THE DECAY RATE OF ORTHOPOSITRONIUM

By

Blake K. Winter

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

Bachelor of Science

Houghton College

May 2005

Signature of Author……..……..
Department of Physics
May 9, 2005

……..
Dr. Mark Yuly
Professor of Physics
Research Supervisor

……..
Dr. Ronald Rohe
Associate Professor of Physics

THE DECAY RATE OF ORTHPOSITRONIUM

By

Blake K. Winter

Submitted to the Department of Physics
on May 9, 2005 in partial fulfillment of the
requirement for the degree of
Bachelor of Science

Abstract

Until recently, measurements of the orthopositronium decay rate disagreed with the value predicted by quantum electrodynamics. This paper describes a novel attempt to measure this decay rate. Positrons from the decay of ^{22}Na were slowed in a vacuum chamber containing a sample of aerogel. The release of a positron by the ^{22}Na source was signalled by the detection of a 1.27 MeV gamma ray by a plastic scintillation detector. Orthopositronium typically decays into three gamma rays, which were detected by three NaI detectors. By detecting the annihilation gamma rays in coincidence and using energy information from the events, systematic effects due to contamination by parapositronium decay, which plagued previous measurements, may be reduced. Preliminary analysis of data taken over 200 hours has yielded a value for the decay rate of $7.83 \pm 0.5\ \mu s^{-1}$. Currently, problems with Compton scattering are preventing the use of the energy information in analysis. Possible solutions to this problem will be discussed.

Thesis Supervisor: Dr. Mark Yuly
Title: Professor of Physics

TABLE OF CONTENTS

TABLE OF FIGURES

Chapter 1

INTRODUCTION AND POSITRONIUM THEORY

1.1 History of Positronium

Positrons were first predicted by Dirac [1], using his theory of the spinor field. He initially believed that his theory had predicted protons, despite the prediction that the mass should be the same as that of the electron. The theory required that positrons should exist to preserve causality. The experimental discovery of positrons was made by Anderson [2] in 1932, when measurements of particle tracks through a magnetic field indicated positively charged particles with a very low mass, as would be expected from Dirac's theory. This confirmed Dirac's prediction of a heretofore unknown form of matter. The positron has the same mass as the electron (511 keV/c^2), and the same spin of ½, but the opposite electrical charge, that is +e, while the electron has the electrical charge -e. Electrons and positrons may annihilate, producing gamma rays.

Positronium, the bound state of one positron and one electron, was the first "exotic atom" to be discovered. The discovery was made in 1951 by Deutsch [3]. Deutsch measured the distribution of decay times between the emission of a positron from a ^{22}Na source and the detection of the annihilation gamma rays from positronium formed in N gas. It was discovered that there was a decay curve component inversely proportional to pressure, as would be expected if the positrons were annihilating with atomic electrons. However, there was also a decay curve which was primarily independent of pressure, which was the positronium. Once electrons and positrons bind, their decay rate is independent of pressure, unless the pressure is high enough that there would be many collisions with atoms before the positronium decayed.

Positronium can be found in two spin states. The state with total spin s=0 is known as parapositronium. The state with s=1 is known as orthopositronium. Parapositronium decays very

rapidly, with a measured decay rate of approximately 8 ns^{-1} [4], while orthopositronium has a decay rate closer to 7 μs^{-1}. This time delay is easily measurable with current technology, which combined with the fact that positronium involves only electromagnetic interactions, makes measurements of the decay rate of orthopositronium a good testing ground for Quantum Electrodynamics (QED). The gamma ray spectrum of the orthopositronium decay has been measured and the distribution of the energies of the gamma rays has been confirmed to be in agreement with the phase space predictions [5]. The energy levels of the positronium atoms have also been investigated [6].

1.2 The Orthopositronium Decay Rate Problem

Although QED saw a great deal of success after the work of Feynman [7], Schwinger [8], and Tomonaga [9], its prediction of the decay rate of orthopositronium, initially made using perturbation theory in QED to first order in α by Ore and Powell [10], was not in agreement with the experimentally measured value. The calculation by Ore and Powell gave a decay rate of 7.14 μs^{-1}. However, the experimental results yielded higher results than the theoretical predictions, by as much as 9.4 standard deviations [11]. Table 1 shows some of the previous experimental results.

The current best theoretical prediction [12] of the vacuum decay rate is $7.039934 \pm 0.000010\ \mu s^{-1}$, using perturbation theory in QED to second order in α, while the experimental results have generally been somewhat higher than this. The first modern experiment by Gidley *et al* [13] in 1975 formed positronium in a low density SiO_2 powder, and measured the annihilation gamma rays using three NaI scintillator detectors. The decay rate was measured at varying pressures and the vacuum decay rate was given by extrapolating to zero density. The 1976 measurement by Gidley *et al* [14] was made by forming positronium in an evacuated chamber coated with MgO, to enable the measurement to be made directly in a vacuum. The 1982 measurement by Gidley *et al* [15] used isobutane gas for positronium formation, along with a magnetic field, which increased the path length allowing for more positronium to form. The magnetic field caused a fine-splitting of the states, which was accounted for in analysis. The 1989 measurement by Westbrook *et al* [16] used various gasses, including N, Ne, isobutane, and a mixture of the three, to form the positronium, and then extrapolated to zero pressure.

In 1990 Nico *et al* [17] used evacuated MgO cavities, similar to the 1976 experiment, of varying volume and then extrapolated to infinite cavity size for their measurement.

Table 1. Previous experimental results. Notice that there has been an increase in the precision of the experiments, and that newer techniques have given values closer to the QED prediction.

GROUP	YEAR	METHOD	RESULT	REFERENCE
Gidley *et al*	1975	SiO_2 powder	$7.104 \pm 0.006\ \mu s^{-1}$	13
Gidley *et al*	1976	MgO coated chamber	$7.09 \pm 0.02\ \mu s^{-1}$	14
Gidley *et al*	1982	Isobutane gas	$7.051 \pm 0.005\ \mu s^{-1}$	15
Westbrook *et al*	1989	N, Ne, and isobutane gas	$7.0514 \pm 0.0014\ \mu s^{-1}$	16
Nico *et al*	1990	MgO coated cavities	$7.0482 \pm 0.0016\ \mu s^{-1}$	17
Asai *et al*	1995	SiO_2 powder	$7.0398 \pm 0.004\ \mu s^{-1}$	18
Vallery *et al*	2003	Silica film	$7.0404 \pm 0.0018\ \mu s^{-1}$	19
Asai *et al*	2004	SiO_2 powder	$7.0396 \pm 0.0023\ \mu s^{-1}$	20

All these experiments found values with which Quantum Electrodynamics disagreed. The disagreement was suspected by some to indicate that there must be a large correction from the second order in α terms in QED perturbation theory, but this did not turn out to be the case [19]. This led to speculation that there might be new physics present. There have been searches for decays into neutral bosons [21], neutrinos [22], two photons [23], photons plus neutral particles [24], four photons [25], photons and vector bosons [26], and other exotic three body decays [27]. The only result of these experiments was to place severe upper bounds upon the number of orthopositronium atoms which decay via these channels.

Beginning in 1995, however, several experiments have been done for which the theoretical prediction and the measured value agree. In 1995 Asai *et al* [18] measured the decay rate of orthopositronium in an SiO_2 powder. By using a high-efficiency germanium detector, they were able to eliminate most of

the effects of the orthopositronium annihilating through collisions with nearby atoms. Their results were the first to show good agreement with QED. Vallery *et al* [19] found good agreement in 2003, with a measurement using a nanoporous silica film to form positronium. The silica film slowed the positronium down sufficiently to eliminate most of the interaction with nearby atoms. Finally, Asai *et al* [20] in 2004 performed another experiment similar to the 1995 experiment, but with roughly half of the previous statistical uncertainty.

1.3 Theory of Positronium Formation and Decay

Positronium formation occurs when a positron captures an atomic electron to form a bound state. The theory of positronium formation is reviewed in Ref. [28]. A simplified model for formation of positronium may be understood as follows. Positrons which enter a gas are slowed by collisions with the gas molecules until their energy is lower than the energy required to disassociate positronium, around 6.8 eV. At this energy the positrons can form positronium by capturing atomic electrons. However, if the energy of the positron becomes too low, it is also unable to capture an electron. The energy range in which positronium formation is possible is known as the "Ore gap." Generally around 20% of the positrons entering a gas will form positronium [29]. If the positron's energy is below the "Ore gap," however, it cannot capture an electron and will annihilate directly with an atomic electron. The decay rate for this process is faster than that of orthopositronium, however, and can be distinguished in analysis. Positronium formation generally occurs in less than 1 ns, so it is negligible when measuring the decay rate of orthopositronium [28].

When positronium decays, it obeys the charge-conjugation selection rule,

$$(-1)^{\ell+s} = (-1)^n, \qquad (1)$$

where ℓ is the orbital angular momentum, s is the total spin, and n is the number of gamma rays emitted. Since $\ell = 0$ in the ground state, parapositronium (with s=0) must decay into an even number of gamma rays, while orthopositronium (with s=1) must decay into an odd number. For parapositronium the most likely decay channel is therefore into two gamma rays, and from energy-

momentum conservation these two gamma rays must be back-to-back, each with half the energy of the positronium atom, or about 511 keV. Since decay into a single gamma ray is forbidden by momentum conservation, orthopositronium must decay into at least three gamma rays. Momentum conservation also requires that each gamma ray must have less than 511 keV of energy.

As shown diagrammatically in Figure 1, the energy of photon p_2 may be calculated given the energy of photon p_1 and the angle θ, for the decay of orthopositronium into three gamma rays.

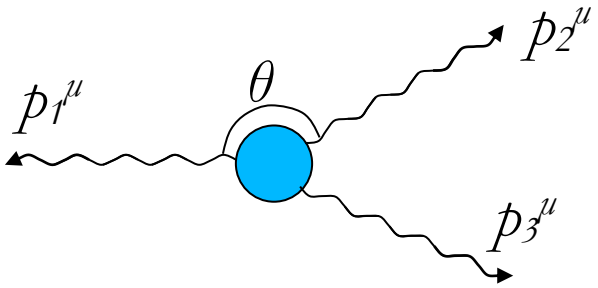

Figure 1. The three gamma decay of orthopositronium. In the center is the positronium atom. The three gamma rays $(p_1^{\mu}, p_2^{\mu}, p_3^{\mu})$ are shown being emitted. The angle between $\vec{p}_1$ and $\vec{p}_2$ is θ.

To derive this equation, start with the conservation of four-momentum for photons,

$$p_1^{\mu} + p_2^{\mu} + p_3^{\mu} = p^{\mu},$$

where the p_i are the four-momenta of the three annihilation gamma rays, i=1, 2, 3, which is given in terms of the three momentum $\vec{p}_i$ by $p_i^{\mu} = (p_i, \vec{p}_i)$. The energy of a photon is given by $E_i = p_i c$. Since the positronium is at rest, its four-momentum is approximately given by $p^{\mu} = (2m_e, \vec{0})$, where m_e is the mass of the electron, neglecting the binding energy of the positronium atom. Rearranging the above equation in order to eliminate p_3^{μ}, which is not measured, yields

$$p_1{}^{\mu} + p_2{}^{\mu} - p^{\mu} = -p_3{}^{\mu}.$$

Next p_3 may be removed from the equation by squaring. Letting $M = 2m_e$,

$$M^2c^2 + 2p_1{}^{\mu}p_{2\mu} - 2p_1{}^{\mu}p_{\mu} - 2p_2{}^{\mu}p_{2\mu} = p_3{}^{\mu}p_{3\mu} = 0.$$

Now we know that $p_1{}^{\mu}p_{2\mu} = p_1p_2(1-\cos\theta)$, and $p_1{}^{\mu}p_{\mu} = p_1Mc,$ so solving for p_2 yields

$$p_2 = \frac{p_1Mc - \frac{1}{2}M^2c^2}{(1-\cos\theta)p_1 - Mc}. \qquad (2)$$

This relationship will be useful in analyzing the data, because when two or more gamma rays are detected in coincidence, their energies can be checked against this relationship to confirm that they are actually from an orthopositronium annihilation event.

Once the positronium has formed, it will decay according to the exponential decay law,

$$N = N_0e^{-\lambda t}.$$

In this formula λ is the decay rate, N_0 is the initial number of positronium atoms, and N is the number of positronium atoms remaining at time t. The decay rate in a gas is always higher than the decay rate in the vacuum, however, because the positron may annihilate with a nearby, bound electron. The usual procedure used in prior experiments in gas has been to measure the decay rate at several pressures and use a linear fit to extrapolate to zero pressure.

Another problem which can occur (especially in a system contaminated with an oxygen-rich gas such as air) is called spin-exchange. Molecular oxygen has two unpaired spins, and it therefore efficiently converts orthopositronium into parapositronium. The parapositronium will decay almost immediately, which makes it appear that the orthopositronium has a faster decay rate than it actually does. This

makes it important to avoid having oxygen or similar molecules in the gas. Another issue is the thermalization of the positron. If the positron loses energy until it is below the "Ore gap," then it can only lose further energy by scattering elastically, and will take a relatively long time to reach thermal energies. After this, the positron will annihilate with a relatively low decay rate. This can distort the measurement of the decay rate as well. Often isobutane (whose electron configuration does not cause spin conversions) will be used, as it exhibits this effect much less than many other gases. Using a solid material instead of a gas, and using energy information to distinguish the different types of annihilation events, can help to reduce these types of systematic uncertainty.

Chapter 2

DESCRIPTION OF EXPERIMENTAL APPARATUS

2.1 Overview of the Experiment

The novel idea in this experiment is to attempt to eliminate the accidental parapositronium events by measuring the gamma ray energies and using the energy relationship in Eq. (2). Using this relationship it should be possible to check events with multiple gamma ray detections to confirm that they came from an orthopositronium annihilation event, and not from a parapositronium conversion event or accidental coincidence (e.g. from a cosmic ray) by examining the energies of the detected gamma rays.

In order to create positronium, a positron source is needed. In this experiment, the positron source was ^{22}Na. The decay scheme for ^{22}Na is shown in Figure 3. The half life of ^{22}Na is 2.6 years, and it decays into ^{22}Ne. For 90.5% of these decays, the channel is positron (β+) emission and decay to an excited state of ^{22}Ne, followed by a subsequent 1.275 MeV gamma ray. Therefore about 90.5% of the time, the detection of a 1.275 MeV gamma ray signals the release of a positron into the system.

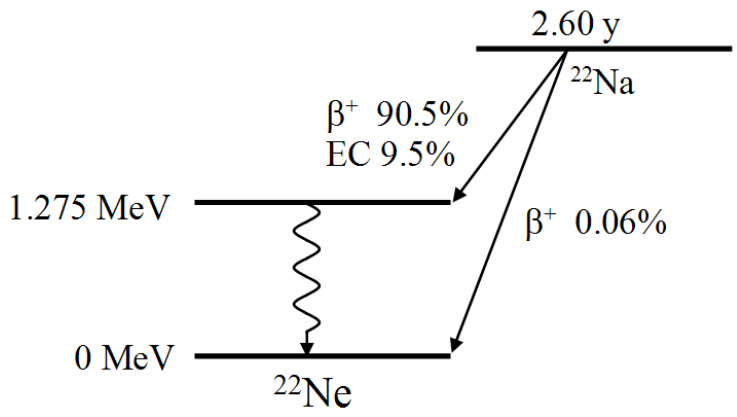

Figure 2. The decay scheme for 22Na [30]. About 90.5% of these decays involve the release of a positron and a 1.275 MeV gamma ray. The diagram is taken from Ref. [31].

2.2 Detectors and Vacuum System

In order to conduct the experiment, a vacuum chamber to form positronium and detectors to look for annihilation gamma rays were needed. The experimental chamber and the detectors are shown in Figure 4. The large clear dome in the center of the photograph is the vacuum chamber, a Nalgene Vacuum Chamber (Cat. No. 5305-0509). The chamber had a height of 23.7 cm, a diameter of 17 cm, and a wall width of 2 mm. This was pumped down with a Sargeant Welch Duo Seal vacuum pump, connected with rubber hoses to the chamber, and an air intake valve. The pressure was reduced to less than 10^{-4} torr over a period of a day. The pressure gauge (an Infinicon FL 9496 Hot Ion Combi Gauge) uses a thermocouple gauge and an ion gauge, but the ion gauge was broken. This prevented the actual pressure from being measured, since it resulted in the gauge not being able to operate at pressures lower than 10^{-4} torr. In the center of the vacuum chamber there was a small plastic cup, with a diameter of 5 cm, filled to a height of about 5 cm with aerogel fragments. Aerogel [32] is a very light solid, and is composed of thin cell walls with pockets between them. Once the air was removed from the vacuum chamber, these pockets were empty. Underneath the cup there was a clear disk, which contained ^{22}Na, with an activity of slightly less than 1 μCi. The source was constructed by evaporating a solution of ^{22}NaCl and hydrochloric acid onto the plastic disk. Between the source and the cup there was a thin layer of plastic wrap, which was used to slow the positrons before they entered the aerogel. Below the ^{22}Na source and outside the vacuum chamber was a plastic scintillator detector, Bicron Corp. Bc400, with a diameter of 5 cm and a length of 15 cm. The detector cannot be seen in Figure 3, but the photomultiplier tube (PMT) base to which it was attached may be seen protruding out to the right. Around the plastic scintillator there were lead bricks to reduce the accidental events in the plastic scintillator and support the vacuum chamber. The plastic detector was used to detect the 1.275 MeV gamma rays which signal the probable release of a positron from the ^{22}Na source. The rise time on the photomultiplier tube for the plastic scintillator was about 4 ns [33]. The ^{22}Na source was sitting on top of two lead rings, which allow the gamma ray to reach the plastic scintillator.

The experiment operated as follows. The positrons traveled up from the ^{22}Na source and into the cup. There they lost their kinetic energy in collisions with the cell walls until they were moving slowly

enough to capture an atomic electron. Once this occurred, the positronium drifted into the vacuum pockets in the aerogel and decayed there.

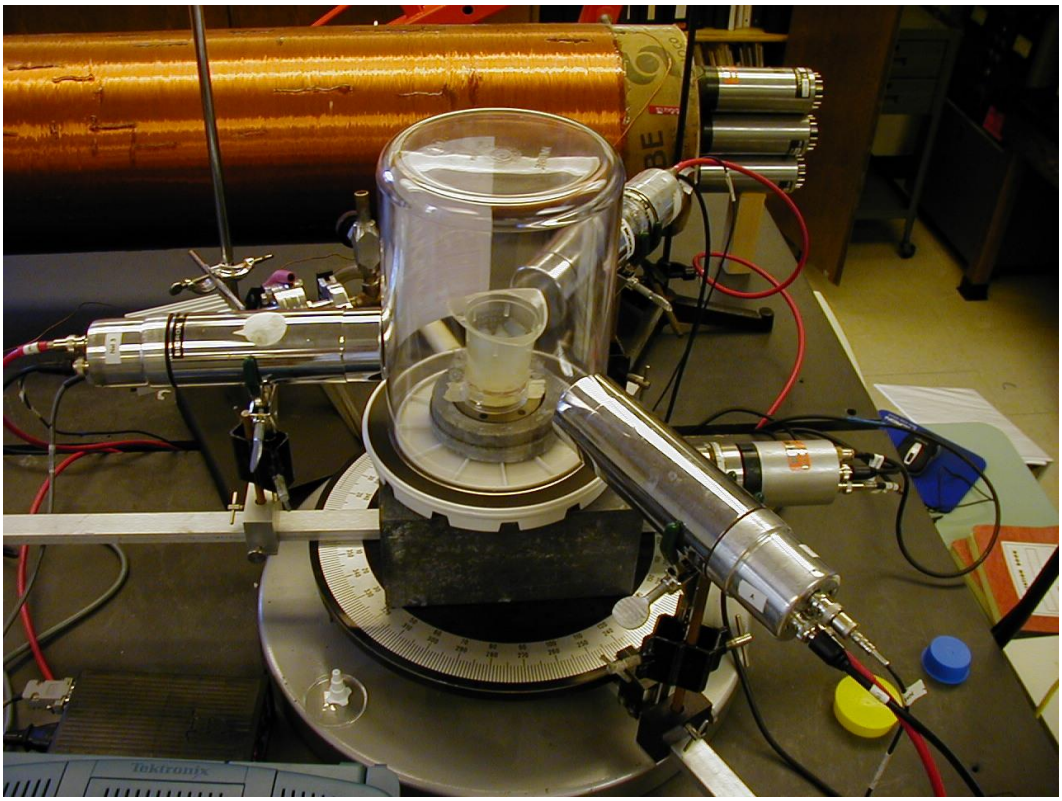

Figure 3. A photograph of the detectors, the vacuum chamber, the source, and the aerogel (used for positronium formation). Positrons released by the ^{22}Na source under the plastic cup entered the aerogel where they were slowed and formed positronium. The annihilation gamma rays were detected by the surrounding NaI scintillators. The 1.27 MeV gamma ray released by the ^{22}Na were detected by the plastic scintillator underneath the vacuum chamber.

Around the vacuum chamber were three NaI scintillator detectors (Bicron Corp. Model 2M1/2), with a diameter of 2" and a thickness of 1". These detected the annihilation gamma rays from the positronium decay events. For all the data which have been collected, the detectors were each placed

with a 120 degree angle between them. The angle was measured by using the calibrated angle plate underneath the detectors, shown in Figure 3. The plate had a mark at each degree. Figure 4 and Figure 5 show diagrams of the experimental setup.

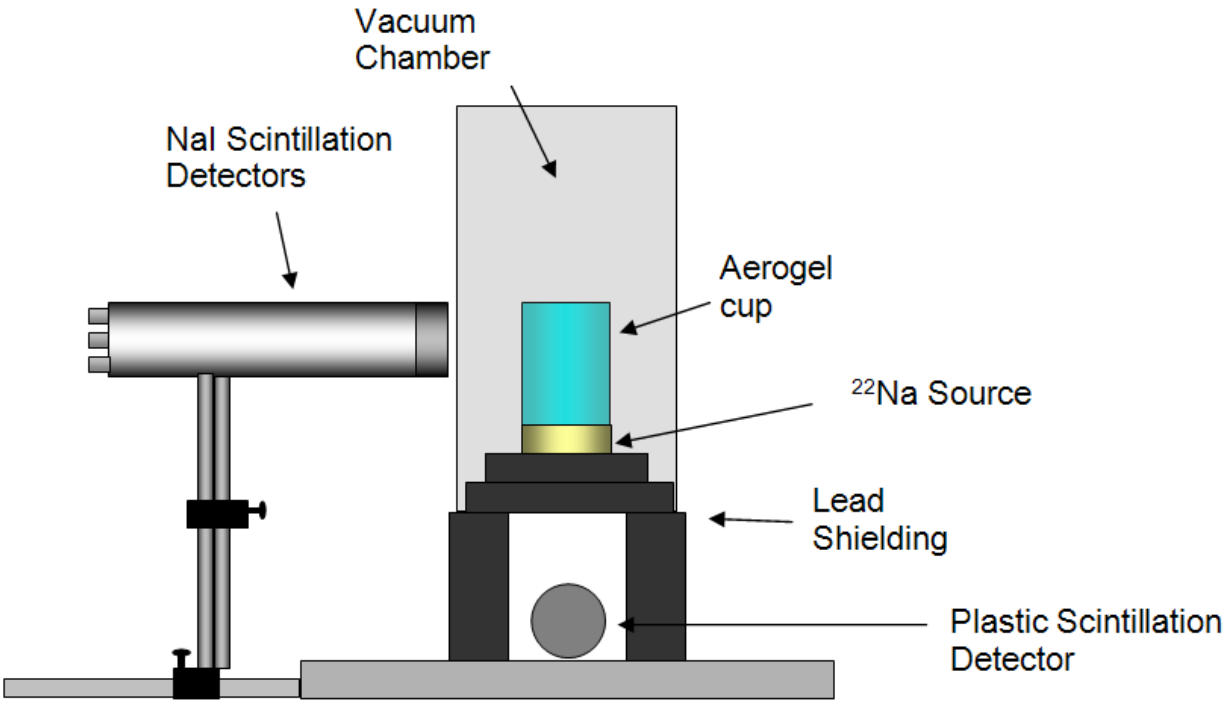

Figure 4. Side view of the experimental setup. The plastic detector is shown at the bottom, underneath the ^{22}Na source. Above the ^{22}Na source is the cup filled with aerogel. Positronium was formed in the cup and the annihilation gamma rays were detected by the NaI detectors.

2.3 Scintillator Detectors

2.3.1 Operation

The gamma ray detectors used in this experiment were scintillation detectors. A diagram of a scintillator is shown in Figure 6.

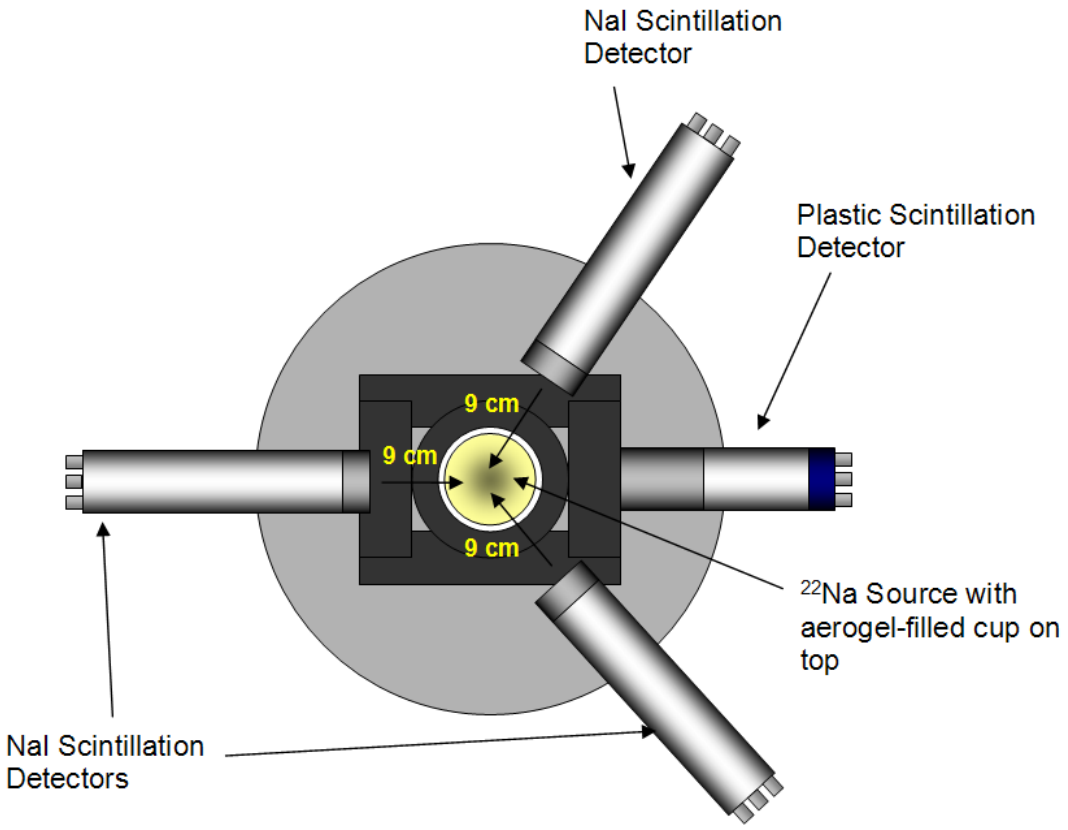

Figure 5. Top view of the experimental setup. The ^{22}Na source is shown in the center, over the plastic scintillator. Above the ^{22}Na source was the cup filled with aerogel. Positronium was formed in the aerogel and the annihilation gamma rays detected by the surrounding NaI scintillation detectors, which were placed with an angle of 120 degrees between them.

The gamma ray travels into the scintillator, where it may interact through Compton scattering or the photoelectric effect, knocking out an atomic electron. In an inorganic scintillator such as the NaI crystal scintillator, this raises electrons to the conduction band, and the following deexcitation process releases low energy photons [34]. In an organic scintillator such as the plastic scintillator, the ionizing radiation causes an excitation of the molecules, which then release low energy photons in deexcitation [34]. The scintillator material is transparent to allow the photons to travel freely within it. The scintillator's index of refraction can cause near total internal reflection so that the light does not escape. Also, the scintillator is surrounded by a reflective foil which reflects light back into the

scintillator. The scintillator is connected to a light pipe, which is a transparent plastic through which the light can travel into the photomultiplier tube.

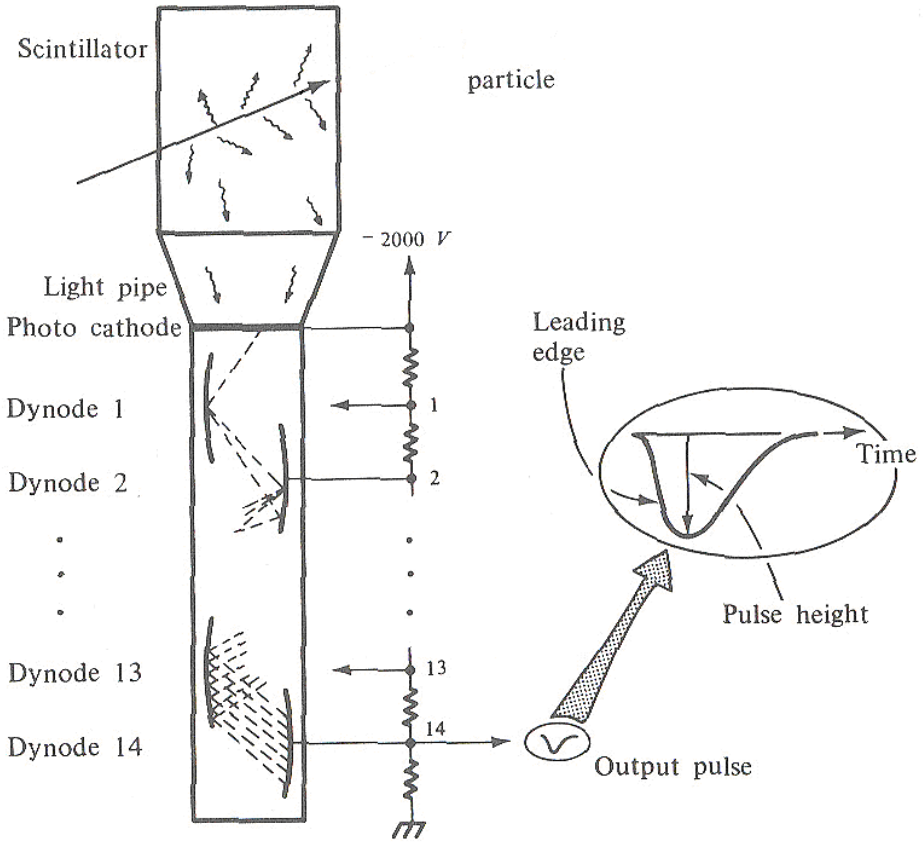

Figure 6. Diagram of a scintillator detector with attached photomultiplier tube. The pulse shown is a voltage/time graph. The diagram is taken from Ref. [35].

The photomultiplier tube has a photocathode which releases electrons by the photoelectric effect. The cathode is at a negative potential. In the PMT there are a series of dynodes, each of which is at a higher (that is, less negative) potential than the one preceding it. The electrons released from the

photocathode gain energy from the potential difference as they travel to the first dynode. They strike the dynode which causes more electrons to be released, because the incident electrons have the added energy from the potential difference. The increased number of electrons travels to the second dynode, where the process repeats itself in an "electron cascade." The current is turned into a voltage pulse using a resistor. The voltage as a function of time of the resulting pulse is also shown in Figure 6. The "tail" is caused by capacitance in the PMT, which causes current to flow for a short time after the photo cathode has stopped releasing electrons. The height of the pulse is proportional to the energy lost by the particle in the scintillator, since more light will cause more electrons to be released. Diagrams of the PMT bases are shown in Appendix B.

2.3.2 Energy Resolution

The energy resolution of the detectors was measured by using a ^{22}Na source and measuring the width of the energy peak at half of the maximum height (the "Full Width at Half Maximum," or FWHM). The PMT for the plastic scintillator was run -1200 V, the PMT's for the NaI detectors 1 and 2 were run at -1060 V, and the PMT for the NaI detector 3 was run at -1000 V. It was found that the plastic detector has a FWHM of 20% at an energy of 1.27 MeV. The peaks from the NaI detectors have a FWHM of 9.6% at an energy of 511 keV. The energy spectra which were used to determine the energy resolution are shown in Figure 11 and Figure 12. Sources of known energy were used to calibrate the energy measurements, by locating their peaks in the spectra from the detectors.

2.3.3 Timing Resolution

The timing resolution and calibration was particularly important, since it determined the decay rate. The calibration of the timing was done by placing a ^{22}Na source between the plastic scintillator and each NaI detector. The positron annihilations result in back-to-back 511 keV gamma rays which were detected in each detector simultaneously. The signal from the plastic detector was then delayed by amounts varying from 100-800 ns with a Lecroy 2323A gate generator. The signal from the NaI detector was used to start the Lecroy 3377 Time-to-digital converter (TDC), and the signal from the plastic was used to stop it. The timing was found to have a calibration of 1 ns/channel to within an

uncertainty of less than 1% (see Figure 7). The timing resolution was found to be 9 ns, by measuring the FWHM of the peaks from Figure 7, as in Figure 8.

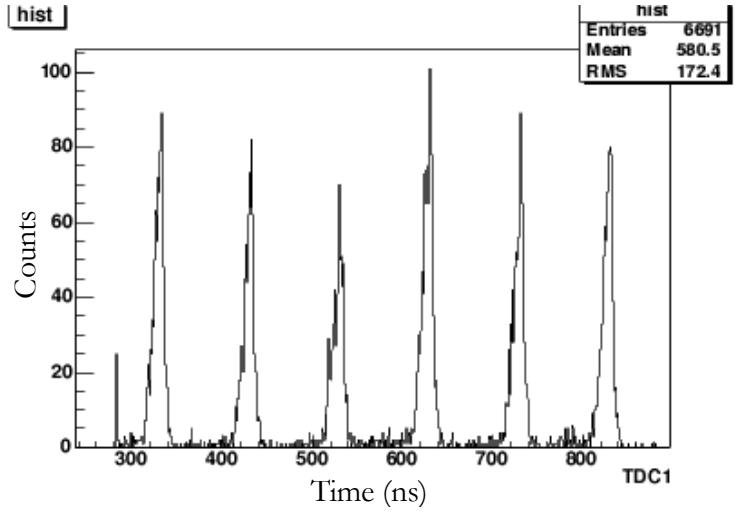

Figure 7. The calibration of the timing. This plot was made with the ^{22}Na source placed between the NaI and plastic. The signals were delayed by 300 to 800 ns using a gate generator. The calibration was found to be 1 ns/channel.

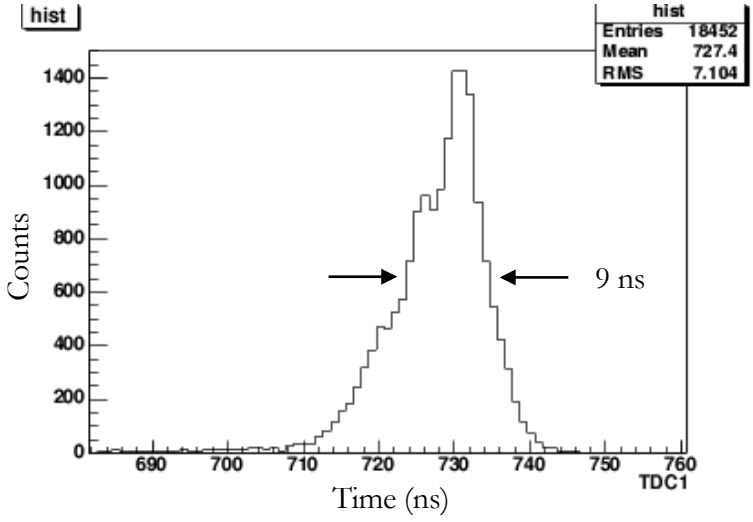

Figure 8. Timing resolution. The FWHM of one of the coincidence peaks from Figure 7 was used to obtain the timing resolution, approximately 9 ns.

2.4 Electronics

The electronics system was designed to record any events which might correspond to a positronium decay. It used NIM modules, CAMAC modules, and a CAMAC crate, which was read out by a computer which recorded the data. A diagram of the electronics is given in Figure 10.

2.4.1 Pulse Height

The photomultiplier tubes (PMTs) on detectors NaI 1 and NaI 2 were powered by a Tennelec TC 952A at -1060 V. The PMT for NaI 3 was powered by an Ortec 456 set at -1000 V, and the PMT for the plastic scintillator was powered by a Mechtronics 257 high voltage supply at -1200 V.

The PMT bases have been modified to have dynode and anode outputs (see Appendix B), in order to match the output impedance with the input impedance of the electronics. The signal from the dynode of each PMT was shaped by an Ortec 113 preamplifier and amplified with Ortec 485 amplifier. The signal then went into an Ortec AD413A Amplitude-to-Digital Converter (ADC). The ADC digitized the height of the pulse, which was proportional to the energy deposited in the detector, and records the height in digital form.

2.4.2 Timing

The anode outputs were used for two purposes, one of which was for timing information. The anode outputs of the NaI detectors are amplified using a Lecroy 612A photomultplier amplifier and then discriminated using an Ortec 473A constant fraction discriminator (CFD). The CFD output a logic pulse if it received an input pulse over a certain voltage. The output of the CFD comes at a constant fraction of the pulse height, which makes the timing independent of the energy of the pulse. This increases the timing resolution of the system. The discriminators were set to their lowest possible setting, because low energy gamma rays can signal an orthopositronium decay event. The logic pulses were then converted from NIM to ECL using a Lecroy 4616 ECL-NIM converter. The signals were used to start a Lecroy 3377 Time to Digital Converter (TDC). The TDC was operated in Common Stop mode, which meant that it measured the time between each start signal and a common stop signal, which was provided by the signal from the plastic detector.

The plastic detector's anode output was discriminated using an Ortec 436 100 MHZ discriminator. The signal from the discriminator triggered a Lecroy 2323A gate generator, which produced a delayed logic pulse. This pulse was converted from NIM to ECL by a Lecroy 4616 converter, and then went into the Common Stop on the TDC. The pulse was delayed in the gate generator sufficiently that it always came after the signals from the NaI detectors had started the TDC timers. Since the plastic detector received the 1.27 MeV event before the NaI scintillators detected the annihilation gamma rays, the signal from the plastic was delayed so that it always triggered the stop after the NaI detectors have triggered the start. The measured time difference was linearly correlated with the time difference between the detection in the plastic detector and the detection in the NaI detector. However, the value reported clearly decreased as the time difference between events increased, so a small number in the TDC corresponded to a large time difference between the plastic event and the NaI event.

2.4.3 Trigger

The trigger was designed so that data were collected whenever there was an event in the plastic detector followed within 1000 ns by an event in any one of the NaI detectors.

The discriminated signals from the NaI detector's anode outputs were ORed with a Lecroy 365AL 4-Fold Logic Unit, and the output triggered an Ortec 416A gate generator. The discriminated signal from the plastic detector's anode output triggered an Ortec 416A gate generator, and the output of this was ANDed with the logic pulse from the NaI detectors. The output of the AND gate was used to gate the Ortec 413 ADC, which signaled the computer whenever data was ready to be read out.

In order to make certain the trigger would function properly, the timing was set using an oscilloscope, as in Figure 9. Line 3 is the trigger from the plastic detector. Line 2 is the trigger from the NaI detectors. These must come at the same time in order to make a gate for the ADC (line 1). Line 4 is the pulse whose energy was to be measured. Since the Ortec 413 measures the height of the pulse, only the peak needed to come inside of the gate.

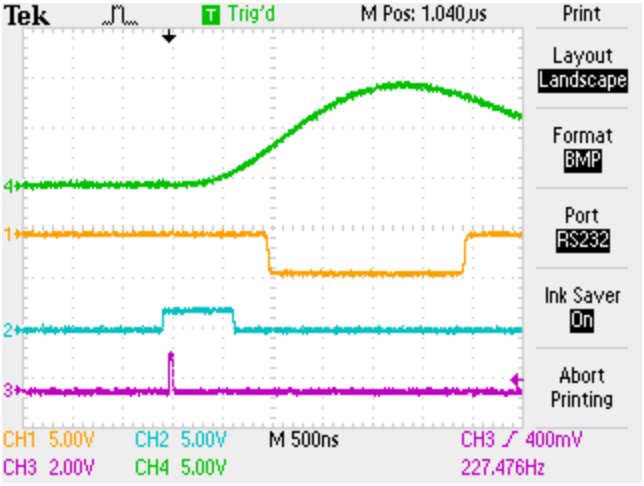

Figure 9. The timing of the pulses on the oscilloscope. Line 3 is the trigger from the plastic detector. Line 2 is the trigger from the NaI detectors. Line 1 shows their combined gate for the ADC. Line 4 is the ADC input which was to be measured. Since the Ortec 413 functions by measuring the pulse height, only the peak needed to come inside the gate.

2.4.4 Computer Acquisition

When the ADC received a gate and digitized the pulse height, it triggered the computer that data was ready to be collected by setting the CAMAC Look-At-Me (LAM). The ADC and the TDC were in a Jorway Minicrate Model 465 CAMAC crate, with a Jorway 73A Crate Controller [36] which was connected to a computer with a SCSI interface. The computer, a Brite ISO 9002 with a Pentium II 400 MHz processor, used Red Hat Linux version 7.3 with the 2.4.18 kernel. The data acquisition program was written in C++ using ROOT [37] histogramming and GUI classes, with the sjy package [38] for CAMAC dataway routines. It read the data from the ADC and TDC out through Jorway 73A controller using the SCSI interface with an Adaptec AID 7880 SCSI card. The computer then recorded the data on hard disk for later analysis. Files created for two-day running periods were approximately 3 MB in size. The code used to read the data is given in Appendix A.

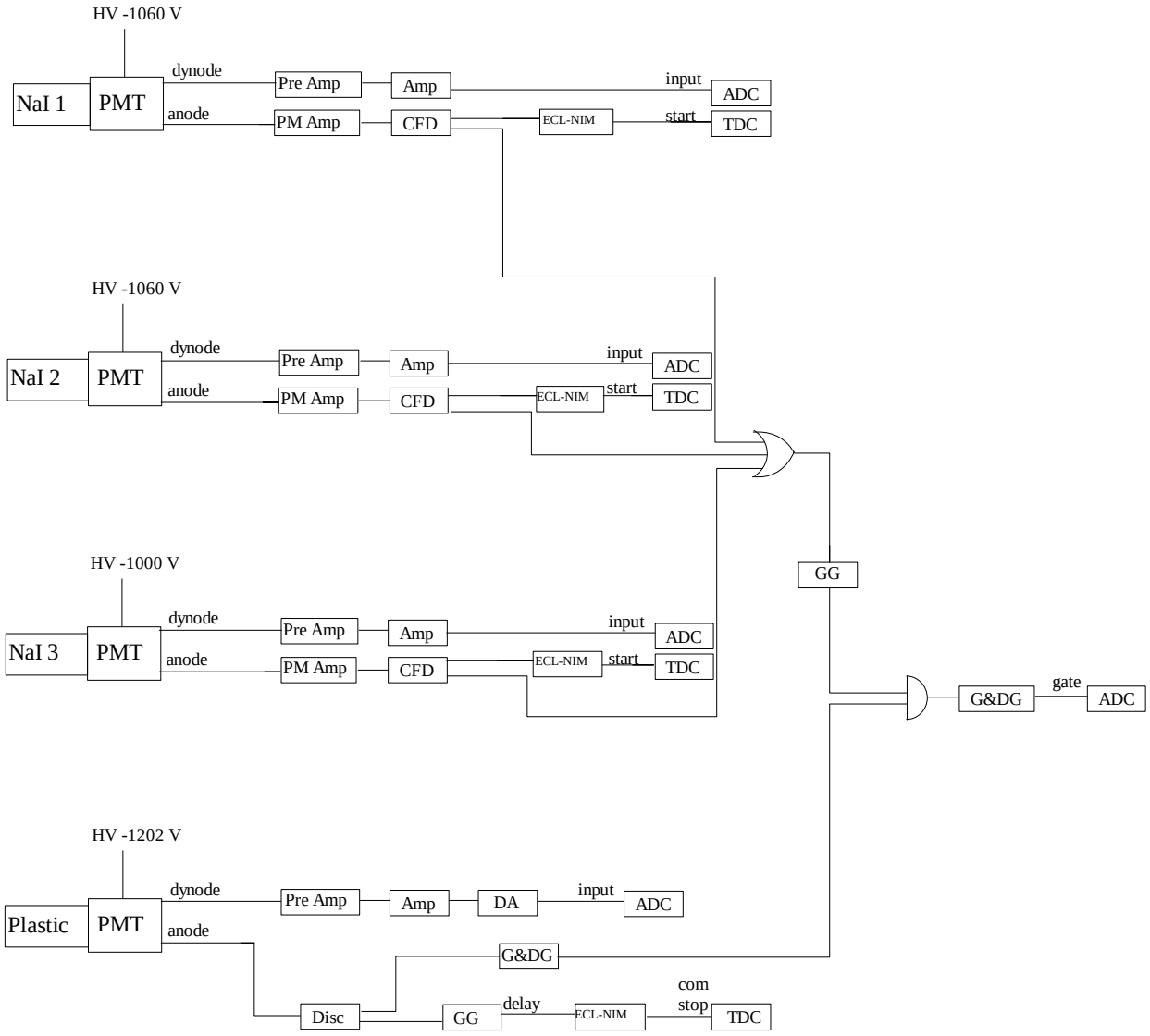

Figure 10. Circuit diagram of the electronics. The scintillator detectors produced light pulses which are converted to electrical pulses by the photomultiplier tubes (PMT). The dynode signals were amplified (AMP) and digitzed by an analog-to-digital converter (ADC). The anode signals from the NaI detectors were amplified (pre-amp) and discriminated (CFD), then used to start the time-to-digital converter (TDC). The anode signal from the plastic detector was discriminated (Disc) and delayed in a gate generator (GG), then used to stop the TDC. Data collection was triggered by an event in the plastic detector and at least one NaI detector simultaneously.

Chapter 3

EXPERIMENTAL RESULTS AND CONCLUSION

3.1 Decay Rate Results

The operation of the experiment and the results of the preliminary analysis are described in this section. Data were collected for approximately 200 hours in February and March of 2005. The decay rates were determined with a ROOT [37] interpreter C++ code (see Appendix A). In the preliminary analysis, the decay rate was calculated for each detector separately, and a weighted average was taken to find the final decay rate. The data were tested so that events would only be used in the analysis of the decay rate if the plastic scintillator received a 1.27 MeV gamma ray and if the gamma rays in the NaI detectors had an energy less than 511 keV. Since parapositronium decays into two gamma rays with an energy of 511 keV, but orthopositronium emits three gamma rays each with energy less than 511 keV, this reduced the number of background parapositronium events which were included in the measured orthopositronium decay curve. Unfortunately with 9.6% energy resolution this resulted in losing some orthopositronium events.

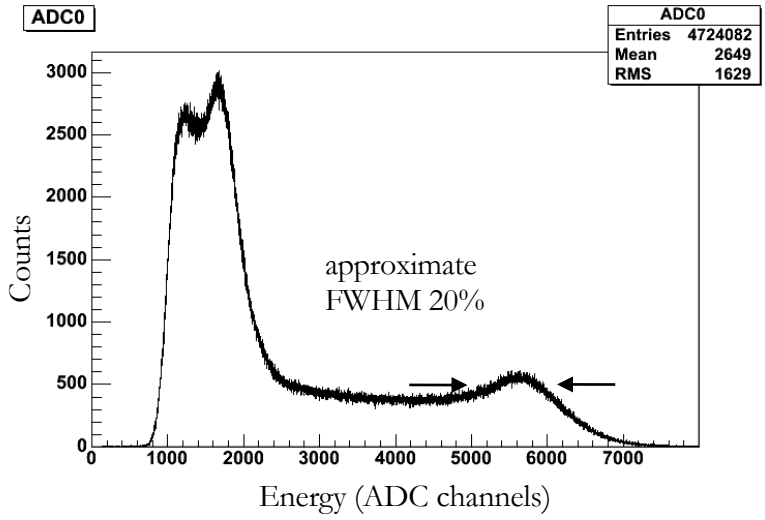

Figure 11. The energy spectrum from the plastic scintillator. The peak around channel 5600 corresponds to the 1.275 MeV peak of ^{22}Na which signals the probable release of a positron. The low energy peak was due to the 511 keV gamma rays from positron annihilations.

In Figure 11 the energy spectrum produced by the plastic detector may be seen. The left peak is the 511 keV peak from the positron annihilations, while the peak on the right around channel 5600 is from the 1.275 MeV gamma rays which signal the probable release of a positron into the system.

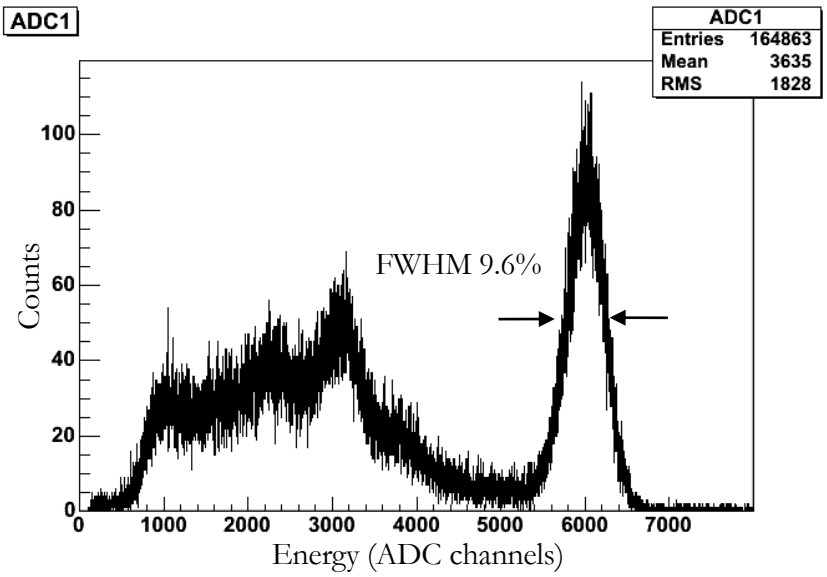

Figure 12. Energy spectrum of one of the NaI scintillators. The peak near channel 6000 is the 511 keV peak from parapositronium annihilations.

Figure 12 is a spectrum of the energy from one of the NaI scintillators. The only events plotted are the ones for which there was also a 1.27 MeV event in the plastic detector signaling the probable release of a positron within 1000ns. Figure 13 is the spectrum from one of the NaI scintillators, for events in which the plastic scintillator had an event between 4000 and 8000 channels (the 1.27 MeV range) and for which the decay time of the event was longer than that of parapositronium but less than several half-lives of orthopositronium.

Figure 14, Figure 15, and Figure 16 show the plots for the decay time spectra. The events in these plots required a simultaneous 1.27 MeV event in the plastic scintillator and a NaI event with less than 511 keV. The decay rate was found by using a ROOT interpreter C++ computer program (shown in Appendix A) to fit a function of the form $Ae^{\lambda t} + B$ to the data. The *B* in this equation accounted for

the background events. The exponent is positive and not negative since the decay time is backwards because of the common-stop electronics.

The analysis of data from the first NaI detector (shown in Figure 14) found a lifetime of 122±11ns. Since the decay rate is the inverse of the lifetime, the decay rate from detector one was therefore 8.2±0.74 μs^{-1}. This is a slightly higher decay rate than predicted by QED or measured in previous experiments.

In Figure 15 the decay curve taken from the second NaI detector is shown. The fit for this curve was found to be 7.5±0.76 μs^{-1}. This shows good agreement with the results from detector 1, and also agrees with previous experiments and with QED.

The results from the third NaI scintillator are shown in Figure 16. The decay rate was found to be 7.56±1.5 μs^{-1}. This had a much higher statistical uncertainty than either of the other detectors, perhaps because of problems with the detector, which also produced an energy spectrum slightly different than the others.

Taking a weighted average of the results yielded a decay rate of 7.83±0.5 μs^{-1}. This is a larger uncertainty than other modern measurements, in part due to the short time over which the data was collected. However, the results so far are encouraging, since they are consistent with previous measurements of the decay rate of orthopositronium. It is to be hoped therefore that as more statistics are gathered the uncertainty can be reduced to a degree where it will be possible to make a meaningful comparison of the predictions of QED to the measured decay rate.

3.2 Energy Conservation Results

The unique aspect of this measurement was to be the technique of checking energy and momentum conservation using Eq. (2) for events in which multiple gamma rays were detected. To this end, events with multiple annihilation gamma rays were analyzed and the measured energies were compared to the predictions of Eq. (2).

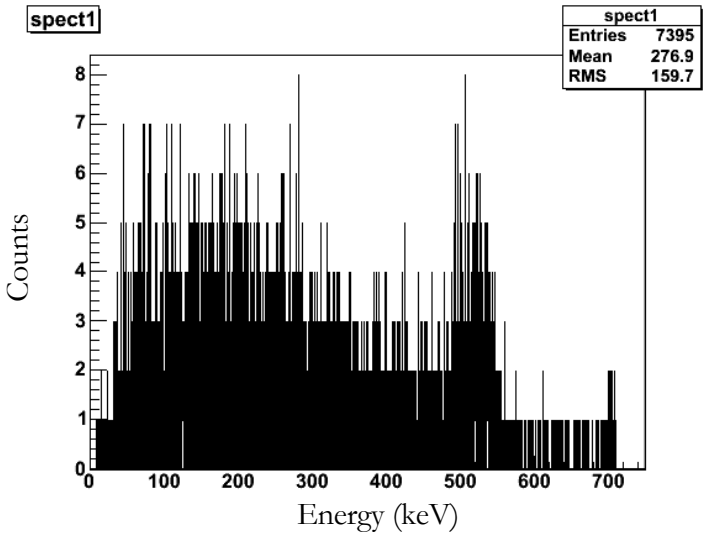

Figure 13. The energy in one of the NaI detectors, for events in which the plastic detector had a 1.27 MeV gamma ray and the decay time was consistent with an orthopositronium decay.

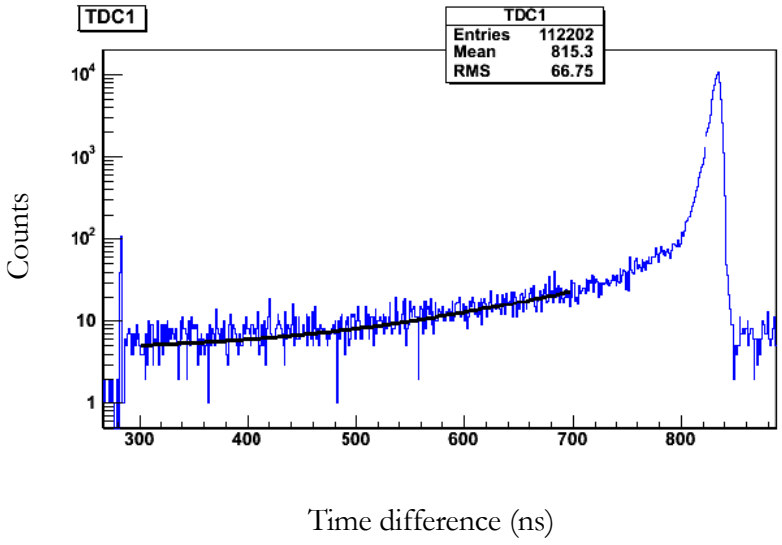

Figure 14. Decay curve from detector 1. The peak around 800 ns is the fast decay curve from the parapositronium. The black curve is the fit. The decay rate was found to be 8.2±74 μs^{-1}. Points used were those between channels 300 and 700, which correspond to orthopositronium decay.

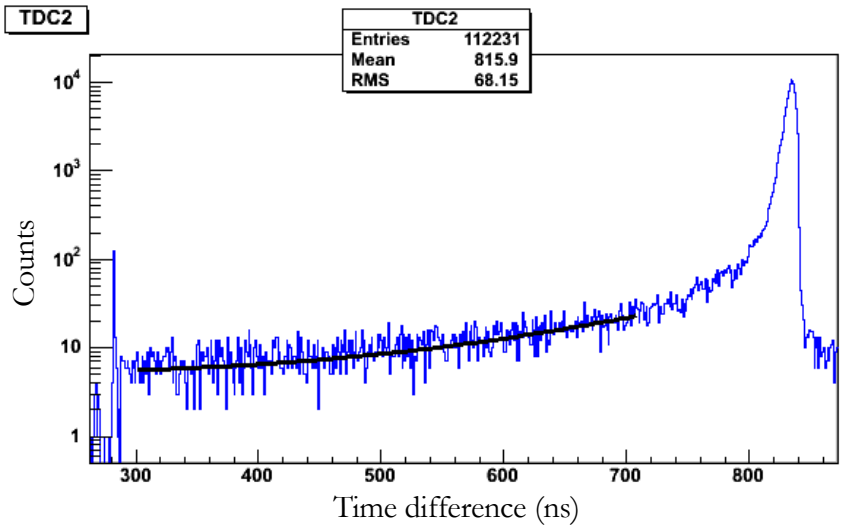

Figure 15. Decay curve for detector 2. The points used for the fit were between channels 300 and 720. The black line is the fit.

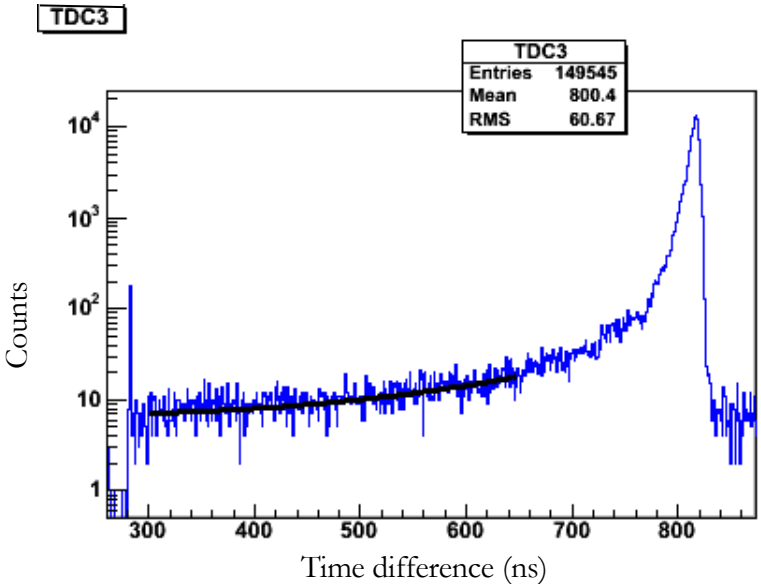

Figure 16. Decay curve from NaI detector 3 (counts/ns). The points used in the fit were those between channels 300 and 720. The black line is the fit.

The first step in this process was to check that the decay gamma rays were coming from the same orthopositronium decay event. To do so, the time difference between the signals from the NaI detectors which had recorded an event was plotted to make sure that both detectors had fired at the same time. An example of the results is shown in Figure 17. Almost all of the events came within a few ns of each other, which shows that they were unlikely to have come from accidental coincidences.

Having checked to make sure that coincidence events were coming from positronium decay, it was possible to predict the energy in one detector given the energy in another detector and the angle between the detectors, using Eq. (2). The difference between the energy in detector NaI 1 and the energy predicted by Eq. (2) using energy information from detector NaI 3 is plotted in Figure 18.

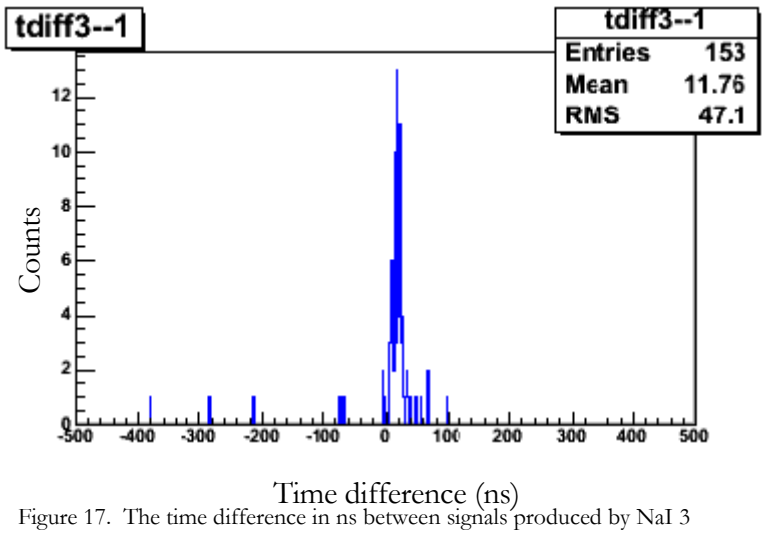

Figure 17. The time difference in ns between signals produced by NaI 3 and NaI 1, for events in both detectors satisfying energy and timing requirements to be an orthopositronium event. Notice that there are a fairly small number of coincident events. Also notice that most events come within approximately 50 ns of each other. This shows that most coincidences are coming from positronium decay events.

The plot shows very poor agreement between the predicted and the measured energy. The prediction is roughly 100 to 200 keV higher than the measured energy. It is thought that this is a result of Compton scattering out of the detectors. If this occurs then only part of the gamma ray energy will be

detected. Compton scattering in either of the detectors will result in the predicted energy being higher than the detected energy. There are two cases: Compton scattering in the detector being used to make the prediction, and Compton scattering in the detector whose energy is predicted. If Compton scattering occurs in the detector which is being used for the prediction, then the energy used for predictions will be lower than the energy of the gamma ray, which from Eq. (2) shows that the predicted energy will be higher than the energy which is measured. If the gamma ray in the detector whose energy is predicted Compton scatters, then clearly the measured energy will be less than the predicted. Either way, the predicted energy will be higher if there is Compton scattering in either detector.

This means that using the current setup, it will be very difficult to use energy information from coincidence events to check whether the events are from orthopositronium. If this technique is to be implemented, a new system will be needed. Possible solutions are discussed in section 3.3.

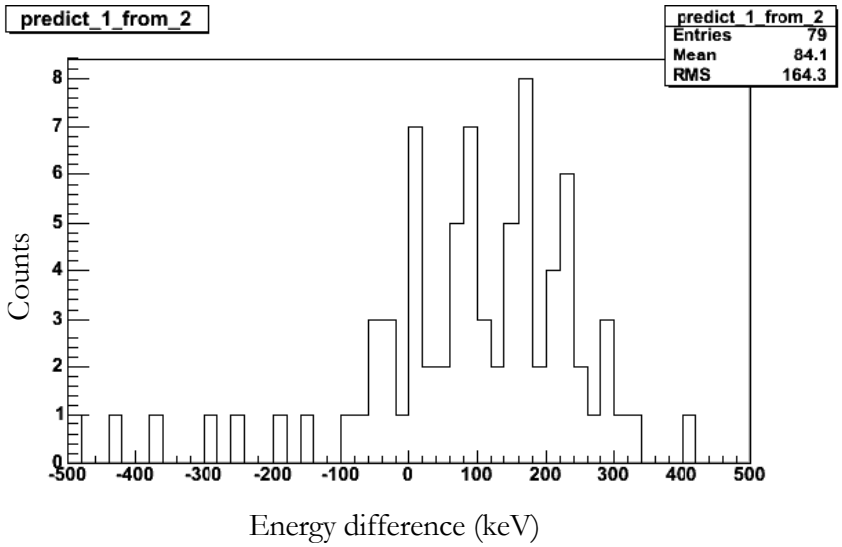

Figure 18. Histogram of the difference between the predicted energy and the measured energy for the NaI 1 scintillator. The peak would be expected to be around zero. Its shift may, however, be explained by Compton scattering.

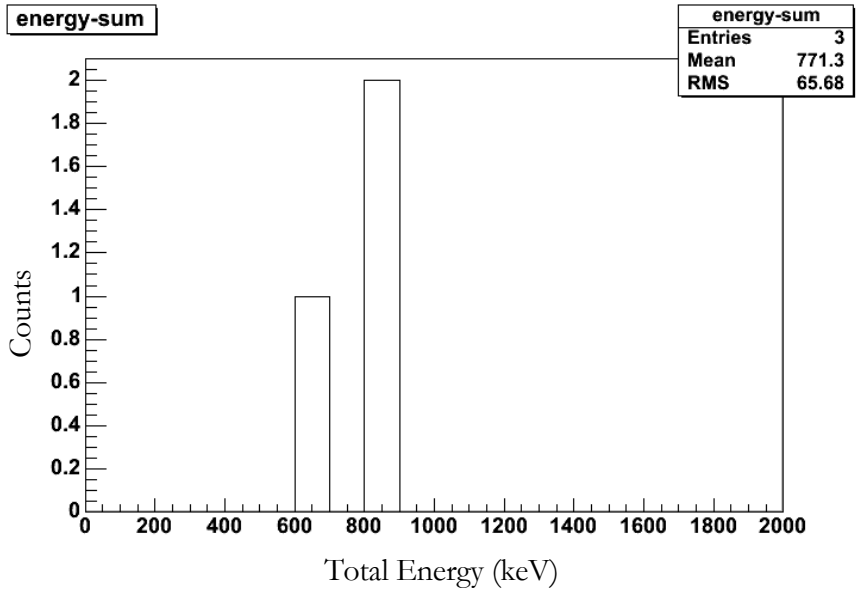

Figure 19. The energies of the triple-coincidence events. Notice that the total energy was not 1.27 MeV as expected, which signals that Compton scattering took place.

It would be easier to detect all three gamma rays from an annihilation, and simply add their energies to verify that they came from the same positronium annihilation. However, over the 200 hours of data collection, only three triple gamma ray coincidences were detected. They also exhibited total energies lower than the expected 1.27 MeV, signaling that Compton scattering took place. The plot of these is shown in Figure 19.

3.3 Future Plans

The next goal is to find overcome the problems associated with Compton scattering in order to use the energy information in the data analysis, since currently Compton scattering is preventing this. Several solutions are being considered:

3.3.1 Shielding

It is possible that using energy information to discriminate between orthopositronium and parapositronium decays will not help to eliminate many parapositronium events, and may therefore not be necessary. Since parapositronium decays into back to back gamma rays, these will never trigger

a coincidence event provided that the detectors are not separated by an angle of π radians. Therefore by avoiding such a detector configuration, and requiring a coincidence in at least two detectors, the only way in which a parapositronium event could appear to be an orthopositronium event would be if a decay gamma Compton scattered in the source or nearby material (thereby resulting in the gamma rays no longer traveling back to back), or if it Compton scattered from one detector to the other. Compton scattering cannot be controlled, but the probability of scattering from one detector to another can be reduced by placing lead shielding between the scintillators; however, Compton scattering in the lead itself could make it worse.

3.3.2 Compton Suppression

Another solution would be to implement a Compton suppression scheme, in which another detector is placed around each NaI scintillator. This surrounding detector would detect most gamma rays which Compton scatter out of the NaI scintillator, and such Compton scatter events could then be ignored in data analysis. However, there is no practical advantage gained by doing this instead of simply using a larger detector. The only reason to do so would be if the primary detector was too expensive to obtain a larger detector. Since this is not the case for NaI, it would make more sense to increase the size of the NaI detectors instead, although this would increase the angular uncertainty.

3.3.3 Germanium Detectors

An ideal solution would be to use Ge scintillation detectors in a near-4π configuration, such as GEANIE [39] or the Gammasphere [40]. Their high energy resolution would help to decrease systematic uncertainties in the measurement. However, these detectors would require Compton suppression, and this combined with their high cost makes prohibits their use at this time.

3.3.4 CsI Scintillation Detectors

Another solution is to use CsI scintillators instead of NaI. CsI scintillators are very dense and can be made with a large volume. Their density combined with a large volume would make it less likely that Compton scattering out of the detector will occur. The possibility of obtaining CsI detectors to replace the NaI detectors is currently being investigated. The CsI detectors would cover a larger solid

angle, which could help the count rate, but also presents some further problems. Having a larger solid angle for each detector would increase the uncertainty in the angle, which would result in a larger uncertainty in the predicted energy. It would be possible to move the detectors further away from the source, however, reducing the solid angle and the probability of detector-to-detector scattering.

3.3.5 Liquid Xe Proportional Counters

Finally it may be possible to use liquid Xenon drift chambers to detect the annihilation gamma rays. These would allow for very accurate angular measurements, much better than that provided by discrete scintillators, have a high efficiency because of its high density, and would allow for very large solid angle coverage. It might be possible to use two drift chambers, one inside the other, in order to detect Compton scattering out of the inside detector and prevent such events from being used in analysis. Liquid Xenon drift chambers have a very high efficiency of near 100% for detecting gamma rays [41], which combined with the solid angle coverage would increase the count rate significantly. Mutiwire liquid Xenon gamma ray proportional detectors have been successfully implemented [42], and the electron avalanche behavior has been investigated [43]. A summary of some of the work which has been done may be found in Ref. [44]. Using drift chambers would also allow for the detection of all three annihilation gamma rays in coincidence. However, implementing such a system would present a large number of technical difficulties which would need to be explored. Using CsI scintillators or implementing a direct Compton suppression scheme as discussed above would be easier to implement.

Appendix A

COMPUTER CODES

A.1 Introduction

In this section, the code used to read the data from CAMAC crate and the code used in analysis are presented.

A.2 Data Acquisition Code

The computer code to read out the ADC and the TDC using the Jorway 73A CAMAC crate controller [36] was written using the ROOT [37] histogramming and GUI classes and the sjy [38] CAMAC routines.

A.2.1 Makefile

Here is the makefile, which is used to compile and link the program. The makefile compiles example.c, main.cxx, and gui.cxx. The compilation of example.c is a carryover from an older version of the program; all functions are now kept in main.cxx and gui.cxx. These are presented below, and their uses described. The resulting object files are then linked together into the executable "example".

```
ROOTCFLAGS    = $(shell root-config --cflags)
ROOTLIBS      = $(shell root-config --libs)
ROOTGLIBS     = $(shell root-config --glibs)

# Linux with egcs
CXX           = g++
CXXFLAGS      = -O -Wall -fPIC
LD            = g++
LDFLAGS       = -g
SOFLAGS       = -shared


CXXFLAGS     += $(ROOTCFLAGS) -I/root/camac/include
LIBS          = $(ROOTLIBS)
GLIBS         = $(ROOTGLIBS)
CAMACLIB      = -L/root/camac/lib -lsjy
#------------------------------------------------------------
--------------
#example: example.cxx
# g++ -o example.o -c example.cxx $(CXXFLAGS)
```

```
# g++ -g example.o $(GLIBS) $(CAMACLIB) -o example
# @echo "example done"

example: example.c example.h main.cxx gui.cxx gui.h
 gcc -c example.c -o example.o $(CXXFLAGS)
 gcc -o main.o -c main.cxx $(CXXFLAGS)
 gcc -o gui.o -c gui.cxx $(CXXFLAGS)
 g++ -g main.o example.o gui.o $(GLIBS) $(CAMACLIB) -lThread -o
example
 @echo "example done"
gui:
 gcc -o gui.o -c gui.cxx $(CXXFLAGS)
 g++ -g main.o example.o gui.o $(GLIBS) $(CAMACLIB) -lThread -o
example


clean:
 @rm -f *.o core
```

A.2.2 Main.cxx

The file main.cxx is used to create the application which reads out the data from the CAMAC crate.

```
#include "gui.h"
//#include <TThread.h>
#include <iostream.h>
#include <stdlib.h>
TROOT root("GUI", "GUI test environement");
int argm;
int main(int argc, char **argv)
{
if (argc >1) argm = atoi(argv[1]);
else argm = 1;
   TApplication theApp("App", NULL,NULL);
   if (gROOT->IsBatch()) {
      fprintf(stderr, "%s: cannot run in batch mode\n", argv[0]);
      return 1;
   }

   TestMainFrame mainWindow(gClient->GetRoot(), 700, 420);
   //TThread *prog=new TThread(&threadfunc,(void*)NULL);
   //prog->Run();
   //mainWindow.CAMACTalk();
   //theApp.Run(true);
   //mainWindow.CAMACTalk();

   return 0;
}
```

A.2.3 Gui.cxx

Gui.cxx is used to create the window for the application. It initializes the SCSI system, initializes the CAMAC crate, and initializes the individual CAMAC modules. It also performs the function of reading out the data in response to the trigger from the CAMAC crate. Finally it stores all the data in a ROOT Tree, in a file called "test33.root". The window is updated periodically so that the events can be visualized. The file gui.cxx has its own header file, gui.h, which may be found in section A.2.4. The header file primarily serves to include all the other necessary header files.

```
/*
 * GUI.CXX -- gui class implementation file for the combined
program
 */

#include "gui.h"
#include <fstream>
#include <stdlib.h>
#include <iostream.h>
#include <unistd.h>
#include <string.h>
//#include <sstream>
extern "C"{//the "example.h" file contains all the actual
functions for talking to the CAMAC crate
#include"example.h"
#include "ieee_fun_types.h"
}
extern int argm;
  //initialize the window
  bool run=true;
  bool rexit=false;
TString ConvertIntToString(int code){//used by DataFileName
method to generate the file names

 int xx = code;
 //for (int tint=0;tint<64;tint++){str[tint]=" ";
 //}
 TString retr;;
 while (xx>0)
 {
 retr.Prepend((char)((xx % 10) +48));
 xx=xx/10;
 }
 return retr;
```

```
}

int dat1[4];
TestMainFrame::TestMainFrame(const TGWindow *p, UInt_t w, UInt_t
h)
      : TGMainFrame(p, w, h)
{
run = true;
filecode=1;
//cout<<DataFileName()<<endl;;
  // Create container
  //TreeData = TTree("FileData","Treedata"/*implicitly use split
level 99*/);

  //appears we want ("name",thing to fill with, "name");
  //TBranch *BranchData = TreeData->Branch("events","Event",void
* addobj, bufsize, splitlevel);
  //not sure what to put in for those particular things

  TGCompositeFrame *pContainer = new TGCompositeFrame(this, 200,
100); //Main window's container
  pContainer->SetLayoutManager(new TGVerticalLayout(pContainer));
  pTab = new TGTab(pContainer,w-100,h-100); //the tab widget,
which manages the tabs
  AddFrame(pContainer, new TGLayoutHints(kLHintsNormal, 1, 1, 1,
1));  //we want the container in the main window

  // Create canvas
  pContainer->AddFrame(pTab, new TGLayoutHints(kLHintsTop |
kLHintsExpandX,1, 1, 1, 1));//and the tab widget in the main
container in the main window
 pTab->Associate(this);//associate pTab with the TestMainFrame
object
 //initialize the buttons: new TGTextButton(parent, label, id
number);
  Buttons[0] = new TGTextButton(this,"Start/Stop",tStartStop);
  this->AddFrame(Buttons[0], new
TGLayoutHints(kLHintsCenterX,5,5,3,4));
  Buttons[1] = new TGTextButton(this,"Exit",tExit);
  this->AddFrame(Buttons[1], new
TGLayoutHints(kLHintsCenterX,5,5,3,4));
  names[0]="TDC 1";
  names[1]="TDC 2";
  names[2]="TDC 3";
  names[3]="ADC 0 - Plastic";
  names[4]="ADC 1";
  names[5]="ADC 2"; //for easy, no-compile custimization, we
should have a function here reading a data file
  names[6]="ADC 3";
```

```
  for (int i=0;i<NCHAN;i++){tab_frame[i] = pTab-
>AddTab(names[i]);} //initializes all tab_frames to point to tabs
contained by pTab
  for (int i=0;i<NCHAN;i++){pRHistCanvas[i] = new
TRootEmbeddedCanvas(names[i], tab_frame[i],  400, 300);} //gives
each tab_frame a histogram canvas
  //note that it has to go with tab_frame as the parent!!!
  pTab->MapSubwindows();
  pTab->Layout();
  //tab_frame->AddFrame(Buttons[0],new TGLayoutHints(kLHintsTop,
             //| kLHintsExpandX | kLHintsExpandY,
             //1, 1, 1, 1));
  //tab_frame->AddFrame(Buttons[1],new TGLayoutHints(kLHintsTop
 //             | kLHintsExpandX | kLHintsExpandY,
 //            1, 1, 1, 1));
 for (int i=0;i<NCHAN;i++){     //Actually give each tab_frame its
respective histogram canvas
  tab_frame[i]->AddFrame(pRHistCanvas[i], new
TGLayoutHints(kLHintsBottom
             | kLHintsExpandX | kLHintsExpandY,
             1,1,1,1));}
  pContainer->AddFrame(Buttons[0], new
TGLayoutHints(kLHintsNormal, 500, 100, 1, 1));
 for (int i=0;i<NCHAN;i++){
  pRCanvasHist[i] = pRHistCanvas[i]->GetCanvas();  //set up each
histogram canvas
  pRCanvasHist[i]->Divide(1,1);//appears to be vertical
divisions, horizontal divisions
  }


  // Create histograms with proper initialization for each.
 // Make sure there's the same number as for NCHAN!  should have a
function here
    h_L[0] = new TH1F("TDC 1", "TDC 1", 1050, 0, 1050.0);
    h_L[0]->Reset();
    pRCanvasHist[0]->cd(1);
    h_L[0]->Draw();

    h_L[1] = new TH1F("TDC 2", "TDC 2", 1050, 0, 1050.0);
    h_L[1]->Reset();
    pRCanvasHist[1]->cd(1);
    h_L[1]->Draw();

    h_L[2] = new TH1F("TDC 3", "TDC 3", 1050, 0, 1050.0);
    h_L[2]->Reset();
    pRCanvasHist[2]->cd(1);
    h_L[2]->Draw();

    h_L[3] = new TH1F("ADC 0", "ADC 0", 8000, 1.0, 8000.0);
```

```
    h_L[3]->Reset();
    pRCanvasHist[3]->cd(1);
    h_L[3]->Draw();

    h_L[4] = new TH1F("ADC 1", "ADC 1", 8000, 1.0, 8000.0);
    h_L[4]->Reset();
    pRCanvasHist[4]->cd(1);
    h_L[4]->Draw();

    h_L[5] = new TH1F("ADC 2", "ADC 2", 8000, 1.0, 8000.0);
    h_L[5]->Reset();
    pRCanvasHist[5]->cd(1);
    h_L[5]->Draw();

    h_L[6] = new TH1F("ADC 3", "ADC 3", 8000, 1.0, 8000.0);
    h_L[6]->Reset();
    pRCanvasHist[6]->cd(1);
    h_L[6]->Draw();


  h_L[0]->SetFillColor(3);
  h_L[1]->SetFillColor(6);

  MapSubwindows();
  Resize(700,500);  //set the window size to 600x400
  SetWindowName("Positronium");
  MapWindow();
  CAMACInit();
 while (!rexit){
 if (run)CAMACTalk();
 gSystem->ProcessEvents();
 }
 CloseWindow();
}//End TestMainFrame::TestMainFrame

TestMainFrame::~TestMainFrame()
{
   // Delete all created widgets.

}
//to gracefully exit the program on a button click
void TestMainFrame::CloseWindow()
{
   // Got close message for this MainFrame. Calls parent
CloseWindow()
   // (which destroys the window) and terminate the application.
   // The close message is generated by the window manager when
its close
   // window menu item is selected.
   TGMainFrame::CloseWindow();
```

```
   gApplication->Terminate(0);
   exit(0);
}
//this method does all the processing of input which is
nonstandard; ie not built into the ROOT classes
Bool_t TestMainFrame::ProcessMessage(Long_t msg, Long_t parm1,
Long_t)
{
   // Handle messages send to the TestMainFrame object. E.g. all
menu button
   // messages.

   switch(GET_MSG(msg)){
  case kC_COMMAND:
         switch (GET_SUBMSG(msg)) {

            case kCM_BUTTON:
               //printf("Button was pressed, id = %ld\n", parm1);
        //for button events, parm1 is the button ID.
               if (parm1 == tExit){rexit=true;}
        if (parm1 == tStartStop){run=!run;

         }//end parm1 == tStartStop
              break;
  }
 }
   return kTRUE;
}
/*Bool_t TestMainFrame::HandleButton(Event_t *event1){
run=!run;
}*/
//this is the function which takes user input and collects data
void TestMainFrame::CAMACInit(){
int i;
int treedat[4];
int ext_Z, ext_C, ext_inhibit, ext_read_lam;
int ext_TDC3377_0, ext_TDC3377_1, ext_TDC3377_2;
int ext_TDC3377_3, ext_TDC3377_4, ext_TDC3377_5;
int ext_ADC413_0, ext_ADC413_1, ext_ADC413_2, ext_ADC413_3;
int q,lam;
int counter;
int block[4]={0};
int updatecheck;
int data;

/* open the SCSI device */
if((cdchn(0,1,0) &1) != 1){
 perror("cdchn error");
 exit(2);
 }
```

```
/* set the controller type */
ccctype(0,0,1); /* parallel - Jor 73A is considered parallel */


cdreg(&ext_Z,0,1,28,8);
cdreg(&ext_C,0,1,28,9);
cdreg(&ext_inhibit,0,1,30,9);
cdreg(&ext_read_lam,0,1,30,0);
cdreg(&ext_TDC3377_0,0,1,9,0);
cdreg(&ext_TDC3377_1,0,1,9,1);
cdreg(&ext_TDC3377_2,0,1,9,2);
cdreg(&ext_TDC3377_3,0,1,9,3);
cdreg(&ext_TDC3377_4,0,1,9,4);
cdreg(&ext_TDC3377_5,0,1,9,5);
cdreg(&ext_ADC413_0,0,1,8,0);
cdreg(&ext_ADC413_1,0,1,8,1);
cdreg(&ext_ADC413_2,0,1,8,2);
cdreg(&ext_ADC413_3,0,1,8,3);


// Flash the inhibit light -- let us know its working
//for (i=1;i<2;i++) {
//cssa(26,ext_inhibit,&dummy,&q); // set dataway inhibit
//sleep(1);
//cssa(24,ext_inhibit,&dummy,&q); // remove dataway inhibit
//sleep(1);
//}//include this if you need to test whether the computer can
communicate
//with the crate at all.

// Clear and Reset the dataway
cssa(26,ext_inhibit,&dummy[0],&q); // set dataway inhibit
cssa(26,ext_Z,&dummy[0],&q); // dataway Z
cssa(26,ext_C,&dummy[0],&q); // dataway C
//cssa(24,ext_inhibit,&dummy,&q); // remove dataway inhibit
//DO ADC SETUP
//Set control registers
  dummy[0]=29440;
  cssa(16,ext_ADC413_0,&dummy[0],&q);//ctrl reg 1
  dummy[0]=15;
  cssa(16,ext_ADC413_1,&dummy[0],&q);//ctrl reg 2
  cssa(9,ext_ADC413_0,&dummy[0],&q);//clear module
//Enable LAM
cssa(26,ext_ADC413_0,&dummy[0],&q);
cssa(10,ext_ADC413_0,&dummy[0],&q);//clear lam
//END ADC SETUP
// Set up TDC module for single word common start mode one

//CAMAC reprogramming xilinx chip
```

```
//Enable Xilinx program
cssa(9,ext_TDC3377_0,&dummy[0],&q);

//Reset xilinx
cssa(30,ext_TDC3377_0,&dummy[0],&q);

//Select EPROM mode 0, common stop, single word
//cssa(20,ext_TDC3377_0,&dummy[0],&q);

//Begin xilinx programming se&quence
cssa(25,ext_TDC3377_0,&dummy[0],&q);

// wait for 500 ms (or actually 1 seconds)
sleep(1);

//Test Xilinx programming done
q = 0;
while (q == 0)
 {
 cssa(13,ext_TDC3377_0,&dummy[0],&q);
 }


//Reset PAL, clear buffers, disable xilinx program mode
cssa(9,ext_TDC3377_0,&dummy[0],&q);


//Set registers

//Register 0
// module ID = FF
// 1 ns resolution
// leading edge
// CAMAC readout
// multievent buffer
// header

buf[0]=0x11FF;//4351
cssa(17,ext_TDC3377_0,&buf[0],&q);


//Register 1
// event number starts at zero
// no MPI
// normal FAST FER
// max trigger width and delay

buf[0]=0x00FF;//0
cssa(17,ext_TDC3377_1,&buf[0],&q);
```

```
//Register 2
// full scale 1016 ns
// 1 hit allowed max

buf[0]=0x07E1;//2017
cssa(17,ext_TDC3377_2,&buf[0],&q);

//Register 3
// zero offset
// no request delay

buf[0]=0x0000;//1008
cssa(17,ext_TDC3377_3,&buf[0],&q);

//Register 4
//time out at 550ns
buf[0]=0x000B;//11
cssa(17,ext_TDC3377_4,&buf[0],&q);

//Register 5
//no test mode
buf[0]=0x0000;//0, 0x101 for test mode
cssa(17,ext_TDC3377_5,&buf[0],&q);

//All Registers are now set up

//Enable LAM
cssa(26,ext_TDC3377_0,&dummy[0],&q);

//Enable acquisition mode
cssa(26,ext_TDC3377_1,&dummy[0],&q);

//clear the LAM
cssa(10,ext_TDC3377_0,&dummy[0],&q);

// remove dataway inhibit
cssa(24,ext_inhibit,&dummy[0],&q);
return;
}


void TestMainFrame::CAMACTalk(){

//This method is really the entire point of the program.  Once it
starts running,
//this method is invoked and runs for the rest of the time.


int i;
int treedat[4];
```

```
int ext_Z, ext_C, ext_inhibit, ext_read_lam;
int ext_TDC3377_0, ext_TDC3377_1, ext_TDC3377_2;
int ext_TDC3377_3, ext_TDC3377_4, ext_TDC3377_5;
int ext_ADC413_0, ext_ADC413_1, ext_ADC413_2, ext_ADC413_3;
int q,lam;
int counter;
int block[4]={0};
int updatecheck;
int info;

TStopwatch *stopwatch = new TStopwatch();

TFile FileData("test33.root","RECREATE");
TTree *TreeData = new TTree("positronium","positronium-tree");


bool fileisopen=false;


  TreeData->Branch("ADC0",&dummy[0],"ADC0/s");//s=unsigned 16bit
integer
  TreeData->Branch("ADC1",&dummy[1],"ADC1/s");
  TreeData->Branch("ADC2",&dummy[2],"ADC2/s");
  TreeData->Branch("ADC3",&dummy[3],"ADC3/s");
  TreeData->Branch("TDC1",&dat1[1],"TDC1/I");//I=normal integer
  TreeData->Branch("TDC2",&dat1[2],"TDC2/I");
  TreeData->Branch("TDC3",&dat1[3],"TDC3/I");

/* open the SCSI device */

/* set the controller type */
ccctype(0,0,1); /* parallel - Jor 73A is considered parallel */


cdreg(&ext_Z,0,1,28,8);
cdreg(&ext_C,0,1,28,9);
cdreg(&ext_inhibit,0,1,30,9);
cdreg(&ext_read_lam,0,1,30,0);
cdreg(&ext_TDC3377_0,0,1,9,0);
cdreg(&ext_TDC3377_1,0,1,9,1);
cdreg(&ext_TDC3377_2,0,1,9,2);
cdreg(&ext_TDC3377_3,0,1,9,3);
cdreg(&ext_TDC3377_4,0,1,9,4);
cdreg(&ext_TDC3377_5,0,1,9,5);
cdreg(&ext_ADC413_0,0,1,8,0);
cdreg(&ext_ADC413_1,0,1,8,1);
cdreg(&ext_ADC413_2,0,1,8,2);
cdreg(&ext_ADC413_3,0,1,8,3);

//main loop
```

```
updatecheck = 0;
stopwatch->Start();//start the clock!  No more than 2 hours on
one file

int timercount=0;
int nevents=0;
while(run)
 {
 // Process windows events
 gSystem->ProcessEvents();

 //check the LAM pattern
 // 128 = slot 8 (TDC)
 // 32 = slot 6 (ADC)
        // 160 = 32 + 128 both TDC and ADC
 //
 // 256 = slot 9 (TDC)
 // 384 = 256 + 128 both slot 8 and 9

 counter = 0;
 lam = 0;
 while ((lam & 128) != 128){
  cfsa(0,ext_read_lam,&lam,&q);

   counter++;
   if (counter>500){
   gSystem->ProcessEvents();  // do this every once in a while in
case locked up
    counter=0;
   //cout<<"failure"<<endl;
   }
   }
        //cout<<"LAM: "<<lam<<endl;
 for (i = 0; i < 34; i++){buf[i]=10000;}
 //Read the data in
 block[0]=4;//34;//maximum number of transfers to make
 block[1]=0;//
 csubc(0,ext_TDC3377_0,buf,&block[0]);

 //Read back data
 //for (i=0;i<=block[1];i++) cout<<"i =  "<<i<<"  data =
"<<buf[i]<<endl;
 //cout<<buf[1]<<" ";//endl;

 //Read out TDC

        dat1[1]=-1;dat1[2]=-1;dat1[3]=-1;
 for (int k=1; k<4;k++){
 // Fill the histograms
 if ((buf[k]&0x7c00)/0x400==1){
```

```
    dat1[1] = buf[k] & 0x03FF;
 }
if ((buf[k]&0x7c00)/0x400==2){
    dat1[2] = buf[k] & 0x03ff;
 }
if ((buf[k]&0x7c00)/0x400==4){
    dat1[3] = buf[k] & 0x03ff;
 }
}

   h_L[0]->Fill(dat1[1]);
   h_L[1]->Fill(dat1[2]);
   h_L[2]->Fill(dat1[3]);
  for (i = 0; i < 4; i++){if(buf[i]==10000)cout<<buf[i]<<endl;}
nevents++;//TAKE OUT LATER TO PREVENT TROUBLES

 //Read out ADC
cssa(2,ext_ADC413_0,&dummy[0],&q);
h_L[3]->Fill(dummy[0]);

cssa(2,ext_ADC413_1,&dummy[1],&q);
h_L[4]->Fill(dummy[1]);

cssa(2,ext_ADC413_2,&dummy[2],&q);
h_L[5]->Fill(dummy[2]);

cssa(2,ext_ADC413_3,&dummy[3],&q);
h_L[6]->Fill(dummy[3]);

//cout<<"ADC "<<dummy[0]<<" "<<dummy[1]<<" "<<dummy[2]<<"
"<<dummy[3]<<"  ";
//cout<<"TDC "<<dat1[1]<<" "<<dat1[2]<<" "<<dat1[3]<<endl;

TreeData->Fill();

//then clear dat1
dat1[1]=0;dat1[2]=0;dat1[3]=0;

cssa(9,ext_ADC413_0,&dummy[3],&q);//not needed in singles mode?

// Update the histogram display
// clear all data and lam
 cssa(9,ext_TDC3377_0,&dummy[3],&q);


updatecheck++;
 if (updatecheck>9){//update the window every 10 events.
 updatecheck=0;
 for (int i=0;i<NCHAN;i++){
   pRCanvasHist[i]->cd(1);
```

```
     h_L[i]->Draw();
     pRCanvasHist[i]->Modified();
     pRCanvasHist[i]->Update();
   }
    pTab->DrawBorder();

 }

 } //end main while

/*  TString fname;
  while(!fileisopen){
   fname.Append("run");
   fname.Append(ConvertIntToString(argm));
   fname.Append(".root");
   if(FileData-
>Open(fname.Data(),"NEW",fname.Data()))fileisopen=true;
    fname="";
 argm++;
  }

*/

cout<<"Number of events = "<<nevents<<endl;


FileData.cd();
TreeData->Write();
FileData.Write();
FileData.Close();
exit(0);

}//end CAMACTalk method
```

A.2.4 Gui.h

Gui.h is the header file for gui.cxx. Its function is to load the necessary C++ header files, and to declare certain classes and variables.

```
/*
 * GUI.H -- gui class declaration file
 */

#include <stdlib.h>

#include <TROOT.h>
#include <TApplication.h>
```

```
#include <TVirtualX.h>

//#include <TGListBox.h>
#include <TGClient.h>
#include <TGFrame.h>
#include <TGIcon.h>
#include <TGLabel.h>
#include <TGButton.h>//its important to use this so that we can
redefine Clicked
#include <TGTextEntry.h>
//#include <TGMsgBox.h>
#include <TGMenu.h>
#include <TGCanvas.h>
#include <TGComboBox.h>
#include <TGTab.h>
#include <TStopwatch.h>
//#include <TGSlider.h>
//#include <TGDoubleSlider.h>
//#include <TGFileDialog.h>
#include <TRootEmbeddedCanvas.h>
#include <TCanvas.h>
#include <TH1.h>
#include <TF1.h>
#include <TFile.h>
//#include <TH2.h>
//#include <TRandom.h>
#include <TSystem.h>
#include <TEnv.h>
#include <TGListBox.h>
#include <RQ_OBJECT.h>//needed for the widgets (buttons, etc.)
#include <TTree.h>
//#include <TBranch.h>//I believe this is not needed

//enumerate the various button ids:
enum buttonidset{
tExit,tStartStop
};
class Page {       //I think this class is unneeded and unused

public:
  Page (TCanvas *c, char *line);
  ~Page();
  int isValid() {return done;}
  char *Name() {return name;}
  int Draw(void);
  TCanvas *Canvas() {return page_canvas;}
  TH1F *Hist(int x, int y) {return page_hist[x][y];}
  int nx, ny;

private:
```

```
  char name[128];
  int done;
  TCanvas *page_canvas;
  TH1F ***page_hist;
  int log_flag[100][100];
};


//Special class to handle tabs, I think it ends up not being used

class Tab {
public:
  Tab(char *tname, TGCompositeFrame *tab_frame, TGMainFrame
*parent, int tid);
  ~Tab() {};
  Page *page[1000];
  int ReadPages(char *fname);
  TCanvas *canvas;

private:
  char name[128];

  TGListBox *list_box;
  TRootEmbeddedCanvas *emb_can;
  int tab_id;

  int n_pages;
};
//NCHAN is the number of histograms which we want to look at.
#define NCHAN 7
//NCHAN is the number of things we want to look at
/*
 * Main Window class
 */
 //this class is what contains almost all the actual, working,
functions of the program.


class TestMainFrame : public TGMainFrame {
RQ_OBJECT("TestMainFrame");//necessary for signal/slots to work
here

private:
  TGCompositeFrame    *pContainer;
  TGTab    *pTab;
  TRootEmbeddedCanvas *pRHistCanvas[NCHAN];
  TGButton      *Buttons[1];
  TCanvas             *pRCanvasHist[NCHAN];
  TH1F                *h_L[NCHAN];
```

```
  TF1                 *fit_L;
  TF1                 *fit_R;

public:
  TGCompositeFrame *tab_frame[NCHAN];
  //bool run;
   TestMainFrame(const TGWindow *p, UInt_t w, UInt_t h);
   //void StartStop();
   virtual ~TestMainFrame();
   void CAMACTalk();
   void CAMACInit();
   virtual void CloseWindow();
   virtual Bool_t ProcessMessage(Long_t msg, Long_t parm1,
Long_t);
   int filecode;//code for generating the file name
   Tab *tabs[32];
   char * names[NCHAN];
   unsigned short dummy[7], buf[34];//dummy is for ADC, buf for
TDC data
   TString *fname;
   //TFile *FileData;
   //TTree TreeData;
   //TBranch *ADC1Data, *ADC2Data, *ADC3Data, *ADC4Data,
*TDC1Data, *TDC2Data, *TDC3Data;
   //TBranch
*BranchData1,*BranchData2,*BranchData3,*BranchData4,*BranchData5,
*BranchData6,*BranchData7;//let's hope this works - its based on
WC
};
```

A.3 Fitting Code

A.3.1 Fitfun.c

The data analysis is done using ROOT's built-in C++ interpreter. The first step was to load the function which was to be fit to the data. This was accomplished using the file "fitfun.C" which was loaded in the interpreter using the command ".L fitfun.C". This allows a function of the form

$$f(t) = Ae^{\frac{t}{\tau}} + B$$

to be fit to the data. In this equation τ is the lifetime. The decay rate λ is given by $\lambda = \tau^{-1}$. In the function itself, A is par[0], B is par[2], and τ is par[1].

```
Double_t fitfun(Double_t *var, Double_t *par)
{
Double_t y;
y=par[0]*exp(var[0]/par[1])+par[2];
return y;
}
```

A.3.2 Fittdc1.c

Finally, the function loaded in "fitfun.C" must actually be fit to the data. There are three files used for this, "fitTDC1.C," "fitTDC2.C," and "fitTDC3,C." Only fitTDC1.C is shown below; the others are identical except for switching the detector and using slightly different bounds for the fit. The function was executed in the ROOT interpreter using the command ".x fitTDC1.C." In this function, the channels to be used in analysis (here those between 300 and 800) are passed, and the names of par[0], par[1], and par[2] (see fitfun.C) are given. Values are guessed for each of these parameters (here 40, 140, and 8). The function then numerically fits the parameters and prints the results and the uncertainties. It also puts a fit curve on the histogram.

```
// .L fitfun.C
int fitTDC1() {
TF1 *func=new TF1("fitfun",fitfun,300,800,3);
func->SetParNames("norm","decay","bkgd");
func->SetParameters(40.,140.,8.);
TDC1->Fit("fitfun","r");
}
```

A.4 Analysis Code

The next step in analysis was to collect all the stored data files and create all the histograms needed for analysis. This was accomplished using the file "tdelay6.C." This program collected each stored data file and put all the data into one set of histograms. Histograms were created of the energy in the plastic scintillator, the energy in each NaI detector cut on the energy in the plastic detector being between channels 4000 and 8000, and the decay times for each NaI detector cut on the plastic detector's energy being between channels 4000 and 8000 and the NaI detector having an energy of less than 511 keV. Also, the energy of each NaI detector was displayed cut on the decay time being

consistent with an orthopositronium event, as well as a plot of the difference between the detection times for coincident events, and a plot of the difference between the energy predicted to be in detector NaI 1 using energy information from detector NaI 2 and the measured energy in NaI 1.

```
/**************************************************************
***********
                          tdelay.C  -  description

Plots and fits decay curve for posit_tree experiment.

                             -------------------
    begin                : Fri Oct 8 2004
    copyright            : (C) 2004 by Mark Yuly
    email                : mark.yuly@houghton.edu

**************************************************************
**********/

/**************************************************************
***********
 *
*
 *   This program is free software; you can redistribute it
and/or modify  *
 *   it under the terms of the GNU General Public License as
published by  *
 *   the Free Software Foundation; either version 2 of the
License, or     *
 *   (at your option) any later version.
*
 *
*

**************************************************************
**********/



gROOT.Reset("a");



//*****************************************************


int tdelay6()
{
```

```
int i, j, nentries;
float energy[4], time[4], total_energy, P;

//Create branch and leaf pointers
TBranch *ADC_branch[4], *TDC_branch[4];
TLeaf *ADC_value[4], *TDC_value[4];

// Energy Calibration constants

float m[4], b[4];

m[1]=0.08661; b[1]=0.;
m[2]=0.07802; b[2]=0.;
m[3]=0.08517; b[3]=0.;

// Rest mass of posit_tree (keV/c^2)

float MM = 2. * 511.;

// detectors that go together for predictions, along with angles
// see logbook for how angle is defined (in deg)

int det[4], ang[4];

det[1]=3;  ang[1]=120;
det[2]=1;  ang[2]=120;
det[3]=2;  ang[3]=120;

// Make up the names used for adc and tdcs
TString adc_name[4], tdc_name[4], spect_name[4], predict_name[4],
tdiff_name[4];

for (i=0;i<4; i++) {
  adc_name[i] = "ADC";
  adc_name[i].Append('0'+i);
  tdc_name[i] = "TDC";
  tdc_name[i].Append('0'+i);
  spect_name[i] = "spect";
  spect_name[i].Append('0'+i);
  predict_name[i] = "predict_";
  predict_name[i].Append('0'+det[i]);
  predict_name[i].Append("_from_");
  predict_name[i].Append('0'+i);
  tdiff_name[i] = "tdiff";
  tdiff_name[i].Append('0'+det[i]);
  tdiff_name[i].Append("--");
  tdiff_name[i].Append('0'+i);
  }

//Set the cuts
```

```
int adc_l[4],  adc_h[4],  tdc_l[4], tdc_h[4], tdiff_l[4],
tdiff_h[4];

adc_l[1]=500; adc_h[1]=5500;
adc_l[2]=500; adc_h[2]=5500;
adc_l[3]=500; adc_h[3]=5500;
adc_l[0]=4500; adc_h[0]=8000; // plastic


tdc_l[1]=300; tdc_h[1]=750;
tdc_l[2]=300; tdc_h[2]=750;
tdc_l[3]=300; tdc_h[3]=750;


// Time diff cuts

tdiff_l[1]=5; tdiff_h[1]=30;  // 1-3
tdiff_l[2]=-10; tdiff_h[2]=10;  // 2-1
tdiff_l[3]=-30; tdiff_h[3]=5;  // 3-2



// Open the root file
TFile *f[9];

 f[1]= new TFile("/home/public/camac/Combined2/02-11-05.root");
 f[2]= new TFile("/home/public/camac/Combined2/02-16-05.root");
 f[3]= new TFile("/home/public/camac/Combined2/02-17-05.root");
 f[4]= new TFile("/home/public/camac/Combined2/02-22-05.root");
 f[5]= new TFile("/home/public/camac/Combined2/02-24-05.root");
 f[6]= new TFile("/home/public/camac/Combined2/02-26-05.root");
 f[7]= new TFile("/home/public/camac/Combined2/03-01-05.root");
 f[8]= new TFile("/home/public/camac/Combined2/03-03-05.root");

 TTree *posit_tree;

//TChain posit_tree("posit_tree");
//posit_tree.Add("/home/public/camac/Combined2/02-11-05.root");
//posit_tree.Add("/home/public/camac/Combined2/02-16-05.root");
//posit_tree.Add("/home/public/camac/Combined2/02-17-05.root");
//posit_tree.Add("/home/public/camac/Combined2/02-22-05.root");
//posit_tree.Add("/home/public/camac/Combined2/02-24-05.root");
//posit_tree.Add("/home/public/camac/Combined2/02-26-05.root");
//posit_tree.Add("/home/public/camac/Combined2/03-01-05.root");
//posit_tree.Add("/home/public/camac/Combined2/03-03-05.root");


// Create the canvas
TCanvas *c1 = new TCanvas("c1","Decay Time",10,10,1000,750);
```

```
c1->Divide(4,5);


//Creating histograms

TH1F *adc_hist[4], *tdc_hist[4], *espect_hist[4],
*predict_hist[4], *tdiff_hist[4], *esum_hist, *test_hist;

// plastic
cout<<"Creating "<<adc_name[0]<<" histogram"<<endl;
  c1->cd(1);
  adc_hist[0] = new TH1F(adc_name[0],adc_name[0],8000,0,7999);
  adc_hist[0]->Draw();

// energy-sum
cout<<"Creating energy-sum histogram"<<endl;
  c1->cd(17);
  esum_hist = new TH1F("energy-sum","energy-sum",20,0.,2000.);
  esum_hist->Draw();

  // test
cout<<"Creating test histogram"<<endl;
  c1->cd(13);
  test_hist = new TH1F("test","test",2050,0.,2050.);
  test_hist->Draw();

for (i=1; i<4; i++) {

// raw adc spectra
  cout<<"Creating "<<adc_name[i]<<" histogram"<<endl;
  c1->cd(1+i);
  adc_hist[i] = new TH1F(adc_name[i],adc_name[i],8000,0,7999);
  adc_hist[i]->Draw();

// log tdc spectra
  cout<<"Creating "<<tdc_name[i]<<" histogram"<<endl;
  c1->cd(5+i);
  tdc_hist[i] = new TH1F(tdc_name[i],tdc_name[i],2050,0,2049);
  //tdc_hist[i]->SetMinimum(1);
  //tdc_hist[i]->SetMaximum(100);
  (c1->GetPad(5+i))->SetLogy(1);
  tdc_hist[i]->Draw();


// energy spectra (in keV)
  cout<<"Creating "<<spect_name[i]<<" histogram"<<endl;
  c1->cd(9+i);
  espect_hist[i] = new
TH1F(spect_name[i],spect_name[i],8000,0.,750.);
  espect_hist[i]->Draw();
```

```
  // time-difference spectra
  cout<<"Creating "<<tdiff_name[i]<<" histogram"<<endl;
  c1->cd(13+i);
  tdiff_hist[i] = new TH1F(tdiff_name[i],tdiff_name[i],1000,-
500.,500.);
  tdiff_hist[i]->Draw();

// predicted spectra (in keV)
  cout<<"Creating "<<predict_name[i]<<" histogram"<<endl;
  c1->cd(17+i);
  predict_hist[i] = new TH1F(predict_name[i],predict_name[i],50,-
500.,500.);
  predict_hist[i]->Draw();
}


c1->Update();
c1->Draw();

for (i_file=1; i_file < 9; i_file++)
{
cout<<"+++++++++++++++++++++++++++  File #"<<i_file<<endl;
posit_tree = (TTree*) f[i_file]->Get("positronium");
nentries = posit_tree.GetEntries();


// Set Branches and leaves to read in the data
for (i=0; i<4; i++) {
  cout<<"Setting branches and leaves for "<<adc_name[i]<<endl;
  ADC_branch[i] = posit_tree.GetBranch(adc_name[i]);
  ADC_value[i] = ADC_branch[i]->GetLeaf(adc_name[i]);
  }

  for (i=1; i<4; i++) {
  cout<<"Setting branches and leaves for "<<tdc_name[i]<<endl;
  TDC_branch[i] = posit_tree.GetBranch(tdc_name[i]);
  TDC_value[i] = TDC_branch[i]->GetLeaf(tdc_name[i]);
  }

for (j=0; j<nentries; j++) {
//for (j=0; j<100; j++) {

  // Fill the raw plastic adc histogram
  ADC_branch[0]->GetEntry(j);
  if (ADC_value[0]->GetValue(0)>0) {adc_hist[0]-
>Fill(ADC_value[0]->GetValue(0));}

  // Get the entries
  for (i=1; i<4; i++) {
```

```
    ADC_branch[i]->GetEntry(j);
    TDC_branch[i]->GetEntry(j);
    }

  for (i=1; i<4; i++) {

   energy[i] = 0.;
   time[i] = 0.;

    // cut on plastic ADC for 1.27 MeV gamma
   if (ADC_value[0]->GetValue(0) > adc_l[0] && ADC_value[0]-
>GetValue(0) < adc_h[0])

      {

      // Fill raw ADC histograms cut on plastic ADC for 1.27 MeV
gamma only3
      if (ADC_value[i]->GetValue(0)>0) { adc_hist[i]-
>Fill(ADC_value[i]->GetValue(0)); }

      // Fill TDC Histogram, cut on plastic ADC for 1.27 MeV
gamma and NaI adc for less than 511 keV
      if (ADC_value[i]->GetValue(0)>adc_l[i] && ADC_value[i]-
>GetValue(0)<adc_h[i])
         {
         if (TDC_value[i]->GetValue(0)!=-1) { tdc_hist[i]-
>Fill(TDC_value[i]->GetValue(0)); }
         }

      // Fill the "espect" histograms cut on3
      //   1.  plastic ADC fpr 1.27 MeV gamma
      //   2.  tdc in proper range
      //3
      // Use energy calibration.


      if (TDC_value[i]->GetValue(0)>tdc_l[i] && TDC_value[i]-
>GetValue(0)<tdc_h[i])

        {
 energy[i] = m[i]*ADC_value[i]->GetValue(0)+b[i];
 time[i] = TDC_value[i]->GetValue(0);
// cout<<energy[i]<<endl;3
        if (ADC_value[i]->GetValue(0)>0) { espect_hist[i]-
>Fill(energy[i]); }
        }

      } //if plastic adc

    } // for i
```

```
   // Fill the energy-sum histogram
   if ( (energy[1]!=0.) && (energy[2] != 0.) && (energy[3] != 0.)
)
     {
     total_energy = energy[1] + energy[2] + energy[3];
     cout<<total_energy<<"  =  "<<energy[1]<<"  +
"<<energy[2]<<"  +  "<<energy[3]<<endl;
     esum_hist->Fill(total_energy);
     }

   // Fill the time difference histograms
   for (i=1; i<4; i++)
     {
     if ( (energy[i] > 0.) && (energy[det[i]] > 0.) &&  (time[i]
> 0.) && (time[det[i]] > 0.) )
       {
       // cout<<"Time difference: "<<time[i]-time[det[i]]<<"
D"<<i<<" = "<<time[i]<<"   D"<<det[i]<<" = "<<time[det[i]]<<endl;
       tdiff_hist[i]->Fill(time[i]-time[det[i]]);
       }
    }

   // Fill the prediction histograms
   for (i=1; i<4; i++)
     {
     if ( (energy[i] > 0.) && (energy[det[i]] > 0.) && ((time[i]-
time[det[i]])>tdiff_l[i]) && ((time[i]-time[det[i]])<tdiff_h[i])
)
       {
       P = (energy[i]*MM - 0.5 * MM*MM) / ( ( 1.-
cos(ang[i]*3.1415/180.) )*energy[i] - MM );
       cout<<"Prediction: D"<<det[i]<<" from D"<<i<<"
P1="<<energy[i]<<"   Predicted:"<<P<<"
Measured:"<<energy[det[i]]<<endl;
       predict_hist[i]->Fill(P-energy[det[i]]);
    //   if (i==1) {test_hist->Fill(time[i]);}
       }

     }

   // only update the window every 10000 entries
   if (  ((j/10000)*10000) == j )
     {
     c1->Draw();
     c1->Update();
     cout<<"Processing entry: "<<j<<endl;
     } //if

  } // for j
```

```
} //i_file

c1->Draw();
c1->Update();
}
```

Appendix B

SCHEMATICS OF THE PHOTOMULTIPLIER BASES

The photomultiplier tube bases were modified to have both a dynode and an anode output. Initially one output was to be used, and the signal split. However, the input to the PM Amplifier had a 50 Ω impedance. This caused the signal to become too small to use. It therefore became necessary to use two different outputs. Notice that the anode output has a 1.5 kΩ resistor, while the dynode has a 1 MΩ resistor. The schematic for the NaI PMT base is shown in Figure 20, while the plastic PMT base is shown in Figure 21.

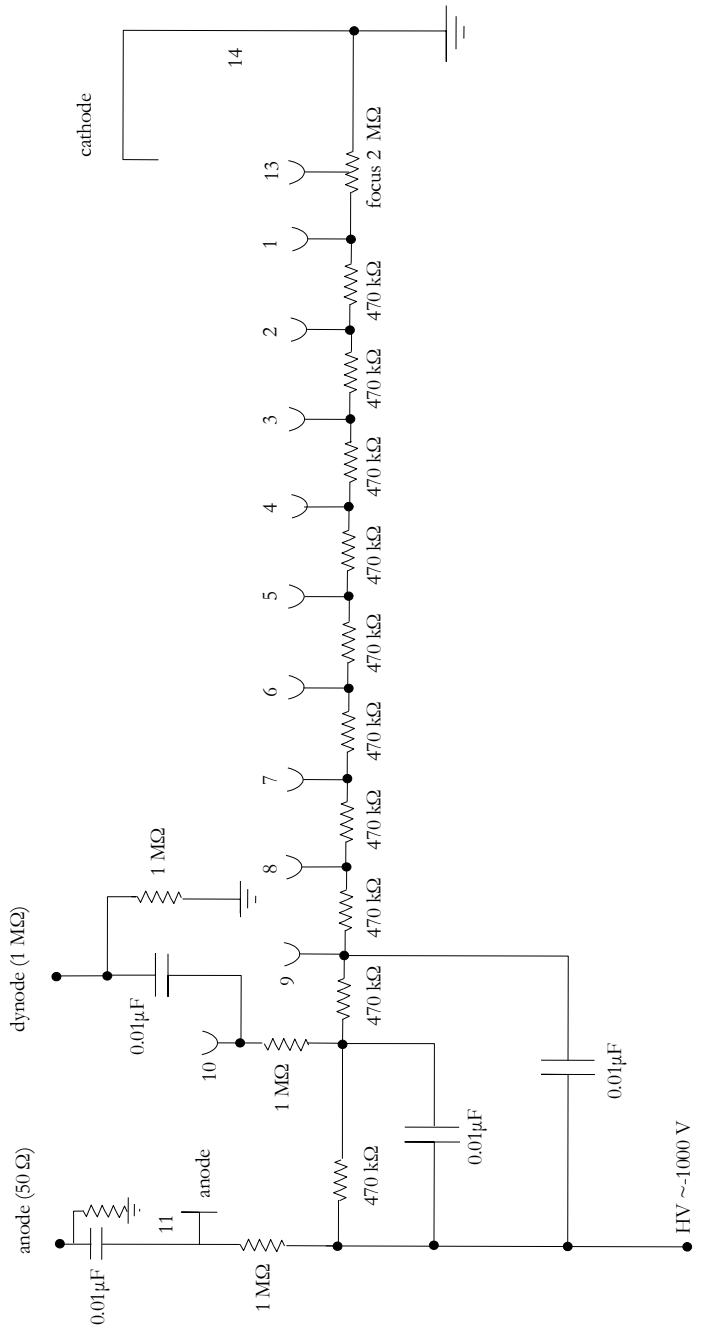

Figure 20. The schematic of the NaI scintillator PMT base. The electrons cascade from the cathode through the dynodes to the anode and dynode outputs.

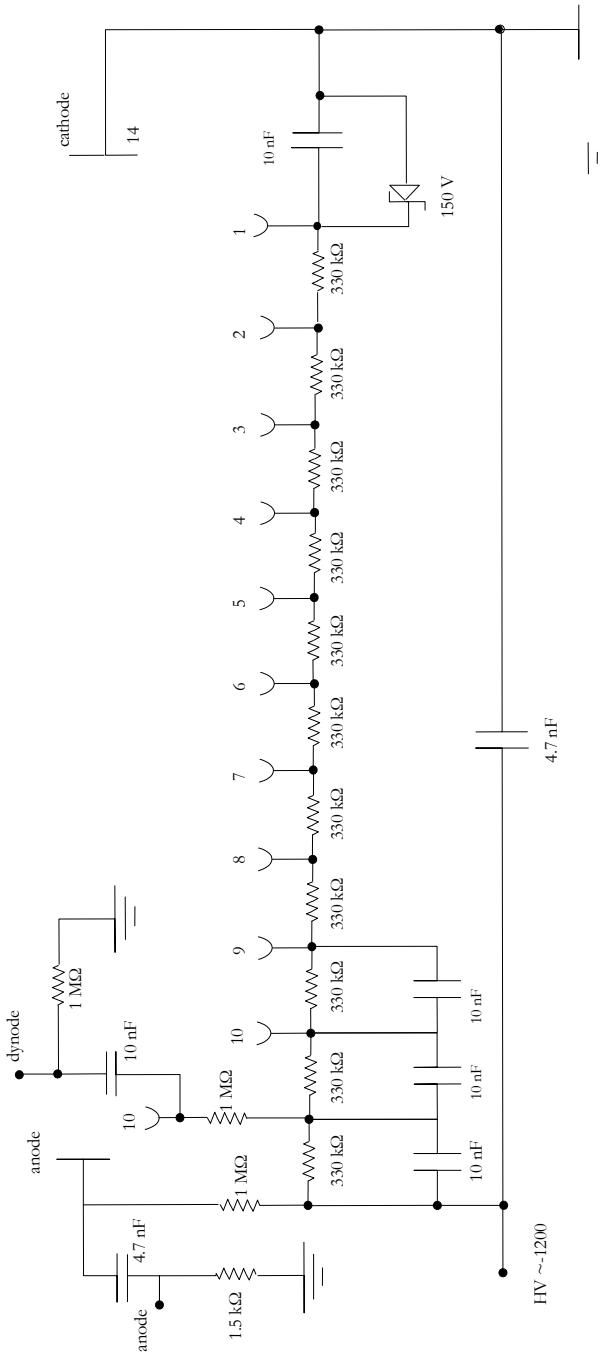

Figure 21. Schematic of the PMT base for the plastic scintillator. Again the electrons cascade from the cathode through the dynodes.

References

[1] P.A.M. Dirac, P.A.M., "A Theory of Electrons and Protons," Proc. Roy. Soc. (London) Series A, **126**, 360 (1930).

[2] C.D. Anderson, "The Positive Electron," Phys. Rev. **43**, 491 (1933).

[3] M. Deutsch, "Evidence For the Formation of Positronium in Gasses," Phys. Rev. **82**, 455 (1951).

[4] E.D. Theriot, *et al.*, "Precision Redetermination of the Fine-Structure Interval of the Ground State of Positronium and a Direct Measurement of the Decay Rate of Parapositronium," Phys. Rev. A **2**, 707 (1970).

[5] T. Chang, H. Tang, Y. Li, "Gamma-Ray Energy Spectrum From Orthopositronium Three-Gamma decay," Phys. Lett. **157B**, 357 (1985).

[6] K.P. Ziock, *et al.*, "First Observation of Resonant Excitation of High-n States in Positronium," Phys. Rev. Lett. **64**, 2366 (1990).

[7] R.P. Feynman, "Space-Time Approach to Quantum Electrodynamics," Phys. Rev. **76**, 769 (1949).

[8] J. Schwinger, "On Radiative Corrections to Electron Scattering," Phys. Rev. **75**, 898 (1949).

[9] S. Tomonaga, "On Infinite Field Reactions in Quantum Field Theory," Phys. Rev. **76**, 790 (1949).

[10] A. Ore, J.L. Powell, "Three-Photon Annihilation of an Electron-Positron Pair," Phys. Rev. **75**, 1696 (1949).

[11] V.V. Dvoeglazov, R.N. Faustov, Y.N. Tyukhtyaev, "Decay Rate of a Positronium. Review of Theory and Experiment," Mod. Phys. Lett. A **8**, 3263 (1993).

[12] G.S. Adkins, R.N. Fell, J. Sapirstein, "Order α^2 Corrections to the Decay Rate of Orthopositronium," Phys. Rev. Lett. **84**, 5086 (2000).

[13] D.W. Gidley, K.A. Marko, A. Rich, "Precision Measurement of the Decay Rate of Orthopositronium in SiO_2 powders," Phys. Rev. Lett. **36**, 395 (1976).

[14] D.W. Gidley, P.W. Zitzewitz, K.A. Marko, A. Rich, "Measurement of the Vacuum Decay Rate of Orthopositronium," Phys. Rev. Lett. **37**, 729 (1976).

[15] D.W. Gidley, *et al.*, "New Precision Measurements of the Decay Rates of Singlet and Triplet Positronium," Phys. Rev. Lett. **49**, 525 (1982).

[16] C.I. Westbrook *et al.*, "Precision Measurement of the Orthopositronium Vacuum Decay Rate Using the Gas Technique," Phys. Rev. A **40**, 5489 (1989).

[17] J.S. Nico, *et al.*, "Precision Measurement of the Orthopositronium Decay Rate Using the Vacuum Technique," Phys. Rev. Lett. **65**, 1344 (1990).

[18] S. Asai, S. Orito, N. Shinohara, "New Measurement of the Orthopositronium Decay Rate," Phys. Lett. B **357**, 475 (1995).

[19] R.S. Vallery, P.W. Zitzewitz, D.W. Gidley, "Resolution of the Orthopositronium-Lifetime Puzzle," Phys. Rev. Lett. **90**, 203402-1 (2003).

[20] S. Asai, O. Jinnouchi, T. Kobayashi, "Solution of Orthopositronium Lifetime Puzzle," Int.J.Mod.Phys. **A19**, 3927 (2004).

[21] S. Asai, *et al.*, "Search for Long-Lived Neutral Bosons in Orthopositronium Decay." Phys. Rev. Lett. **66**, 2440 (1991).

[22] A. Czarnecki, S.G. Karshenboim, "Decays of Positronium," in Proc. of the 14th International Workshop on High Energy Physics and Quantum Field Theory, (MSU-Press, 2000) pp. 538 – 544.

[23] S. Asai, *et al.*, "Direct Search for Orthopositronium Decay Into Two Photons," Phys. Rev. Lett. **66**, 1298 (1991).

[24] S. Orito, *et al.*, "New Limits on Exotic Two-Body Decay of Orthopositronium," Phys. Rev. Lett. **63**, 597 (1989).

[25] J. Yang, *et al.*, "Four-Photon Decay of Orthopositronium: A Test of Charge Conjugation Invariance," Phys. Rev. A **54**, 1952 (1996).

[26] T. Mitsui, *et al.*, "Limit on an Exotic Three-Body Decay of Orthopositronium," Europhys. Lett. **33**, 111 (1996).

[27] A. Badertsher, *et al.*, "Search For an Exotic Three-Body Decay of Orthopositronium," Phys. Lett. B **542**, 29 (2002).

[28] S.J. Tao, "The Formation of Positronium in Molecular Substances," Appl. Phys. **10**, 67 (1976).

[29] S. DeBenedetti, H.C. Corben, "Positronium," Ann. Rev. Nucl. Sci. **4**, 191 (1954).

[30] C. M. Lederer, J. M. Hollander, and I. Perlman, Table of Isotopes, 7th ed. (Wiley, New York, 1978), pp. 35-36.

[31] M. Yuly, "Positronium in the Undergraduate Laboratory," Am. J. Phys. **67**, 880 (1999).

[32] M. Ayers, http://eande.lbl.gov/ECS/aerogels/satoc.htm, (2004).

[33] Thorn EMI Electron Tubes Photomultipliers and Accessories, (1993).

[34] N. Tsoulfanidis, Measurement and Detection of Radiation, 2nd Ed., (Taylor & Francis, Washington, DC 1995).

[35] R.B. Firestone, Table of Isotopes, Vol I, 8th Ed., (John Wiley & Sons, Inc., New York, 1999).H.

[36] Jorway Corporation, User's Manual Model 73A, 73A-1 SCSI Bus CAMAC Crate Controller, (Westbury, NY).

[37] R. Brun and F. Rademakers, "ROOT - An Object Oriented Data Analysis Framework," Nucl. Inst. & Meth. in Phys. Res. A **389**, 81 (1997). See also http://root.cern.ch/.

[38] J. Streets, D. Slimmer, "Linux IEEE CAMAC library for the Jorway 411S SCSI CAMAC Driver and the Jorway 73A SCSI CAMAC Crate Controller," Fermilab internal communication PN540, 2002 (unpublished).

[39]R.O. Nelson, *et al.*, "GEANIE at WNR/LANSCE – A New Instrument for Neutron Science," Conference Proceedings, **59**, Part I, Nuclear Data for Science and Technology, Bologna, (1997).

[40] C.J. Lister, "How Far From Stability Can We God Using Gammasphere and the FMA?," Nucl. Phys. A. **654**, 691c (1999).

[41] R.A. Muller, *et al.*, "Liquid-Filled Proportional Counter," Phys. Rev. Lett. **27**, 532 (1971).

[42] H. Zaklad, *et al.*, "Initial Images From a 24-Wire Liquid Xenon γ–Camera," IEEE Trans. Nucl. Sci. **NS-20**, 429 (1973).

[43] S.E. Derenzo, *et al.*, "Electron avalanche in Liquid Xenon," Phys. Rev. A. **9**, 2582 (1974).

[44] A.J.P.L. Policarpo, *et al.*, "Electron Multiplication and Secondary Scintillation in Liquid Xenon: New Prospects," ICFA Instrumentation Bulletin **15**, 51 (1997).