nds.iaea.org/exfor/endf.htm
      % get H cross section data
      scalexAxis = xlsread('D:\Code\ToFSpec\Resources\HnCrossSectionData.xlsx','A:A');
      scaleyAxis = xlsread('D:\Code\ToFSpec\Resources\HnCrossSectionData.xlsx','B:B');
      % find ROI
      [~, scaleMinROIIndex] = min(abs(scalexAxis - min(henergyFromArea.BinEdges)));
      [~, scaleMaxROIIndex] = min(abs(scalexAxis - max(henergyFromArea.BinEdges)));
```

```
            scalexAxis = scalexAxis(scaleMinROIIndex:scaleMaxROIIndex);
            scaleyAxis = scaleyAxis(scaleMinROIIndex:scaleMaxROIIndex);
            % normalize to 1 for scaling
            scaleyAxis = scaleyAxis/max(scaleyAxis);
            % match vector size with energyFromArea histogram BinEdges
            scalexq = linspace(henergyFromArea.BinEdges(1),
               henergyFromArea.BinEdges(end-1),bins);
            scalevq1 = interp1(scalexAxis,scaleyAxis,scalexq,'pchip');
            % build scaling vector
            scaleVector = 1./scalevq1;
            % scale BinValues
            scaleEnergyFromArea = (henergyFromArea.Values).*scaleVector;
            % make plots
            figure
            scatter(scalexAxis,scaleyAxis,'ro')
            hold on
            scatter(scalexq,scalevq1,'b.')
            scatter(scalexq,scaleVector,'y.')
            scatter(scalexq,scaleVector.*scalevq1,'g.')
            legend('effect vector','interpolated effect vector','correction
               vector','corrected effect vector','Location','best')
            hold off

            % refit to corrected calculated energy spectrum
            % fit energy function to histogram
            xvalues = henergyFromArea.BinEdges(1:end-1);
            yvalues = scaleEnergyFromArea;
            [xData, yData] = prepareCurveData( xvalues, yvalues );
            % find and remove empty bins
            emptyBins = find(yData == 0);
            xData(emptyBins) = [];
            yData(emptyBins) = [];
%             ft = fittype( 'a*x.^(1/2).*exp(-x/1.42)', 'independent', 'x', '
    dependent', 'y' ); % MCNP manual - Maxwellian Spectrum
            ft = fittype( 'a*exp(-x/1.025).*sinh((2.926*x).^(1/2))', '
               independent', 'x', 'dependent', 'y' ); % MCNP manual - Watt
               Spectrum
            opts = fitoptions( 'Method', 'NonlinearLeastSquares' );
            opts.Display = 'Off';
            opts.StartPoint = 10;
            [fitresultEnergyHist, ~] = fit( xData, yData, ft, opts );
            figure
            h = plot(fitresultEnergyHist, 'b-');
            hold on
            set(h,'LineWidth',2)
            plot(xData, yData, 'r-');
            legend('Watt Spectrum','Data for fit')
%             xlim([minEnergy maxEnergy])
            title('Energy Histogram from Area - Corrected')
            xlabel('Energy (MeV)')
            ylabel('Count');
            hold off

            choiceLocal=' ';
```

```
%__________________________________________________________________________
        otherwise %Invalid input
            choiceLocal=' ';
    end
end
```

Bibliography

Dubeau, J., Hakmana Witharana, S. S., Atanackovic, J., Yonkeu, A., & Archambault, J. P. 2012, Radiation Protection Dosimetry, 150, 217

ElseNuclear. 2016, Bonner Spheres Spectrometer

ENDF, Nuclear Data Section, I. A. E. A. 2016, Evaluated Nuclear Data File (ENDF)

Knoll, G. F. 2010, Radiation detection and measurement, 4th edn. (John Wiley, Hoboken, N.J.)

Nicholishiell. 2013, Nested Neutron Spectrometer Moderating Cylinder Assembly (nested together)

Qwerty123uiop. 2013, PhotoMultiplierTubeAndScintillator

X-5 Monte Carlo Team. 2003, "MCNP - Version 5, Vol. I: Overview and Theory", LA-UR-03-1987

Zaitseva, N., et al. 2012, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, 668, 88

Index