University of Washington

Imaging Research Laboratory

MiCES Module Diagnostic and Calibration Tools

Reference Manual

Craig Dowell, University of Washington, Seattle

MiCES Module Diagnostic and Calibration Tools

Reference Manual

Version Information

Version	Date	Description
0.1	10/15/2009	Final Draft for Review
0.2	10/25/2009	Expand Introduction
0.3	10/28/09	Connect with more Physics in Introduction
0.4	10/29/2009	Add Orthogonal Displays and Final Maps
0.5	11/3/2009	Why PET section. Radiation loss not important with 511 KeV photons.

Table of Contents

Table of Figures

Introduction

Positron Emission Tomography (PET) is a medical diagnostic process that takes advantage of nuclear (β+) decay to reconstruct images reflective of specific metabolic processes. In contrast to Magnetic Resonance Imaging (MRI), PET measures metabolism instead of physical structure. Since PET measures metabolism, it is often used to detect the presence of metabolic changes (both increased and decreased) before permanent physical changes occur (at which point the changes will become visible in MRI images).

Cancer cells generally have higher metabolic rates than normal cells, so a PET scan will reveal denser than normal metabolic activity at locations in the body where cancer cells are found. This increased metabolic activity may precede the physical changes detectable in MRI or X-ray. Therefore, a very small but highly metabolically active tumor can be detected by PET but may be missed by MRI. On the other hand, a large tumor with minimal metabolic changes may be missed by PET, but detected by MRI. PET is complementary to other forms of Computed Tomography.

Part of the process used in evaluating the safety and effectiveness of pharmaceuticals for human use is by studying their effects on animals. PET is useful for determining the uptake of pharmaceuticals and also their effects on these animals, however, PET scanners designed for humans must be large enough to accommodate the human body and must undergo stringent testing and certification processes, making them very expensive. Animal studies using "full sized" pet scanners are further hindered by their relatively poor resolution. Small animal PET scanners have been developed to address the cost, size and resolution problems inherent in adapting PET for to animal research. Small animal PET scanners are typically sized to be installed in a small room and provide enough resolution to detect metabolic changes in small laboratory animals.

The University of Washington MiCES PET scanner, which is the subject of this document, is a high-resolution small animal PET scanner.

Tracking Metabolism

PET begins with the selection of a metabolic process and a compound that will be used to indicate the amount, or density, of that process. Typically, a radiopharmaceutical compound is created that is an analog of a known biologically active molecule. This is done by substituting a radioactive atom into the molecules as they are synthesized.

A commonly imaged process is that of glucose metabolism. The biologically active form of glucose – dextrorotatory or D-glucose – is used as an energy source in organisms from bacteria to humans. Fluorodeoxyglucose (2-fluoro-2-deoxy-D-glucose – commonly abbreviated FDG) is a glucose analog used in PET studies. The fluorine atoms in FDG molecules are replaced by ^{18}F atoms (typically created in a medical cyclotron) which are β+ sources. ^{18}F is preferred since it has a relatively long half-life (110 minutes). The resulting compound is called ^{18}F-FDG.

The radioactive ^{18}F-FDG is injected into a subject, and participates in normal metabolic processes. High-glucose-using cells will take up more ^{18}F-FDG than other cells. FDG lacks a hydroxyl group which prevents the FDG from leaving the cell before radioactive decay occurs. As a result, the distribution of ^{18}F-FDG in the body becomes reflective of the distribution of glucose uptake in the body.

The ^{18}F atoms in the ^{18}F-FDG eventually undergo β+ decay and produce positrons which are indirectly detected as described below. This converts the fluorine into $^{18}O^-$ which combines with an available hydronium ion in the cells to produce a hydroxyl group. The presence of the hydroxyl group enables metabolism to proceed normally, producing non-radioactive end-products.

Glucose metabolism is but one of many different processes which can be imaged using PET. Different radioactive tracers (^{11}C, ^{13}N, ^{15}O, ^{22}Na, for example) when substituted into different analogs of biologically active compounds are used to image various metabolic processes.

Positron Emission

As described above, the fundamental physical process used in PET imaging is β+ decay, also called positron emission – the Positron Emission part of the PET acronym.

A nucleus with an over-abundance of protons (such as ^{18}F in the example above) can reduce its nuclear charge by one unit by emitting a positron (and a neutrino). This process can be represented by:

$$ {}^{A}X^{Z} \rightarrow {}^{A}Y^{Z-1} + e^{+} + \nu_e . $$

Conservation of energy implies that

$$ T_e + T_\nu = \left(M_P - M_D - m_e - m_\nu \right) c^2 = Q \geq 0 , $$

where T_e is the kinetic energy of the positron, T_ν is the kinetic energy of the neutrino, M_P is the mass of the parent nucleus, M_D is the mass of the daughter nucleus, m_e is the mass of the positron and m_ν is the mass of the neutrino. Q must be positive for decay to occur and therefore the kinetic energy of the emitted positron can vary from zero up to a maximum endpoint energy $\left(T_e\right)_{\max} = Q$.

In the case of the decay mechanism for ^{18}F-FDG described above, the fluorine-18 atom decays into oxygen-18,

$$ {}^{18}F^{9} \rightarrow {}^{18}O^{8} + e^{+} + \nu_e , $$

and emits a positron with a maximum endpoint energy of 630 KeV.

There are many radionuclides that decay by positron emission. Table 1 shows the maximum endpoint energies of a number of radioduclides commonly used in PET imaging.

Radionuclide	Half-Life	$(T_e)_{\max}$
^{11}C	20.4 min	0.96 MeV
^{13}N	9.97 min	1.20 MeV
^{15}O	122 sec	1.73 MeV
^{18}F	109.8 min	0.63 MeV

Table 1: List of Radionuclides Relevant to PET Imaging. Reproduced from Phelps ME, PET: Physics, Instrumentation and Scanners, Springer, New York. 2006.

Annihilation

Positrons are charged particles. In PET, these particles are emitted into tissues composed of atoms. These particles deposit energy into their surroundings by collision or through electromagnetic radiation (bremsstrahlung). The total energy loss is composed of two parts,

$$\frac{dE}{dx} = \left(\frac{dE}{dx}\right)_{rad} + \left(\frac{dE}{dx}\right)_{coll}$$

For each absorbing material, a critical energy, E_c, can be defined at which the radiation loss equals the collision loss.

Above this critical energy, radiation loss dominates. For water, the critical energy is 92 MeV, and so radiation loss is not an important factor in PET with its 511 KeV annihilation phtons.

As they pass through tissues, positrons lose energy, primarily through collision loss until most of their kinetic energy is dissipated. They then combine with electrons to form hydrogen-like states called positronium. This state lasts for a very short time and positrons and electrons then annihilate. Since the positrons and electrons are almost at rest when this happens, conservation of energy and momentum dictate that two 511 keV photons are emitted in essentially opposite directions as shown in Figure 1.

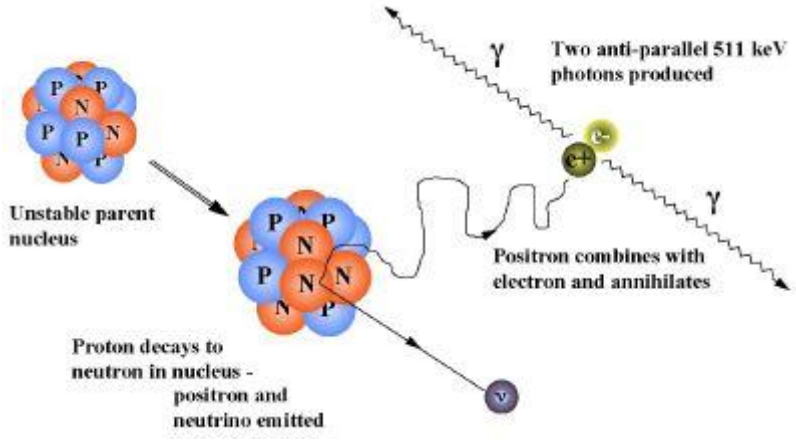

Figure 1: Beta+ Decay. Reproduced from Introduction to PET Physics, <http://depts.washington.edu/nucmed/IRL/pet_intro>

When charged particles move through the Coulomb fields of atoms, they undergo accelerations. Because of these accelerations, positrons follow a tortuous path through the materials through which they pass. It is therefore difficult to estimate the range of the positrons based on energy and material alone, and so empirical measurements are typically performed to determine a *positron range* in a specific material.

Figure 2 shows distributions of distances of annihilation events from a source for selected radionuclides in water. The distance travelled by the positron between the emission and annihilation points depends on its kinetic energy and the matter through which it travels. Note from Figure 1 that the photon-producing events (annihilations) are displaced with respect to the nucleus of the atom of interest; and from Figure 2 how this distance can vary. Positron range is a primary factor in the limiting resolution of a PET scanner. Table 2 shows mean positron ranges for several sources useful in PET.

Radionuclide	Mean Positron Range in Water
^{11}C	1.1 mm
^{13}N	1.5 mm
^{15}O	2.5 mm
^{18}F	0.6 mm

Table 2: Mean Positron Ranges of Selected Sources. Adapted from Bailey DL, Positron emission tomography: basic sciences, Springer, New York. 2005.

The radionuclide ^{18}F is commonly used in PET studies due to its combination of relatively long half life and low mean positron range. The long half life allows time for transport of the tracers and completion of imaging sessions, and low mean positron range results in sharper images than those of other radionuclides.

Another source of limitations in PET resolution is due to the fact that annihilation events occur when the positrons are not completely at rest. In this case, photons do not exit in precisely opposite directions. There is a small angular deviation, required by conservation of momentum, that averages $\pm 0.25°$. This is called non-colinearity. Since the deviation is angular, the effect on resolution is dependent on the size of the

detector itself. In a small animal PET scanner, the bore (diameter) of the scanner is on the order of 15 cm, and so the error is on the scale of a third of a millimeter.

The spatial resolution of PET images is ultimately limited by the positron range and non-colinearity of annihilation photons.

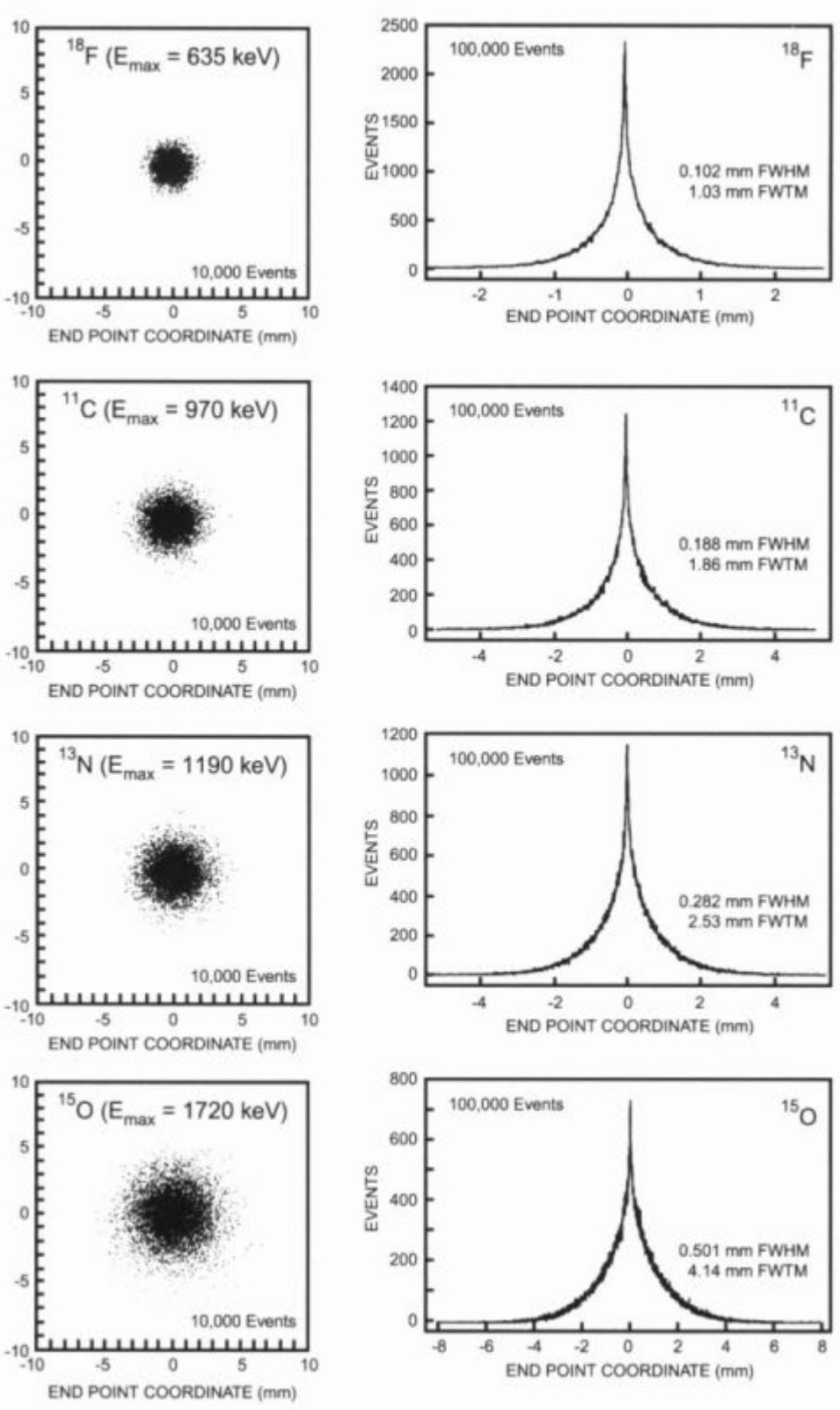

Figure 2: Distribution of Annihilation Sites. Reproduced from Phelps ME, PET: Physics, Instrumentation and Scanners, Springer, New York. 2006.

Transmission

Once an annihilation event happens, two 511 KeV photons travel away in essentially opposite directions. These photons are also travelling through matter and will also suffer interations. The primary mechanisms are the photoelectric effect and Compton scattering.

Interactions with matter will cause the annihilation photons to be deflected. This prevents the establishment of a correct line of response (discussed below) which prevents the annihilation event to be correctly located. For this reason, in PET only the detection of 511 KeV (unscattered) photons are considered.

The probability of an interaction in matter is described by a linear attenuation coefficient, such that,

$$\mu \approx \mu_{compton} + \mu_{photoelectric} .$$

Table 3 lists attenuation coefficients for selected matierials of interest in PET.

Material	$\mu_{compton}$	$\mu_{photoelectric}$	μ
Soft Tissue	~0.096 (cm^{-1})	~0.00002 (cm^{-1})	~0.096 (cm^{-1})
Bone	~0.169 (cm^{-1})	~0.001 (cm^{-1})	~0.17 (cm^{-1})
Bismuth Germinate (BGO)	0.51 (cm^{-1})	0.40 (cm^{-1})	0.96 (cm^{-1})

Table 3: Linear Attenuation Coefficients of Selected Materials. Reproduced from Phelps ME, *PET: Physics, Instrumentation and Scanners, Springer, New York. 2006.*

The probability that a photon will not be scattered over a distance x can be determined using,

$$\frac{I(x)}{I(0)} = e^{-\mu x}$$

In a symmetrical situation, the probability that both annihilation photons will be unscattered is the square of this number. For example, the probability of a single photon escaping unscattered from a mass of tissue over a depth of 5 cm is roughly 0.62; and the probability of both annihilation photons escaping unscathed is roughly 0.38.

Detection

The annihilation photons are detected using a combination of scintillating crystals and photomultiplier tubes.

When an annihilation photon enters a scintillator crystal, it can lose energy in three kinds of interactions: photoelectric emission; Compton scattering; and pair production. Pair production does not occur in the case of 511 KeV photons and so is not a mechanism here. In Compton scattering, the inbound annihilation photon (gamma-ray) strikes an electron in the scintillator material and possibly knocks it loose. This scattered electron also scatters and loses energy until it can be absorbed into an atom with an electron vacancy. When the electron is absorbed, visible or UV radiation is produced.

Once the inbound photon has been reduced in energy sufficiently, it can be completely absorbed into an atom via the photoelectric effect. This excites the atom which then relaxes, releasing radiation. The absorption of the gamma ray can also ionize the atom which releases an electron which can then be scattered and eventually absorbed as described above.

In a NaI scintillator, for example, one photon is generated for roughly every 100 eV that is deposited in the scintillator.

If all of the energy of a 511 keV photon is absorbed by the scintillator, it corresponds to a full-energy peak in a resulting energy spectrum called the phototpeak. This is labeled in Figure 3 as "511 keV."

Not all of the energy of an inbound gamma ray is necessarily absorbed by the scintillator, however. If, during a single Compton scattering event, a photon transfers the maximum kinematically allowed energy (the scattering angle is zero) and then immediately exits the scintillator, a specific energy is transferred. This produces a feature in a resulting energy spectrum called the Compton edge which is the rising edge to the left of the photopeak in Figure 3.

If the inbound gamma ray proceeds through the scintillator crystal without depositing all of its energy, but is "scattered back" into the crystal through an interaction with an external mass, it deposits the remainder of its energy and produces a so-called backscatter peak. This is seen in Figure 3 as the second, lower energy peak to the left of the photopeak.

There is a continuum of energies between the Compton edge and the backscatter peak reflecting photons that exit the scintillator crystal without depositing all of their energy. This feature is called the Compton shelf.

A final feature in Figure 3 is the 1274.58 kEv decay of the ^{22}Ne that results from the β+ decay of ^{22}Na.

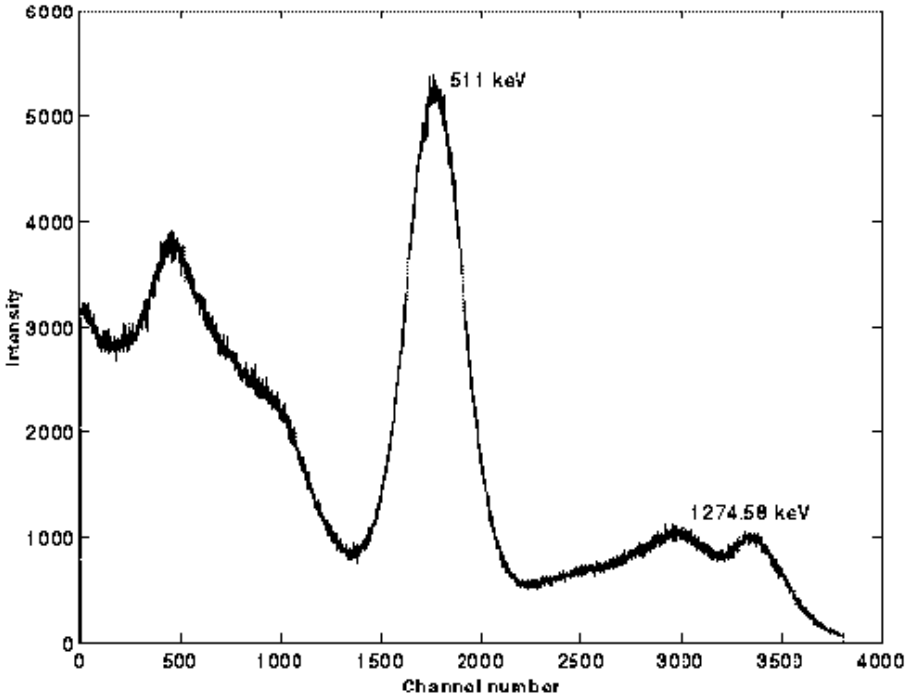

Figure 3: Energy Spectrum of Na-22. Reproduced from Development of a Time-of-Flight 3-D Demonstration Set-up for Positron Emission Tomography, M. Steen and P. Uhlen, RIT, Stockholm.

Position Information

As mentioned above, scintillator crystals are used to produce a flash of light with an intensity proportional to the energy deposited by the annihilation photon. These flashes of light are amplified and detected using photomultiplier tubes (PMTs).

A photomultiplier tube consists of a photocathode, a number of dynodes and an anode as shown in Figure 4. Light enters the PMT from the left, passes through an input window and strikes a photocathode. The light excites the photocathode material which emits electrons into the vacuum inside the PMT.

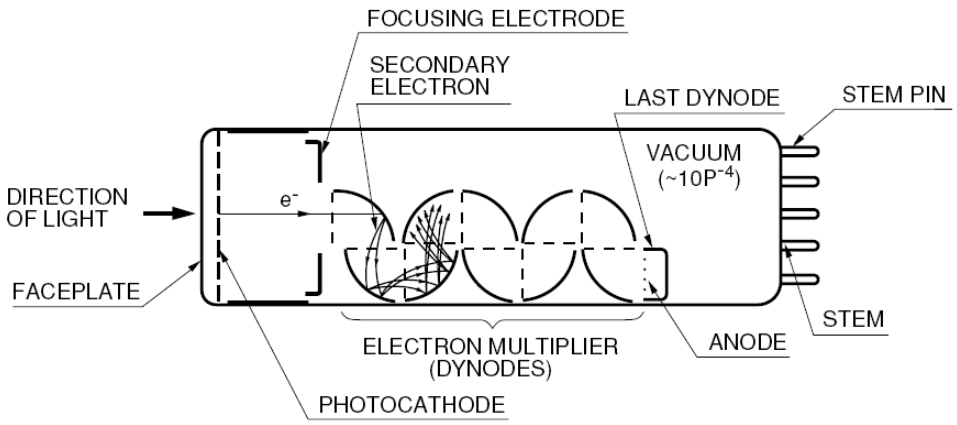

Figure 4: Schematic View of a PMT. Reproduced from Photomultiplier Tubes: Basics and Applications, Hamamatsu Photonics, 2007

These electrons are focused and accelerated onto the first dynode. This impact dislodges secondary electrons, which are accelerated onto a second dynode. Additional secondary electrons are generated there. This process is repeated with more and more dynodes, resulting in an avalanche of electrons which are eventually collected on an anode. The resulting signal is amplified and processed by the system electronics.

Position sensitive photomultiplier tubes (PSPMTs) contain an array of anodes (either a grid or crossed strips) which create output signals related to the position of the incident light on the photocathode. The basic operation of a PSPMT is essentially identical to that of a miniaturized array of individual PMTs as shown schematically in Figure 5. This is shown, schematically, in Figure 5 as a single electron directed downward from the photocathode. The electron avalanche is generated in the dynode array and is eventually directed to one of the anodes of the PSPMT, giving x-y coordinate information.

If a photon strikes the photocathode between dynode arrays, more than one anode will detect avalanche electrons. There is a characteristic response of the PSPMT to such events, shown in Figure 6. This plot shows the response of seven anodes to a spot of light moved across the face of the PSPMT. There is always some amount of crosstalk between anodes. If a spot of light directed between anodes PX1 and PX2, notice that the signature of this event is that both anodes PX1 and PX2 will fire at a relative output of slightly over 60%.

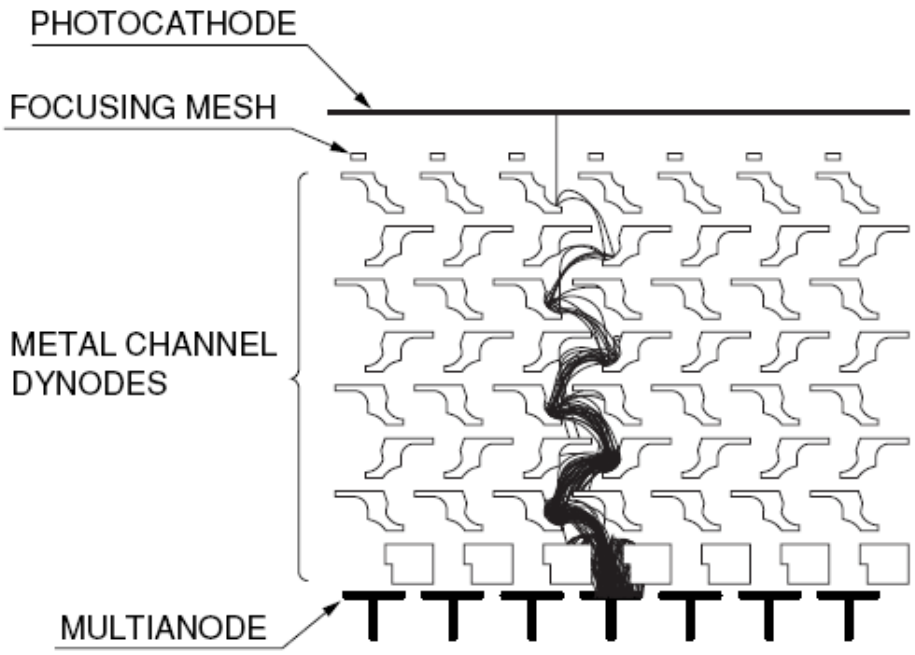

Figure 5: Schematic View of a PSPMT. Reproduced from Photomultiplier Tubes: Basics and Applications, Hamamatsu Photonics, 2007

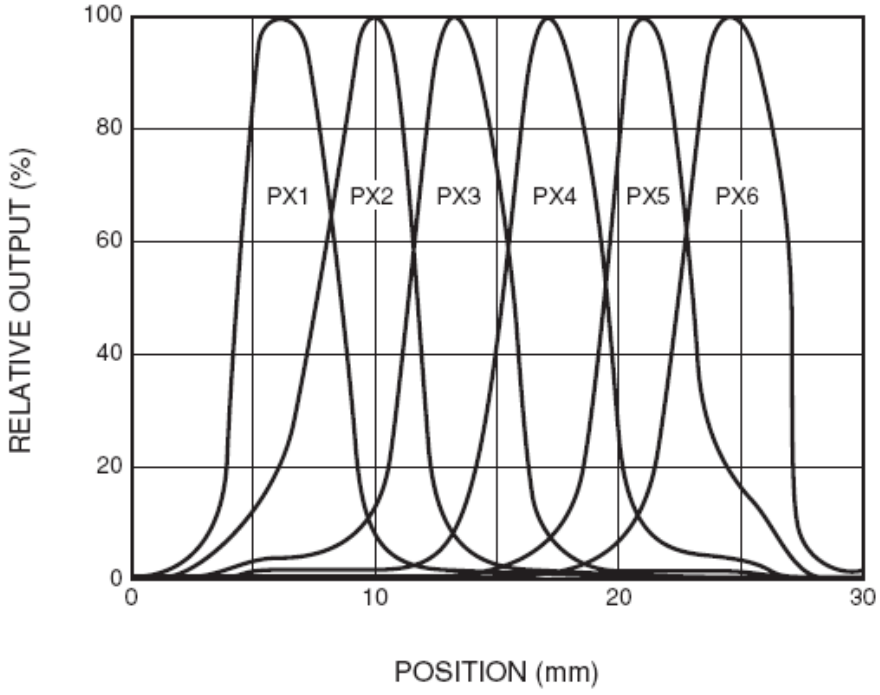

Figure 6: Position Response of Anodes. Reproduced from Photomultiplier Tubes: Basics and Applications, Hamamatsu Photonics, 2007

In a typical detector, an array of scintillator crystals is bonded to the input window of a PSPMT. A generic arrangement is shown in Figure 7 (left). In the MiCES scanner, a 22 x 22 array of scintillator crystals (each measuring 0.8 x 0.8 x 10 mm) is held in place by an opaque comb structure and bonded to the input window of a Hamamatsu Photonics PSPMT.

A scintillation event in an individual crystal then produces a spot of light on the entrance window of the PSPMT. This light that strikes the photocathode is relatively localized to a point under the crystal, which then produces a characteristic distribution of electrons in the PSPMT anodes. The position information determined from the distribution of signals in the anodes allows the system to determine which of the 484 crystals was involved in the detection event.

These crystals and associated PSPMTs are arranged in rings as shown in Figure 7 (right). Since two annihilation photons (recall Figure 1) travel in essentially opposite directions, paths of these photons define a chord in the ring of PSPMTs. The chord line is called the line of response (LOR).

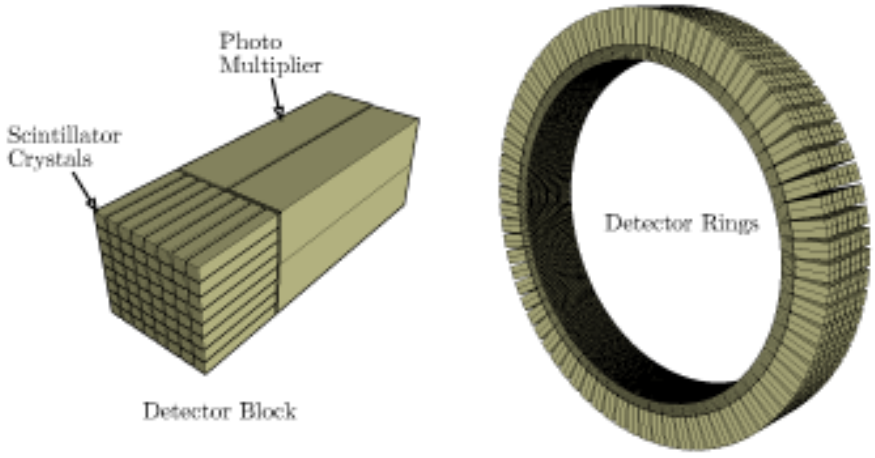

Figure 7: Scintillator Crystals are mounted to PSPMT (left) and arranged in rings (right). Taken from Basic Physics of Nuclear Medicine <http://en.wikibooks.org/wiki/Basic_Physics_of_Nuclear_Medicine/Scintillation_Detectors>

If two photons are detected along a reasonable chord line with energies of approximately 511 keV (to exclude scattered photons) and within a small time window, they are treated as resulting from a single valid annihilation event by the system and it is the LOR that is the basic unit of measurement that is remembered by a PET system (in some coordinate system).

Since there are unavoidable gaps between the detector crystals themselves, and larger gaps between the detector blocks, the entire detector ring is typically mounted in a moveable gantry and rotated to fully expose the subject.

Figure 8 shows a picture of the MiCES small animal PET scanner with the PSPMT devices and scintillator crystals surrounding a central bore. The scanner module electronics are arranged radially. The entire system can be rotated to compensate for gaps in coverage due to the physical arrangement of scintillator crystals in the bore.

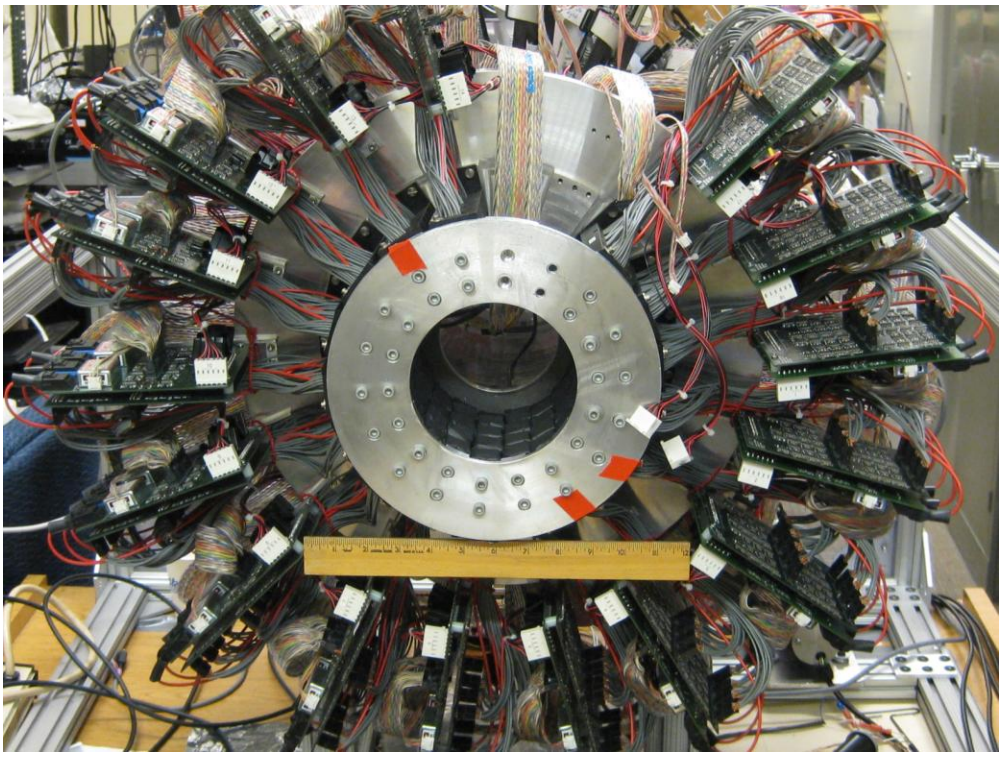

Figure 8: The MiCES Scanner on its Rotating Gantry

Image Reconstruction

The goal of image reconstruction is to build a cross-sectional image of a subject according to an inferred radiopharmaceutical density, based on detected radiation arising from annihilation events occurring in the subject.

The most basic algorithm used in image reconstruction is called *linear superposition of backprojections*. In this method, an image matrix is defined. For each valid detector event, a line of response is determined and projected through the image matrix. A value is then added to each matrix pixel that is weighted by the path length through the voxel (if a voxel is just "nicked" a small value is added, if the path is along a diagonal, the largest value is added). Figure 9 shows an example of how an image matrix could be developed using projections of lines of response.

If the single ring shown in Figure 9 is repeated axially, it can be seen that a 3-dimensional volume can be reconstructed.

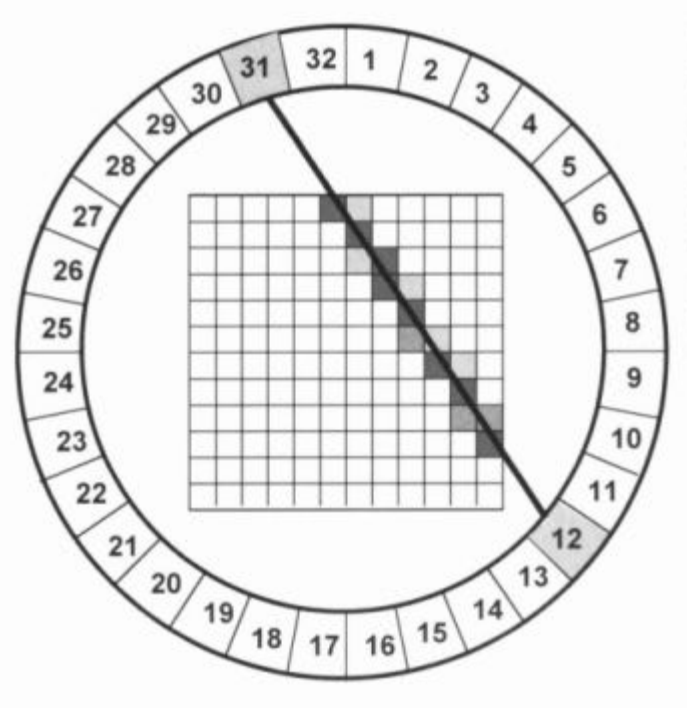

Figure 9: Illustration of Backprojection. Reproduced from Phelps ME, PET: Physics, Instrumentation and Scanners, Springer, New York. 2006.

If one takes a horizontal or vertical slice through the image matrix, a resulting image composed of the image matrix pixels represents the concentration of annihilation events along that plane in the scanner. Since the number of annihilation events correlates with the density of radiopharmaceutical tracers at that location, the image reflects the metabolic activity at that location, and finally, perhaps, the location of a tumor in a patient.

Example

Figure 10 shows a PET image of a human brain. Imagine the image shown as a portion of the image matrix slice illustrated in Figure 9. Red sections represent parts of the image matrix which have had many lines of response cross them, and therefore have been determined to be the source of many annihilation events.

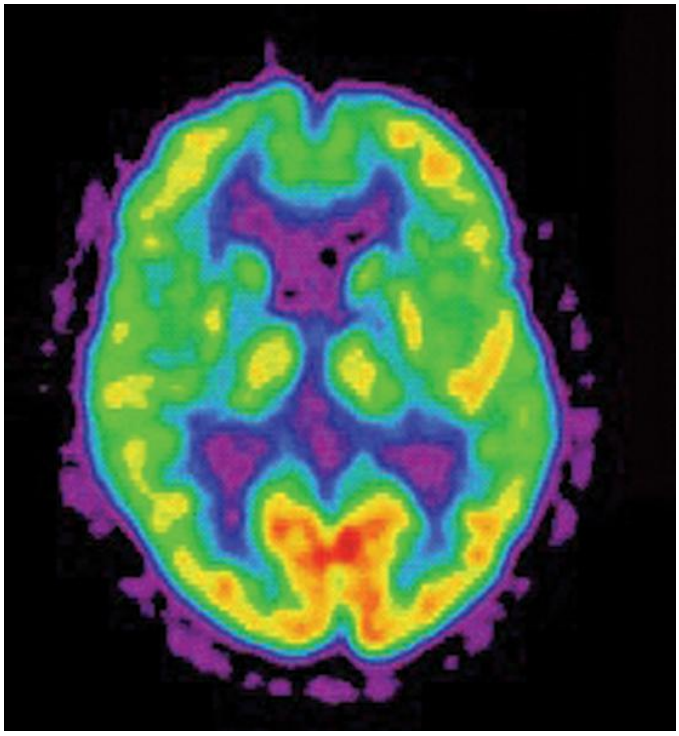

Figure 10: Axial FDG Brain Image. Reproduced from von Schulthess GK, Controversies and Consensus in Imaging and Intervention.

It is clear from the image that PET has a limited spatial resolution as compared to MRI images. It is often the case, however, that the best of both worlds is achieved by co-registering PET and MRI images and creating what is called a fused CT-PET image that depicts the structure detail available in an MRI image with the metabolic information available in a PET image.

Calibration

A PET scanner is composed of a large number of components. These components can exhibit considerable variation in their operating characteristics. For example, different PSPMT devices operating at the same voltage may have different efficiencies, and the different anodes within a PSPMT may exhibit variations in gain. Detector electronics are built with components, and these components are manufactured within certain tolerances. These tolerances combine to make each piece of the detector behave slightly differently with respect to energy and time determination.

The process of adjusting individual detector components, or determining factors to compensate for, normal variation in detector performance is called calibration. A further calibration process is used to relate count densities to an activity concentration in the subject; and is not addressed here.

There are typically several "tuning knobs" that can be varied in a given detector and several kinds of measurements that can be made in order to discover correct settings for those "knobs." In many cases there is a one-to-one correspondence between a potential measurement and a control.

For example, in the MiCES scanner hardware there is a control register that allows one to adjust the high voltage supplied to the two PSPMTs of a particular module. One can adjust the high voltage to arrange it so that the photopeak of an acquired histogram falls at a particular bin. There are, however, two entirely different signal paths for the two PSPMT devices; and there is a single high voltage supply. Fortunately, in the MiCES hardware, there is a gain adjustment per-PSPMT that can be used to compensate for differences between signal paths. There is, however, no way to compensate for the differences in gain in the anodes within a PSPMT.

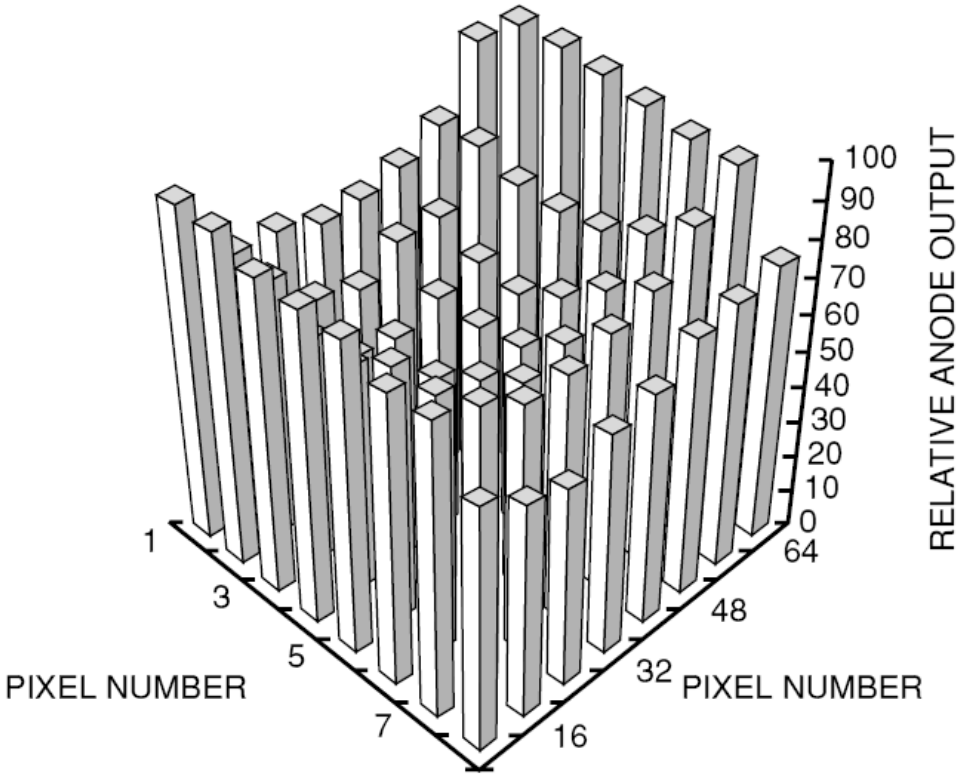

Figure 11: Gain Variations Across PSPMT. Reproduced from Photomultiplier Tubes: Basics and Applications, Hamamatsu Photonics, 2007

Figure 11 illustrates the kind of variation in gain that can be expected across the anodes in a single PSPMT. In order to control hardware costs, the MiCES scanner hardware allows histograms to be taken in four regions. The best that can be done in this situation is to align the average of the photopeaks found in the individual regions – that is, due to hardware limitations a compensation factor for each anode region cannot be computed using the on-board microcontroller.

MiCES Scanner Hardware

The University of Washington Micro Crystal Element Scanner (MiCES) device uses four rings of detectors. Each ring is composed of 18 PSPMTs for a total of 72 detectors. Each detector uses 484 crystals, for a total of 34,848 "pixels" in the PET camera. This arrangement is shown in Figure 12, below.

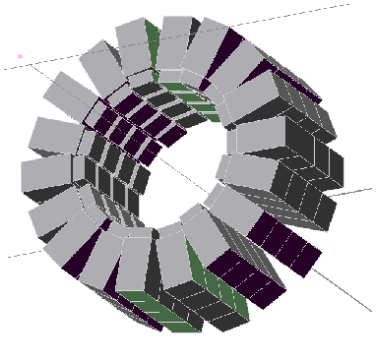

Figure 12: MiCES detector configuration

The central bore of the scanner is five inches (12.7 cm) in diameter. Figure 13 is a photograph of the bore of the prototype scanner. The PSPMT and scintillator assemblies are clearly visible, wrapped in an opaque material to keep out ambient light.

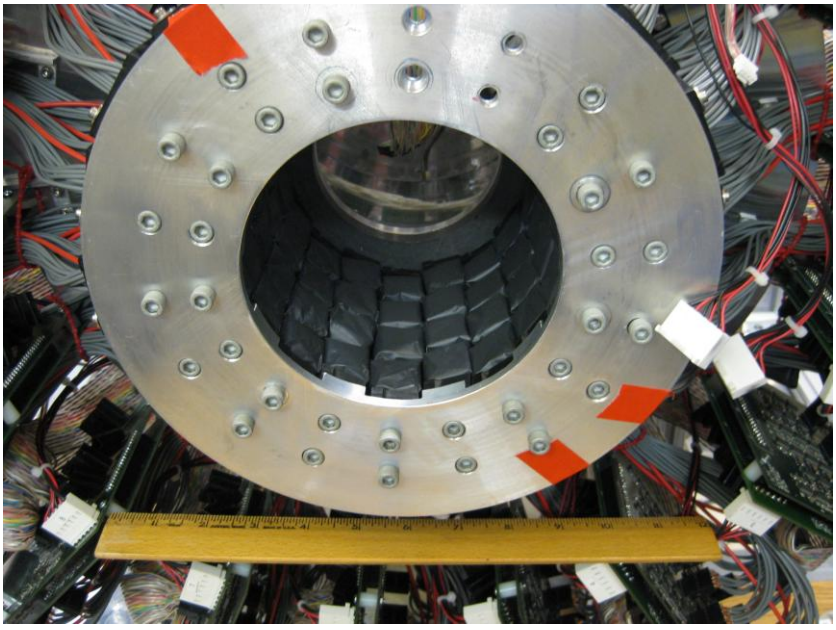

Figure 13: Photograph of the Detector Ring of the MiCES Scanner

There is one circuit board that controls each row of four PSPMT devices. The so-called Rabbit digital board is further logically divided into two modules – one called the master and one called the slave. Two PSPMT devices are assigned to the master

module, and two PSPMT devices are assigned to the slave. As mentioned above, there is one high voltage power supply for each module, so two PMT devices share a single HV supply. PSPMTs on a given module must therefore be matched for their response characteristics. The four PMT devices connected to one digital board are physically mounted together in what is called a cassette and mounted to the scanner gantry as shown in Figure 13.

Each module has a similar block diagram as shown in Figure 14, below.

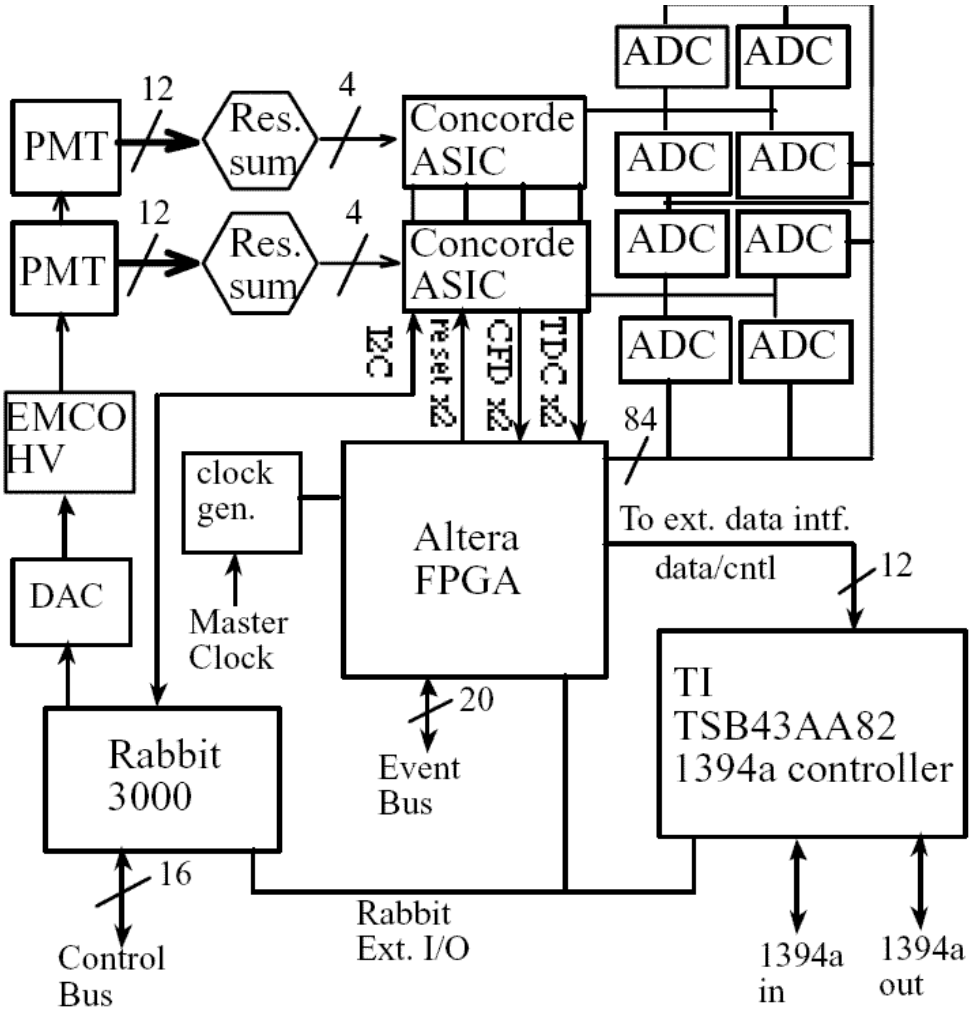

Figure 14: Block Diagram of a MiCES Module

At the lower left corner of the block diagram, one finds the Rabbit 3000 microcontroller. This is essentially the core of a Z80 microprocessor. The processor will run calibration code as well as control the gathering of histograms used by some of the diagnostic tools.

The Rabbit processor controls a Digital to Analog Converter (DAC) which controls the high voltage power supply to both of the PMT devices. The value used to initialize the DAC is one of the primary configuration items to be discovered.

Each PSPMT device (usually called simply a PMT) provides twelve outputs. There are six outputs for X-direction anodes, and six outputs for Y-direction anodes. These twelve outputs are converted into four signals, called X+, X-, Y+ and Y- by a summing network. These four signals are sufficient to define a position on the PMT corresponding to one of the 484 (22 x 22) scintillator crystals.

The summed signals from each of the PMTs are fed into a corresponding Application-Specific Integrated Circuit (ASIC). For PMT0 there is a corresponding ASIC0, and for PMT1 there is an ASIC1 on each module. Each ASIC provides a gain setting for each of the X+, X-, Y+ and Y- signals described above. Setting of the gain control registers of the ASICS is also a primary configuration goal. Each ASIC also includes a constant fraction discriminator (CFD) arming threshold used for setting event trigger levels. This CFD threshold is also a primary configuration item. Finally there is a time to digital converter (TDC) which must be configured for correct timing information. There are two configuration items which control the TDC in each ASIC.

We therefore have fifteen "knobs" that must be adjusted in order to properly configure a MiCES module:

- The HV DAC setting;
- The X+, X-, Y+, Y- Gain settings for PMT0/ASIC0;
- The CFD threshold for PMT0/ASIC0;
- The TAC/DAC settings for PMT1/ASIC1;
- The X+, X-, Y+, Y- Gain settings for PMT1/ASIC1;
- The CFD threshold for PMT1/ASIC1;
- The TAC/DAC settings for PMT1/ASIC1.

The ASICs talk to a Field Programmable Gate Array (FPGA) which controls the data flow from the ASIC to the FireWire, and which accumulates histogram data during calibration. There are no configurable settings in the FPGA.

Test Environments

Typically in the calibration and test environment, one or two digital boards are placed in a test stand with a coincidence board. The test stand is wired to accept all detection events as being in coincidence, thus defeating the coincidence detection, but allowing for histogramming with a single module.

The system can be operated in a stand-alone mode, in which all operations are controlled by the module Rabbit processors; or it can be run in a data acquisition mode which involves another Macintosh system receiving data packets over Firewire. In stand-alone mode, the primary graphical diagnostic tools are energy histograms. Data for these histograms is gathered by the FPGA and transferred to the Windows host computer by capturing a log file via by the Rabbit console (using the HyperTerminal or uCon packages). In data acquisition mode, graphical tools are available to generate both energy histograms and position histograms based on the processed event data gathered by the Macintosh computer running the MiCES software.

Stand-Alone Mode

In stand-alone mode, a MiCES digital board is connected to a coincidence board on the test stand. The coincidence board is required for correct operation of a module. Typically the coincidence board is jumpered to fake coincidences – that is to consider any detection event a coincident event,

As mentioned above, there are two "halves" to a Rabbit digital board – a master and a slave . In the test harness, four PSPMT devices are connected to the digital board and four Na22 sources are typically placed roughly six inches from the PSPMT devices in order to flood the tubes with annihilation photons -- see Figure 50.

The Rabbit processors are controlled via a console program running on a Windows host machine. Typically HyperTerminal or uCon is used for this purpose. A serial cable is connected to either the master or slave console connector and commands and responses are given and gathered using the console program.

By using HyperTerminal or uCon log files, data gathered from the FPGA histogramming commands can be transferred to the Windows host computer for detailed analysis.

Data Acquisition Mode

In data acquisition mode, the stand-alone setup is extended by connecting a Macintosh computer via a FireWire network. When commanded, the Rabbit processor will orchestrate collecting detection events into FireWire packets and sending them down the network connection to the Macintosh computer. The Mac is running a program which captures these packets and saves them to a file.

The resulting data is reduced by Mac software into a map file. These map files may be viewed using diagnostic tools described here, or by other graphical visualization tools that are part of the MiCES tool suite.

Data Visualization Tools

There is a lot of data flowing around the system in a PET scanner. It is important to be able to visualize these data in ways that make finding problems easier. We provide visualization tools in two broad categories to make this easier: Energy Histograms and Position Histograms,

Energy Histograms

An energy histogram divides the x-axis into a number of bins corresponding to an energy range. The y-axis corresponds to the number of events that lie within a given energy range. Energy histograms may be generated from log files captured in stand-alone mode or by processing map files generated in data acquisition mode.

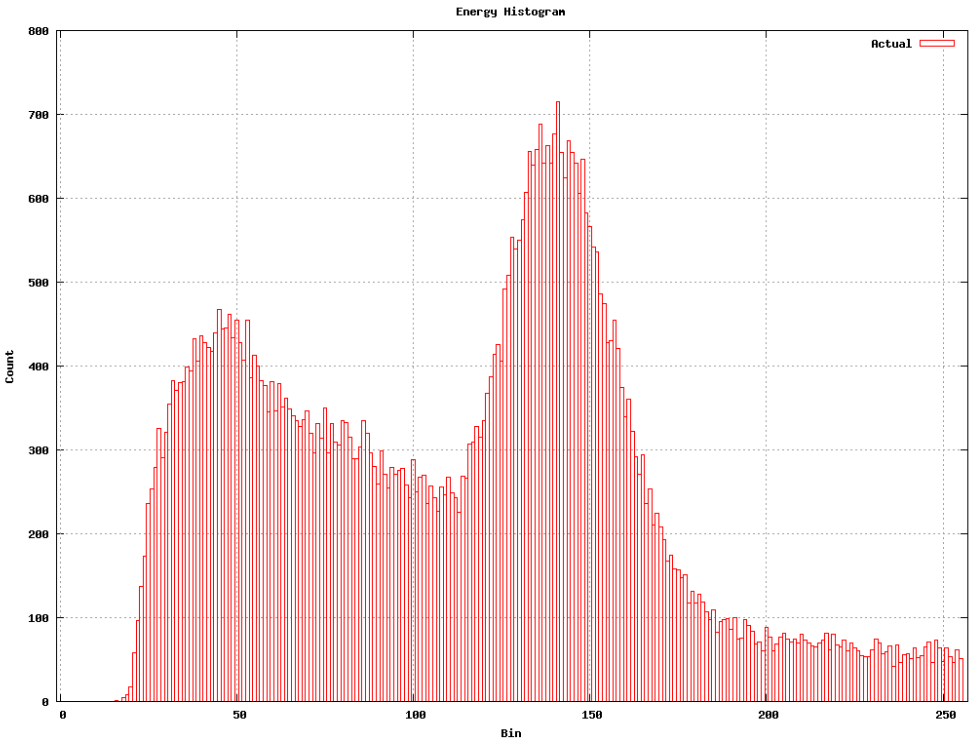

Figure 15: Energy Histogram

Figure 15 shows an example of an energy histogram generated by one of the diagnostic tools from data gathered by a Rabbit module in stand-alone mode. The Rabbit histogrammer separates energies into a 256 bin domain. This shape may be familiar to you from your physics laboratory classes. You may recall that the peak in the histogram at roughly bin 145 corresponds to a photopeak. The bins composing this peak correspond to complete deposition of the energy of the annihilation photons in the detector.

When speaking of energy histograms, one must also understand the set of pixels used to gather the data. In stand-alone mode, the FPGA histogrammer is in charge of gathering the data. It will process the X+, X-, Y+ and Y- signals of captured events

into 256 distinct addresses in the x and y directions. This creates what is effectively a 256 x 256 pixel camera.

Rather than saving the data for all 65536 pixels separately, the histogrammer separates the PMT pixels into quadrants and accumulates hits for all pixels in a quadrant into a single histogram. Since there are undesirable effects present at the edges of the PMT, not all pixel data are included in the final histogram. Instead, the quadrants are 64 x 64 pixel groups as shown in the following figure.

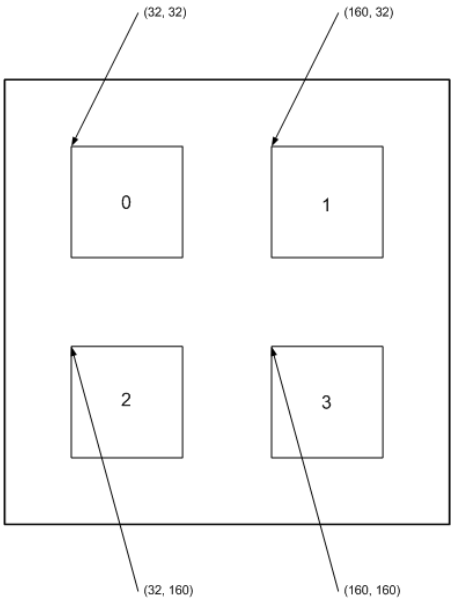

Figure 16: Layout of Histogram Quadrants

Figure 15, as it happens, shows a histogram of the events collected from all pixels in quadrant zero of a particular PMT.

Similar histograms may be generated from map files gathered in data acquisition mode. As mentioned above, these histograms are rebinned and so the domain of energy bins becomes (0, 75) rather than the (0, 256) domain seen in the stand-alone histograms. Additionally, the positioning is remapped into a 200 x 200 array. Rather than accumulate ranges of pixels into single histograms, the individual histograms are retained.

As a result of this rebinning and repositioning, these map files are collections of 40,000 separate histograms (a 200 x 200 matrix) each with 75 bins. The graphical tools in the diagnostic toolkit allow a user to select a region for display. In this way, reasonable connections may be made between regions of the map file and the FPGA histogrammer quadrants to ensure that data was faithfully transmitted from one environment to another.

For example, if a 50 x 50 pixel region beginning at point (25, 25) is plotted from a map file, it should correlate to the 64 x 64 pixel region beginning at point (32, 32) in a

histogram quadrant from a stand-alone acquisition.[1] Figure 17 shows what an energy histogram calculated from a map file might look like.

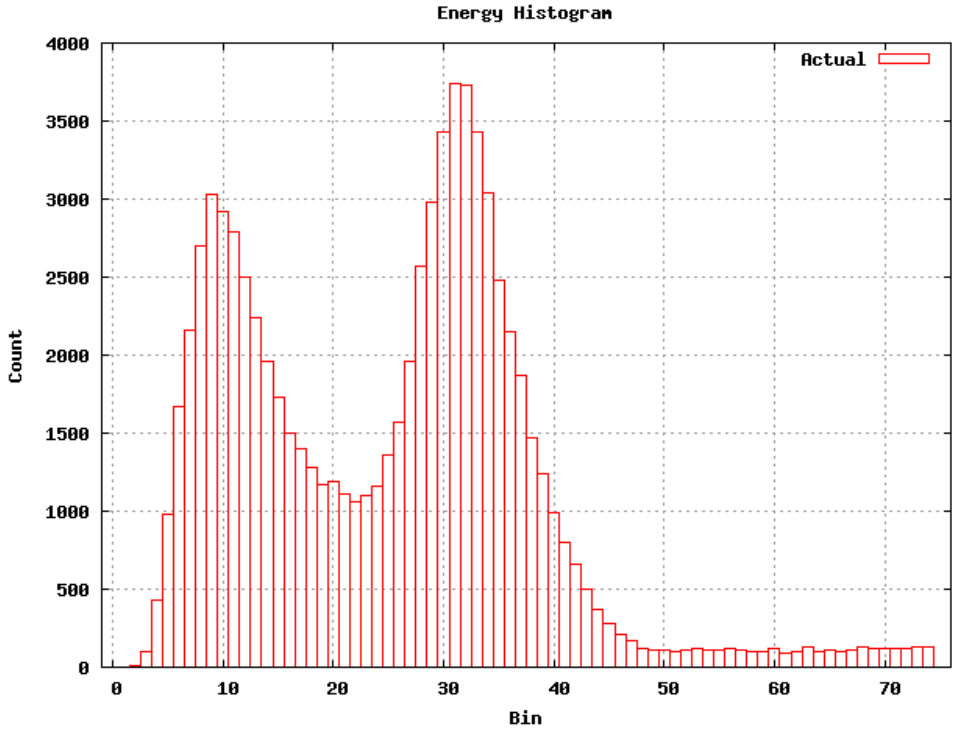

Figure 17: An Energy Histogram from a Map File

It is possible in the map file histogrammer to select arbitrary regions for accumulation. For example, a single pixel may be histogrammed or pixels may be grouped arbitrarily up to a histogram of the entire PMT.

[1] Since the map file is a 200 x 200 array and the histogrammer uses a 256 x 256 array, use 200/256 * 32 to get pixel (25,25) and 200/256 * 64 to get a 50 x 50 pixel region in the map file.

Position Histograms

It is often useful to visualize the event count and event positioning information in a single high-level display. This kind of display is usually called a crystal map since the connection between individual scintillator crystals mounted on the PMT and the display are easily correlated.

As mentioned above, when the system is run in data acquisition mode, detection events are processed by MiCES software into map files. These map files are collections of 40,000 separate histograms (in a 200 x 200 matrix) each with 75 energy bins.

Crystal Maps

The test and diagnostic package compresses this information into a single plot to try and convey a picture of the overall "health" of a PMT at a glance. Figure 18 shows an example of this output.

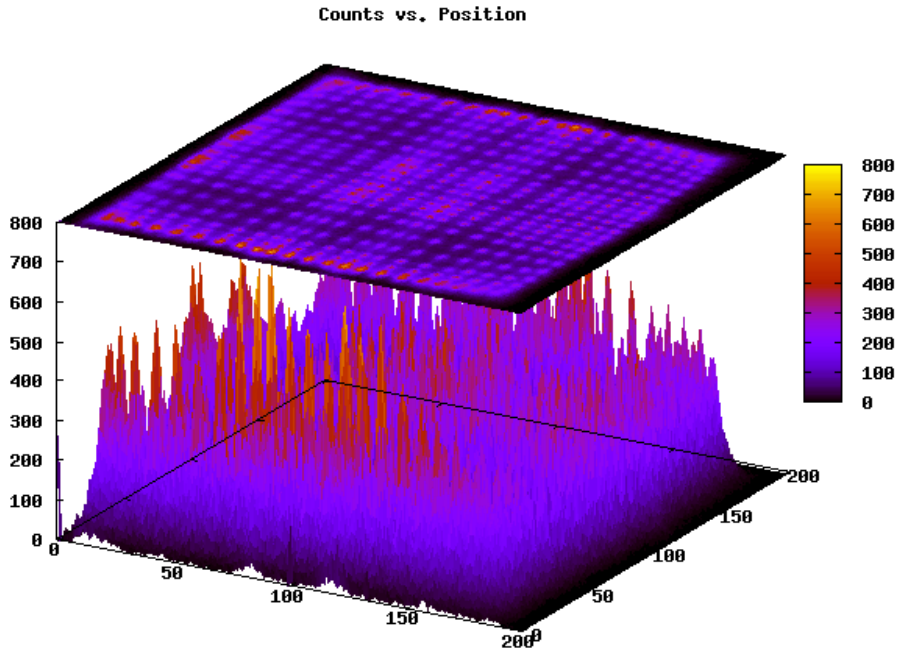

Figure 18: A Crystal Map Generated from Map File Data

The top part of the plot is a contour. It shows the total number of events in the histogram of one of the 200 x 200 addressable locations in the map file. If you look closely, you will find 22 spots across and 22 spots down on the surface which correspond to the 22 x 22 array of scintillator crystals mounted on the PMT.

The lower part of the display is a plot of the total counts assigned to each position of the map. The pattern reminiscent of a "#" character is commonly seen and is probably related to nonlinearities due to the architecture of the PSPMT itself.

Contour Plots

In addition to the three dimensional plot, a surface contour plot is also available to allow closer examination of the count patterns of the PMT. Figure 19 is an example of this kind of display.

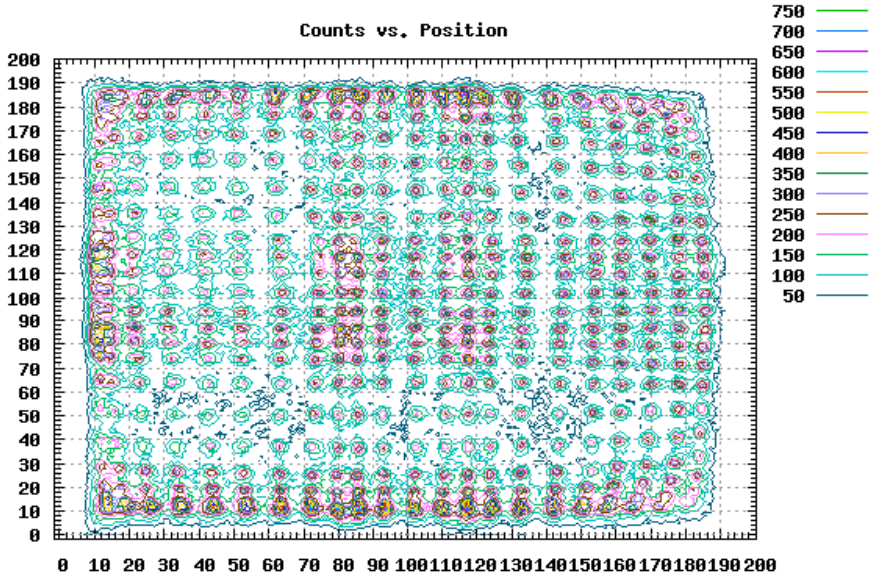

Figure 19: A Contour Plot Generated from Map File Data

Visualization Tools for the Stand-Alone Environment

Whenever we speak of the stand-alone environment we are talking about a Windows environment. The development environment chosen for Windows is based on Cygwin and the GNU toolchain.

The Build Environment

Cygwin is a Unix-like environment for Windows that provides access to GNU development toolchain as well as Unix command line support. In order to enter this environment, click the "cygwin" link icon shaped like a black "C" with a green arrow on the "Quick Launch" part of the Windows Taskbar.

You will see a new window appear which is a Unix bash shell. The prompt will look like,

```
[positron} ~>
```

If you type

```
cd craig/irl-code
ls
```

you will see a number of directories corresponding to the visualization tools. The directory `irl-code` is actually a symbolic link to the latest code. The directories with names beginning with "rabbit" are those that you run on files gathered in stand-alone mode. The remainder are run on map files generated in data-acquisition mode.

if you change into the `rabbit-histo-plot` directory and list its contents,

```
cd rabbit-histo-plot
ls
```

you will see a `makefile`, a `rabbit-histo-plot.cc`, a `rabbit-histo-plot.exe` and a text file (among possibly others).

If you type make,

```
make
```

the GNU make utility will run and attempt to build the rabbit-histo-plot.exe executable and copy it into the `irl-code/bin` directory which contains all of the built tools. Most likely, rabbit-histo-plot.exe is up to date and you will see a confirmation message,

```
make: Nothing to be done for `all'.
```

Each individual tool is built in this manner and the resulting executable is copied into `~/craig/irl-code/bin` which is set in the Windows `PATH` environment variable.

I prefer to run these tools out of the Cygwin bash shell, but since the executables are in a directory referenced by the PATH environment variable, they can be run from the Windows command shell as well.

The visualization functions all use gnuplot to generate their output, so this package was installed as well as the gnu toolchain which provides gcc, g++, make, gdb, etc.

The rabbit-histo-plot Tool

The rabbit-histo-plot tool is used to convert data gathered from a Rabbit FPGA histogrammer run into a graphical display. There is no direct way to write histogram data from a digital board directly to Windows, so the data must be transferred using the console program (HyperTerminal or uCon) log feature.

Starting a Log Session

To gather data in this way you must first start the log session. In a HyperTerminal session, select the menu item "Transfer->Capture Text" and browse for a location to store the log session data. Since the goal is ultimately to run a Cygwin program against this data, the file must be accessible to Cygwin. In order to find a place to save the data you should select "My Computer" in the browse dialog. You should then select the "C:" drive. There should be a "cygwin" directory at the root of the drive and a "home" directory inside there. In "c:\cygwin\home" you should find a directory named "glenxxx" which corresponds to the owner name of the machine. Go ahead and create a directory of your choosing and navigate there (still in the browse dialog of the HyperTerminal). Finally choose a descriptive name for your log file (something like M15-PMT0.TXT for module 15 histogram using PMT 0) and select "Save" to create the file.

Once you have the file created, you must start logging characters to this file by selecting the "Start" button on the HyperTerminal dialog.

Capturing a Histogram

Once the log session has been started, you can actually capture data. In order to do this, you need to turn on the high voltage power supply to the module. This is done with the 'j' option and should be done in two stages to protect the PMTs

```
j a0
j d0
```

If you have samples positioned in the test stand, you can then start a capture session using the 'm' (for PMT0) or 'n' (for PMT1) commands. If you wait for a couple of minutes to gather data and then stop the capture session using the 'o' (for PMT0) or 'p' (for PMT1) commands, you are ready to transfer the bits.

Writing a Histogram

Use the 'q' command to print out all of the histogram bins of all of the quadrants, and select the '0' option since you are just interested the data. This will print out hundreds of numbers which will be transferred to the log file you created above.

Ending the Log Session

To complete your "data acquisition" you must close the log session on the HyperTerminal. Select "Transfer->Capture Text->Stop" to close the log file. You now

have a text file with all of the histogram data contained in it. It is this file that the rabbit-histo-plot utility will use as its input.

Plotting the Histogram

If you now open a bash shell as described in the "Build Environment" section, you can process the data. Go to the bash shell and change into the directory in which you saved the log session text file. You should see a file of the name you selected above ending in ".TXT" in your directory. If you typed in the suggestion above, this file name will be "M15-PMT0.TXT" and you can take a look at the contents if you are interested.

```
more M15-PMT0.TXT
```

will just show all of the characters that were displayed on the HyperTermnal during the capture session.

You are now ready to run the `rabbit-histo-plot` tool. The first thing to do is to ask for some help. If you type,

```
rabbit-histo-plot -?
```

you will get usage information:

```
usage: rabbit-histo-plot infilename quadrant
```

This tells you that the arguments to `rabbit-histo-plot` are a file name followed by a quadrant number. The infilename is the name of the file you created in the HyperTerminal log session. The quadrant is a number from one to four. Yes, that's one through four. It turns out that low level FPGA documentation and internal Rabbit code speaks of quadrants zero through three, but the output routine of the histogrammer talks about quadrants one through four. Sorry, but that is something you'll just have to be aware of and deal with. In this case, quadrant zero as shown in Figure 16: Layout of Histogram Quadrants, above, corresponds to quadrant one here.

If you enter

```
rabbit-histo-plot M15-PMT0.TXT 1
```

the plot tool will generate a couple of files for you. You may see a message,

```
Cannot find DISPLAY variable: is it set?
```

This is of no concern and may be safely ignored.

The file,

M15-PMT0.TXT.1

is a data file containing the bin counts of each of the 256 bins of the histogram. This file can be imported into a plotting program (Excel, for example) directly. You will also find a file named

M15-PMT0.TXT.1.PNG

which is a PNG file of the resulting histogram. Figure 20, below, was generated using the rabbit-histo-plot visualization tool.

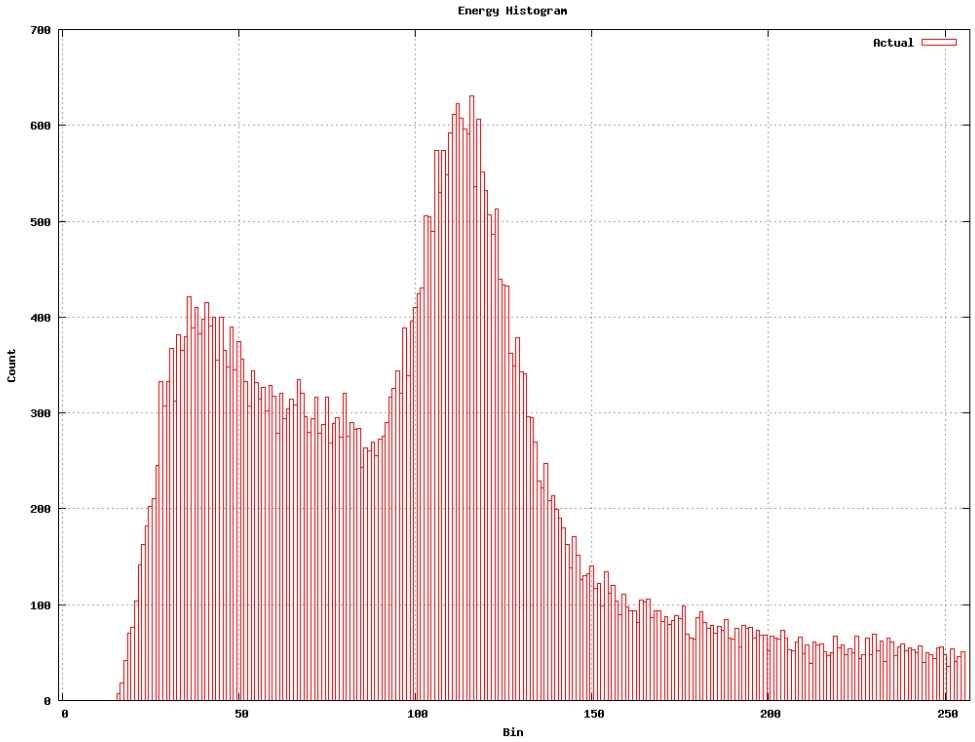

Figure 20: Example of rabbit-histo-plot Output

In order to generate histograms for the other quadrants, you would use the following commands:

```
rabbit-histo-plot M15-PMT0.TXT 2
rabbit-histo-plot M15-PMT0.TXT 3
rabbit-histo-plot M15-PMT0.TXT 4
```

It may not be obvious where the Cygwin directories are. To display the histograms you can start "`My Computer`" and then follow through the folders:

"`C:\cygwin\home\glenxxx\craig`"

to the `irl-code` directory. Typically one can just double-click on the PNG file to launch the Windows Picture and Fax Viewer.

The rabbit-histo-fit Tool

The rabbit-histo-fit tool takes the output of rabbit-histo-plot one step further and tries to determine shape information from the histogram. This is done for testing purposes to determine what is or is not a well-shaped histogram.

A number of features of the histogram are extracted and displayed as shown in Figure 21 below.

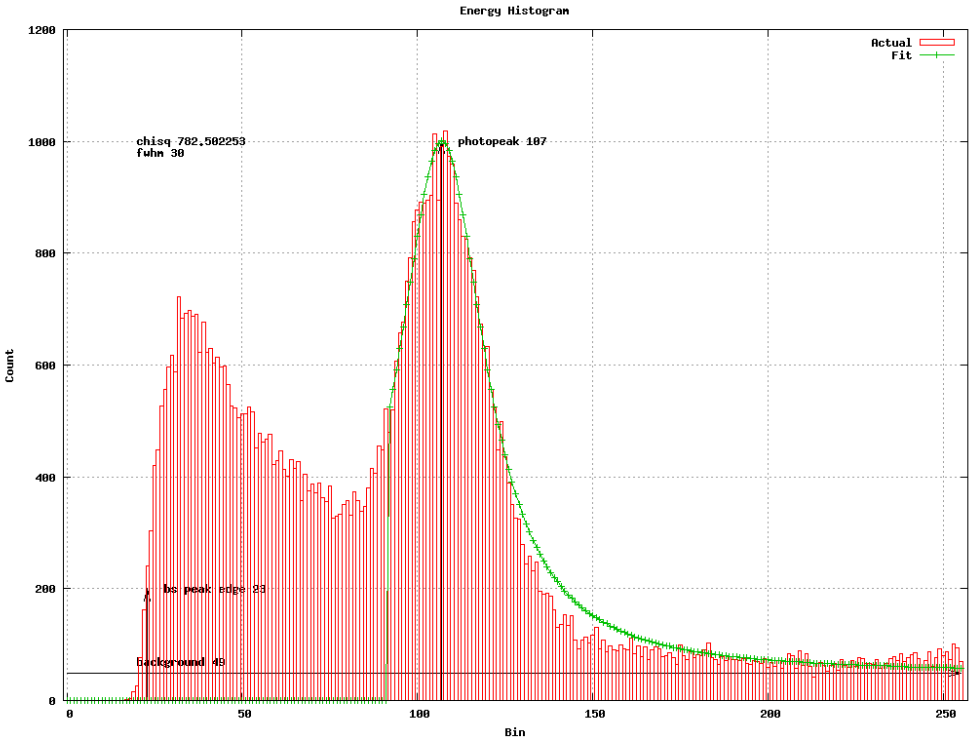

Figure 21: Example of rabbit-histo-fit Output

The first feature to notice is the photopeak. The photopeak is fit to a Lorentzian shape to determine the energy bin corresponding to the actual photopeak. It turns out that the photopeak is not always guaranteed to be colocated with the bin with the highest count. In the example above, the photopeak was determined to lie at bin 107 even though bins 105 and 108 actually have higher counts. A chi-squared test of goodness of fit of the photopeak with a Lorentzian shape is performed and the resulting chi-squared statistic is shown at the upper left of the plot. The better the fit, the lower this number will be found to be. It is expected that for a good shape, this value will be in well below one thousand.

The width of the photopeak is important in determining the quality of the histogram. The width is characterized by a Full Width at Half Maximum (FWHM) value. In figure 11 the FWHM of the photopeak was determined to be 30 energy bins and this result is printed below the chi-square statistic.

Every histogram will have some background on it. The background should never be zero in a correctly operating system, so the fit function determines this background and presents it as the horizontal line toward the bottom of the plot. In the example above, the background value was found to be 49 counts.

In every histogram there should be a photopeak, some form of Compton shelf and a backscatter peak. There should be a low-energy edge to the backscatter peak that is quite distinct from the low energy edge of the photopeak. This edge is called out by the vertical arrow at the lower left corner of the plot.

Capturing a Histogram

Data is gathered for rabbit-histo-fit in exactly the same way as it was gathered in the rabbit-histo-plot case.

Plotting the Histogram

Just as in the rabbit-histo-plot case, the first thing to do is to ask for some help. If you type,

```
rabbit-histo-fit -?
```

you will get usage information:

```
usage: rabbit-histo-fit infilename quadrant
```

This tells you that the arguments to `rabbit-histo-fit` are a file name followed by a quadrant number. The infilename is the name of the file you created in the HyperTerminal log session. The quadrant is a number from one to four, just as in rabbit-histo-plot.

If you enter

```
rabbit-histo-fit M15-PMT0.TXT 1
```

the plot tool will generate the same couple of files as did rabbit-histo-plot.

M15-PMT0.TXT.1

is a data file containing the bin counts of each of the 256 bins of the histogram. This file can be imported into a plotting program (Excel, for example) directly. This file adds data for the Lorentzian curve displayed in green as shown above. You will also get a file named

M15-PMT0.TXT.1.PNG

which is a PNG file of the resulting histogram and figures of merit. Figure 11, above, was generated using the rabbit-histo-fit visualization tool.

In order to generate histograms for the other quadrants, you would use the following commands:

```
rabbit-histo-fit M15-PMT0.TXT 2
rabbit-histo-fit M15-PMT0.TXT 3
rabbit-histo-fit M15-PMT0.TXT 4
```

Interpreting rabbit-histo-fit Plots

Running histograms through rabbit-histo-fit allows us to quantify strange looking histograms. For example, the histogram below (Figure 22) looks fine on the high energy side, but the backscatter peak seems cut off, resulting in the backscatter peak edge being found at bin 65. This is entirely too high and is the result of the CFD threshold being set too high. Being able to quantify this shape parameter allow us to come up with a cut range and then a PASS-FAIL indication.

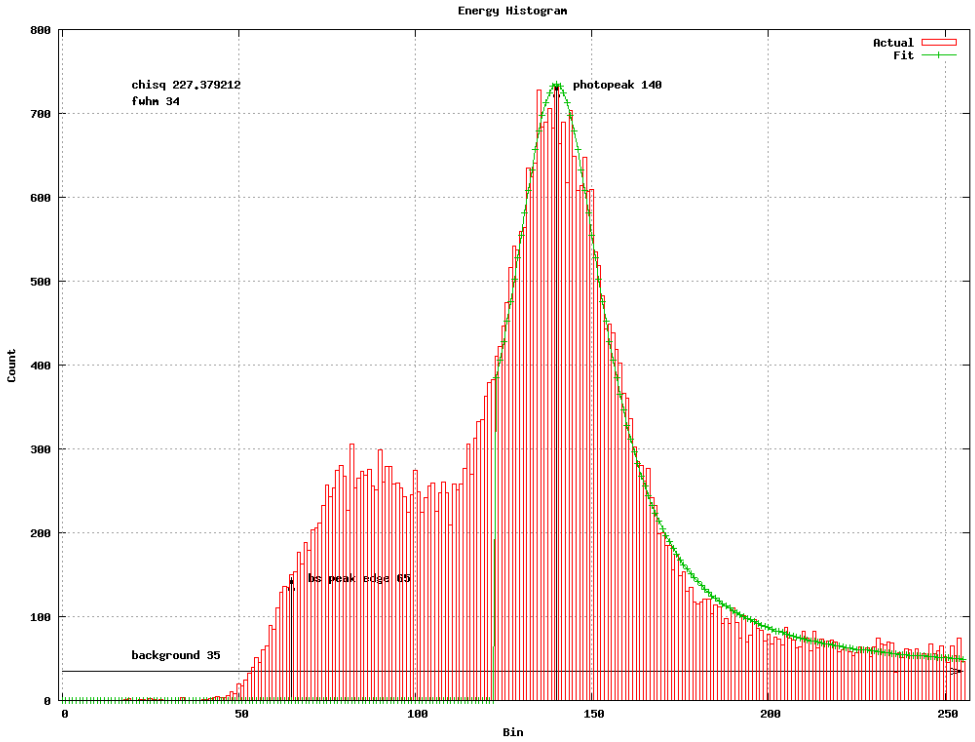

Figure 22: CFD Set Too High

A backscatter edge found too close to energy bin zero indicates a CFD value that is not set high enough and a system that will be letting in noise. A backscatter edge found to be too high is going to cut off desired parts of the histogram.

Figure 22 shows an example of a histogram of a module with the CFD threshold set too high. Notice how the backscatter peak is "eaten away" and the backscatter peak edge is correspondingly too far right at bin 65. Notice that setting the CFD too high doesn't necessarily result in increasing the number of bins with counts of zero. There may always be some low level of noise at the low levels past a module-dependent cutoff value. Below this cutoff value are bins that will never contain any hits no matter what is done with respect to calibration.

The next example, Figure 23 , shows a high voltage setting that is too high. Notice that the shape looks fine on the low energy side, with the backscatter edge being found at bin 24. There is a cutoff at about bin 18 below which we will never see any counts. The rest of the histogram is skewed toward the high energy side. The photopeak is found at bin 203, and the FWHM of the shape is also larger than usual, found here to be 54. The Lorentzian shape doesn't have any room to flatten out and so there is no room for a background level, which is reflected in the reorted background of one.

This is the signature of either a high voltage power supply being set too high or an ASIC gain setting being too high.

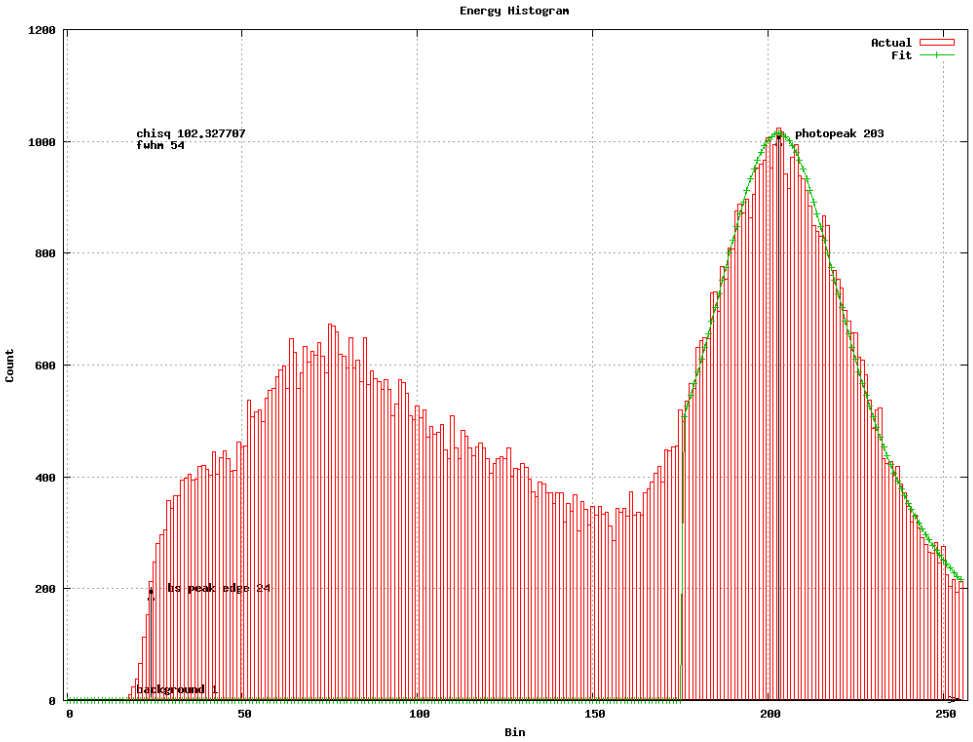

Figure 23: High Voltage Set Too High

The opposite case is shown in Figure 24, below. For this particular histogram, the ASIC gain was set too low. Notice that the photopeak is found at an unreasonably low value, as might be expected with a low gain setting, but also notice that the shape of the photopeak is "thinner" than a Lorentzian. The fitter will not try and fit to too sharp of a peak and so limits itself to a FWHM of 30. This results in an unreasonably high chi-squared statistic of almost four thousand. Further notice that the background level exhibits a "drop off" at about energy bin 200. This is presumably after the 1274.58 kEv peak of the ^{22}Ne that results from the β+ decay of ^{22}Na as shown in Figure 3.

These are collectively the signature for either a High Voltage set too low or an ASIC gain set too low. Since these two settings control essentially the same behavior in the resulting histogram it is quite difficult to tell the two cases apart.

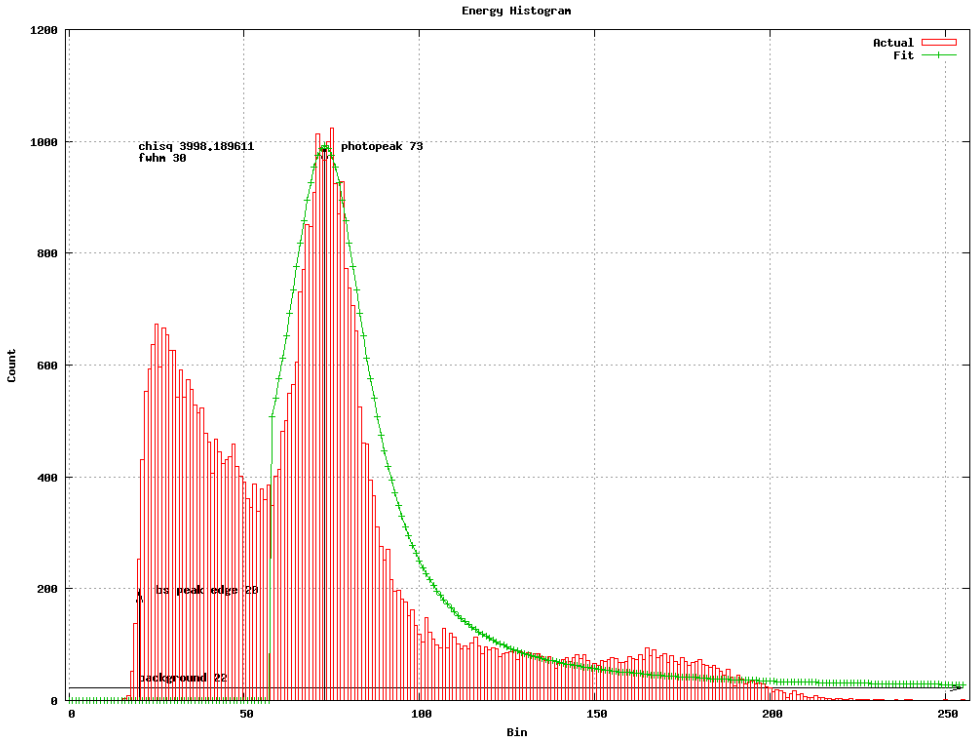

Figure 24: ASIC Gain Set Too Low

With a practiced eye, however, it is possible. The differences between the cases become more apparent at the limits. With the ASIC gain set quite low and the HV set to a typical value, the resulting histogram is a compressed version of a good histogram, as shown below in Figure 25 (left).

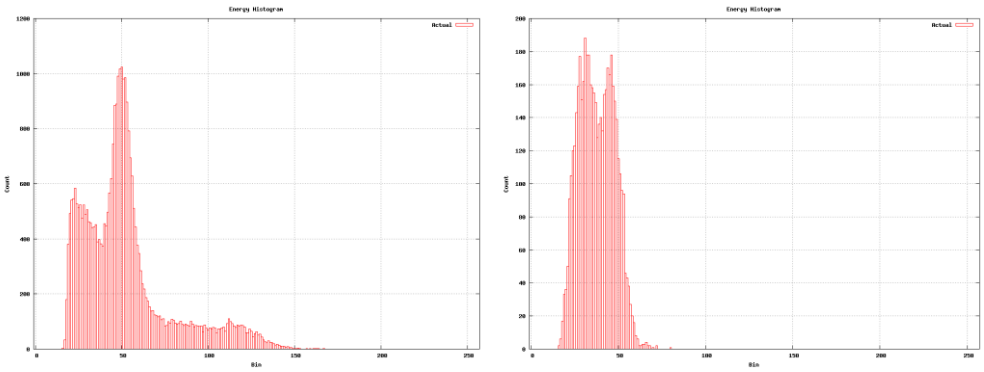

Figure 25: Low Gain (left) and Low HV (right)

Figure 25 (right) shows a histogram with the high voltage set too low. The too-low high voltage shape results in increased noise to an extent that the backscatter peak is higher than the photopeak. The tail of the photopeak is completely cut off in this case.

The rabbit-tdc-plot Tool

The `rabbit-tdc-plot` tool is used to convert time to digital converter data gathered from a Rabbit FPGA histogrammer run into a graphical display.

Capturing a Histogram

Data is gathered for `rabbit-tdc-plot` in exactly the same way as it was gathered in the rabbit-histo-plot case. In fact, the console log files that were used to gather the histogram data include the information required for the TDC plots.

Plotting the Histogram

Just as in the `rabbit-histo-plot` case, the first thing to do is to ask for some help. If you type,

```
rabbit-tdc-plot -?
```

you will get usage information:

```
usage: rabbit-tdc-plot infilename
```

This tells you that the argument to `rabbit-tdc-plot` is a file name. The infilename is the name of the file you created in the console log session.

If you enter

```
rabbit-tdc-plot M15-PMT0.TXT
```

the plot tool will generate a couple of files for you just as `rabbit-histo-plot` did

M15-PMT0.TXT.TDC

is a data file containing the bin counts of each of the 32 bins of the TDC histogram. This file can be imported into a plotting program (Excel, for example) directly. You will also get a file named

M15-PMT0.TXT.TDC.PNG

which is a PNG file of the resulting histogram. Figure 26, below, was generated using the rabbit-tdc-plot visualization tool. The histogram of the fine-grain timestamps is shown in red. There are 32 possible values. An average is taken of the counts which exclude the lowest and highest values (bins 0-11 and 20-31 are excluded). The exclusions are made since the goal of the TDC calibration program will be to bring the lowest and highest values into agreement with a uniform distribution. The part of the distribution that does not change during the calibration sequence is bins 12-19. The average shown is then the value that the calibration code will attempt to cause the highest and lowest bins to take on by varying some of its control "knobs."

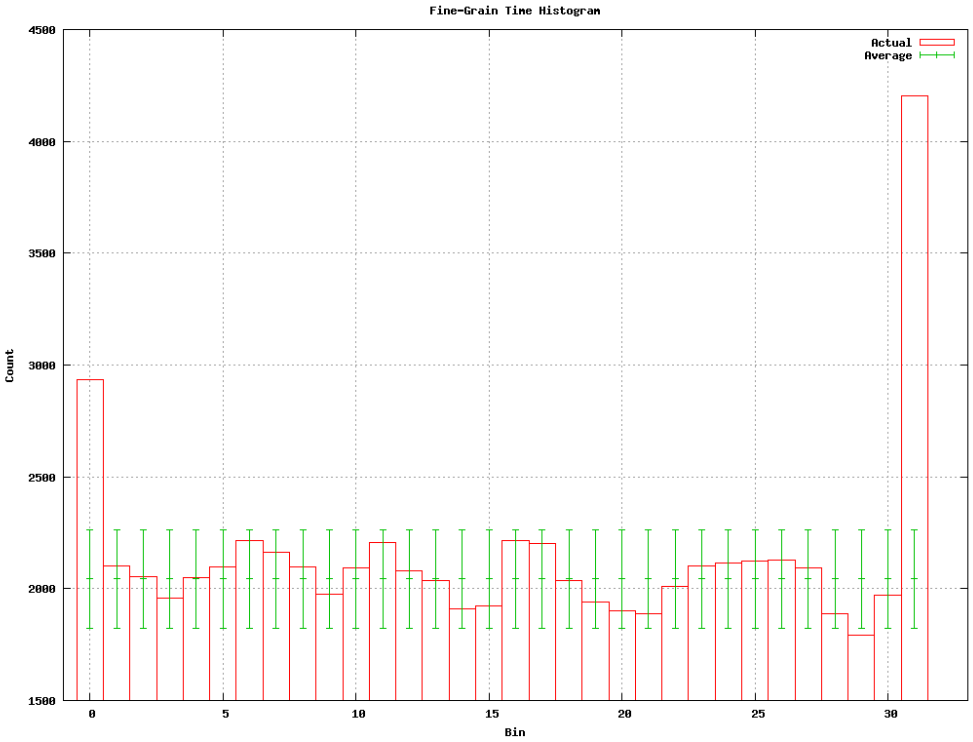

Figure 26: Example of rabbit-tdc-plot Output

The error bars on the average value correspond to twice the standard deviation of the measured bins.

The goal of the TDC calibration is to cause the TDC histogram to form a uniform distribution. In Figure 26, both the lowest and highest bins contain too many hits to be considered a uniform distribution. In order to reduce the number of counts in the lowest bin, the value stored in the TDC gain register must be decreased. In order to reduce the counts in the highest bin, the TDC offset register must be *increased*.

Visualization Tools for the Acquisition Environment

Whenever we speak of the data acquisition environment we are talking about a Macintosh environment. The development environment chosen for Macintosh is based on XCode and its associated toolchain.

The Build Environment

XCode runs in the native MAC environment using either the term or xterm applications. Just like the Windows environment, term and xterm provide a bash shell to work in. In order to enter this environment, click the "Terminal" icon in the Macintosh Dock. You will see a new window appear which is a Unix bash shell. The prompt on the Macintosh system will be different than that on the Windows system. You should see,

```
rabbitTestStation:~ physics$
```

Change into the "craig" directory in the desktop to find the irl-code directory which contains the visualization tools

```
cd ~/Desktop/craig/irl-code
ls
```

you will see a number of directories corresponding to the visualization tools. The directories with names beginning with "rabbit" are those that you run on files gathered in stand-alone mode. The remainder are run on map files generated in data-acquisition mode.

if you change into the crystal-map directory and list its contents,

```
cd crystal-map
ls
```

you will see a `makefile`, a `crystal-map.cc`, and a `crystal-map` executable (among possibly others).

If you type make,

```
make
```

the XCode (GNU) make utility will run and attempt to build the `crystal-map` executable and then copy it into the `irl-code/bin` directory which contains all of the built tools. Most likely, `crystal-map` is up to date and you will see a confirmation message,

```
make: Nothing to be done for `all'.
```

Each individual tool is built in this manner and the resulting executable is copied into `irl-code/bin` which is set in the shell $`PATH` environment variable.

The visualization functions all use gnuplot to generate their output, so this package was installed using `fink`. Xcode provides a Darwin port of the GNU toolchain which provides gcc, g++, make, gdb, etc.

The crystal-map Tool

The rabbit-histo-plot tool is used to convert data gathered from a MiCES data acquisition into a graphical summary display. There is no direct way to write histogram data from a digital board directly to the Macintosh, so the MiCES application must be used to accumulate and process digital board packets received over the FireWire.

When the acquisition and reduction process is complete, you will have a file named XYE_map0 corresponding to PMT0 of the digital board, and another file named XYE_map1 corresponding to data from PMT1.

Generating the Map

If you open a bash shell using the term or xterm applications, you can generate the crystal maps. Go to the shell and change into the directory in which you saved the XYE_map0 and XYE_map1 files.

There is one argument to `crystal-map` which is a file name. The file name is the name of one of the XYE_map files you created.

If you enter

```
crystal-map XYE_map0
```

the crystal map tool will generate a couple of files for you.

```
XYE_map0.crystal-map
```

is a data file containing the raw data used to generate the plot. This file can be imported into a plotting program (Excel, for example) directly.

You will also get two graphics file named

```
XYE_map0.crystal-map.png
XYE_map0.crystal-map-3d.png
```

which are PNG files of the resulting crystal map. Figure 27 shows an example of a 3-D crystal map file. As described above, the surface shows how the 22 x 22 array of scintillator crystals iare physically placed on the surface of the PSPMT. There are nonlinearities in the crystal map due to the physical architecture of the underlying PSPMT. Crystal positions tend to be compressed around the periphery of the PMT and also in a '#' pattern across the center. This is completely normal. Because the representations of the crystal positions are compressed in space, the number of counts present in these compressed locations is increased. In a "perfect" crystal map there would be 22 x 22 = 484 distinct spikes perfectly arranged across the surface of the PMT. In the real world, edge effects cause the crystal maps to be rounded and distorted as they approach the edges of the PMTs.

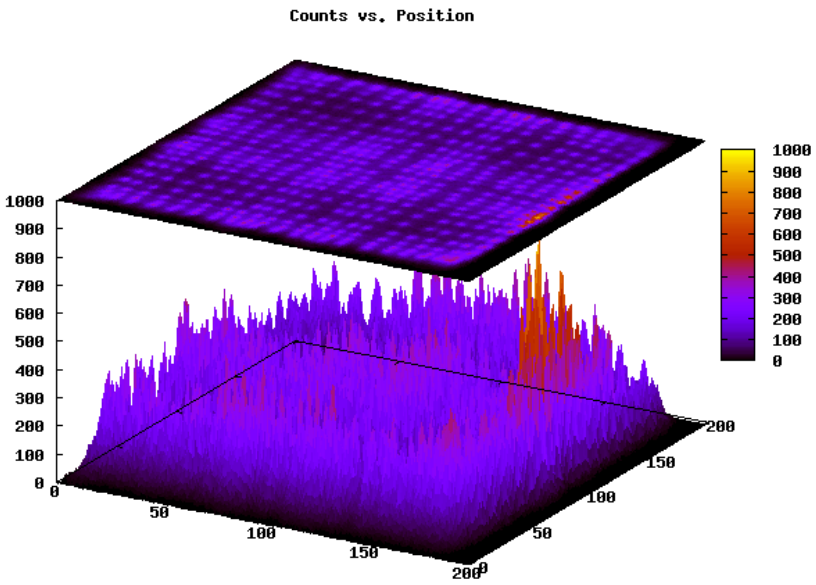

Figure 27: Examle Crystal Map

Figure 28 shows an example of a crystal map of a PMT with a distortion in the crystal lattice, presumably due to a bent comb (the structure that holds the crystals in place).

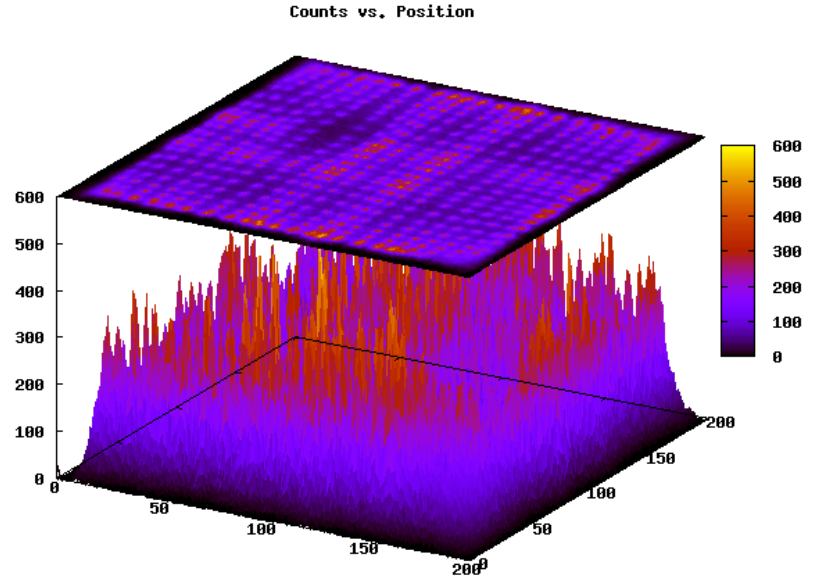

Figure 28: Distorted Crystal Lattice

Figure 29 shows a view of a crystal map in which the entire scintillator crystal array seems to have shifted during the bonding process. You can see quite nice distinct spikes corresponding to crystals, but the plot has been cut off in the front edge and many of the crystals have been cut off or merged together due to edge effects.

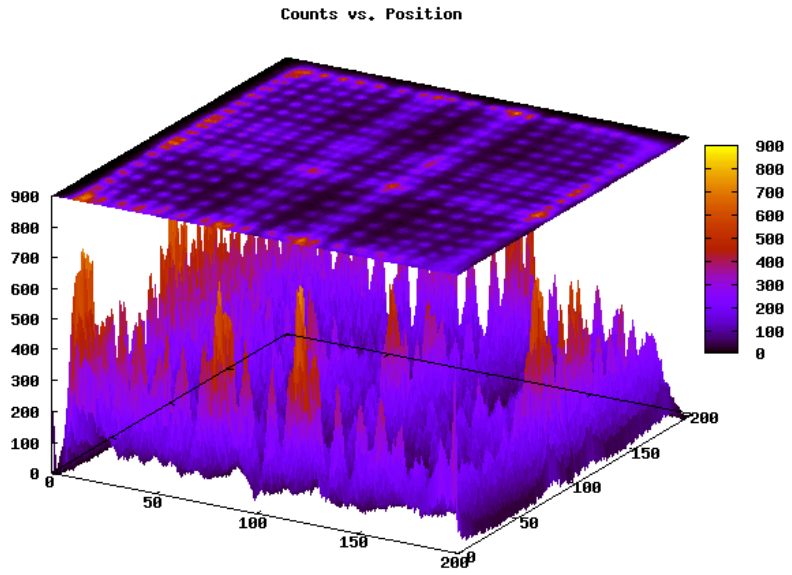

Figure 29: Shifted Crystal Lattice

Figure 30 shows a crystal map with indistinct areas where the individual crystals blur together at the left and right sides, possibly due to separation of the scintillator crystals and/or comb from the PSPMT surface.

One can imagine that the inability to precisely identify a scintillator crystal position in the cyrstal map would lead to a corresponding inability to identify a position in a Line of Response if this PSPMT were to find its way into a scanner. Just as the crystal positions seem to be blurred, the position of a feature in the subject would be blurred as well.

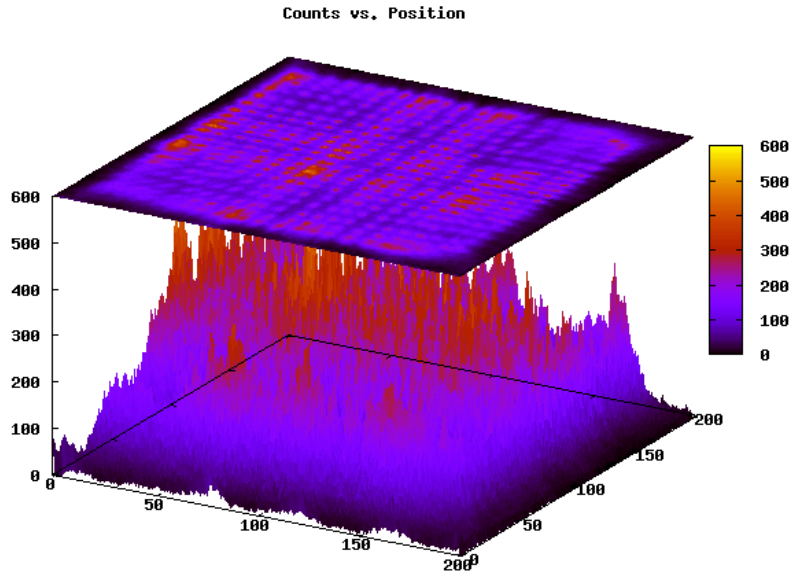

Figure 30: Indistinct Areas of Crystal Map

One final example shows a PSPMT which presumably has presumably suffered a serious delamination of the scintillator crystal assembly is shown in Figure 31. Notice that the number of events on the right third of the PMT is markedly reduced with respect to the rest of the device.

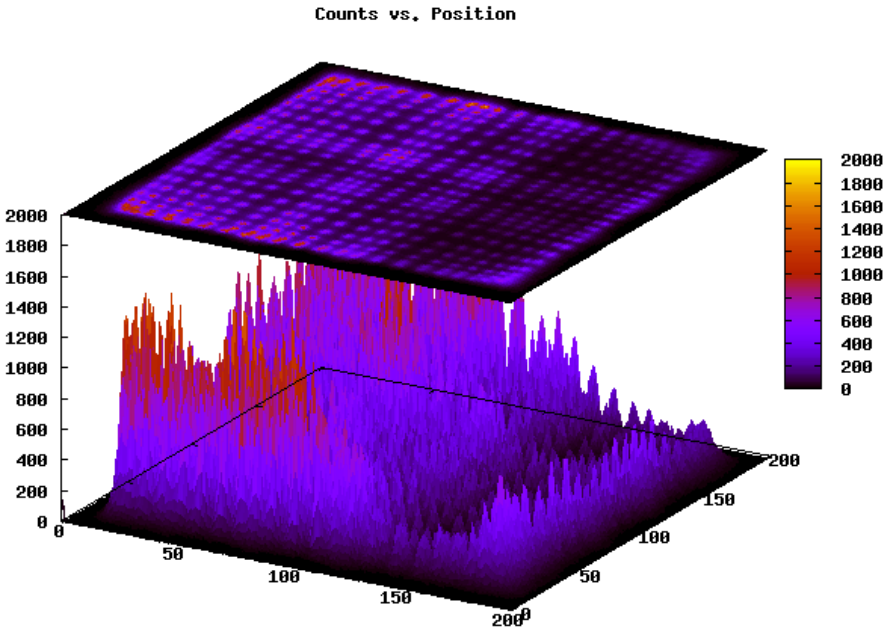

Figure 31: Presumptive Delamination

As mentioned above, in addition to creating the 3-D crystal maps, the `crystal-map` tool will generate a 2-D surface plot. This is simply a more detailed rendering of the surface plot shown in the 3-D renderings above.

Figure 32 shows an example of this kind of plot corresponding to the 3-D plot shown in Figure 31. Notice that crystals toward the left side are receiving up to 1,800 hits in their corresponding histograms, and the crystals on the right are getting as few as 100 hits in their histograms.

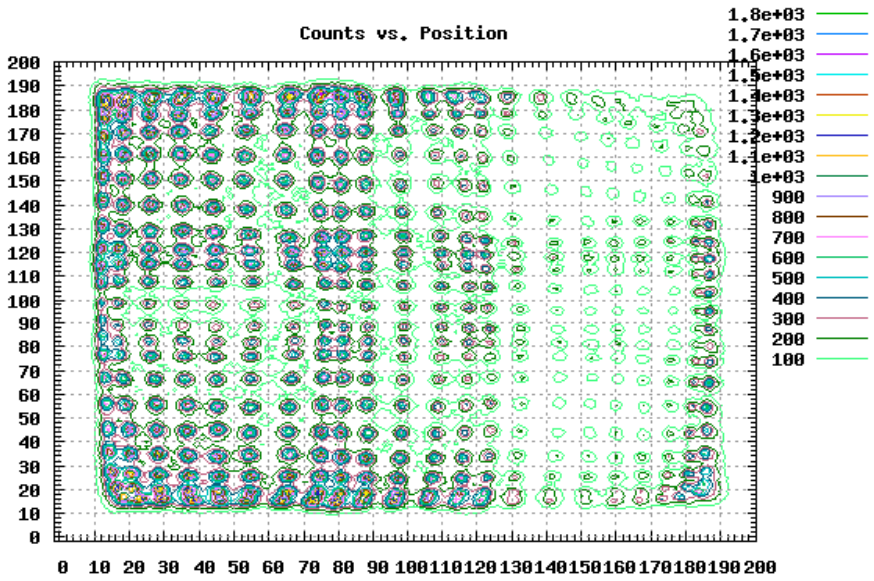

Figure 32: Surface Plot Showing Delaminated Area

The map-read Tool

The XYE_map files that were used to construct the crystal maps are actually composed of a 200 x 200 array of 75 bin histograms. It is sometimes useful to be able to investigate details of those histograms. The map-read tool provides that functionality.

See the crystal-map tool documentation for details regarding how to acquire and convert event data to XYE_map files. To understand how the data is organized, refer to Figure 32 – the surface plot shown above. There are 200 x 200 "pixels" in the surface plot. Each one of those pixels is actually a 75-bin energy histogram. To generate the surface plot, the program adds together the event counts in all 75 bins of each histogram and displays the total count. There are no hits in any of the histograms around the edges of this plot. The points representing crystal centers have lots of hits, up to 1,800 in some cases. The map-read tool allows one to pull out the actual histogram data and display it.

Generating the Histograms

If you open a bash shell using the term or xterm applications, you can generate the histograms. Go to the shell and change into the directory in which you saved the XYE_map0 and XYE_map1 files.

The `map-read` tool takes five parameters. The first parameter is a file name, and the next four parameters are an x-coordinate, a y-coordinate, a width and a height.

If you enter

```
map-read XYE_map0 0 0 10 10
```

the `map-read` tool will accumulate all of the histograms in the 10 x 10 pixel region starting at x=0, y=0. It will generate two files:

```
XYE_map0.000.000.010.010
```

and

```
XYE_map0.000.000.010.010.png
```

The first file is a data file containing the 75 bin entries of the generated histogram, suitable for importing into a graphing or analysis program such as Excel. The second file is the graphics file generated by gnuplot containing the histogram.

From looking at the crystal map of your data, you might expect very few hits in this area. This is generally true, and so you should not be surprised to find a histogram that looks like Figure 33.

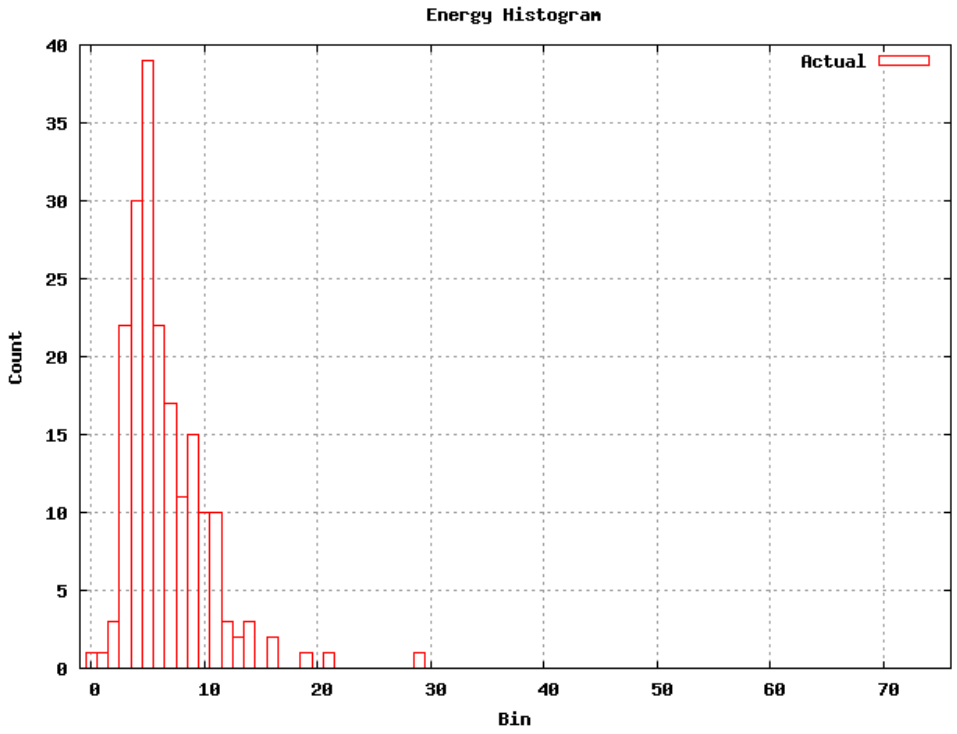

Figure 33: Histogram of Sparse Region

One of the most interesting things to do is to try and correlate the histograms found in the map files with those from the FPGA histogrammer (cf. rabbit-histo-plot). Recall that the FPGA histogrammer breaks up the PSPMT into a 256 x 256 pixel region. Figure 7 shows that it accumulates a 64 x 64 pixel section starting at the point (32, 32) into a histogram it calls "Quadrant 0." What we need to do is to scale these numbers and ask `map-read` to histogram them from the XYE_map file. The scale conversion factor is 200/256 applied to each of our numbers 32 32 64 64. This maps to 25 25 50 50. So if you enter

```
map-read XYE_map0 25 25 50 50
```

you should get a histogram that has the same shape as the one you found using `rabbit-histo-plot`. Perhaps you would find something like Figure 34.

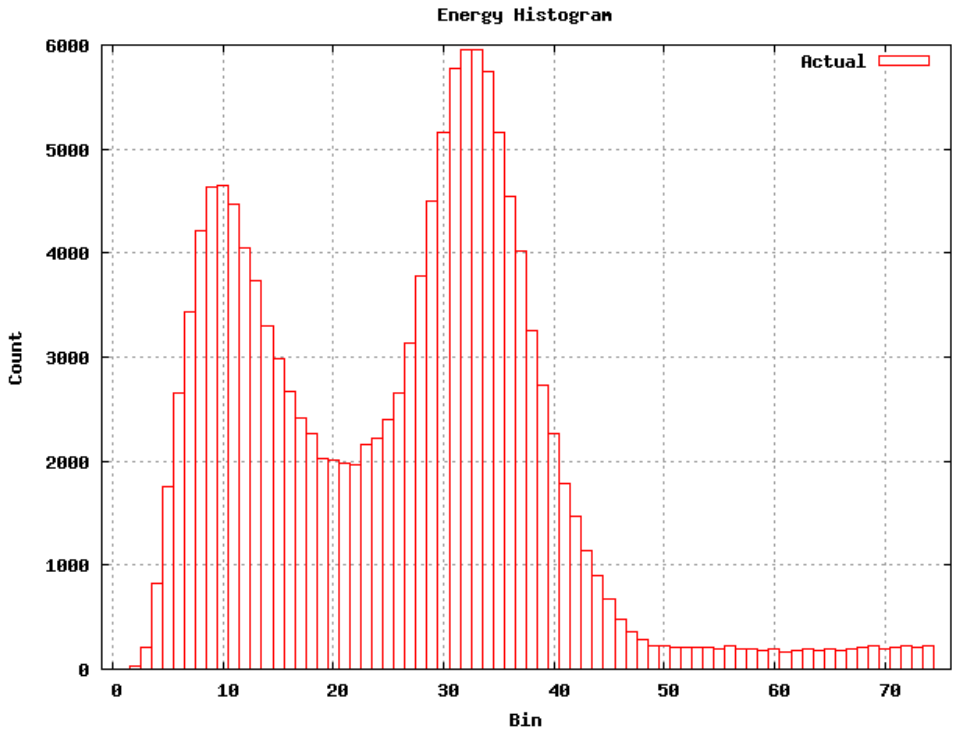

Figure 34: Histogram of FPGA Quadrant 0 Region

You can explore the histograms of a particular PSPMT and find perhaps surprising results. Figure 35 is an example of a surface contour plot generated by crystal-map. This is another example of a PSPMT which apparently has a delaminated scintillator crystal assembly. There are very few hits on the vertical section at the left third of the device and through a section across the device just below the middle.

One can examine histograms from various regions of the underlying map file and see if they can be related to the features on the contour plot and perhaps to defects on device itself. You may have a similar XYE_map file lying around with which you can experiment.

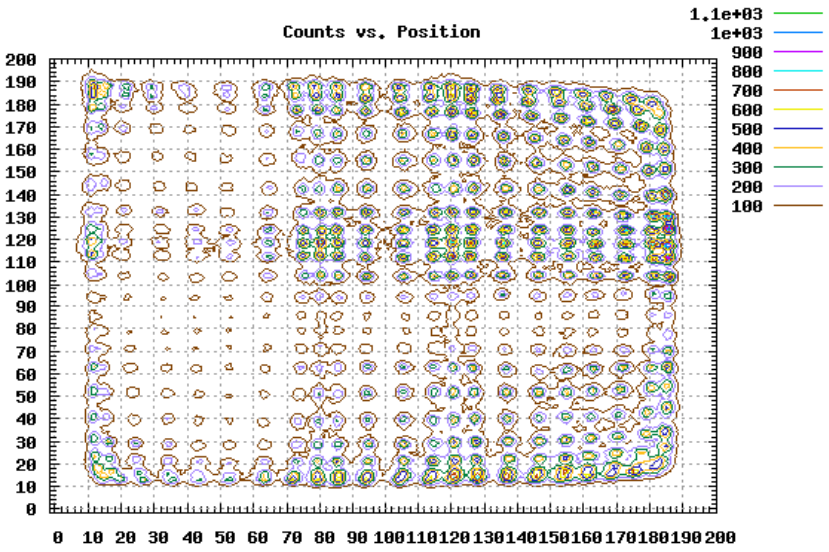

Figure 35: Example Countour Plot

From the countour plot shown in Figure 35, we expect to see lots of hits and nice looking histograms in the FPGA quadrant 3 area. Recall (see Figure 16) that this is 160 160 64 64, which maps to 125 125 50 50 in the XYE_map file. Try

```
map-read XYE_map0 125 125 50 50
```

This results in a file named XYE_map0.125.125.050.050.png which, when examined results in a beautiful histogram, shown in Figure 36.

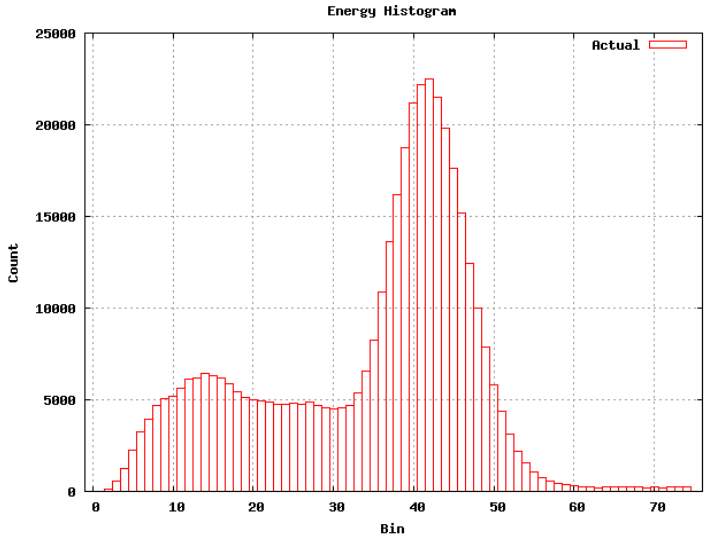

Figure 36: Histogram of Quadrant 3 Area

It turns out that Figure 34 is a histogram of the quadrant 0 area of the same device used in the contour plot. Notice that the number of lower energy events is relatively much higher in Figure 34 as compared to Figure 36. In extreme cases, this "backscatter peak" can actually be larger than the photopeak as shown in Figure 37.

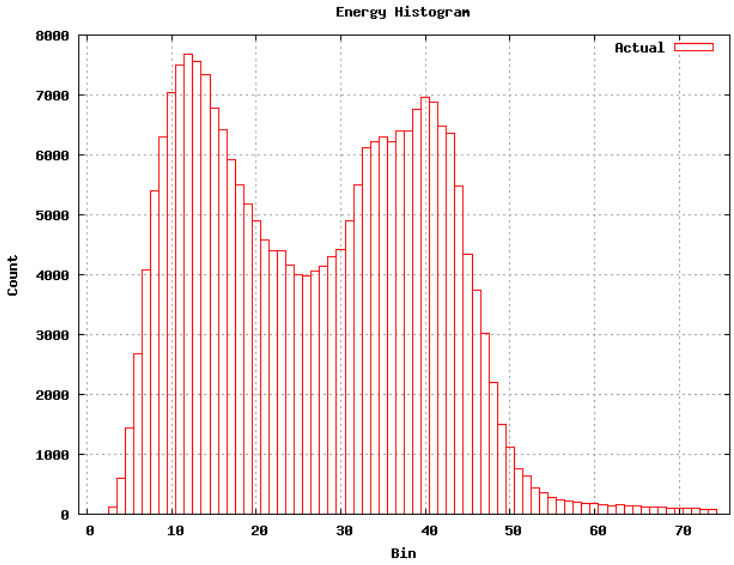

Figure 37: Backscatter Peak Larger than Photopeak

One can imagine how such a histogram could be formed. Imagine that there are several anodes present in the histogrammed area with different gain characteristics present such that the low energy hits add constructively, while the high energy hits are separated. In Figure 37 there do seem to be two photopeaks separated slightly in energy; however there seems to be a single, narrow backscatter peak. This histogram makes sense in the light of that explanation.

Conditions that produce histograms with the backscatter peak larger than the photopeak have been deemed "okay" in the scanner project, so the test and calibration software must be able to handle these cases gracefully. This complicates the calibration functions since it is not possible to identify photopeaks simply by looking for the largest count.

Let's take a closer look at another map file. Just because the crystal map looks fine doesn't mean that the histograms inside are good. The crystal map shown in Figure 38 is made from data taken in a system test of the MiCES system. It looks fairly nice. There are distinct peaks corresponding to the 22 x 22 scintillator crystal array.

The scintillator array looks nicely centered on the PSPMT and the edge effects are not too serious. The peaks in the crystal maps only reflect the total counts of the histograms in the file, though.

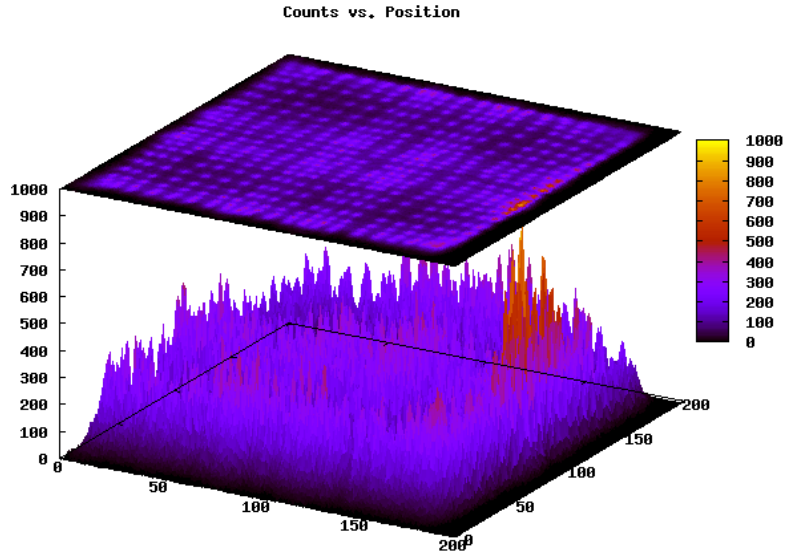

Figure 38: A Good Crystal Map?

What do the actual histograms look like? Let's take a look at one of the quadrants. Figure 39 shows the accumulated histograms from quadrant 0 of the PMT.

There is a recognizable photopeak, but the shape of the histogram is hideous. There is something clearly wrong with this run that results in a widening of the shapes, perhaps noise in the system. The cause of this effect is unknown at this time.

The energy histograms that lie underneath a good looking crystal map can be extremely "deformed." It is always a good idea to verify the histogram shapes even though a crystal map looks good. There is no guarantee that the underlying data represents good histogram shapes.

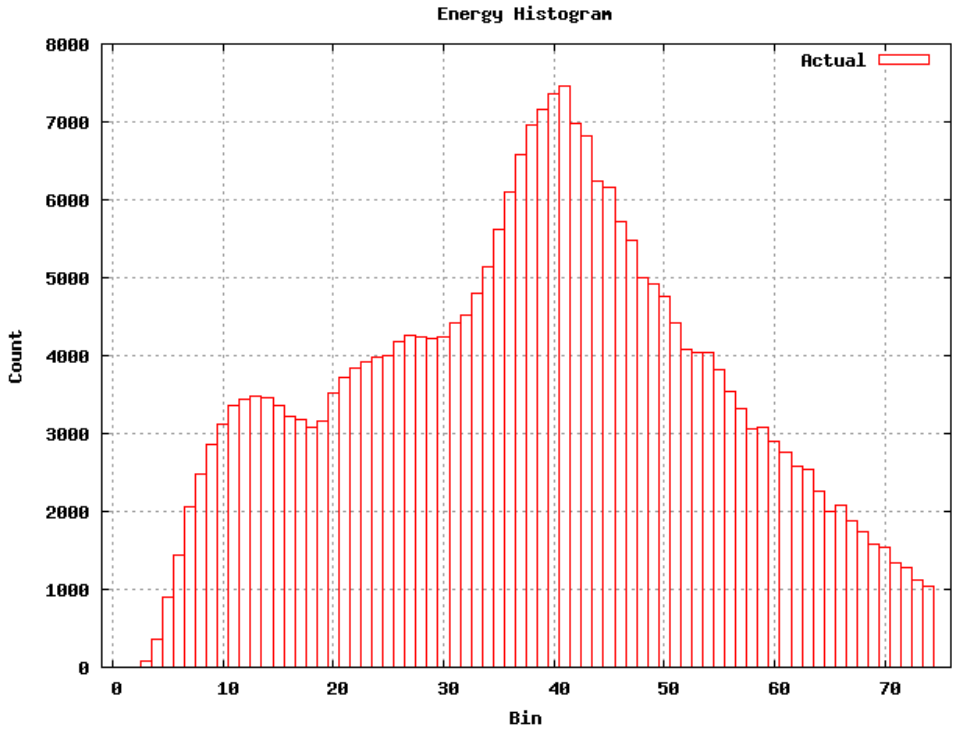

Figure 39: Ugly Histogram in Good Looking Cyrstal Map

Figure 40 shows another example of a histogram found in a PSPMT with a very nice looking crystal map. A closer indicates that the high voltage was set too high, resulting in a photopeak shifted too far to the high energy side of the histogram. The noise at the low energy side indicates that the Constant Fraction Discriminator threshold was probably set too low. This histogram exhibits the shape widening effect that may be due to system noise but possibly just reflects the high voltage or ASIC gain setting.

The important thing to take away from this example, however, is that the crystal map of all of these histograms looked great. It is important to examine the histograms of the maps to ensure that they are also fine (i.e. they should look more like Figure 36 than Figure 40).

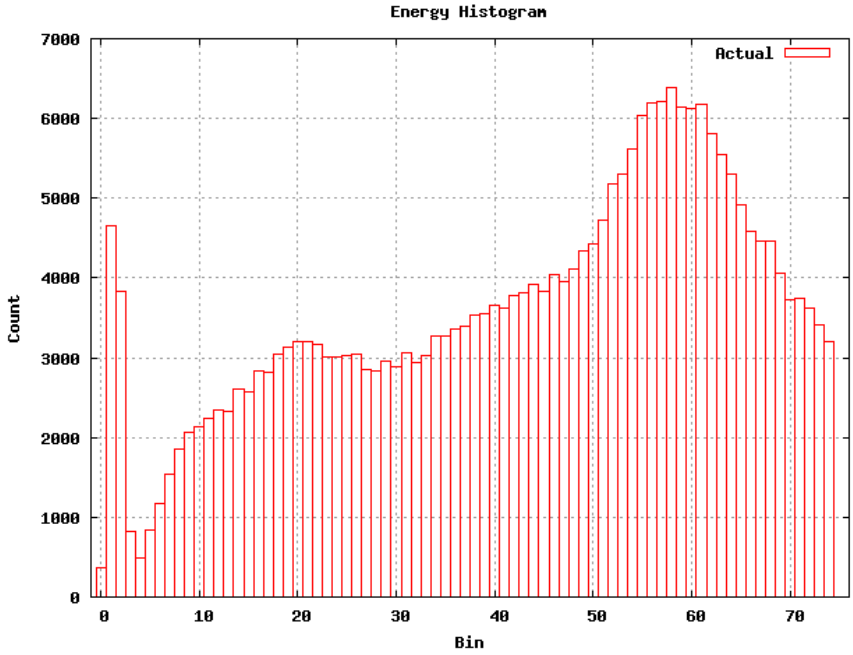

Figure 40: Can a Histogram be Worse?

Automatic Calibration and Test Tools for the Rabbit

The primary goal of the project was to develop automatic calibration routines for the Rabbit processor. Again, whenever we speak of the Rabbit processor, we imply the stand-alone environment and 256 bin histograms of the four quadrants defined by the FPGA histogrammer.

As mentioned in the Scanner Hardware section above, we have fifteen electronic "knobs" that must be adjusted in order to properly configure a MiCES module:

- The HV DAC setting;
- The X+, X-, Y+, Y- Gain settings for PMT0/ASIC0;
- The CFD threshold for PMT0/ASIC0;
- The TAC/DAC settings for PMT1/ASIC1;
- The X+, X-, Y+, Y- Gain settings for PMT1/ASIC1;
- The CFD threshold for PMT1/ASIC1;
- The TAC/DAC settings for PMT1/ASIC1.

The automatic calibration and test tools are separated into two broad categories. Calibration tools attempt to turn those electronic "knobs" in a way that results in optimum operation of the modules. These are further divided into "quick" and standard calibration processes. The difference between the approaches is that virtually no time taken to analyze histograms in the quick versions. The standard versions spend quite a bit of time analyzing the histograms to determine if they meet acceptability criteria. Also the standard versions can tolerate some relatively non-standard shaped histograms while achieving acceptable results.

The second category is a test process. These tools accumulate histograms and decide if the resulting histograms meet programmed criteria for goodness. The output is simply a PASS or FAIL indication on PSPMT0/ASIC0 or PSPMT1/ASIC1 (see the scanner hardware description).

Automatic Calibration Operation

Recall that there are two modules on each Rabbit digital board – a master module and a slave module. There are two PSPMT devices connected to each module, called PMT0 and PMT1. The high voltage supply on each module affects both PMTs which have been matched with respect to their responses to supply voltage. There are two ASICs on each module, each handling a PMT. The ASIC handling PMT0 is called ASIC0 and the ASIC handling PMT1 is called ASIC1.

Each ASIC has a gain setting for each of the X+, X-, Y+ and Y- channels from the summing board. Typically these four gain settings per ASIC are set to a single value. Each ASIC has a CFD threshold and TAC/TDC controls.

Since the high voltage power supply is shared between the two PSPMTs on each module, the gain settings on the ASICs are the only way to separately adjust the responses of the two devices. The high voltage and ASIC gains (if four settings changed as a group) do essentially the same thing: they cause the histogram to be horizontally expanded. If one holds the ASIC gain setting constant and sweeps the high voltage setting from low to high, it has the effect of leaving the low energy features of the histogram the same, but moves higher energy features off to the right. The changes in the histogram are quite linear over a relatively large range. If the high voltage is set too low, the histogram becomes distorted, as shown in Figure 29.

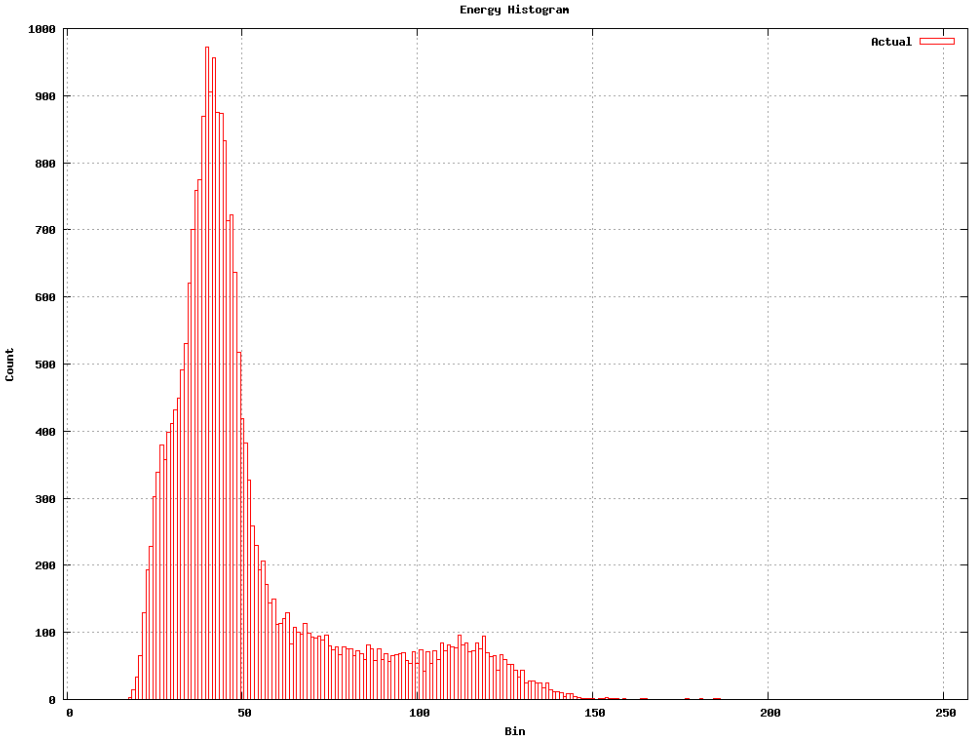

Figure 41: High Voltage Set Too Low

When the HV setting is changed in the linear region, the histogram is expanded horizontally when the voltage is increased and contracted as the voltage is decreased, as shown in Figure 42. The low energy part of the histogram is dominated by the CFD threshold setting and remains virtually unchanged as the HV is changed. Compare the lowest energy bins of the left and right sides of the figures.

Notice that even slightly past the cutoff region (where the histogram counts are always zero), the histogram shows that some bins are not affected by a high voltage change. Slightly past this constant region, bins do begin to be affected by changes in the high voltage. Past this point, the shape of the histogram on the left side is stretched fairly linearly into the shape on the right side.

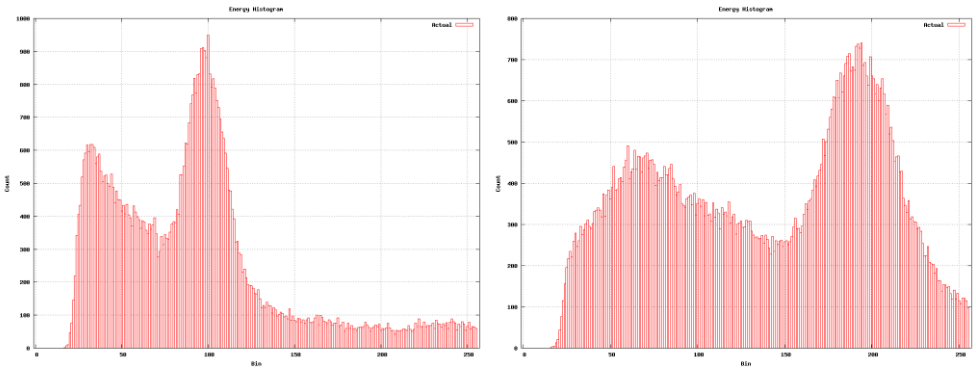

Figure 42: Response to Changes in HV

There are a number of desired values that correspond to the various settings. These are constants defined for the calibration code. For example, the principal investigators have requested that the photopeak of a histogram be adjusted to happen at bin number 155 in FPGA histograms. One way to arrange this would be to set the high voltage supply to a value in between those settings shown in Figure 42.

This is done easily enough. We choose a starting value for the high voltage supply and run the equivalent of rabbit-histo-fit on it to determine where the photopeak occurs. An example is shown in Figure 43. We then increment the high voltage setting by one bit in the DAC that controls the power supply and do a similar fit. Results of such a process are shown in Figure 44. Notice that changing the HV by one bit stretched the histogram slightly. The photopeak changed from bin 140 in Figure 43 to bin 146 in Figure 44. The FWHM of the photopeak changed from 34 in Figure 43 to 35 in Figure 44. We have relatively fine grained control over the position of the photopeak available.

With information in hand regarding the effects of changing the high voltage by one bit, it is straightforward to calculate how many bits worth of change would be required to move the photopeak into the desired energy bin.

It turns out, however, that placement of photopeaks is limited by the physical hardware in other ways.

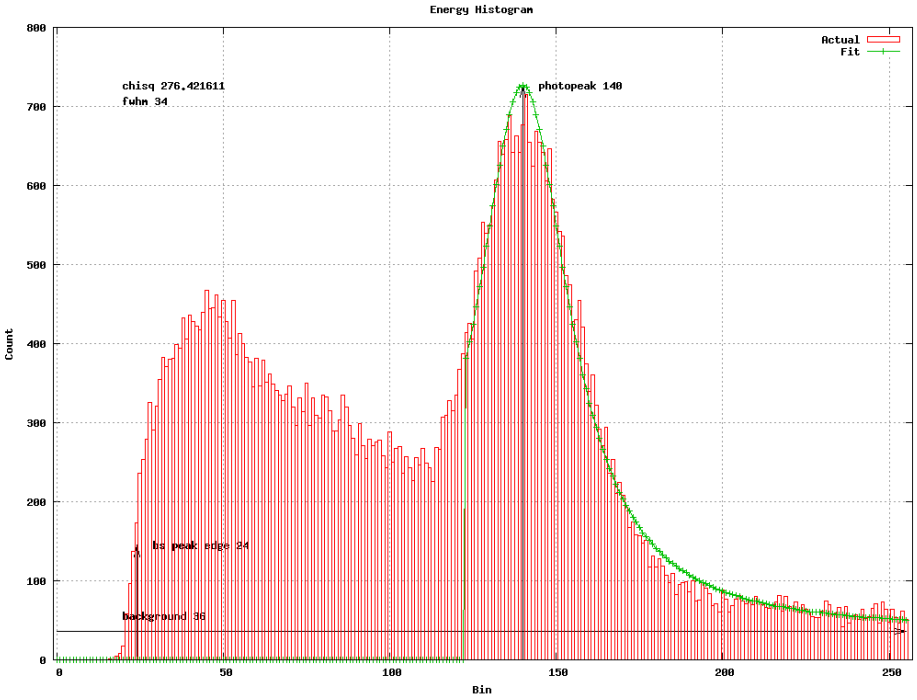

Figure 43: Fit Data from Reference HV Setting

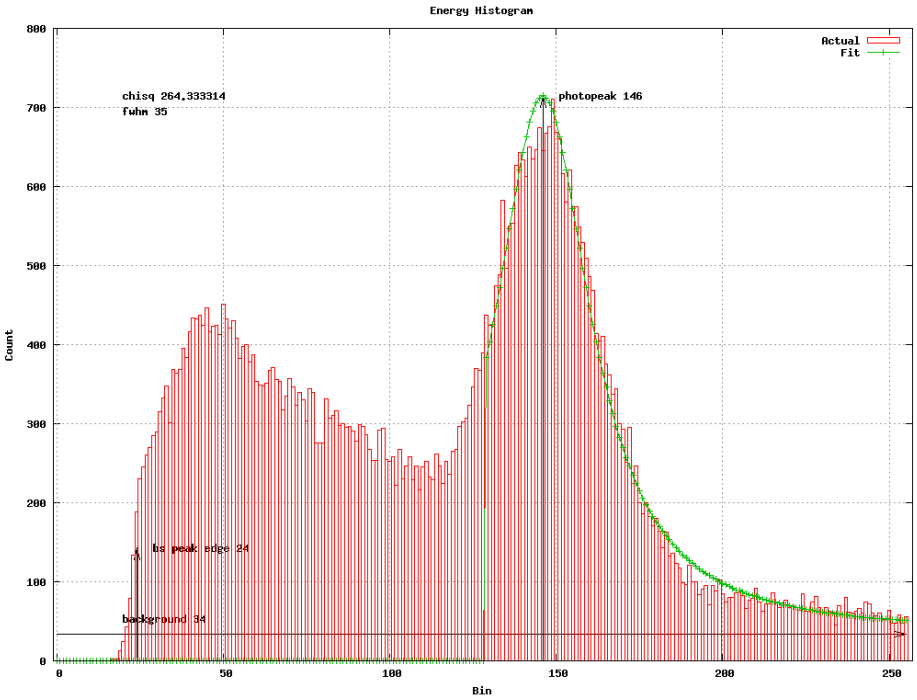

Figure 44: Fit Data from Incremental HV Setting

Figure 45 shows histograms from two quadrants taken on the same PMT in the same histogramming session. Notice that the histogram on the left (quadrant 1) has a photopeak located at bin 140, while the histogram on the right (quadrant 4) has a photopeak located at bin 166. This is presumably due to variations in the anode gain characteristics within the PMT. Figure 11 shows how significant these variations can be. The difference in photopeak energy shown in Figure 45 is about 17%. This is well within the variations shown in Figure 11.

Because of the kinds of gain variations seen within a PSPMT, we can only calibrate the ***average*** photopeak locations found across all four quadrants of the PMT. Users should not be surprised to see rather large excursions in locations of the photopeaks in fully calibrated modules.

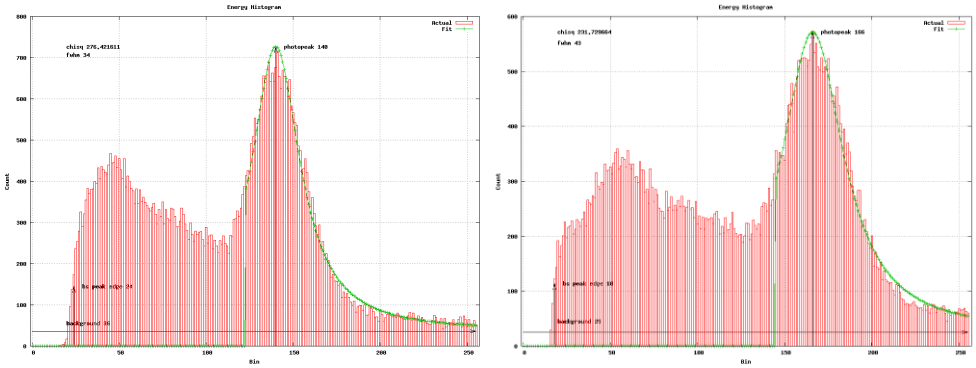

Figure 45: Different Quadrants on the Same PMT

Differences in anode gains can also be seen within a single histogram. This manifests as a histogram with multiple photopeaks. An example of this can be seen in Figure 52 (bottom). One photopeak appears to be at approximately bin 154 and the other is at approximately bin 173. This can be explained by gain variations within the quadrant, with one group of anodes clustered at a gain such that their average photopeak is at bin 154, and another group clustered such that their gains produce an average photopeak at bin 173.

Differences in anode gains can also be seen within a single histogram. This manifests as a histogram with multiple photopeaks. An example of this can be seen in Figure 46. One photopeak appears to be at approximately bin 154 and the other is at approximately bin 173. This can be explained by gain variations within the quadrant, with one group of anodes clustered at a gain such that their average photopeak is at bin 154, and another group clustered such that their gains produce an average photopeak at bin 173.

Figure 47 is an example that shows how gain variations within quadrants can blur the photopeaks and backscatter peaks considerably. One can make out three different photopeaks. Since there are 6 x 6 anodes in a PSPMT, each anode covers roughly a 43 x 43 pixel region; and each quadrant histogram covers nine anodes. It can be readily seen that the energy resolution of an uncompensated system such as this is poor. This is the primary drawback of using a PSPMT-based system.

There are no hardware adjustments available in the modules to tighten this up.

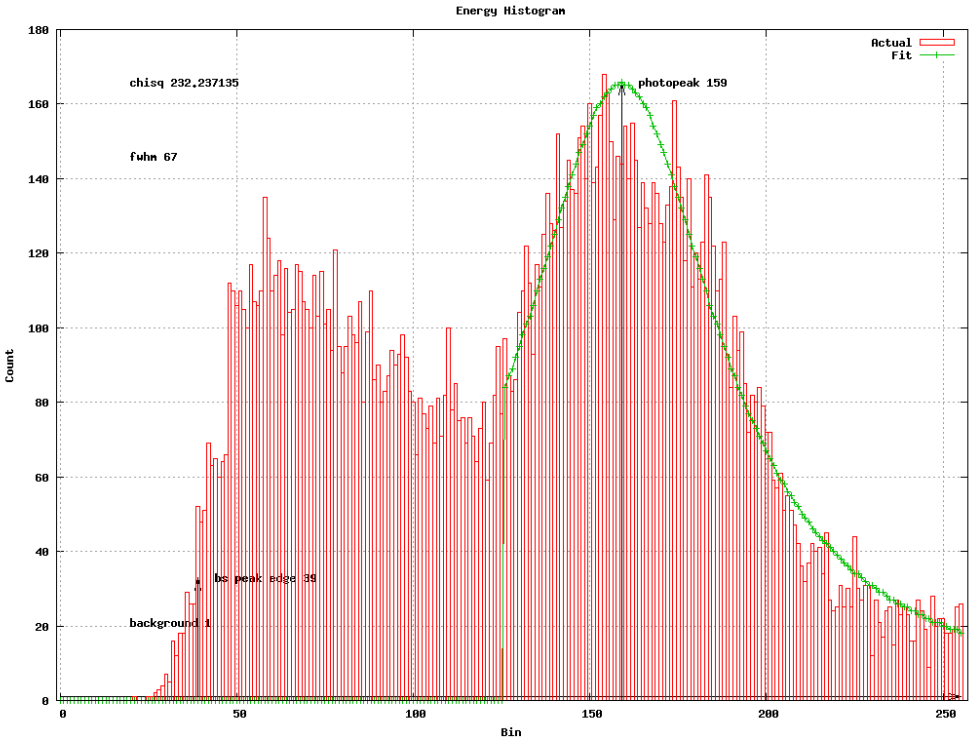

Figure 46: Two Photopeaks in a Quadrant

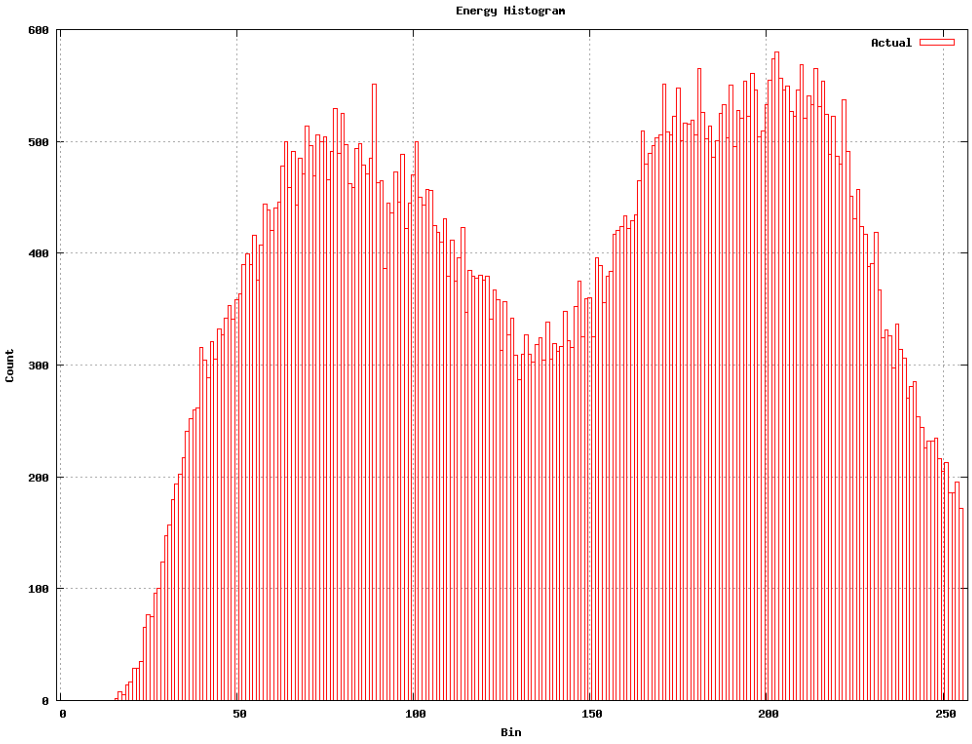

Figure 47: Three Photopeaks in a Quadrant?

Calibration of the high voltage setting is done using PSPMT0/ASIC 0. It is possible, since two PSPMT devices share a comon high voltage supply, that the second PMT will exhibit an average photopeak location away from the desired location. If this is the case, we use the ASIC1 gain setting to position the photopeak of PSPMT1.

It turns out that the gain settings of the ASICs are not as fine grained as the high voltage setting.

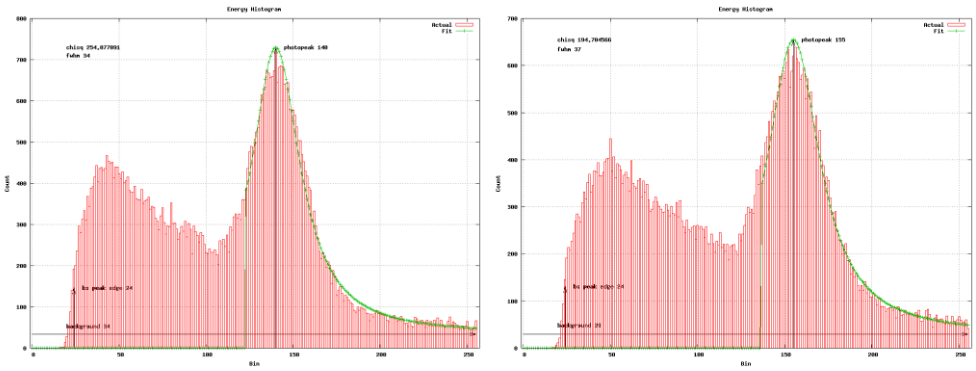

Figure 48: Change of One Bit in Gain

That is, a change of one bit in the ASIC gain results in a movement of the photopeak. A change in the HV setting of one bit produces a 6 bin change in photopeak location, while a change in gain setting of one bit produces a 15 bin change in phototpeak location as shown in Figure 48. Using the same approach as used in the high voltage setting, the effect of a one bit change in gain is assessed and the required change in order to bring the average photopeak bin to the desired bin is calculated.

Using the above process, the common high voltage setting for a module is found, the gain setting for ASIC0 remains at the default level and the gain setting for ASIC1 is calculated.

Calibration Testing

If a module has been previously calibrated and has become suspect for some reason, calibration tests are provided to allow for getting a PASS or FAIL result on the module. These tests acquire histograms for a given PSPMT and run the usual statistical fit functions on them.

Cut values are provided in the program to determine what are good values for:

- The average photopeak bin location'
- The chi-squared statistic for a fit of a Lorentzian peak to the photopeak;
- The Full Width at Half Maximum (FWHM) value of the photopeak;
- The background level;
- The location of the backscatter peak low energy edge.

The calibration tests are automatically performed at the end of a calibration run, but are also provided as stand-alone tests for use in module debugging or in-situ testing.

Calibration Results

Once a module has been calibrated, the results of a calibration are stored to the Rabbit flash memory. At this time, the results are not automatically restored when a module is brought up (the automatic calibration is not yet fully integrated into the mainline code). An option is provided to restore saved configuration information.

```
Digital board version: 25 July 09 Rev D (modified by CMD)

ID=06
Master

FPGA data -> RAM in progress.
Init ASICs
ACQ debug is off
Write test data via ext. IO
Lun 0 logged out
Lun 1 logged out
1) read 4 TI bytes       a) read Rabbit port     i) write cmd reg
2) write 4 TI bytes      b) toggle acq           j) write DAC
3) read IO device        c) toggle DTF mode      k) READ MODULE ID
4) write IO device       d) set ASIC defaults    l) WRITE MODULE ID
5) initilize rabbit      e) read ASIC reg        m) start historgram 0
6) run autotune          f) write ASIC reg       n) start histogram 1
7) load TI ROM A         g) read RAM             o) stop histogram 0
8) Start run - E exit    h) write RAM            p) stop histogram 1
9) write Rabbit port     q) read histo           r) load FPGA bin file
s) start FireWire        t) stop FireWire        u) calibrate
v) write default cals    w) restore saved cals   x) write current cals
y) print saved cals      z) calibrate HV         -) calibrate gain
=) calibrate CFD 0       _) calibrate CFD 1      +) calibrate TDC 0
.) calibrate TDC 1
enter option (1..k):  w
Restoring saved configuration
  Restoring configuration to ASIC 0
  Restoring configuration to ASIC 1
Bringing up HV to configured value
  Set initial HV = 0xa0
  Set configured HV = 0xd0
Configuration restored.
```

As seen above, option ‘w’ will load the saved calibration configuration from the Rabbit flash memory.

An option is also available to examine the saved calibration configuration using option ‘y’ in the main menu

```
enter option (1..k):  y
Current saved configuration
  Reading configuration from flash
  ASIC 0 configuration settings:
    magic = CAL_CONFIGURED_MAGIC
    channel_a_gain = 0x8c
    channel_b_gain = 0x8c
    channel_c_gain = 0x8c
    channel_d_gain = 0x8c
    cfd_threshold = 0xca
    cfd_control = 0x20
    tdc_gain = 0x85
    tdc_offset = 0x76
    tdc_control = 0x17
    ab_test_gain = 0x0
    cd_test_gain = 0x0
  ASIC 1 configuration settings:
    magic = CAL_CONFIGURED_MAGIC
    channel_a_gain = 0x8c
    channel_b_gain = 0x8c
    channel_c_gain = 0x8c
    channel_d_gain = 0x8c
    cfd_threshold = 0xca
    cfd_control = 0x20
    tdc_gain = 0x85
    tdc_offset = 0x76
    tdc_control = 0x17
    ab_test_gain = 0x0
    cd_test_gain = 0x0
  HV configuration setting:
    hv_dac = 0xda
End configuration.
```

It is also possible to manually tweak the results of a calibration process. If a user decides that a calibration value needs to be manually changed, he or she can do so using the "`write ASIC reg`" command, "f", or the "`write DAC`" command, "j". The calibration code will track the changes made. As soon as the changes are complete, one can select the "x" option – "`write current cals`" to save the new values to the Rabbit flash memory.

```
enter option (1..k):  x
Write current configuration to flash
  Updating current configuration of ASIC 0
  Updating current configuration of ASIC 1
  Writing to flash
Current configuration written to flash
```

Since the automatic calibration is not infallible in the presence of unusual module states, it is also possible to run the calibrations piecewise. For example, if a particular configuration step fails, it is possible to run all of the other steps automatically. Then the user can manually configure the problem step and save the results. The calibration steps build on each other in general, so the following order should be preserved

1. calibrate HV – option;
2. calibrate gain – option;
3. calibrate CFD 0 or calibrate CFD 1;
4. calibrate TDC 0 or calibrate TDC 1.

For example, if the automatic calibration of the constant fraction discriminator on PMT 0 is producing unusual results, a combination of automatic and manual calibration may be used. The user should first automatically calibrate the high voltage using option 'z', then automatically calibrate the gain using option option '-' (a minus sign). The user may then manually configure CFD 0 using the "read ASIC reg" and "write ASIC reg" commands ('e' and 'f'). The user can then return to automatic calibration by using the "calibrate CFD 1" option, followed by "calibrate TDC 0" and calibrate "TDC 1".

Example Calibration Run and Discussion

The test station has two power supplies on the top shelf of the bench (shown in Figure 49), both of which must be turned on. (Turn them on left to right, and off right to left)

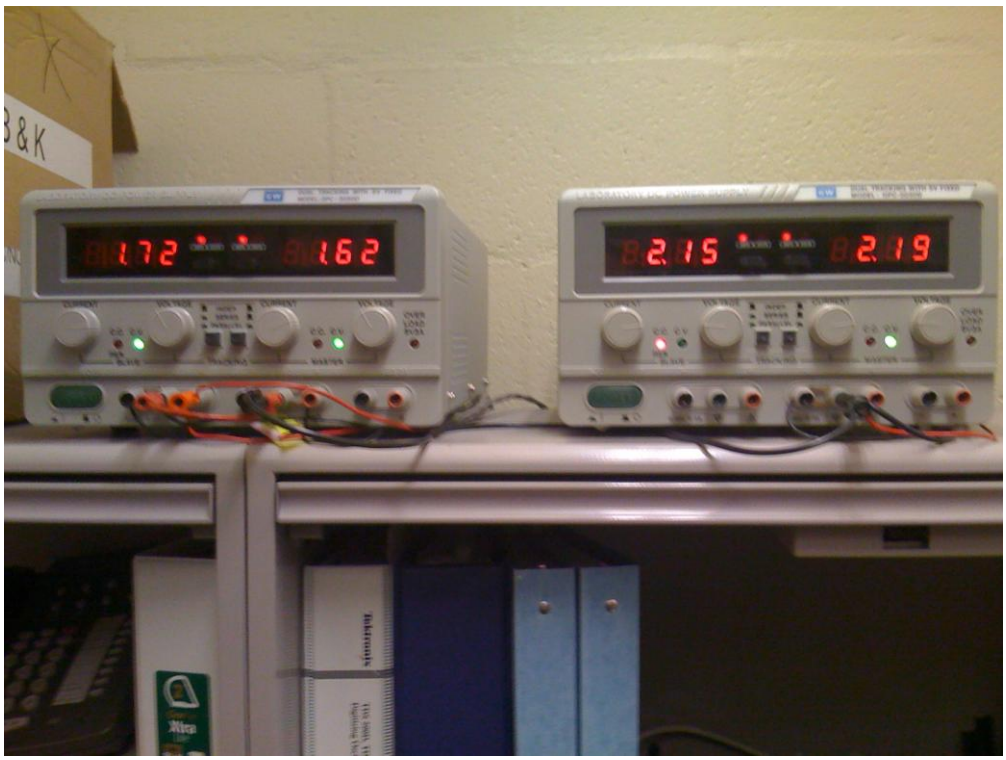

Figure 49: Test Station Power Suppleis

The sources should be arranged as shown in Figure 50. Typically we use from four to eight Na-22, 20 μCi sources distributed evenly between the PSPMTs.

If you use HyperTerminal console to look at digital board output you will quickly discover that scrolling doesn't work very well. For this reason, I provided an alternate console called uCon which can be found in the quick start section of the Windows taskbar. I prefer to use uCon over HyperTerminal in almost all cases since it allows one to scroll backwards and read the verbose output of the configuration code.

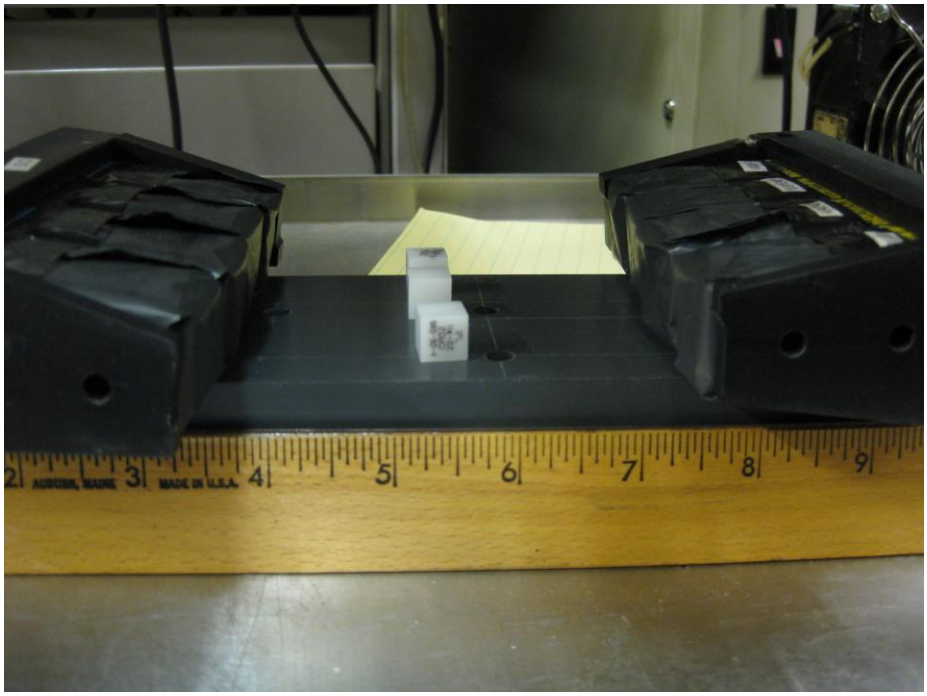

Figure 50: Arrangement of Sources

The following section is a walk-through of the output of a sample calibration run, which took roughly eleven minutes to complete. The output of the calibration code is shown in a `fixed-point` font, and a discussion of the output will follow.

The output of the calibration function is extremely verbose. This is intentional for two reasons: First, the more information that is displayed while running the calibration, the better chance that something will pop out that will turn out to be useful to a technician looking for an underlying problem; and second, the calibration functions sometimes need to "go away" and accumulate hits or do lengthy calculations. If these times turn our to be longer than a few seconds, a user will have no way of knowing if things are progressing normally or if something "bad" has happened to cause a program crash. During calculations, the calibration code will print out many intermediate results as a kind of keepalive message, and while accumulating hits, the code will print a spinning clock that ticks away the seconds.

```
Digital board version: 25 July 09 Rev D (modified by CMD)

ID=06
Master

FPGA data -> RAM in progress.
Init ASICs
ACQ debug is off
Write test data via ext. IO
Lun 0 logged out
Lun 1 logged out
1) read 4 TI bytes       a) read Rabbit port     i) write cmd reg
2) write 4 TI bytes      b) toggle acq           j) write DAC
3) read IO device        c) toggle DTF mode      k) READ MODULE ID
4) write IO device       d) set ASIC defaults    l) WRITE MODULE ID
5) initilize rabbit      e) read ASIC reg        m) start historgram 0
6) run autotune          f) write ASIC reg       n) start histogram 1
7) load TI ROM A         g) read RAM             o) stop histogram 0
8) Start run - E exit    h) write RAM            p) stop histogram 1
9) write Rabbit port     q) read histo           r) load FPGA bin file
s) start FireWire        t) stop FireWire        u) calibrate
v) write default cals    w) restore saved cals   x) write current cals
y) print saved cals      z) calibrate HV         -) calibrate gain
=) calibrate CFD 0       _) calibrate CFD 1      +) calibrate TDC 0
.) calibrate TDC 1
enter option (1..k):  u
```

When the digital board is first brought up, it identifies the version that is running. It then prints the module ID followed by an indication that you are looking at the Master or Slave side of the digital board.

The board is initialized – a process that can take a couple of minutes to complete – and then the main menu is displayed.

The output of the calibration messages are indented to help keep the messages organized. For example, in the following listing the Reset ASIC function is composed of four parts: setting ASIC gains, setting ASIC CFD thresholds, setting TDC gains and setting TDC offsets. These are indented to indicate they are part of the Reset ASIC function. Inside the code to reset the ASIC gains, messages are further indented to group them. The same convention is used throughout the code.

```
Calibrate Module
  Reset ASIC
    Set ASIC gains to default values
      ASIC 0 channel 0 = 0x8c
      ASIC 0 channel 1 = 0x8c
      ASIC 0 channel 2 = 0x8c
      ASIC 0 channel 3 = 0x8c
      ASIC 1 channel 0 = 0x8c
      ASIC 1 channel 1 = 0x8c
      ASIC 1 channel 2 = 0x8c
      ASIC 1 channel 3 = 0x8c
    Set ASIC CFD thresholds to default values
      ASIC 0 CFD = 0x68
      ASIC 1 CFD = 0x68
    Set ASIC CFD modes to default values
      ASIC 0 CFD mode = 0x20
      ASIC 1 CFD mode = 0x20
    Set ASIC TDC gains to default values
      ASIC 0 TDC gain = 0x80
      ASIC 1 TDC gain = 0x80
    Set ASIC TDC offsets to default values
      ASIC 0 TDC offset = 0x80
      ASIC 1 TDC offset = 0x80
    Set ASIC TDC config registers to default values
      ASIC 0 TDC config = 0x17
      ASIC 1 TDC config = 0x17
    Set ASIC AB Test gain registers to default values
      ASIC 0 AB test gain = 0x0
      ASIC 1 AB test gain = 0x0
    Set ASIC CD Test gain registers to default values
      ASIC 0 CD test gain = 0x0
      ASIC 1 CD test gain = 0x0
  ASIC reset
  Reset HV
    Set initial HV = 0xa0
    Set default HV = 0xd0
  HV reset
```

The first thing to be done in the calibration is to set the default, or starting values, for all of the configuration items. This also serves to point out which settings are going to be manipulated during the configuration session. There are eight ASIC gain settings to discover, two ASIC constant fraction discriminator settings, two time to digital converter gain settings and two time to digital converter offset settings to be configured in the ASICs. We also have one high voltage setting which is shared between the two PMTs on the module. There are also control mode register configuration items which are initialized here, but not changed in the calibration process.

The next thing that happens is that the high voltage is turned on. It is generally a good idea with PMT devices to bring up the high voltage gradually. This is done here. First the high voltage DAC is set to 0xa0. The program actually waits for one second

after the high voltage is turned on before proceeding to set the HV up to the initial value of 0xd0.

```
Calibrate HV
  Starting histograms of PMT 0
  Stopping histograms of PMT 0
```

The next step is to begin calibrating the high voltage setting and the first thing to be done in that process is to gather histogram data. This process will last thirty seconds, during which time a "ticking clock" will be simulated by displaying | / – and \ characters. The histograming process involves communication with the FPGA so the fact that you do or do not see certain messages can help isolate problems relating to communication between modules on the digital board. After the histograms have been gathered for thirty seconds, a message indicating that a stop command is being sent to the FPGA is output

```
  Loading histos
    Quadrant 0
    Quadrant 1
    Quadrant 2
    Quadrant 3
```

Here, the process of downloading the histograms from the FPGA to the Rabbit memory happens. It turns out that an unprintable character is output for every histogram bin by the download routine which is not showed here.

```
  Fitting histos
    Fitting quadrant 0
      Iteration 0
      Iteration 1
      Iteration 2
    Fitting quadrant 1
      Iteration 0
      Iteration 1
      Iteration 2
    Fitting quadrant 2
      Iteration 0
      Iteration 1
      Iteration 2
    Fitting quadrant 3
      Iteration 0
      Iteration 1
      Iteration 2
```

The next step in the process is to do the statistical fitting on the histograms. These fits start with a set of default values and iteratively refine the fits. To make a "perfect" fit sometimes many iterations would be required, but we limit this to three iterations here. This results in a much better fit that the possible accuracy of the resulting calibration, primarily due to variations between histogram quadrants within a given PMT.

```
Quadrant 0
  Photopeak maximum = 215
  Photopeak center = 113
  Photopeak center deviation = 19
  Photopeak FWHM = 29
  Background = 15
Quadrant 1
  Photopeak maximum = 156
  Photopeak center = 85
  Photopeak center deviation = -9
  Photopeak FWHM = 29
  Background = 9
Quadrant 2
  Photopeak maximum = 265
  Photopeak center = 93
  Photopeak center deviation = -1
  Photopeak FWHM = 26
  Background = 12
Quadrant 3
  Photopeak maximum = 178
  Photopeak center = 87
  Photopeak center deviation = -7
  Photopeak FWHM = 35
  Background = 9
Average photopeak center for PMT = 94
Variance of photopeak center for PMT = 123
```

This next section prints out details regarding the fit that was calculated for each of the four quadrants of the PMT.

The "Photopeak maximum" is the calculated maximum of a Lorentzian shape fit to the photopeak of the underlying histogram. This may or may not be the bin in the histogram with the highest count. See Figure 24 for an example of a histogram where the calculated fit does not correspond with a histogram maximum.

The "Photopeak center" is the histogram bin that was calculated to be the center of the Lorentzian shape to which the photopeak was fit.

The "Photopeak center deviation" is the number of bins that this photopeak varies away from the average photopeak location of all four quadrants.

The "Photopeak FWHM" is the Full Width at Half Maximum of the Lorentzian shape. If the maximum of the photopeak is at the count 180, then this is the width of the shape as measured at count 90.

The "Background" is the noise level found to the right of the Lorentzian shape. The way to think of this is: the Lorentzian shape normally quickly drops off to zero. Plots of the histograms show that the real shape drops off to some non-zero background level. This count is that background.

After all of the quadrants are listed, a two line summary is printed. The first is the average photopeak center of all of the quadrants. The second is the variance, or mean squared value of the individual deviations. If you take the square root of the variance,

you will find the standard deviation. This is not done for you since the Rabbit has no math processor and we don't want to have to calculate things like this using its native Z80 instructions.

```
  Set incremental HV = 0xd4
    Starting histograms of PMT 0
```

As mentioned above, we need to figure out where to set the high voltage supply to put the average photopeaks at the desired point in the histogram. We do that by making a small change to the high voltage supply and looking to see what happens. You will see a repeat of the above listings, except with slightly different statistics.

```
    HV change of 4 bits produces change of 17 in photopeak bin
    Implies change to default HV of 14 is required
    Setting HV to optimal setting for PMT 0 = 0xde
  HV Calibration complete
```

After the histogram taken at the incremental high voltage is fit, the program notes that a change of one bit to the high voltage DAC results in a certain change in the average photopeaks. It then calculates what change in high voltage would be required to move the photopeak to the desired location and sets the high voltage to that setting. In the example above, it figured out that for every bit increase in the DAC setting, the photopeak moves five bins. It decided that a change of 12 to the DAC with respect to the default setting would be "just right" and made that change.

```
  Calibrate ASIC gain
    Starting histograms of PMT 1
    Stopping histograms of PMT 1
    Loading histos
      Quadrant 0
      Quadrant 1
      Quadrant 2
      Quadrant 3
    Fitting histos
      Fitting quadrant 0
        Iteration 0
        Iteration 1
        Iteration 2
      Fitting quadrant 1
        Iteration 0
        Iteration 1
        Iteration 2
      Fitting quadrant 2
        Iteration 0
        Iteration 1
        Iteration 2
      Fitting quadrant 3
        Iteration 0
        Iteration 1
        Iteration 2
```

The next step is to calibrate the photopeaks of the other PMT. Since the high voltage setting is shared between the two PMTs, the only way to do this is by varying the ASIC gain on PMT 1. The high voltage is left set to the calibrated value found above, and the ASIC gains are all left at the default value. A histogram is taken of PMT1 and the same fit process as described above is performed. A summary of the fit values is output just as in the high voltage calibration section of the test.

```
    Set incremental ASIC gain = 0x8e
    Set ASIC gain
      ASIC 1 channel 0 = 0x8d
      ASIC 1 channel 1 = 0x8d
      ASIC 1 channel 2 = 0x8d
      ASIC 1 channel 3 = 0x8d
    Starting histograms of PMT 1
    Stopping histograms of PMT 1
```

Just as the high voltage setting was changed and a new histogram taken above, here the ASIC gain setting is changed across all of the channels and a new histogram is taken. The same fit is performed and statistics are output just as has been done previously.

```
    Gain change of 2 bits produces change of 33 in photopeak bin
    Implies change to default ASIC gain of 0 is required
    Setting ASIC gain to optimal setting for PMT 1 = 0x8c
    Set ASIC gain
      ASIC 1 channel 0 = 0x8c
      ASIC 1 channel 1 = 0x8c
      ASIC 1 channel 2 = 0x8c
      ASIC 1 channel 3 = 0x8c
  ASIC gain Calibration complete
```

Just as in the high voltage calibration, the code discovers how many histogram bins the phtotoeak changes for a small change in the gain. In this case, it finds that the peak is moved 16 bins in response to a one-bit change to the gain registers. Here, the program decides that no change is required to move the photopeaks of PMT 1 into position – they are already good enough.

```
  Calibrate CFD for ASIC 0
    Set CFD to max = 0xff
    Starting histograms of PMT 0
    Stopping histograms of PMT 0
    Loading histos
      Quadrant 0
      Quadrant 1
      Quadrant 2
      Quadrant 3
```

The next thing to be done is to calibrate the Constant Fraction Discriminator threshold of PMT 0. We want to set this value high enough to cut out any unwanted noise, but low enough to avoid cutting out any wanted events.

In a process that should be familiar by now, we will set the CFD threshold to a particular value and then observe what happens when we make a change. We use information gathered to decide how to position the result where we want it. In this case, though, we turn the CFD threshold all the way up to make the first measurement. This places the low energy edge of the histogram backscatter peak about a third of the way into the histogram. We then gather a histogram in this configuration.

The program calculates an edge position corresponding to the low energy side of the backscatter peak and remembers this value.

```
    Set CFD to 0xf7
    Starting histograms of PMT 0
    Stopping histograms of PMT 0
    Loading histos
      Quadrant 0
      Quadrant 1
      Quadrant 2
      Quadrant 3
```

We then make a change downward in the value of the CFD threshold. It takes a relatively large change in threshold to make a change in the histogram so we drop the threshold by 8 here.

```
    Reduction of CFD threshold of 0x8 produces change of -4 in bin
    Implies CFD threshold change of 98 required
    Setting CFD threshold to optimal setting for PMT 0 = 0x9d
  CFD calibration complete for ASIC
```

After the before and after numbers are gathered, the program displays what it has found. In this case, if the CFD is reduced by 8, the low energy edge of the histogram is moved by -4 bins. The calibration code then calculates what it thinks is required to move the low energy edge down to a reasonable place. In this case, the ASIC 0 CFD setting is changed to 0x9d.

```
  Calibrate CFD for ASIC 1
  CFD calibration complete for ASIC
```

The CFD calibration process is repeated for ASIC 1, which corresponds to PMT1.

```
  Calibrate TDC for ASIC 0
    Set TDC gain to default = 0x80
    Set TDC offset to default = 0x80
    Binary search for TDC offset.  Low = 64, High = 192
      Set TDC offset = 0x80
      Starting histograms of PMT 0
      Stopping histograms of PMT 0
      Loading TDC histo
      Found average = 189, tolerance = 54, count = 1090
```

The next calibration items are the TDC gain and TDC offset for PMT0. The goal of this test is to gather a TDC histogram (see Figure 26 for an example) and to vary the gain and offset so as to bring the lowest and highest energy bins into agreement with the average. We first choose the value that is exactly between 64 and 128, which is 128

(80 in hexadecimal). We calculate an average value of the central bins of the histogram which are not affected by changes in the TDC offset value and determine a tolerance for what is a "good" result. The count found in the lowest bin is 1090 here, and so we discover that the TDC offset value is too high.

```
    Binary search for TDC offset.  Low = 64, High = 128
      Set TDC offset = 0x60
      Starting histograms of PMT 0
      Stopping histograms of PMT 0
      Loading TDC histo
      Found average = 202, tolerance = 56, count = 0
```

Since the TDC offset value was too high, we need to binary search for the correct value in the lower half of the previous range (64 – 192). We know that if we set the TDC value to 128 (0x80) the count is too high, so we want to search between the loest value in the range and 128. The midway point between these two values is 96, or 60 hexadecimal. With a TDC offset register setting of 0x60, we discover that the lowest bin has a count of 0, which is too low. We need to increase it.

This process of narrowing down the possibilities continues until we either find the lowest bin in the tolerance range, or until we run out of bits. It is often the case that there is insufficient resolution to form a proper distribution. That is, the lowest bin is still not close enough. In this case, we just leave the setting at the best possible value and continue.

```
    Binary search for TDC gain.  Low = 138, High = 140
      Set TDC gain = 0x8b
      Starting histograms of PMT 0
      Stopping histograms of PMT 0
      Loading TDC histo
      Found average = 187, tolerance = 54, count = 260
    TDC gain found = 0x8b
    Binary search converged to 139 but did not pass
    Resolution of register insufficient ...
```

In the example above, the binary search has continued until the test range has been reduced to betweeen 138 and 140. The only possibility left is 139 (or 8b in hexadecimal). The average counts in the center of the histogram average to 187, and the calibration code is looking for a value between 133 and 241. The result was a count of 260. This is not within the tolerances, but it is as good as the register precision will allow; so the message "TDC gain found" is shown with an explanation.

```
  Calibrate TDC for ASIC 1
    Set TDC gain to default = 0x80
    Set TDC offset to default = 0x80
    Binary search for TDC offset.  Low = 64, High = 192
    ...
```

The same TDC calibration is done for the TDC gain and offset for ASIC1/PMT1.

```
  Write configuration to flash
    Updating configuration of ASIC 0
    Updating configuration of ASIC 1
    Writing to flash
  Configuration written to flash
Calibration complete
```

Finally, the program writes the found calibration values to the Rabbit flash memory.

```
Calibration complete
 ACQ debug is off
 Write test data via ext. IO
Lun 0 logged out
Lun 1 logged out
1) read 4 TI bytes       a) read Rabbit port     i) write cmd reg
2) write 4 TI bytes      b) toggle acq           j) write DAC
3) read IO device        c) toggle DTF mode      k) READ MODULE ID
4) write IO device       d) set ASIC defaults    l) WRITE MODULE ID
5) initilize rabbit      e) read ASIC reg        m) start historgram 0
6) run autotune          f) write ASIC reg       n) start histogram 1
7) load TI ROM A         g) read RAM             o) stop histogram 0
8) Start run - E exit    h) write RAM            p) stop histogram 1
9) write Rabbit port     q) read histo           r) load FPGA bin file
s) start FireWire        t) stop FireWire        u) calibrate
v) write default cals    w) restore saved cals   x) write current cals
y) print saved cals      z) calibrate HV         -) calibrate gain
=) calibrate CFD 0       _) calibrate CFD 1      +) calibrate TDC 0
.) calibrate TDC 1
enter option (1..k):
```

The calibration is now done and the menu is displayed. If you now take histograms and run then through `rabbit-histo-fit` and `rabbit-tdc-plot` you will see the results of the calibration run. Figure 51 through Figure 55 are a series of before and after histograms illustrating what has happened as a result of all of this work to the response of one PSPMT in the system.

From these figures, you should notice that there are significant variations between quadrants within a PMT / ASIC combination as mentioned earlier. For example, Figure 51 (bottom, quadrant 0) has a photopeak at bin 152, while Figure 54 (bottom, quadrant 3) has the photopeak at bin 184. This is different by a factor of 0.17, which may seem high, but should not entirely surprising given the gain variation characteristics of PSPMT anodes shown in Figure 11.

Figure 52 (bottom) is also an interesting example. It appears that there are actually two photopeaks. One is at approximately bin 154 and the other at approximately bin 173. This can also be explained by gain variations within the quadrant, with one group of anodes clustered at a gain such that their average photopeak is at bin 154, and another group clustered such that their gains produce an average photopeak at bin 173. There are no adjustments available in the modules to tighten this up.

The gain characteristics of each anode could conceivably be addressed by doing an end-to-end scan (using the Macintosh system to gather data over the FireWire) and collecting map files, which contain one histogram for each pixel in the PSPMT crystal

map. It would be straightforward, though time-consuming, to calculate a calibration factor to be applied to the bin numbers of each pixel to compensate for variations in gain across PMTs.

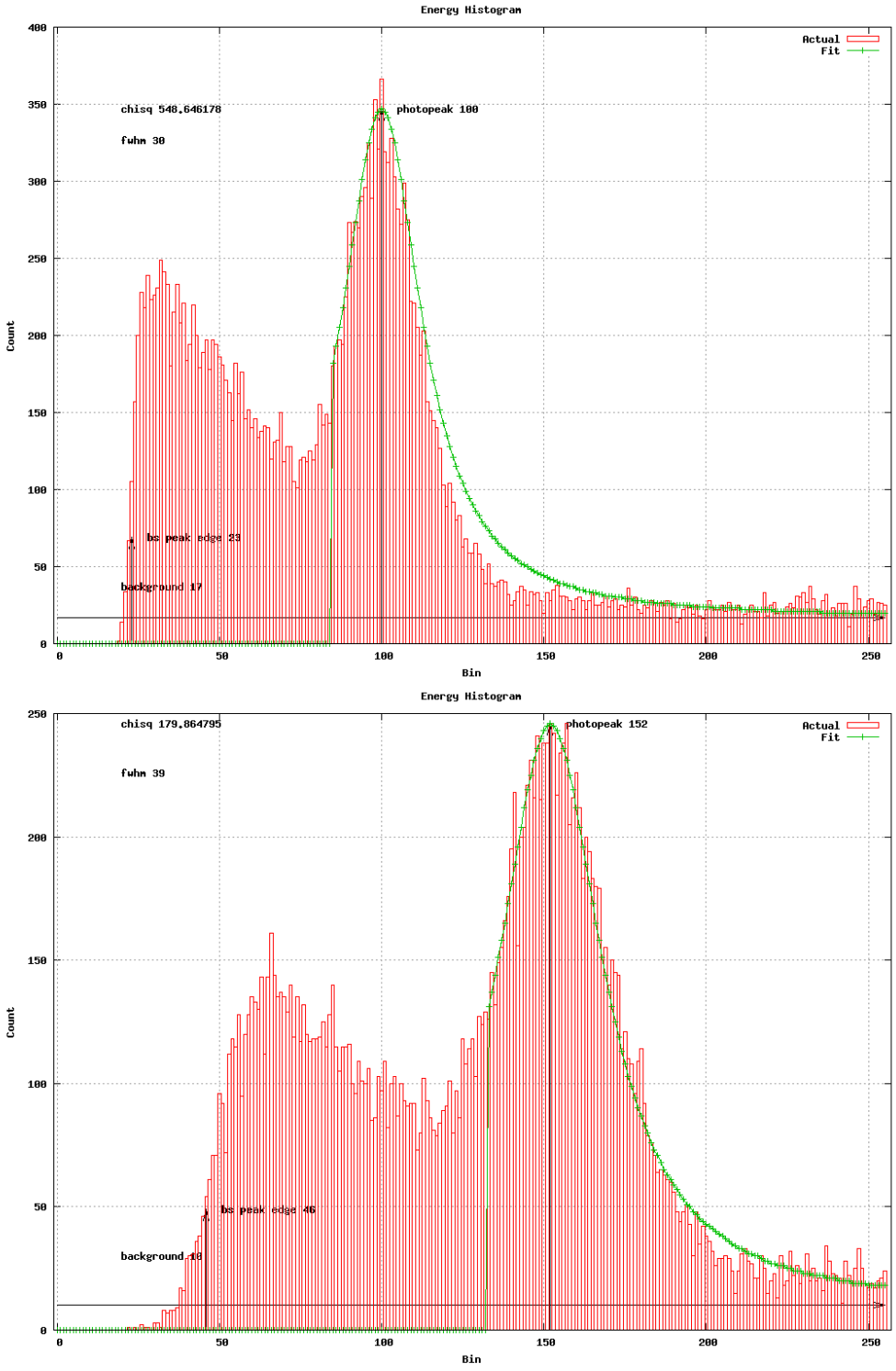

Figure 51: Quadrant 0 Before (top) and After (bottom)

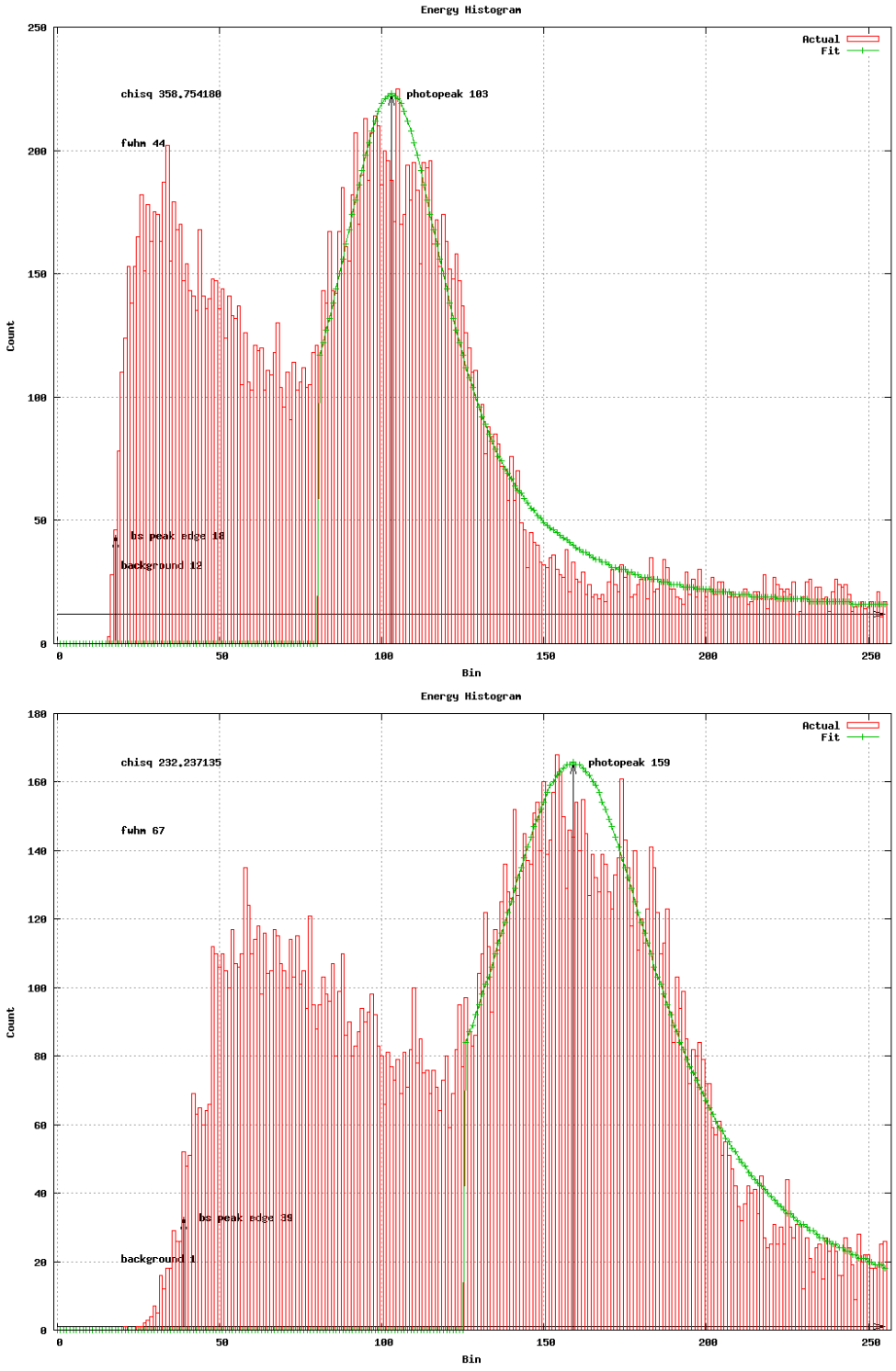

Figure 52: Quadrant 1 Before (top) and After (bottom)

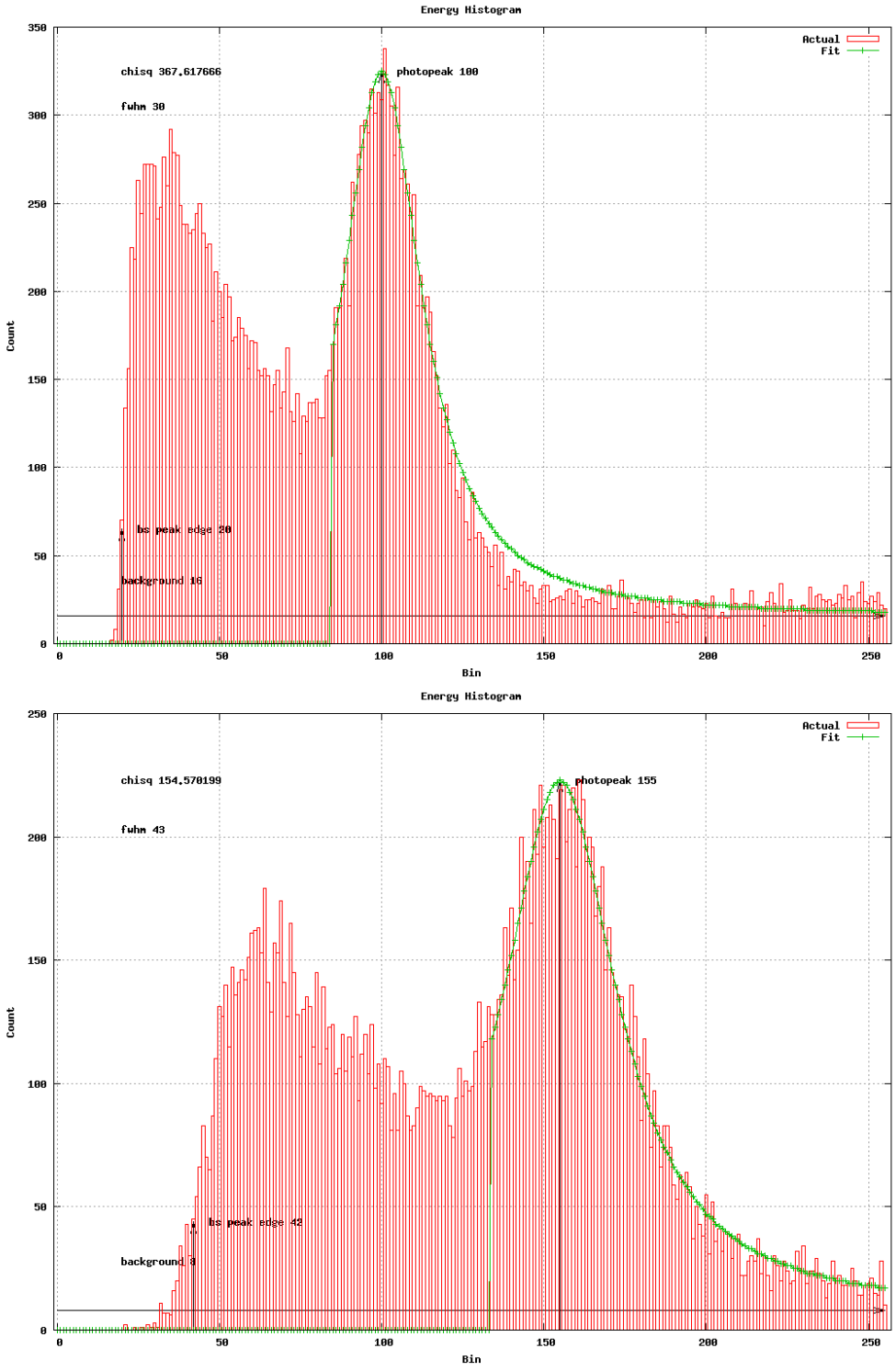

Figure 53: Quadrant 2 Before (top) and After (bottom)

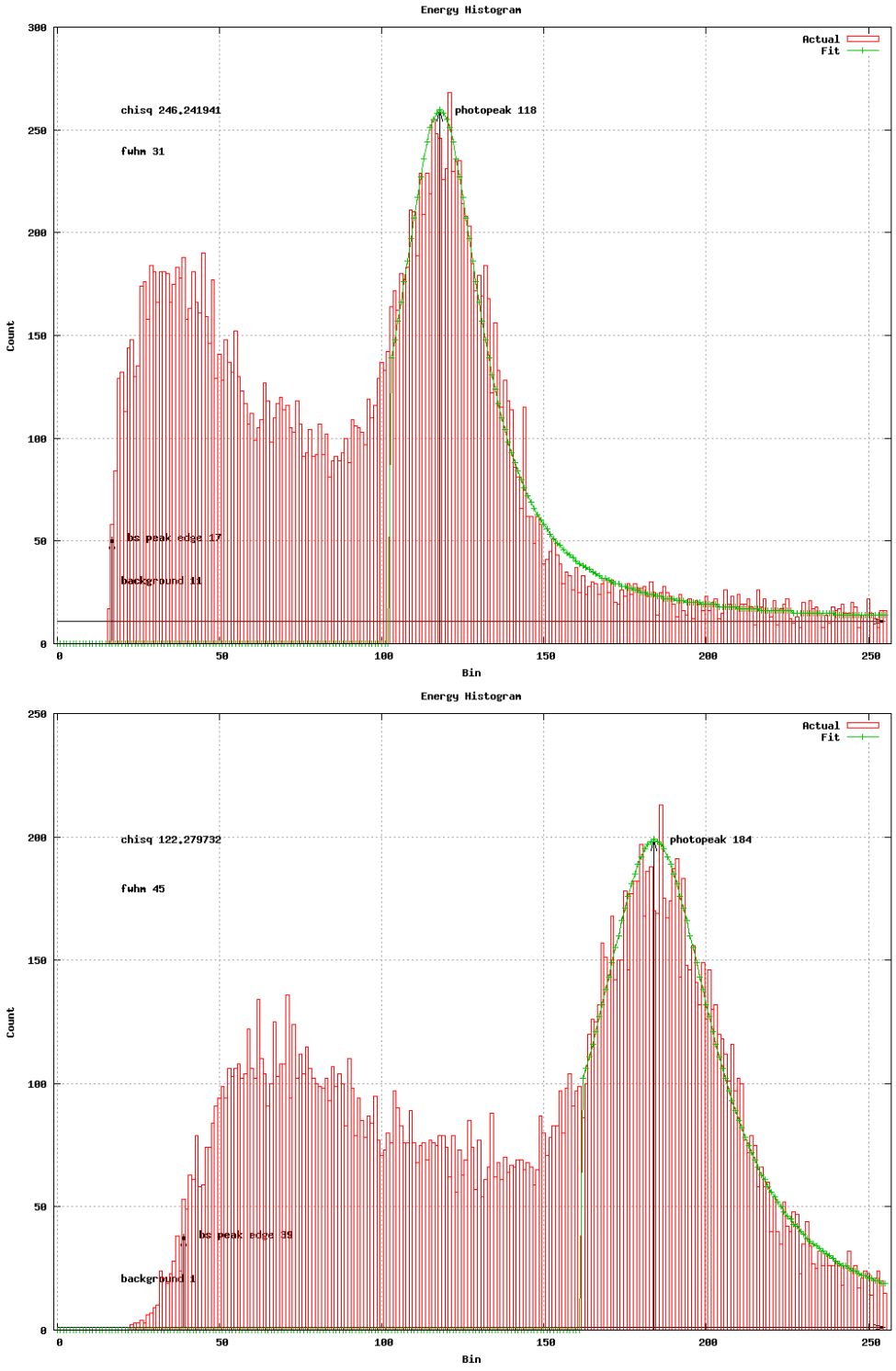

Figure 54: Quadrant 3 Before (top) and After (bottom)

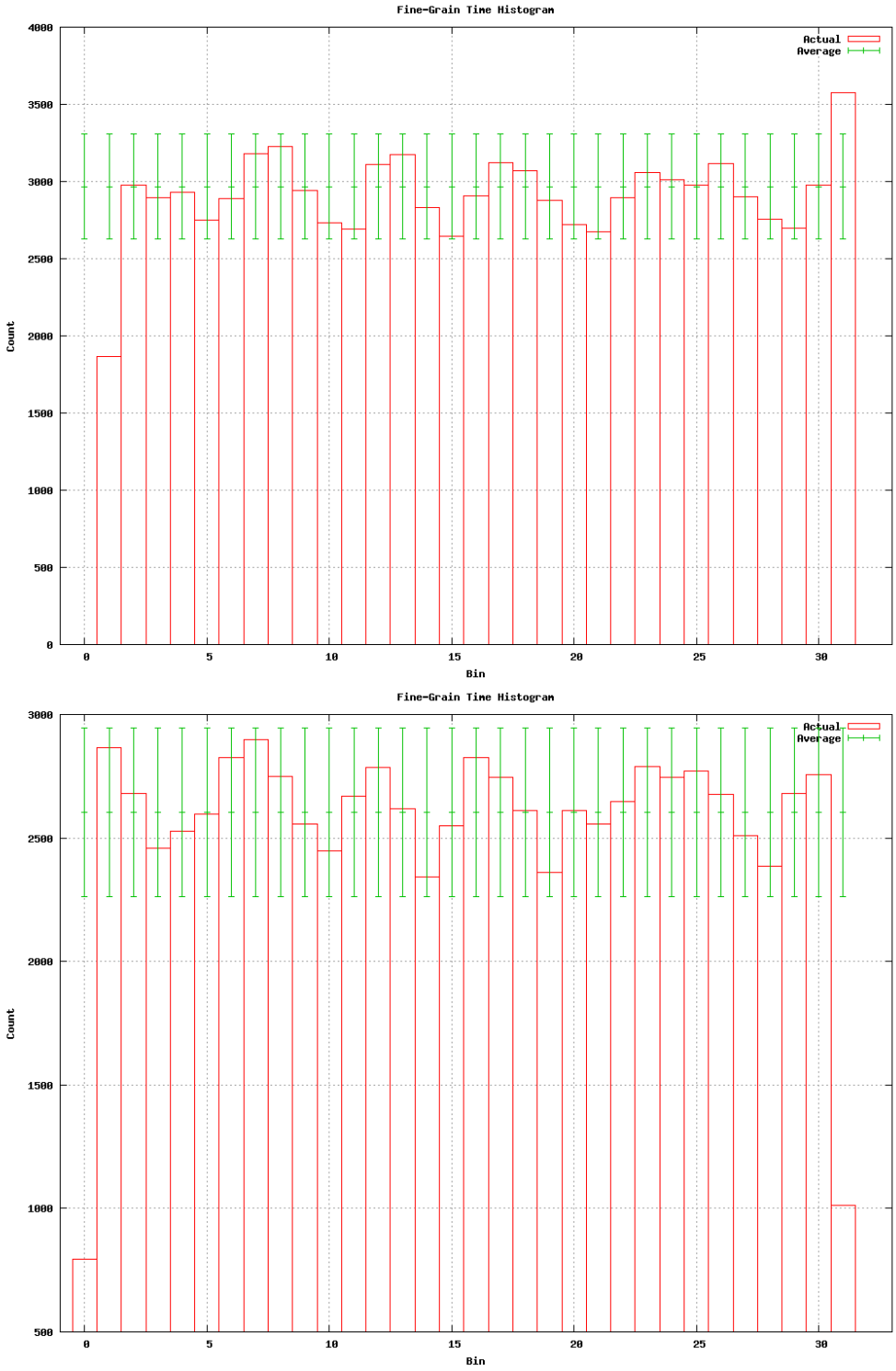

Figure 55: TDC Before (top) and After (bottom)

The final figures are from orthogonal displays. These are another visualization tool used to look at of map files. In some respects they are quite similar to the `crystal-map` tool described in the Data Visualization Tools section. Generating these images is

an involved process and won't be covered here. These displays do, however, put a lot of information in one place and allow us to see the results of our work quite clearly.

There are three graphics shown in an orthogonal display. The image of the "Transverse Slice" at the top is essentially the same kind of display as that of the `crystal-map` tool's contour. That is, you are conceptually looking at a top view of a PSPMT. The spots correspond to scintillator crystal locations. The grid is irregular due to non-linearity in the PSPMT position determination.

The crystal-map tool displays counts of hits in each pixel. In contrast, the orthogonal display tool shows energy ranges. Imagine that a histogram is constructed, but instead of plotting the count of hits versus bin, one plotted a color versus bin. Very low counts such as the background would be dark, and very high counts such as the photopeak would be light. Now imagine a 200 x 200 array of these color-encoded histograms stood on end in an array. That is the Transferse Slice display you see.

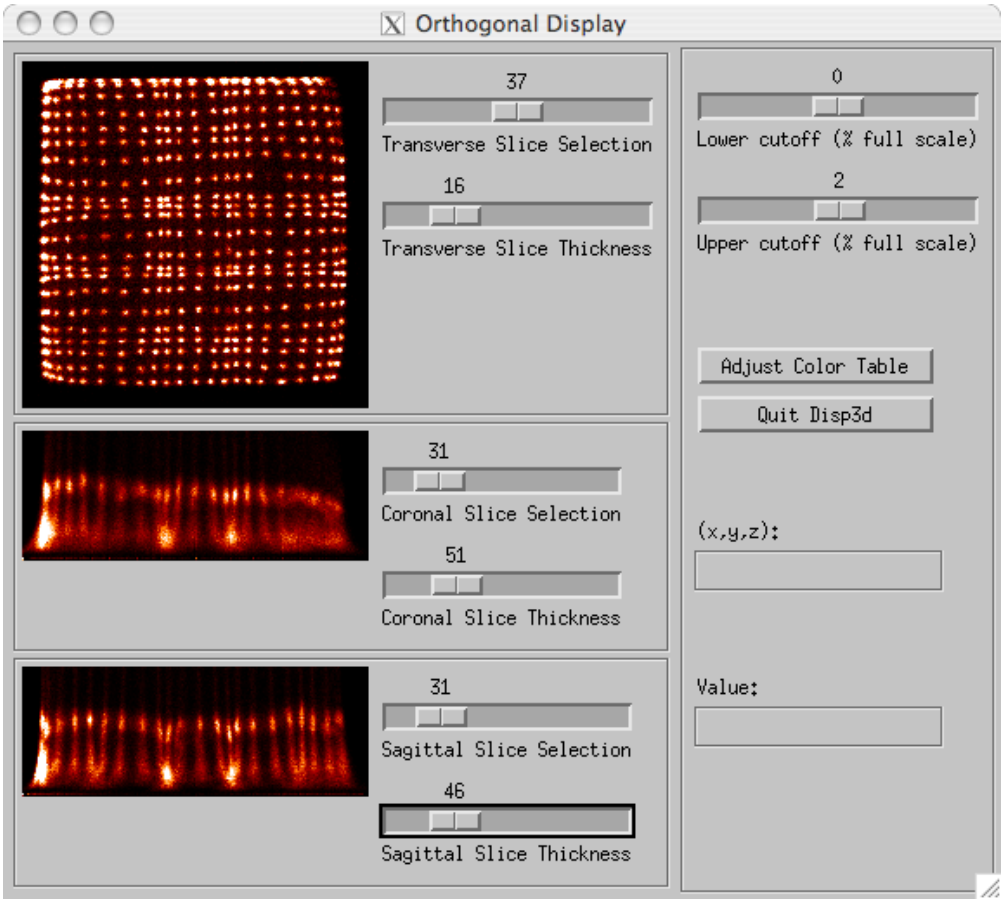

Figure 56: Orthogonal Display of PMT 0 Before Calibration

The Transverse Slice Selection slider selects a histogram bin to use as a base, and the Transverse slice thickness selects a range of histogram bins. Map files have histograms rebinned to 75 bins, so the "Transverse Slice Selection" slider runs from 0 to 75.

In the case of Figure 56, we have selected a histogram range from 37 to 37 + 16 = 53 to be displayed. The average value of the histogram in that range is, "pretty white," indicating that the section of the histogram containing the photopeak is being displayed

If one takes the selection slider and runs it from 75 downward until the spots turn white, one will have imaged the histogram from the high energy side down to the photopeak.

The "Coronal Slice Selection" slider picks out a horizontal section of the top display. This slider ranges from 0 to 199, corresponding to the y-axis coordinate of the pixel. The "Coronal Size Thickness" is the height of the slice that is displayed. The graphic depicts the vertically oriented histograms described above. The top of the graphic section is dark and corresponds to the high energy background section of the histograms. Moving down the image, one sees a row of light sections. These are the photopeaks. Moving further down the image, the histograms grow darker as the photopeak reduces in height down to the Compton shelf level. Further down, the histograms begin lighening as the backscatter peaks develop. The "Saggital Slice Selection" corresponds to a vertical slice through the image

The Coronal slice display in Figure 56 is set to correspond to a horizontal slice across quadrants 0 and 1 of the PMT. The saggital slice display is set to correspond to a vertical slice across quadrants 0 and 2 of the PMT (see Figure 16).

We are now in a position to compare before and after orthogonal displays. Figure 57 (left) shows an orthogonal display captured prior to calibration. Notice in the coronal and saggital slices how the histograms are compressed into the lower half of the possible range. Compare this with Figure 54 (top) which is also a "before" histogram showing the same compression. Figure 57 (right) shows an orthogonal display of the same PMT after calibration. Notice how the histogram now fills the available range. Compare this with Figure 54 (bottom) to see a similar effect in a Rabbit histogram.

Figure 58 (left) shows a "before" orthogonal display of the second PMT on the module. This PMT exhibits much more gain variation than PMT0. Notice in particular the strong dip in gain visible just past the midpoint in the saggital slice. Further notice that the width of the anomaly is about one sixth of the full width of the PMT. This suggests that the problem is due to one particular anode (recall that there are six horizontal anode strips and six vertical anode strips in the PSPMT). Notice that the weak anode corresponds with the dark area in the middle left of the transverse slice.

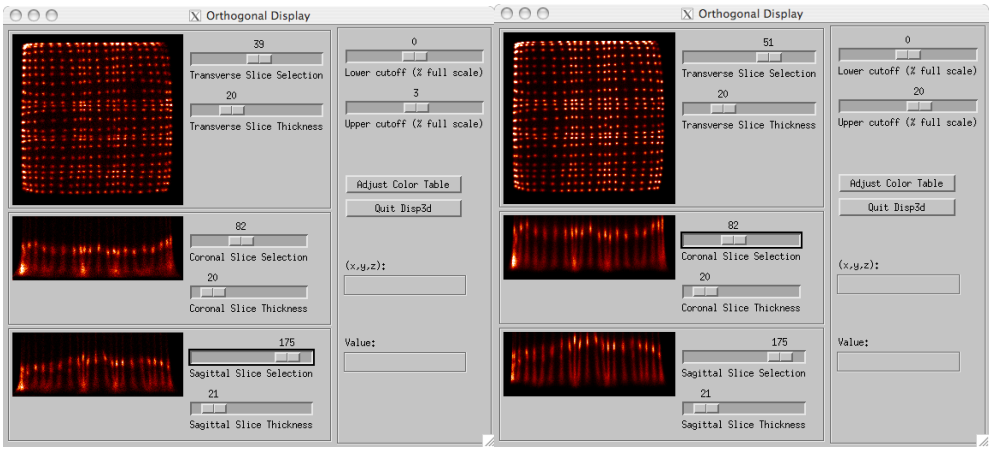

Figure 57: Before (left) and After (right) Module 12, PMT 0

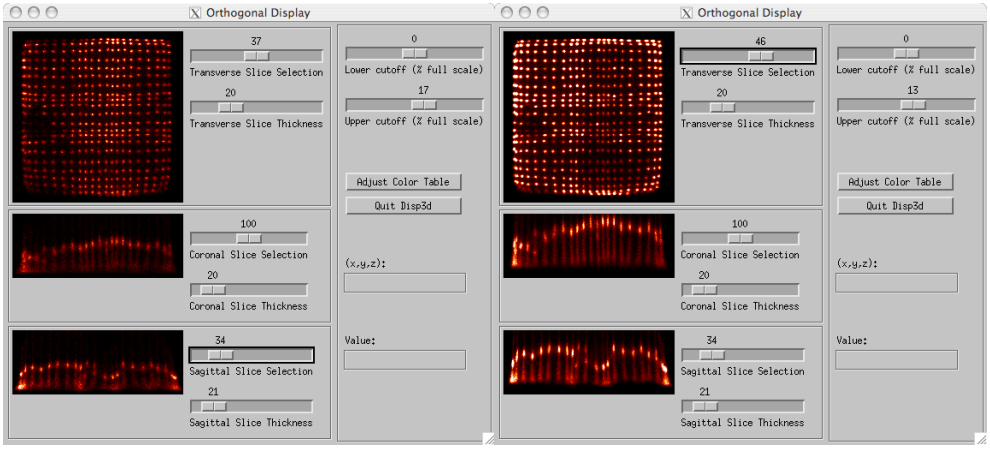

Figure 58: Before (left) and After (right) Module 12, PMT 1

Finally, using the same map files that were used to generate the orthogonal displays of Figure 57 and Figure 58, we generate four quadrant histogram displays. Figure 59 shows the four quadrants of PMT0, and Figure 60 shows the four quadrants of PMT1.

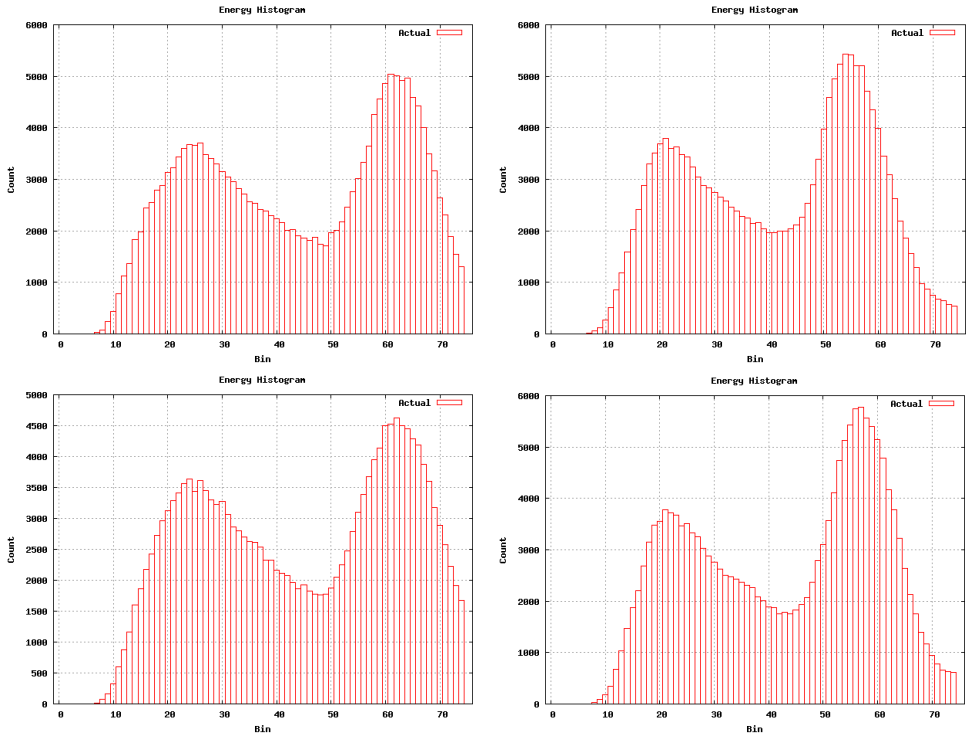

Figure 59: After Histograms Corresponding to Rabbit Quadrants. Module 12, PMT 0

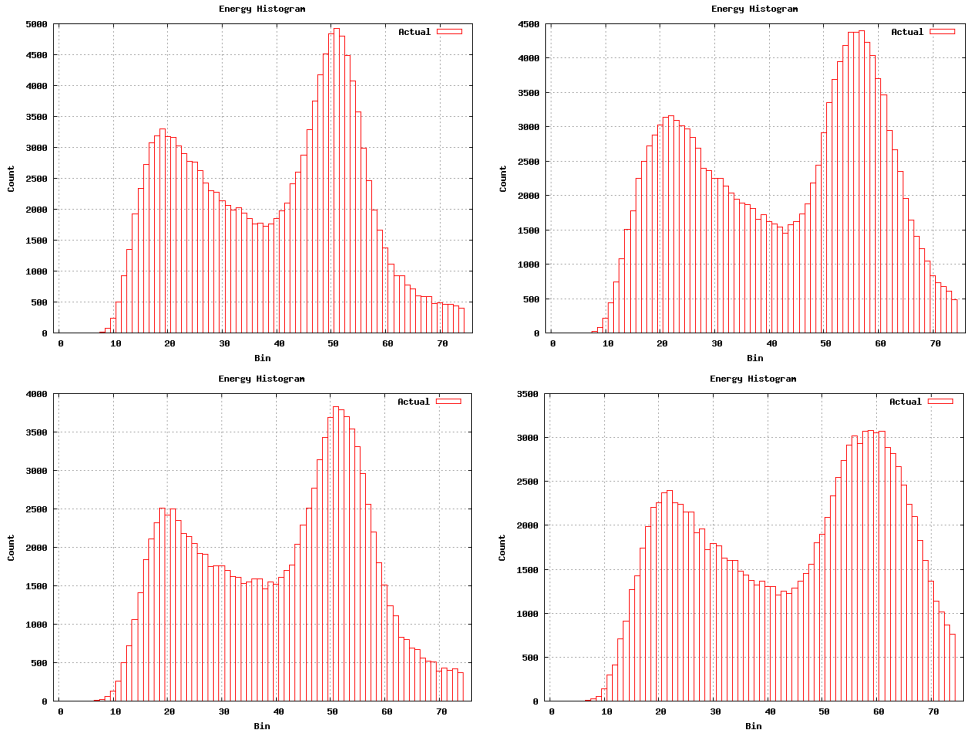

Figure 60: After Histograms Corresponding to Rabbit Quadrants. Module 12, PMT 1

[1] Das A and Ferbel T, Intruduction to Nuclear and Particle Physics, World Scientific, Singapore (2003)

[2] Hamamatsu Photonics, Photomultiplier Tubes: Basics and Applications, Hamamatsu Photonics, Hamamatsu (2007)

[3] Jackson J, Classical Electrodynamics, Wiley, Hoboken (1999)

[4] Leo W: Techniques for Nuclear and Particle Physics Experiments: A How-to Approach, Springer-Verlag, Berlin (1994)

[5] Phelps M, et al., PET: Physics, Instrumentation, and Scanners, Springer, New York (2005)

[6] Saha G, Basics of PET Imaging: Physics, Chemistry, and Regulations, Springer, New York (2005)

[7] Seiden A, Particle Physics, Addison Wesley, San Francisco (2005)

Index