Testing Q_{weak}'s Multiplexing Electronics

Ryan Zielinski REU Program, College of William and Mary
Dr. David Armstrong, College of William and Mary, Physics Dept.
August 5, 2009

Abstract

The Q_{weak} experiment, which will run at Jefferson Laboratory in summer 2010, will be a precision test of the proton's weak charge using parity violating electron scattering. Q_{weak} will place new constraints on the running of the weak coupling; any deviation from the current Standard Model prediction could signal new physics. Q_{weak} will employ four vertical drift chambers, each consisting of two wire planes of 280 wires, in a U - V arrangement to complete the path reproduction of the incident electron. A multiplexing system will allow the determination of both the position and drift time of the path. The goal of this project is to test the time resolution, signal integrity and signal stability of the multiplexing electronics associated with the data readout of the four VDC's.

Supported by NSF REU grant PHY-0755262

I. Introduction

Despite being remarkably consistent between theoretical predictions and experimental results the Standard Model of particle physics is known to be incomplete. Serving as link between the electromagnetic, weak and strong interactions the model fails to include gravity. Beginning in the summer of 2010 the Q_{weak} experiment, running at Jefferson Lab, will provide a further test of the current theory. Any discrepancies in the Standard Model's predictions could signal the possibility of new physics. In order to test the theory Q_{weak} will scatter 85% polarized electrons at 1.165GeV off of a liquid hydrogen target. The aim is to measure the proton's weak charge, which is clearly defined within the Standard Model, using this parity violating electron scattering. The experiment will be highly precise with a combined statistical and systematical error of ~4%.

Essential to the experiment are four vertical drift chambers. The chambers will allow for track reconstruction as well as position and momentum measurements for the scattered electrons. A research group led by Professor David Armstrong here at The College of William and Mary is responsible for the construction and testing of the drift chambers for the Q_{weak} Experiment. My task this summer has been to test the multiplexing electronics associated with the data readout from the chambers. This has included using the CERN developed ROOT data analysis program to check the signal integrity, signal stability and time resolution of the VME crates. Data was taken from cosmic rays and readout from the chamber via the multiplexing crates that will be used in the experiment when it runs at Jefferson Lab. A total of four crates will be needed for the experiment so the goal of this summer's project was to streamline the testing process as much as possible. This included developing a ROOT script that would automate almost the entire analysis procedure with minimal user input needed.

II. The Physics of Q_{weak}

The weak interaction, which Q_{weak} seeks to probe, is mediated by the W and Z bosons. The three bosons, W^+, W^- and Z^0 have charges of +1, -1 and 0 respectively. Due to the nature of their charges each boson acts differently depending on the handedness of the fermions they interact with. The W^+, boson only acts on a fermion with spin parallel to its velocity (right handed fermion). The W^- boson only acts on a fermion with spin anti-parallel to its velocity (left handed fermion). The Z^0 boson because of its electric neutrality acts on both types of fermions but with varying magnitude. This unique property of Z^0 bosons leads to an inherent asymmetry in the probability of left and right handed fermion scattering. Q_{weak} will use this fact to measure the weak charge of the proton.

The asymmetry associated with the fermion scattering is governed by the following equation:

$$A \equiv \frac{\sigma^+ - \sigma^-}{\sigma^+ + \sigma^-} \tag{2.1}$$

Where σ^+ and σ^- are the scattering cross sections associated with the right and left handed fermions experienced under an elastic collision. Quantum Field Theory shows that this asymmetry should also be defined by (as a low order approximation and at low momentum transfers) [2]:

$$A = \frac{1}{P}\frac{-G_F}{4\pi\alpha\sqrt{2}} Q^2 Q_W^P \tag{2.2}$$

Where P is the polarization of the electron beam, Q^2 is the momentum transfer, G_F is the Fermi coupling and α is the fine structure constant. It should also be noted that the momentum transfer is defined as the amount of momentum that is moved between particles during a collision. It is clear then that by combing the above two equations and by calculating the associated asymmetry in the electron-proton scattering Q_{weak} will be able to measure the weak charge of the proton. Q^2 values will be determined via the scattering angles used. It is of note that in order to complete the calculations the polarization of the incident beam will need to be changed periodically to obtain both right and left handed fermions.

III. The Experimental Set-Up

A magnet will be used to bend scattered electrons away from the incident beam after it hits the 35cm liquid hydrogen target. Collimators will be used to select the scattering angle. From there the scattered electrons will go through a three region tracking system before entering the quartz detector bars. The tracking system will be responsible for reconstructing the particles path, making sure that only scattered electrons enter the quartz detectors. Detected particles, creating bursts of light, will be read out via photo-multiplier tubes from the quartz bars. In Figure 3.1 the collimator and magnet set up can be seen.

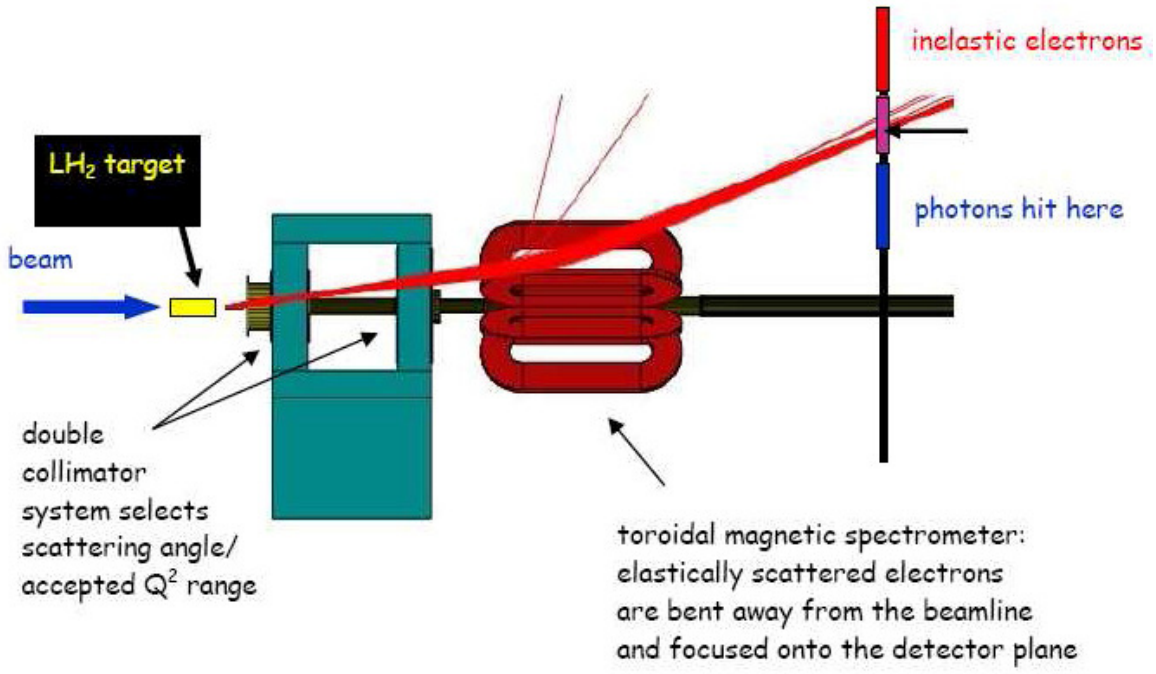

Figure 3.1: The incident electron beam strikes the 35cm long liquid hydrogen target and then passes through the double collimator which selects the scattering angle. A toroidal magnet then bends the scattered electrons away from the beam line so they then can be detected by the quartz bars. [2]

Each region serves as a filter for inelastic, secondary and electron-electron scattering. Region I is responsible for determining the position of the electrons once they exit the double collimator. This region will consist of Gas-Electron Multipliers (GEM's). Region II will determine the position of the scattered electrons before they enter the torus shaped magnet and are bent away from the beam line. It is also responsible for determining the momentum spectrum. This region will consist of horizontal drift chambers as well as the QTOR magnet. Region III will make certain that only elastically scattered particles are entering the quartz detectors. It will determine the position and angle and ultimately the momentum of the particles. In order to do so this region will consist of four vertical drift chambers assembled on a rotating chassis to accommodate the entire detector system. Four measurements will need to be made to accomplish this task.

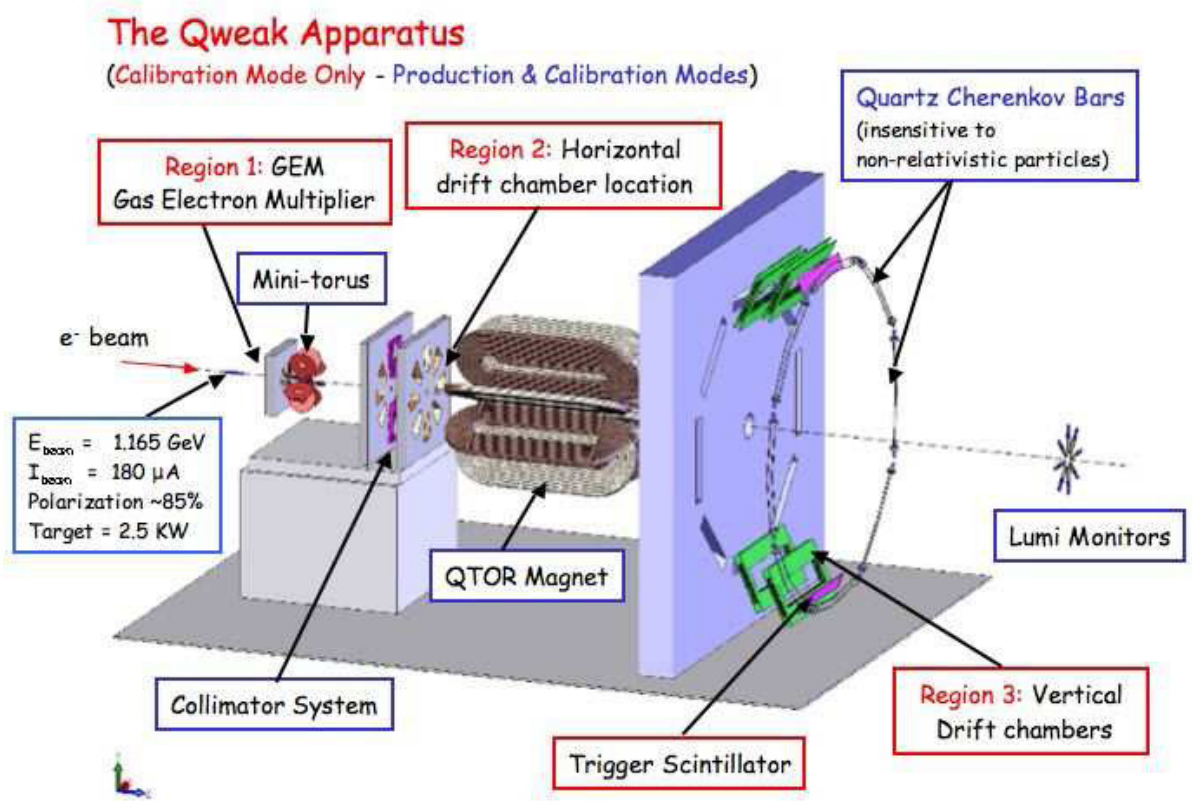

Figure 3.2: The Q_{weak} Tracking System will consist of three regions. The lab group at William and Mary is responsible for the Region III vertical drift chambers which will make sure that only elastically scattered electrons are entering the detectors

IV. Drift Chambers

A drift chamber is a gas filled device containing an array of wires which uses a high electric field to detect charged particles. The wires themselves are held at ground while a high voltage is applied to the chamber. By measuring the drift time of the particles the chamber is be able to determine both the location and angle of the primary electron after the elastic scattering occurs [1]. The drift time is defined as the time for the first electron to hit the detector wire and its associated velocity is the drift velocity. In order to keep the drift velocity as stable as possible a mixture of Argon (65%) and Ethane (35%) gas is used. In total each chamber used in the Q_{weak} experiment will contain an array of 560 wires in a U-V plane arrangement. Two wire planes orientated at 90 degrees to each other (280 wires each) are used to make sure all particles are detected.

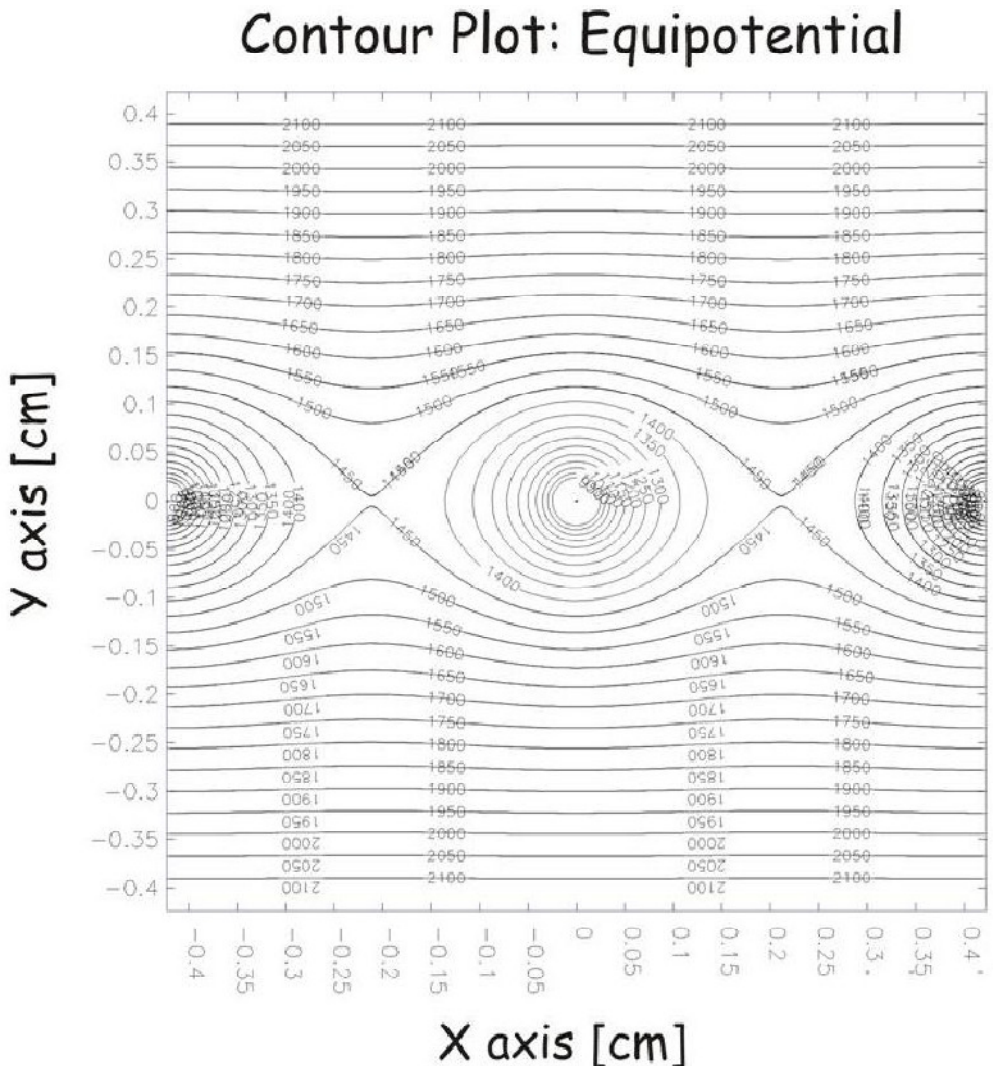

Figure 4.1: The equipotential lines inside the chamber show the increasing electric field as the distance to the grounded sense wires becomes smaller and smaller. [3]

When the elastically scattered electrons enter the drift chamber they will collide with the gas molecules creating molecule-electron ionization pairs. The electrons and ions become accelerated towards the grounded wires by the large electric field within the chamber. The closer to the grounded wires the particles get the larger the electric field (Figure 4.1), and with this the acceleration, becomes. The increased acceleration and energy allow for the creation of more electron-ion pairs. The closer to the wire the primary electron gets the more secondary electron-ion pairs are created. [5] This is known as the avalanche effect (Figure 4.2) and it triggers data readout from the wires. Knowing the drift distances for several wires allows for the path reconstruction of the primary electron.

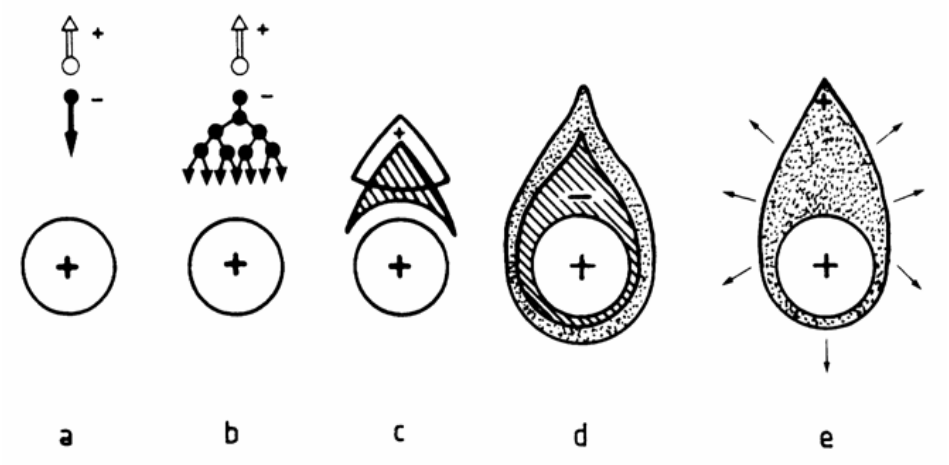

Figure 4.2: Demonstrating the avalanche effect. At (a) the primary electron travels towards the grounded sense wire. On its way to the wire the electron, due to the increasing electric field, undergoes more ionizing collisions (b). The avalanche begins to develop (c) and as more electrons are rapidly collected (d) the remaining cloud of positive ions is stretched out towards the cathode (e). [4]

V. Vertical Drift Chambers

The Q_{weak} experiment will employ a specific kind of drift chamber known as a vertical drift chamber. As the name implies a vertical drift chamber determines the path of a primary electron by creating ionization that occurs in the vertical direction. This vertical drift is achieved by applying a high negative voltage to the chamber. Positively charged ions twill drift towards the cathode planes of the chamber while negatively charged electrons will drift towards the grounded wires. Data is collected in the form of analog pulses on the signal wires. The pulses are then amplified, discriminated and readout via multi-hit time-to-digital converters (TDC's). Working backwards and knowing the relationship between the time and perpendicular distance to the grounded wire the position of the primary electron can be determined.

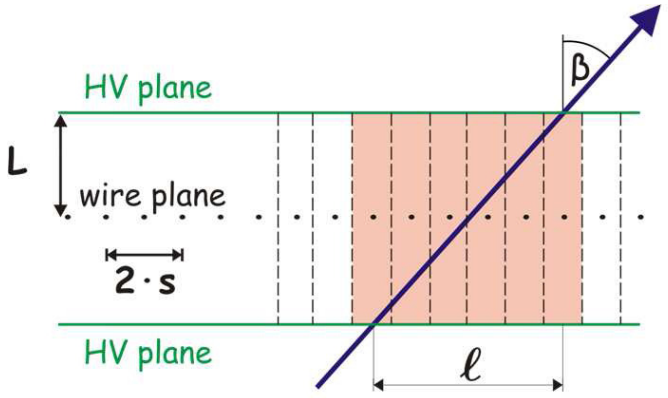

Figure 5.1: A diagram of the vertical drift chamber found in Region III. The top and bottom lines are the high voltage planes. The black dots are the sense wires. The red region denotes the ionization region from the incoming blue track. L is the distance from the wires to the high voltage plane. *l* is the length traversed by the electron from when it enters and exits the chamber with β being the exit angle. [2]

VI. Data readout from the Chambers

For the experiment, data readout will consist of 18 (per plane) CERN developed, 16-channel amplifier discriminator boards known as MAD cards. The cards will output a low-voltage differential format (LDVS) signal. The differential signal is used to help suppress noise. As previously mentioned each drift chamber contains 560 sense wires. Summing over the four chambers it becomes apparent that a total of 2240 wires need to be read out and analyzed. Assuming an individual TDC channel is assigned to each wire and a 64channel TDC is used, 35 individual TDC's would be needed. To save on this considerable expense a multiplexing system consisting of multiple delay lines will be implemented. In the system 18 wires from separate locations in the chamber will be placed onto two signal paths leading to two TDC channels. [8] From the sum of the two signals the drift time can be determined and from the difference of the two signals the triggered wire can be identified. This process reduces the number of TDC's needed by a factor of nine.

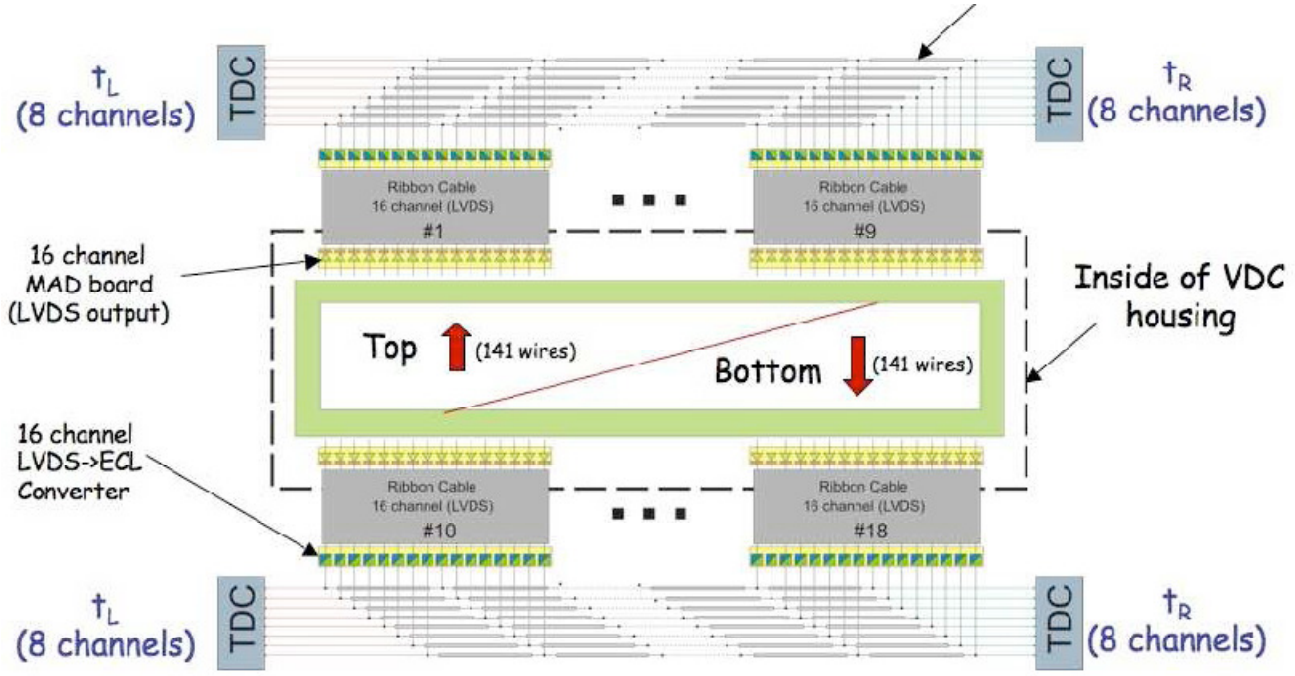

Figure 6.1: Delay scheme schematic showing the readout of a chamber. The multiplexing crate will do the job of the LDVS to ECL conversion. The delay lines denoted by the gray bars feeding into the TDC channels will also be part of the VME crate. [8]

The delay scheme is based on simple ECL logic in which the ECL logic gates act as the delay mechanism. The LVDS signal created by the MAD boards is fed into an external multiplexing (VME) crate. The crate turns the LDVS signal into an ECL pulse via the LEX2 boards developed at Jefferson Lab. As an ECL pulse the signal is split and then fed into the ECL gate delay backplane. One signal will go left and the other will go right along the backplane. The backplane delays every 9^{th} wire by 1.3ns. Meaning every 9^{th} wire is multiplexed into a single delay line where the signals are offset by 1.3ns. The split and delayed signals are then read out on the left

and right side of the delay line where the difference can be taken to determine which wire was hit. Each pair of corresponding TDC channels (IE 0 and 16) correspond to a certain set of wires (IE 1, 9, 17…). Depending upon the time location of the signals within each channel pair, the hit wire can be determined. A total of up to eighteen signals should be seen per channel pair and because of the left minus right measurement the separation between the peaks should be 2.6ns.

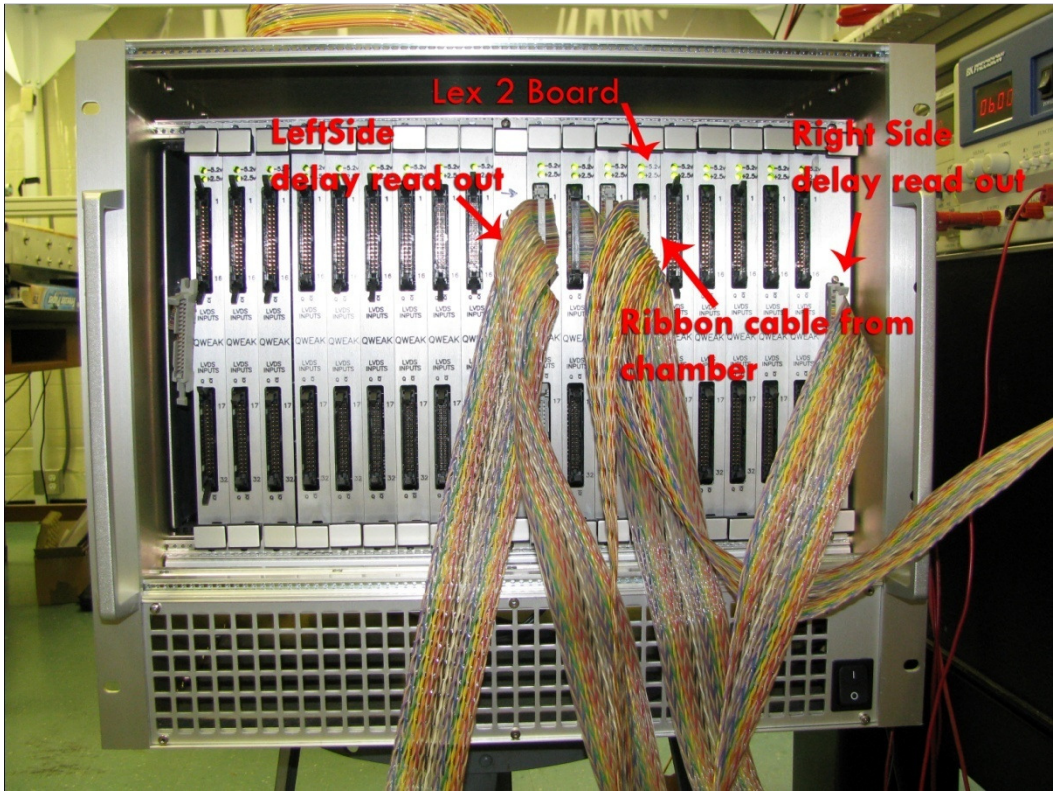

Figure 6.2: An assembled multiplexing crate full of LEX-2 circuit boards. One quadrant of the crate (between the delay readouts and the top half of the LEX-2 board) will read out approximately 140 wires from the chamber. In order to see a full set of peaks all ribbon cables would need to be hooked up to one quadrant of the crate. As currently pictured 8 peaks would be seen in the relevant histograms.

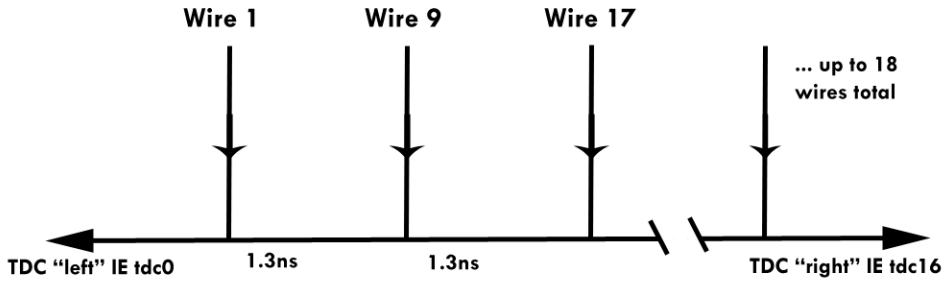

Figure 6.3: Further illustration of the delay line technique. The diagram is repeated for each set of TDC channels. For example the schematic for TDC1 and TDC 17 would contain wires 2, 10, 18 and so on. The number of wires seen is directly related to the number of ribbon cables connected to the LEX-2boards as previously stated

Data from the TDC comes out in a CODA event format file. In order for the data to be analyzed this CODA file is converted into another file type that can be used by the ROOT analysis software. Once the data is stored in a ROOT tree, the left minus right measurements can be taken via the ROOT GUI. The command line interface of ROOT allows for the plotting and fitting of the relevant histograms. Also, ROOT has a built in C/C++ interpreter for developing scripts to aid in data analysis. A thorough ROOT tutorial can be found at the following website.

When the data is graphed Gaussian shaped peaks are expected for each delay line used. As previously stated each peak corresponds to a wire within the chamber. Up to 18 peaks and thusly 18 wires can be identified per each channel pairing. Good data is characterized by peak stability. This means that there is no overlapping between the peaks. The peaks should experience minimal drifting and be confined within a narrow enough time window (they have narrow widths). Gaussian fits are applied to each peak in order to determine its location in the time spectra (the mean of the fit) and also its width (the sigma value of the fit). The signals should appear at the output 100% of the time. There should be signal isolation meaning there should not be ECL outputs on other output channels. Further checks can also be performed to make sure that peaks are 2.6ns apart. Delay line checks (not done this summer) can be implemented by comparing the left and right signals and to verify that they appear 17x1.3ns = 22.1 ns apart.

VII. Results

The first results of the VME crate testing can be found below. The TDC resolution was set at 98.9 ps/bin. Data was taken using two connectors, hence the four peaks. Initial results showed problems with the V775 TDC. The locations and definition of the peaks depends on the time scale examined. This is apparent when Figure 7.1 is examined. If the whole time spectrum is included the peaks are smeared out and are indistinguishable. Wire location is rendered undecipherable. A solution to this problem was found to be that the calibration for each TDC channel varied. Not all of the channels had the same number of ps per bin. The supposedly constant left minus right subtraction was smeared out depending on what the absolute time is, i.e. where on the TDC range the hits arrive. [6]

To fix the problem the relative binning of each pair of TDC channels in the V775 module were measured. In order to do this a duplicated signal was used to “Start” the counting on each channel pair. The “Common Stop” signal was varied and the peaks moved around in the TDC time spectrum. The difference in the bin numbers for the two peaks, as the “Common Stop” varied, should be constant if both channels register the same number of picoseconds per bin. This was found to not be the case. The calibrations varied by approximately 4%. This variation was then measured by using the “sliding scale” mode on the TDC. [7] Figure 7.2 confirms that the problem was fixed.

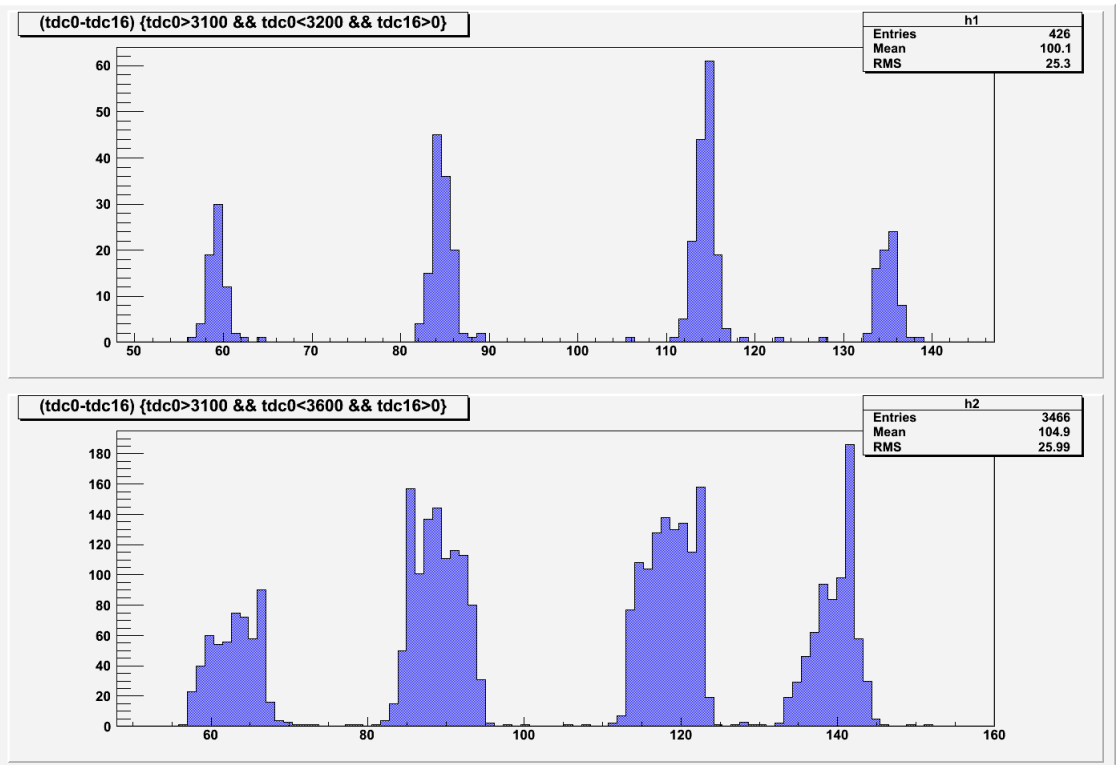

Figure 7.1: An example of the multiplexing smearing. It is clear that as the time window is increased the peaks lose sharpness.

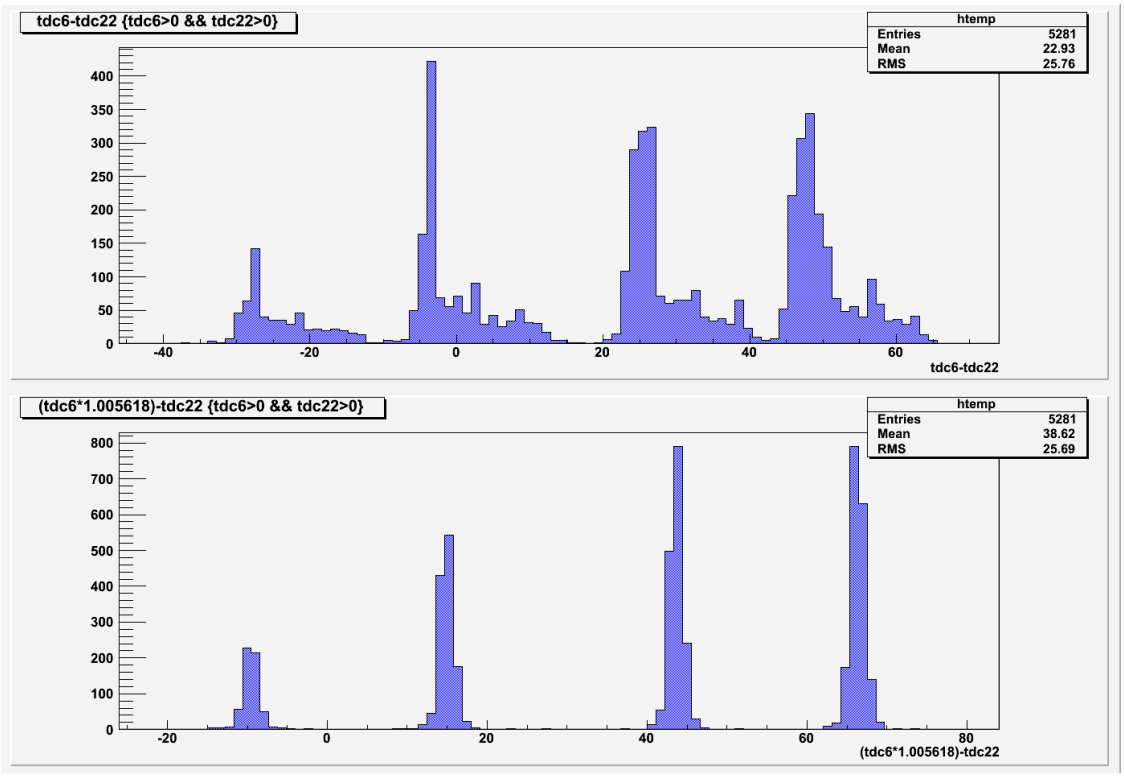

Figure 7.2: An example of the smearing that occurred before the slope correction along with the associated fix. It is clear that once the channel is adjusted for the varying LSB the peaks become well defined regardless of the position in the time spectrum

With the smearing problem solved the next task was to automate the analysis process. Previously all analysis was done by manually loading the ROOT tree, manually plotting the graphs and applying the fits. The process was sped up by creating a C script that does most of the dirty work. The script (found in the appendix) contains a mix of C coding structures and ROOT command line prompts. The script is loaded at the ROOT terminal by typing ".L mux_rev4.c" and then is run by calling "tdc()". The code will prompt the user to input the location of the ROOT and slope configuration files as well as the first TDC channel. From there the script pairs up the second TDC channel with the correct slope value from the input file. The script will automatically find and fit the peaks and output the fit data to a text file for further analysis. A sample output can be found below.

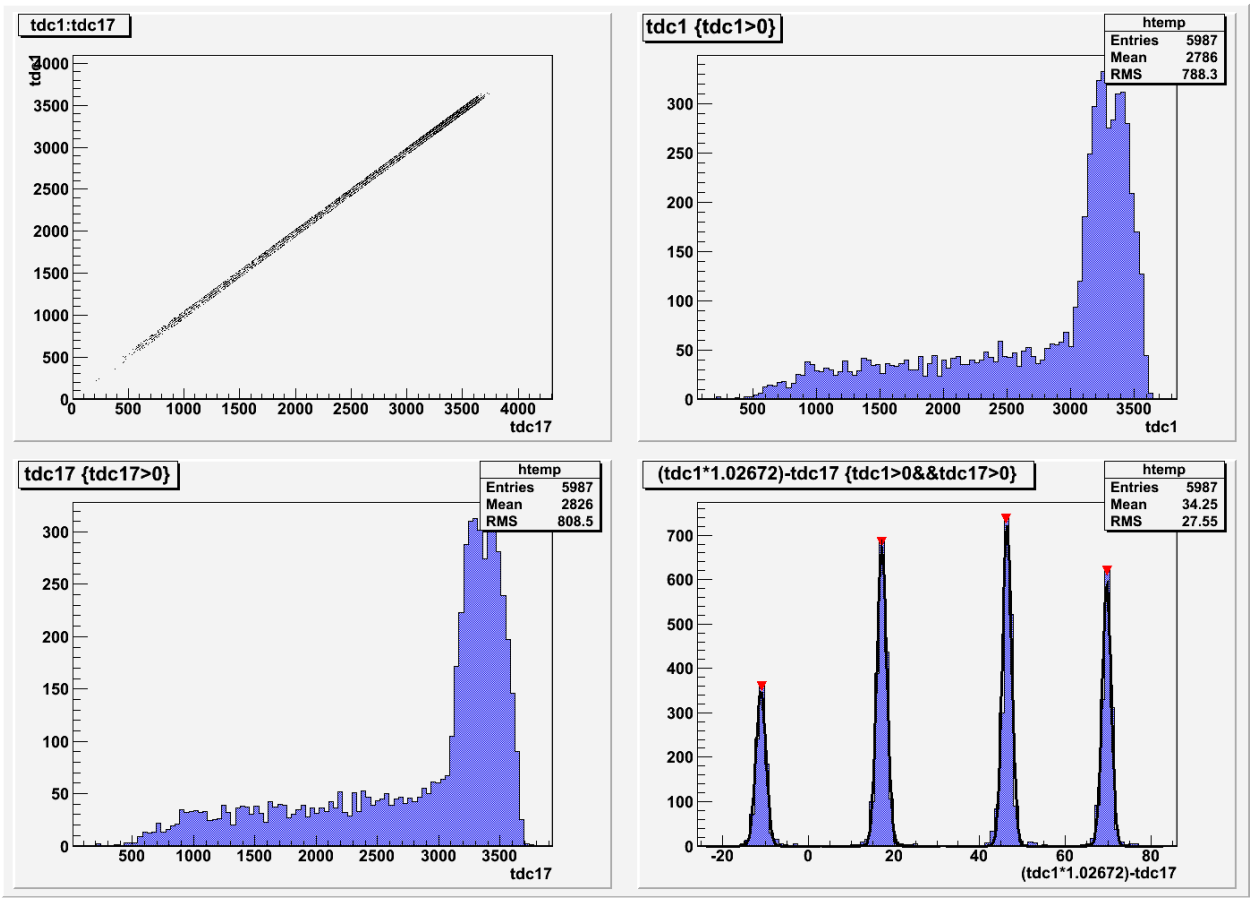

Figure 7.3: An example of the output of the code found in the Appendix. The upper left quadrant shows that there is indeed correlation between the timing of the two TDC channels. The upper right and bottom left quadrants show the range of data for the two channels. For all quadrants except for the upper left the x-axis is time represented in bin number and the y-axis is event number. In the bottom right quadrant the read arrows representing the find if the peaks in the code. Four peaks are show meaning two delay lines were read out.

```
Run number: C775_1132.root
Channel numbers: tdc1:tdc17
 **************

 Peak 1
 Name            Value           Error
 Constant        87.176080       7.933803
 Mean            16.634045       0.074530
 Sigma           1.093570        0.066648

 Peak 2
 Name            Value           Error
 Constant        88.325089       7.940070
 Mean            45.819869       0.075498
 Sigma           1.049514        0.061624

 Peak 3
 Name            Value           Error
 Constant        80.456112       7.500501
 Mean            69.009664       0.079003
 Sigma           0.949084        0.051772

 Peak 4
 Name            Value           Error
 Constant        47.948118       5.714605
 Mean            -11.529989      0.079003
 Sigma           1.165938        0.093029
```

Figure 7.4: An example of the fit data output by the code found in the Appendix. It importantly reports the mean and sigma values with the associated errors.

VIII. Conclusion

The first half of the summer was spent learning ROOT, brushing up on the syntax of the C++ and C programming languages, and learning the ins and outs of the Q_{weak} experiment. This was done while waiting for a complete multiplexing crate to be built by the Jefferson Lab electronics group. Once the crate was built and confirmed working it was moved to the lab at William and Mary. Here at William and Mary the crate was tested using cosmic ray data output from the drift chamber using a V775 TDC. It has been determined this summer that the multiplexing crate is operational and performs as expected. Although there is work that still needs to be done. At the present moment not all channels and delay lines have been confirmed working. The thermal and time-dependent effects, and cross talk (IE is a hit registered on one channel at nearly the same time as a hit on an adjacent channel is there any smearing). It also needs to be checked for what happens if a LEX-2 card is removed. The Jefferson Lab F1 TDC will be used during the experiment and while the F1 has been successfully hooked up at the William and Mary lab no data has been taken with it. It will need to be confirmed if there is the same smearing effect with this TDC. If there is then a similar process as with the V775 TDC will be used to correct for it. Dr. David Armstrong also approved the purchase of the three remaining multiplexing crates so these will need to be tested as well. All of these tasks will be looked at in the fall as I continue my senior research working with the Q_{weak} experiment and associated lab group at The College of William and Mary.

IX. References

1. Brian P. Walsh. W&M Senior Thesis Project: “Development of a Flatness Scanner and Simulation for Qweak Wire Chambers”. 2007. (Unpublished)

2. Douglas C. Dean. W&M Senior Honors Thesis Project: “Constructing, Testing And Characterization of Vertical Drift Chambers for Qweak”. 2009. (Unpublished)

3. Graham K. Giovanetti. W&M Senior Honors Thesis Project: “Development of the Front End Electronics for the Qweak Experiment”. 2006. (Unpublished)

4. G. Charpak. Applications of proportional chambers and drift chambers in high-energy physics and other fields. Nature, 1977. pp 479-482

5. Sauli, F. Principles of Multiwire Proportional and Drift Chambers. European Organization for Nuclear Research [CERN]. Lecture. Geneva 1977.

6. Multiplexing with Chamber Data: Smearing, (http://dilbert.physics.wm.edu/Construction/703)

7. Multiplexing smearing: Solved, (http://dilbert.physics.wm.edu/Construction/706)

8. Armstrong et al. The Qweak Experiment: A Search for New Physics at the TeV Scale Via the Measurement of the Protons Weak Charge. http://www.jlab.org/qweak/. 2007. (Unpublished; Proposal to Jefferson Lab Program Advisory Committee)

APPENDIX: ROOT Code

```
/*  "mux_rev4.c" -- Create histograms from multiplexing crates and
apply Gaussian fits to the peaks  */
/*  Ryan Zielinski 7/17/09 */


#include <stdio.h>
#include <string.h>


void tdc()
{

char data_in_file[200] = "C:/root/"; //location of root file
char slope_in_file[200] = "C:/root/"; //location of slope file
char filename[100];                  //name of root file
char slopename[100];                  //name of slope file
char *data_filename;                 //pointer to root file
char *slope_filename;                 //pointer to slope file
int  chanl_num1;                    //channel numbers
int  chanl_num2;                    //channel numbers
char slope[100] = {'\0'};            //slope correction number
char tdc1[100]="tdc";                //first tdc channel
char tdc2[100]="tdc";               //second tdc channel
```

```
char *channel1;                    //pointer to first channel
char *channel2;                    //pointer to second channel



//Structure to define the upper and lower bounds of the peaks

struct range
{
int lower;
int upper;
};

struct range peak1 = {0,0};
struct range peak2 = {0,0};
struct range peak3 = {0,0};
struct range peak4 = {0,0};

//Load root file

     printf("Enter input data file name (C:/root/***) :\n");
      scanf("%s",filename);
      strcat(data_in_file, filename);
      printf("data file name = %s\n",data_in_file);
      data_filename = data_in_file;

// Load slope calibration file

      printf("Enter input slope file name (C:/root/***) :\n");
      scanf("%s",slopename);
      strcat(slope_in_file, slopename);
      slope_filename = slope_in_file;

// Open and read slope calibration file

FILE *fs;
char line[10] = {'\0'};
float slopes;
int i = 0;
float slopeval[16] = {0};
fs = fopen(slope_filename, "rt");

 while((fgets(line, 80, fs) != NULL) && i < 16)
   {
       sscanf (line, "%f", &slopes);
      slopeval[i] = slopes;
      i++;
   }

fclose(fs);

//Print loaded slope values to screen for verification
```

```
for (k=0; k<16; k++)
{

printf("%f \n", slopeval[k]);

}

//Channel Number Enquiry

    printf("Enter the number of the first tdc channel: \n");
    scanf("%d", &chanl_num1);

//Determines Second Channel Number and Slope Correction Value

switch(chanl_num1)
{
  case 0:
    chanl_num2 = 16;
    sprintf(slope, "%f", slopeval[0]);
    strcat(tdc1, "0");
    strcat(tdc2, "16");
    break;

  case 1:
    chanl_num2 = 17;
    sprintf(slope, "%f", slopeval[1]);
    strcat(tdc1, "1");
    strcat(tdc2, "17");
    break;

case 2:
    chanl_num2 = 18;
    sprintf(slope, "%f", slopeval[2]);
    strcat(tdc1, "2");
    strcat(tdc2, "18");
    break;

case 3:
    chanl_num2 = 19;
    sprintf(slope, "%f", slopeval[3]);
    strcat(tdc1, "3");
    strcat(tdc2, "19");
    break;

case 4:
    chanl_num2 = 20;
    sprintf(slope4, "%f", slopeval[4]);
    strcat(tdc1, "4");
    strcat(tdc2, "20");
    break;
```

```
case 5:
    chanl_num2 = 21;
    sprintf(slope, "%f", slopeval[5]);
    strcat(tdc1, "5");
    strcat(tdc2, "21");
    break;

case 6:
    chanl_num2 = 22;
    sprintf(slope, "%f", slopeval[6]);
    strcat(tdc1, "6");
    strcat(tdc2, "22");
    break;

case 7:
    chanl_num2 = 23;
    sprintf(slope, "%f", slopeval[7]);
    strcat(tdc1, "7");
    strcat(tdc2, "23");
    break;

case 8:
    chanl_num2 = 24;
    sprintf(slope, "%f", slopeval[8]);
    strcat(tdc1, "8");
    strcat(tdc2, "24");
    break;

  case 9:
    chanl_num2 = 25;
    sprintf(slope, "%f", slopeval[9]);
    strcat(tdc1, "9");
    strcat(tdc2, "25");
    break;

case 10:
    chanl_num2 = 26;
    sprintf(slope, "%f", slopeval[10]);
    strcat(tdc1, "10");
    strcat(tdc2, "26");
    break;

case 11:
    chanl_num2 = 27;
    sprintf(slope, "%f", slopeval[11]);
    strcat(tdc1, "11");
    strcat(tdc2, "27");
    break;

case 12:
    chanl_num2 = 28;
```

```
    sprintf(slope, "%f", slopeval[12]);
    strcat(tdc1, "12");
    strcat(tdc2, "28");
    break;

case 13:
    chanl_num2 = 29;
    sprintf(slope, "%f", slopeval[13]);
    strcat(tdc1, "13");
    strcat(tdc2, "29");
    break;

case 14:
    chanl_num2 = 30;
    sprintf(slope, "%f", slopeval[14]);
    strcat(tdc1, "14");
    strcat(tdc2, "30");
    break;

case 15:
    chanl_num2 = 31;
    sprintf(slope, "%f", slopeval[15]);
    strcat(tdc1, "15");
    strcat(tdc2, "31");
    break;

default:
  printf("Invalid channel choice");

}

//Print channel numbers and slope number for verification

    printf("\n *************************\n");
    printf("First tdc channel number: %s\n" , tdc1);
    channel1 = &tdc1;
    printf("Second tdc channel number: %s\n" , tdc2);
    channel2 = &tdc2;
    printf("Slope correction value: %s\n" , slope);
    printf("\n *************************\n\n");

//Load Root tree

TFile f(data_filename);
f.ls();
TTree *tree = R;
tree->Print();

//Create new canvas

TCanvas *MyC = new TCanvas("MyC",data_filename,1);
MyC->Divide(2,2);
```

```
//Fill canvas with relevent histograms


MyC->cd(1);
char argument[100] = {'\0'};
strcpy(argument, channel1);
strcat(argument, ":");
strcat(argument, channel2);
tree->Draw(argument);


MyC->cd(2);
char argument2[100] = {'\0'};
strcpy(argument2, channel1);
strcat(argument2, ">0");
tree->SetFillColor(4);
tree->SetFillStyle(3001);
tree->Draw(channel1, argument2, "",4000, 0);

MyC->cd(3);
char argument6[100] = {'\0'};
strcpy(argument6, channel2);
strcat(argument6, ">0");
tree->SetFillColor(4);
tree->SetFillStyle(3001);
tree->Draw(channel2, argument6, "", 4000,0);

MyC->cd(4);
char temp[100] = "(";
char argument3[100] = {'\0'};
char argument4[100] = {'\0'};
char argument5[100] = {'\0'};
strcpy(argument3, temp);
strcat(argument3, channel1);
strcat(argument3, "*");
strcat(argument3, slope);
strcat(argument3, ")");
strcat(argument3, "-");
strcat(argument3, channel2);
strcpy(argument4, channel2);
strcat(argument4, ">0");
strcpy(argument5, argument2);
strcat(argument5, "&&");
strcat(argument5, argument4);
tree->SetFillColor(4);
tree->SetFillStyle(3001);
tree->Draw(argument3, argument5 , "", 4000, 0);

TH1F *peakloca = (TH1F*)gPad->GetPrimitive("htemp");

//Determine peak location for Gaussian fits code in C++
```

```
      TSpectrum *sp1 = new TSpectrum();
      sp1->Search(peakloca,1.0,"") ;
      Int_t npeaks = sp1->GetNPeaks() ;
      cout << "Peaks found " << npeaks << endl ;

      Float_t *peaks ;
      peaks = sp1->GetPositionX() ;

      for (j=0 ; j<npeaks ; j++)
     cout << "Peak at = " << peaks[j] << endl ;

//Set bounds on location for Gaussian fits

      peak1.lower = peaks[0] -10;
      peak1.upper = peaks[0] + 10;


      peak2.lower = peaks[1] - 10;
      peak2.upper = peaks[1] + 10;


      peak3.lower = peaks[2] -10;
      peak3.upper = peaks[2] + 10;


      peak4.lower = peaks[3] - 10;
      peak4.upper = peaks[3] + 10;

//Create Gaussian fits and apply to canvas

p1 = new TF1("m1", "gaus", peak1.upper, peak1.lower);
p2 = new TF1("m2", "gaus", peak2.lower, peak2.upper);
p3 = new TF1("m3", "gaus", peak3.lower,peak3.upper);
p4 = new TF1("m4", "gaus", peak4.lower, peak4.upper);


tree->Fit("m1", argument3,argument5, "R SAME" ,"", 4000, 0);
tree->Fit("m2", argument3,argument5, "R SAME","", 4000, 0);
tree->Fit("m3", argument3,argument5, "R SAME","", 4000, 0);
tree->Fit("m4", argument3,argument5, "R SAME","", 4000, 0);

//Retrieve fit information for the peaks

Double_t constant1 = p1->GetParameter(0);
Double_t ce1 = p1->GetParError(0);
Double_t mean1 = p1->GetParameter(1);
Double_t me1 = p1->GetParError(1);
Double_t sigma1 = p1->GetParameter(2);
Double_t se1 = p1->GetParError(2);

Double_t constant2 = p2->GetParameter(0);
```

```
Double_t ce2 = p2->GetParError(0);
Double_t mean2 = p2->GetParameter(1);
Double_t me2 = p2->GetParError(1);
Double_t sigma2 = p2->GetParameter(2);
Double_t se2 = p2->GetParError(2);


Double_t constant3 = p3->GetParameter(0);
Double_t ce3 = p3->GetParError(0);
Double_t mean3 = p3->GetParameter(1);
Double_t me3 = p3->GetParError(1);
Double_t sigma3 = p3->GetParameter(2);
Double_t se3 = p3->GetParError(2);


Double_t constant4 = p4->GetParameter(0);
Double_t ce4 = p4->GetParError(0);
Double_t mean4 = p4->GetParameter(1);
Double_t me4 = p3->GetParError(1);
Double_t sigma4 = p4->GetParameter(2);
Double_t se4 = p4->GetParError(2);

//Print fit information to the an output text file

FILE *fp;
char output[100] = {'\0'};
strncpy(output, &filename[5], 4);
strcat(output, "_");
strcat(output, tdc1);
strcat(output, tdc2);
strcat(output,".txt");
fp = fopen(output,"w");

fprintf(fp, "Run number: %s \nChannel numbers: %s \n **************
\n\n Peak 1 \n Name \t\t Value \t\t Error \n Constant\t %f \t %f \n
Mean\t\t %f \t %f \n Sigma\t\t %f \t %f
\n\n Peak 2 \n Name \t\t Value \t\t Error \n Constant\t %f \t %f \n
Mean\t\t %f \t %f \n Sigma\t\t %f \t %f \n\n Peak 3 \n Name \t\t Value
\t\t Error \n Constant\t %f \t %f \n Mean\t\t %f \t %f \n Sigma\t\t %f
\t %f \n\n Peak 4 \n Name \t\t Value \t\t Error \n Constant\t %f \t %f
\n Mean\t\t %f \t %f \n Sigma\t\t %f \t %f", filename, argument,
constant1, ce1, mean1, me1, sigma1, se1, constant2, ce2, mean2, me2,
sigma2, se2, constant3, ce3, mean3, me3, sigma3, se3, constant4, ce4,
mean4, me4, sigma4, se4);
fclose(fp);

}
```