CLAS-NOTE 1999-011
CVS: docs/tof_calib.tex
Original: August 10, 1999
Updated: November 5, 1999
Updated: August 8, 2001
Version: 1.2

Calibration of the CLAS TOF System

E.S. Smith, V. Dharmawardane, H. Egiyan, J. Ball, L. Elouadrhiri, M. Holtrop, K. Loukachine, G. Mutchler, Y. Prok, E. Pasyuk, J. Santoro, S. Taylor, L. Todor

Abstract

We describe the procedure for calibrating the Time-of-Flight(TOF) system of CLAS. This paper is intended as a guide for updating the calibration constants for a new run period. We define the meanings of all calibration constants in the Map and how to run the programs to determine them for both electron and photon beam experiments. We also provide an initial set of quantities which may be used to monitor the quality of the TOF calibration.

Contents

1 Introduction

The purpose of this CLAS-NOTE is to provide the CLAS users with detailed information and a manual on TOF calibration procedure. We do not expect any major changes of the procedure in the future, however the calibration code may be developed and improved.

2 TOF Constants

The reconstructed times and energies in the scintillator are determined in the following way:

$$\begin{aligned}
t_L &= c_{norm} \cdot (t_w - c_{LR}/2 + c_{c2c} + c_{p2p}) & (1)\\
t_R &= c_{norm} \cdot (t_w + c_{LR}/2 + c_{c2c} + c_{p2p}) & (2)\\
\bar{t} &= (t_L + t_R)/2 & (3)\\
E_L &= \frac{k(A-P)}{M0_L} & (4)\\
E_R &= \frac{k(A-P)}{M0_R} & (5)\\
E &= \sqrt{E_L \cdot E_R} & (6)\\
y &= \frac{v_{eff}}{2}\,(t_L - t_R - y_{offset}) & (7)
\end{aligned}$$

The variables t_L and t_R are the adjusted times of the left and right pmts, and E_L and E_R are the normalized pulse heights of the same tubes, respectively. Time calibration constants are adjusted so that the computed times ($\bar{t}$) for particles hitting any scintillator simultaneously are the same. The average time $\bar{t}$ is a position-independent determination of time of particle impact. The normalized pulse heights are determined so that normally incident particles at the center of each scintillator yield a value of $E_{L,R} = k = 10$ MeV. The quantity E is a position-independent measure of the energy deposited in the scintillator. The position in the scintillator, y, measured with respect to the center of the counter can be determined using the time difference between right and left PMTs and the effective propagation velocity of the light in the scintillator ($v_{eff} \approx 16$ cm/ns).

The constants necessary for these computations are the subject of several of the calibration steps and itemized below:

2.1 Definitions

1. Pedestals (P). The pedestal corresponds to the ADC channel when no data is present and is measured by taking the data with a pulser-trigger.

2. TDC calibration constants c$_0$, c$_1$, c$_2$ defined as:

$$t = = c_0 + c_1 T + c_2 T^2 \tag{8}$$

where T is the raw time in units of TDC channels and t is the converted time in ns.

3. Time-walk correction. Time-walk is an instrumental shift in the measured time using a leading-edge discriminator due to the finite rise time of the analog pulse. To correct for time-walk, we perform software corrections of the form

$$t_w = t - f_w\left(\frac{A-P}{Th}\right) + f_w\left(\frac{600}{Th}\right) \tag{9}$$

where Th is the channel corresponding to the leading-edge discriminator threshold of 20 mV (approximately 35 channels), A is the value of the ADC in channels, P is the pedestal, and $f_w(x)$ is the time-walk-correction function given below. (See section A.2 for additional details).

This parameterization has the desirable limit that t and t_w are equal for minimum-ionizing pulses which are set to be in ADC channel 600. The function $f_w(x)$ is a monotonically decreasing function of the pulse height A, since the measured time is late for a pulse with a finite rise time. Our parameterization has three fit parameters w_0, w_2, and w_3. They were determined for each PMT separately using the laser calibration system. Fits to data indicate that the time-walk correction is described by a function which first decreases rapidly as a power law then changes to a slow linear decrease:

$$\begin{aligned} f_w(x) &= \frac{w_2}{x^{w_3}} \quad \text{if } x < w_o \\ f_w(x) &= \frac{w_2}{{w_o}^{w_3}}(1+w_3) - \frac{w_2 w_3}{{w_o}^{w_3+1}}\, x \quad \text{if } x > w_o \end{aligned} \tag{10}$$

4. PMT status - Poor performing tubes are entered into the database or map so that the reconstruction code can properly interpret the data stream.

5. Left-Right delay constants (c_{LR}) - the relative time of PMTs on opposite ends of the same counter.

6. Counter-to-counter offsets (c_{c2c}) - relative time shifts of the measured times from counter to counter. These constants are also referred to as paddle-to-paddle constants, not to be confused with the next item.

7. Panel-to-panel (c_{p2p}) - The offsets between scintillator panels are currently all set to zero, and only preserved for historical compatibility.

8. Attenuation length (λ) - Measured attenuation length of each counter. The measured pulsed heights in each counter are given by

$$A_L - P = \frac{M0_L}{k} E_L \, e^{-y/\lambda} \tag{11}$$
$$A_R - P = \frac{M0_R}{k} E_R \, e^{y/\lambda} \tag{12}$$
$$\tag{13}$$

   Note that the software allows for different attenuation lengths for right and left pmts, but in practice we set them both to the same value. See section A.3 for additional details.

9. Effective velocity (v_{eff}): Measured propagation velocity of scintillator light in each counter. Relative to the time t_0 at the center of the counter, the propagation velocity is defined by the relations

$$t_L = t_0 + y/v_{eff} \tag{14}$$
$$t_R = t_0 - y/v_{eff} \tag{15}$$
$$\tag{16}$$

   As in the case of the attenuation length, the software allows for different propagation velocities for right and left pmts, but we set them equal to each other. See section A.3 for additional details.

10. Pulse height normalization ($\mathrm{M0}_L$ and $\mathrm{M0}_R$): the peak heights of minimum-ionizing particles normally incident at the center of the TOF counter for left and right PMTs, respectively.

11. Pulser normalization (c_{norm}): overall time scale for time measurements. This number (≈ 1) reflects a possible absolute scale shift of the pulser used during TDC calibration runs relative to the accelerator RF. Presently, deviations are typically less than 1%.

12. RF offset: delay time between RF signal and averaged event start time.

For more detailed definition of the constants and description of CLAS TOF system see Ref. [1], the postscript file is available on:

http://www.jlab.org/Hall-B/pubs/

2.2 Location of Constants

All calibration constants for the TOF system are written in two off-line Maps: SC_CALIBRATIONS.map and RF_OFFSETS.map. The Maps are located in: /group/clas/parms/Maps/. The entries in the Maps are arranged by run number, so reconstruction codes can correct data on run-by-run basis.

All information in the map-managing software can be obtained from the off-line web page: "http://www.cebaf.gov/ manak/packages/utilities/maputil/maputil.html", which links to the official "Hall B Off-Line" web page.

To work with the existing map, one can use the following commands which use executables built by the CLAS off-line software librarian:

◇ scan_map [MapName] (optional: -t);
◇ rem_map_arr -m[MapName] -s[subsystem] -i[itemname] -t[time/runNo];
◇ get_map_float -m[MapName] -s[subsystem] -i[itemname] -t[time/runNo] (optional: -l[arraylen]);
◇ get_map_int -m[MapName] -s[subsystem] -i[itemname] -t[time/runNo] (optional: -l[arraylen]);
◇ put_map_float -m[MapName] -s[subsystem] -i[itemname] -t[time/runNo];
◇ put_map_int -m[MapName] -s[subsystem] -i[itemname] -t[time/runNo];

Below we show an example of writing to the SC_CALIBRATIONS.map the TOF counter-to-counters delays arranged in a column in the file DELAY.dat:

```
$CLAS_BIN/put_map_float -m$CLAS_PARMS/Maps/SC_CALIBRATIONS.map -sdelta_T
-ipaddle2paddle -t1000 < DELAY.dat
```
[1]

[1]$CLAS_BIN presently points to the database. For map entries one should point to the proper release for a given data set, e.g. /group/clas/builds/release-2-5/bin/LinuxRH6.

3 Calibration Sequence

In Table 1 we give the order in which TOF calibrations should be performed and the requirements for every calibration step. Every step will be described in details in Sections 4 and 5. We want to emphasize that the order of calibration steps is important. We would like to mention here that the TOF calibration is an iterative procedure. Therefore constants from previous run periods may be acceptable starting values for monitoring the detector. However, to achieve a calibration set for production processing, all calibrations should be redone.

We note that in many cases, we describe two procedures to obtain the same set of constants. For example, the left-right timing offsets and energy loss calibrations can be iterated depending on the status of previous calibrations. One method needs only raw data, so it can be implemented early. The second requires tracking and therefore an initial reliable drift chamber calibration. The latter procedures are generally more accurate and automated, but can sometimes be used only in a second iteration of the calibration sequence.

Table 1: The order and requirements for TOF calibration. RF - accelerator Radio Frequency, TBT - Time-based tracking, SC - SC BOS bank.

Calibration step	Main requirements
Status	Raw Data
Pedestals	dedicated data
TDC calibration	dedicated data
Time-walk correction	laser data
Left-right Adjustment	Raw Data
Energy loss	left-right time alignment at SC level
Attenuation length	left-right time alignment at SC level
Effective velocity	good TBT and all constants above
RF parameters	good TBT and all constants above
Counter-to-counter delays	good TBT and all constants above
RF offsets	good TBT and all constants above
Geometric constants	survey data

4 TOF Calibration Procedures

4.1 Pedestals

The pedestals and TDC calibration constants are obtained by analyzing the data taken with dedicated DAQ configurations during the experimental run [4].

a) The pedestals are analyzed using PEDMAN on-line utility [4]. The results should be loaded in TOF ADCs with sparcification threshold 40. The table of pedestals is archived on CLON01 in: /usr/local/clas/parms/pedman/Tfiles. One can easily put the numbers in the Map by editing the file (removing all comments, checking that the pedestals are within a resonable range), and using the UNIX command "awk" as shown in the example below (the second column in the PED.dat will be written into the Map) :

awk '{print$N}' PED.dat | $CLAS_BIN/put_map_float
-m$CLAS_PARMS/Maps/SC_CALIBRATIONS.map -spedestals -ileft -tRUNnumber,
where PED.dat is the name of the file from which the text has been taken, and 'print$N' specifies the Nth column number of the file. The '-t' option specifies the run number. The column number N must be specified to correspond to the particular subsystem and item as follows:

Table 2: Definition of columns in PED.dat file.

Column Number (N)	Definition	Subsystem (-s option)	Item (-i option)
3	left pedestal	-spedestals	-ileft
5	right pedestal	-spedestals	-iright
4	left pedestal uncertainty	-spedu	-ileft
6	left pedestal uncertainty	-spedu	-iright

4.2 TDC Calibrations

4.2.1 Taking the pulser data

This paragraph is just an introduction to give you an idea of how this works. The real documentation for this belongs in the "Procedures: DAQ" binder in the counting house and on the web. The channel to time calibrations of all the CLAS TDCs, except for drift chamber TDCs, is performed using a special pulser run. To take this data, configure the DAQ for clas_pulser and start a run. These runs take approximately 40 minutes and accumulate 40000 events. The pulser stops automatically after all events are taken, at which point the operator can end the run and reconfigure the DAQ. A file will be written to the raid disks (/raid/stage_in) with a name of the form clas_tdc_012345.A00, where 12345 is the run number. This file will appear on the active raid drive, and gets transferred to the silo automatically. To retrieve the file from the silo see the jget command[2] on the CUE machines.

4.2.2 Structure of pulser data

The data is accumulated by pulsing all the TDC channels of all the crates simultaneously. For each step, a group of 50 pulses is sent with a fixed delay between the TDC start and stop. For each subsequent step the delay time is increased by approximately 2ns (the exact time is 2.139 ns). This process is repeated 200 times, after which the whole thing is repeated with a different bit-mask for the TDCs in the SC crate. There are 4 different masks $0xEEEE$, $0xDDDD$, $0xBBBB$, $0x7777$. Note that all other crates are unmasked, and thus take the same data 4 times. This pattern of events should be 100% predictable, however, various DAQ problems can interfere. One problem is events other than pulser events causing dead-time of the DAQ, thus throwing off the sequence. (Scalar events should have been turned off automatically by the DAQ configuration.) Also note that the raw data banks are readout along with the BOS banks.

4.2.3 Computer program to analyze pulser runs

The program, which is used to analyze the clas_tdc_012345.A00 data file, is found in packages/utilities/TDC_cal, which contains the source code and a standard makefile. It compiles into an executable called TDC_cal. A recently compiled executable can usually be found in holtrop/codes/clas/bin/SunOS/TDC_cal. The program has a built-in help screen which is printed out using the -h option. (Type ./TDC_cal -h at the command line.) Note that the program needs access to the current MAP files, so the $CLAS_PARMS variable needs to be defined.

4.2.4 Quick overview of analysis sequence

A typical analysis sequence has the following steps, each of which will be explained in more detail below. At this point the analysis is still somewhat cumbersome; future improvements of the program may improve this. This analysis can be done for all crates simultaneously or for a single crate using the -only option:

1. Check the correctness of the data sequence using the -spy option.
2. Analyze the run to check data quality and make histograms using the -hist option.
3. Check the histograms, perhaps repeat step 1.
4. Re-analyze the data and put results in the map using the -map option.

[2]Use ‘man jget’ for details.

4.2.5 Step 1: Checking data sequence

The data sequence is checked by printing out each TDC reading for one single channel of one TDC in one crate. The output will then reveal whether the program has the correct "boundaries". To print out a sequence you use the -spy option. An example command line would be:

```
TDC_cal -only 13 -spy 13.12.4 clas_fc_tdc_017840.A00
```

Which gives the following output:

```
Process only crate 13 (1)
Spying on crate 13 slot 12 channel 4
Retrieving ROC13.tab from /group/clas/parms/TT
Running all of clas_fc_tdc_017840.A00
ERROR: NO RAW DATA for crate 13 in event 1 !
ERROR: NO RAW DATA for crate 13 in event 2 !
Data Out of Sequence for : 13.13.55 = 290 pattern: eeee – I'll stop complaining.
Data Out of Sequence for : 13.17.09 = 288 pattern: eeee – I'll stop complaining.
Data Out of Sequence for : 13.15.33 = 41 pattern: eeee – I'll stop complaining.
13.12.04 (peak no: 0): TDC: 202 Expect: 70.59 Mean: 202.00
13.12.04 (peak no: 0): TDC: 204 Expect: 70.59 Mean: 203.00
13.12.04 (peak no: 0): TDC: 205 Expect: 70.59 Mean: 203.67
13.12.04 (peak no: 0): TDC: 204 Expect: 70.59 Mean: 203.75
... many lines deleted ...
13.12.04 (peak no: 0): TDC: 206 Expect: 70.59 Mean: 202.51
13.12.04 (peak no: 0): TDC: 202 Expect: 70.59 Mean: 202.50
13.12.04 (peak no: 1): TDC: 204 Expect: 72.73 Mean: 204.00
13.12.04 (peak no: 1): TDC: 199 Expect: 72.73 Mean: 201.50
13.12.04 (peak no: 1): TDC: 245 Expect: 72.73 Mean: 216.00
13.12.04 (peak no: 1): TDC: 244 Expect: 72.73 Mean: 223.00
```

What you see in this output is that the software has not correctly identified the start of the read data, the peak number increases (from 0 to 1) but the TDC values does not increase until 2 events later. To correct for this you can use the "-skew" switch. In this case you want to skew two events forward so you use "-skew 2". To skew 3 events backwards you would use "-skew -3".

If the data has events missing, you may wish to try the "-autojump" feature. However, care must be taken with this feature, it often makes things worse rather than better. If possible you can also take another run.

4.2.6 Step 2: Analyzing the run.

Now that the amount of skew is determined, you can analyze the whole run and make histograms. An example command line would be:
TDC_cal -only 13 clas_fc_tdc_017840.A00 -hist
(The code will run for some time you can check progress using the "-paddle" switch)
This will create histograms, and produce a text file: $CLAS_PARMS/initfiles/TDC_RC13.cal with all the constants, but it will NOT put information in the map. Note that only this program uses the text files produced, and that it keeps a record of previous calibrations. This allows one to track the changes of TDC constants over time. You can skip writing this file by supplying the "-notdc" switch. Note that the program is rather verbose. It will complain if the masking bits were not working correctly, if there is no data for a TDC channel (because it is not in use), and if the chi-squared of the fit is high.

4.2.7 Step 3: Checking histograms.

You will now find a number of files in your directory with names like: TDCcal.13.12.plt, which are "hvplot" files. These text files contain 128 plots each, and can be viewed with the "hvplot" program (see /home/heddle/Hv/demos/hvplot/SunOS/hvplot. At some later point I will change this to a ROOT based plotting interface, since "hvplot" is too slow for normal use.) To few these files type:
/home/heddle/Hv/demos/hvplot/SunOS/hvplot TDCcal.13.12.plt
at the command line, and have some patience.

For each channel in the TDC you will see two plots: the peaks of the data, and fit to the positions of these peaks. The fit also has the chi-squared and fit parameters plotted. If a fit looks too far off from the data, the chisquared is high, or the data points are very large, there is either a problem with the TDC, a problem with the data, or a problem with the fitting procedure. In that case, try "-spy" on the channel, and if needed contact an expert.

4.2.8 Step 4: Updating the MAP.

If you are satisfied with the results of the program, you need to run it once more with the "-map" option, so that the new values are entered into the MAP database. The standard file that is used for the TOF is $CLAS_PARMS/Maps/SC_CALIBRATIONS.map. An alternate file can be specified with the "-mapfile" option. Note that the older version of the map will be backed up in a file SC_CALIBRATIONS.map.∼ (emacs backup convention) unless the "-nobackup" switch is specified.

When you run the code with the "-map" option, it will verify the old map values (usually for a lower run number or a previous calibration attempt) with the ones just calculated. It they differ by too much, the user is prompted to confirm entry into the MAP database.

4.3 Counter status

When analysing data more useful information can be extracted by acknowledging that one side of a scintillator counter is dead, than by leaving the reconstruction programs to seek for correlated information for both left and right. Therefore, before proceeding to the counter by counter calibration, it is important to check and correctly include the counter status in the map ($CLAS_PARMS/Maps/SC_CALIBRATIONS.map, subsystem: status, items: left and right). For the counter status we use the following convention in off-line Map:

"1" - no ADC;
"2" - no TDC;
"3" - no ADC, no TDC (PMT is dead);
"5" - any other reconstruction problem;

To identify the raw (ADC,TDC) response (SC bank level) of each PMT you can use the program tof_calib. The histogram building program is on version control in :
/packages/utilities/sc_calib/atten/tof_calib.c
The usage of the code can be obtained by giving the argument "-h":

```
Usage: tof_calib -s# [-n#] -[A] -[c] -[R] -[G] [-o<outputfile>] inputfile
 Options:
    -s              Sector number #
    -n[#]           Process only # of events
    -o[filename]    HBOOK output file name
    -c              Checkout mode for left-right offsets
    -A              Attenuation length histos
    -R              Raw data histos
    -G              Geometric Mean histos
    -h              Print this message
```

The "tof_calib" program does not require any reconstruction and can be run on raw data. To determine the status of the counters one should run the program with the map entries for the status all set to zero, so that historical failures do not bias the current run. The code outputs only histograms for one sector at a time. With the -R switch, the program produces a 2-D histogram of the raw TDC and ADC signals versus counter number. For a given sector N these histogram have numbers 10*N+1 for ADC left, 10*N+2 for ADC right, 10*N+3 for TDC left, 10*N+4 for TDC right. By slicing and projecting these histograms you can identify the counters where either or both ADCs or TDCs were not working. An example of the histograms used in analyzing one sector can be seen in Figure 1.

To monitor more complicated reconstruction problems user can use "sc_mon" program, which is under version control in /packages/utilities/sc_mon/. The occupancy plots (histogram numbers 3001-3006, 3011-3016, 3021-3026 and 3031-3036) give information about the raw signals without detailing the ADC and TDC signals. The results of track matching are best checked using the reconstructed pion vertex times (histograms numbers 501-506) and reconstructed masses (histograms numbers 401-406) versus scintillator number. These plots are particularly useful to check proper operation of the counters. The status of reconstructed hits from the SCR bank versus scintillator number is given in histogram numbers 351-355. These are also useful to determine which information was used for each hit (left, right, ADC or TDC).

4.4 Left-Right PMT's time alignment

Establishing the left-right signals time offsets ensures good identification of the hit position in the scintillator paddle. In this phase we set the rough values of these offsets. Together with calibrating the effective velocity of the light in the scintillator (described in Section 4.7), the fine tuning of these time offsets is done. We use the hbooks created with the program **tof_calib** using the '-c' switch. The histograms N+6 (where N is the sector number) show the distribution of TDC left -TDC right with existing calibration offsets. If the offsets are not correct the distribution is not symmetric (see figure 2 middle). The edges of the X-projection for each counter ($edge_L$ and $edge_R$) should be symmetric with respect to zero. These offsets can be determined using the packages/utilities/sc_calib/atten/tdc_lr.kumac kumac. The edges of the distribution as determined by the kumac for one example are shown in Figure 3. The value of left-right offset is then determined as

$$\Delta t = (edge_L + edge_R)/v_{eff} \tag{17}$$

where v_{eff} is effective velocity in scintillator material. For this step, we use the nominal value of 16 cm/ns. [3] This value of Δt must be added to the effective value of left-right constant in the Map for this run: SC_CALIBRATIONS.map -sdelta_T -ileft_right -t(run). Once this calibration is done the left right alignment should look like the lower plot in Fig. 2.

4.5 Energy loss and attenuation length calibration

The next step is to calibrate $\delta E/\delta x$ in scintillator material and determine the attenuation length of each counter. This should be done in three steps:

1. Measuring geometric mean of the MIP peak position for every counter. Described in a) and b) below are two procedures which can be used for this purpose. The first does not use previous calibrations, but requires the user to make decisions based on the data set. The second assumes that a reasonable timing calibration is already in place and can be used to select pions for the energy calibration. The second option is automated and preferred when it can be used.

[3] Final adjustment of the position measurement from left and right pmts is obtained by fitting for v_{eff} and y_{offset} during the calibration of the effective velocity.

a) Before good quality processed (cooked) data becomes available, approximate energy loss calibrations can be accomplished from the raw data. For this use the histogram building program which is on version control in :

/packages/utilities/sc_calib/atten/tof_calib.c

The usage of the code is described in subsection 4.3. To obtain the positions for the geometric means of MIP in TOF counters, run the program with the "-G" switch on. Since the ratio of MIPs and highly ionizing particles depends heavily on experimental conditions (e.g. beam energy, triggers, etc.), the identification of the MIP peaks must be determined judiciously by the user. Gaussian fits can be used in many cases, but are not always satisfactory. The kumac file packages/utilities/sc_calib/atten/mip.kumac can be used to loop through and fit the distributions and produce a file with the necessary information (288 lines, one per counter) arranged in three columns: counter id ($100\times sector + scint$) (int), geometric mean (float) and sigma of geometric mean (float). To invoke the kumac use the PAW command 'exe mip [pre]' where the input histogram file names are '[PRE][SECTOR].hbook,' where [PRE] is usually the run number and [SECTOR] is the sector number. When this procedure is complete the fitted histograms should be checked, since the Gaussian function does not always yield a satisfactory fit to the data.

b) Finding the geometric mean of MIP peak position calibration using cooked data. This procedure uses loose timing cuts to select pions which are used for the energy loss calibration. Therefore, preliminary calibrations must already be available, but the procedure is automatic and more accurate than the one described in a). The programs are under version control in /packages/utilities/sc_calib/gmean/

The following executables need to be made:

make gmean_cooked (builds histograms).
make hscan_means.exe (converts data from HBOOK to ASCII format);
make min_means_main.exe (fitting routine);

The usage of gmean_cooked can be obtained by executing the program with switch "-h". The output of this program is a histogram file[4] which is used as input to "hscan_means." First edit "hscan_means1" to read the correct histogram file, and then edit "min_means_main1" to use the output of hscan_means.exe. Then execute the following scripts to fit the histograms:

hscan_means1
min_means_main1

The first program runs quickly, the second takes a few minutes. Each outputs many messages, indicating scintillators which are dead, or lack of statistics to produce a good fit. Histograms with poor statistics are indicated by a $\chi^2 \geq 10$. For these cases, the program substitutes a default value of 600 ± 600. The latter program exits "normally" with a floating point exception flag which should be ignored. The result of these operations are several files, but two of importance. The first is "means.parm" which contains the histogram number, fitted peak, peak error, χ^2 for fit and the MINUIT status number. Means.parm can be

[4]An example of such a file "means.hbook" is saved under CVS and can be used for testing.

used for updating map entries (see below). The second file is "min_means.kumac". This file can be used to check the quality of the fits. Run PAW, open the original input histogram file, and execute this KUMAC file. All fitted histograms will be shown (24 to a page) with corresponding fits. They are also saved in "means.ps" for printing. An example of a single fit is shown in Fig. 4.

2. To start the $\delta E/\delta x$ and attenuation length calibration one should check out from CVS the directory /packages/utilities/sc_calib/atten/, make the executables:

```
make tof_calib (builds histograms).
make hscan_atten (converts data from HBOOK to ASCII format);
make min_atten (fitting program);
```

To build the set of histograms required for this calibration, use "tof_calib" and the switch "-A" (about 500,000 events are needed for good statistics):

```
tof_calib -n500000 -s1 -A -o[output file ] [input file]
```

The switch "s1" creates the histograms for sector one, so the procedure must be repeated six times, once for each sector. (The program is run once per sector to reduce memory requirements which are considerable for two dimensional histograms.) When the histograms are built one should edit the UNIX script file "atten1" and set up relevant links. The fitted parameters are stored in "min_parm," which a soft link to a file specified in "atten1." This script runs both codes: "hscan_atten" and "min_atten", and creates PAW kumac and parameters output files. To visually verify that the calibration was successful, run PAW and input the hbook file created by tof_calib. The kumac "atten.kumac" will display all fits on the appropriate scatter plot (24 to a page). For an example of one of the plots see Fig. 3. The plots are also output to atten.ps for printing. After repeating the procedure for all six sectors, the six .parm files should be merged into a single file (called "atten.parm" for the purpose of illustration) for further processing.

3. To calculate final values of the constants user should make executable of the packages/utilities/sc_calib/atten/adc_const.f:

```
make adc_const
```

This code combines the results the MIP peaks ("means.parm") and measured attenuation ("atten.parm"), described in steps 1 and 2. The user is prompted for the file-names of geometric mean and attenuation length parameters and output file-names. It will create the output in the format (columns, from left to right):

1. counter # - from 1 to 288,
2. NMIP_ADC(left) - gain balancing constant for all left PMTs;
3. NMIP_ADCu(left) - uncertainty for gain balancing constant for all left PMTs;
4. NMIP_ADC(right) - gain balancing constant for all left PMTs;
5. NMIP_ADCu(right) - uncertainty for gain balancing constant for all left PMTs;
6. atten_length - attenuation length for all 288 PMTs;

7. atten_u - uncertainty of attenuation length for all 288 PMTs;

These constants and uncertainties can be put in the map using “awk” command, showed as example in subsection 4.1. In Fig. 5 we show an example of correctly calibrated attenuation length and energy loss constants for counter 15 in sector 3: the distribution should be symmetric about the origin [1].

Once all these constants are in the map, we recommend the calibration be checked by using the programs described in Section 6. For example, in Fig. 6 we show the results of the calibrated energy loss versus momentum for all particles. One can clearly see the proton and pion bands. It is very important to have a good independent pion identification using dE/dx to select pions used in the counter-to-counter delay calibration below.

4.6 Time-walk correction constants

4.6.1 Data Taking

The time-walk calibration of the TOF is based on special laser data. Presently, the procedure of laser data taking is not completely automated, therefore it is left to the care of TOF experts to provide a good laser data set when the calibration is necessary.

4.6.2 Analysis of Laser Data

The software for obtaining the time-walk correction parameters is located in the cvs repository at:

/packages/utilities/sc_calib/time_walk

Three main programs are used: make_tw_histos makes the histograms needed for the fits, hscan converts the histograms into text files readable by minuit, and min_main steers minuit to obtain the fit parameters.

The program make_tw_histos operates on one side of a sector at a time due to the large size of the two-dimensional histograms it makes. The program displays the following help message if the -h option is used:

```
Usage: make_tw_histos [...options...] file1 [file2] etc....
 Options:
        -n[#]         Process only # number of events
        -o[filename]  Histogram output file
        -s[sector]    Sector number
        -t[offset]    Time offset to get data in range
        -l            Produce left histograms
        -r            Produce right histograms
        -h            Print this message.
        -c            Lower cut on laser diode time
        -C            Upper cut on laser diode time
```

The laser diode for each laser provides the reference time. Because the time range of the histograms is fixed, sometimes the -t offset option is needed to be able to see the data. The -c and -C options are sometimes needed because the quality of the laser diode timing signal can be poor; they allow the user to pick a range in time for which the laser diode timing is usable.

The following is a specific example using the above program. It produces the histograms for the left pmts in sector 3 of the North Clam Shell using the laser calibration file located in /work/clas/disk2/toflaser_020366.A00. The arbitrary time offset (20 ns) is determined to center the data in the histogram window. One should check that data in all plots fall within the defined range.

```
    make_tw_histos -n2000 -o/work/clas/disk2/tw20366_ns3_l.hbook
-s3 -l -t20 /work/clas/disk2/toflaser_020366.A00
```

The name of the output histogram file is arbitrary, but one should make sure it completely specifies the data it contains (e.g. sector number, right or left side and carriage location). The location of all detectors is given in Table 3.

Table 3: Location of detectors on the mechanical support structures.

Mechanical Support Frame	Sectors	Scintillator Number
Forward Carriage	All Sectors (1-6)	1-23
North Clam Shell	Sectors 3, 4, 5	24-42
South Clam Shell	Sectors 6, 1, 2	24-42
Space Frame	All Sectors (1-6)	43-48

The hscan program reads input from a logical file called "hscan_input" and writes to a file called "hscan_data". To point "hscan_input" to your data and "hscan_data" to your desired output file, define symbolic links using the UNIX "ln" command:

```
ln -s [histogram file name] hscan_input
ln -s [text file name] hscan_data
```

Similarly, the min_main program reads input from "min_input" and outputs the parameters of the fits to "min_parm" (extention .parm), a kumac for visually checking the quality of the fits to "min_kumac" (extension .kumac), and the minuit output (containing more information on the results of the fits, including error matrices) to "min_output".

The output parameters are in the .parm file. There must be two such files for each detector, corresponding to right and left pmts. After they have been generated, it is necessary to check them for

- missing detectors: if a detector is missing the corresponding row on the .parm file will be missing. Copy the data of the next (or previous) detector into this row.
- total number of counters: each .parm will contain a specific number of rows depending on the support structure of CLAS; e.g. for Space Frame each .parm will contain 6 rows corresponding to the detectors 43 to 48.

Once files have been produced for all six sectors, the files for each side should be concatenated into a single file. The result will be two files with 288 rows, one for the right and the other for the left. Each row contains the pmt id and fitted time-walk parameters. We note that the offset (first fitted parameter) is arbitrary. It is necessary for fitting the data, but not used during analysis. The fitted parameters were obtained for the ADC range between 0 and 8100 counts and typical values are w_o ~50 (depends on the PMT), w_2 ~15 ns and w_3 ~0.07, with a strong correlation between the last two parameters. The columns of the file contain the following information:

Table 4: Definition of columns in .parm time-walk paramter file. The item (left/right) depends on which file is used. See Eqn (2) for definitions.

Column Number	Definition	Map Subsystem	Map Item
1	100*sector + scint #	unused	
2	offset	unused	
3	w_2	WALK1	left/right
4	w_3	WALK2	left/right
5	w_0	WALK_A0	left/right
6	offset error	unused	
7	w_2 error	WALK1u	left/right
8	w_3 error	WALK2u	left/right
9	w_0 error	WALK0u	left/right
10	Chisquare for Fit	unused	
11	Fit Status	unused	

4.7 Effective velocity calibration

The position of the hit along the length of the scintillator, y, can be determined using the timing information in right and left tubes using Eq. 7. Using a sample of fitted tracks, we can determine the position y from tracking. Therefore, a fit to the time difference between right and left tubes versus y can be used to determine the constants y_{offset} and v_{eff} for each scintillator counter. This procedure is accomplished using the program veff described next.

To measure the effective velocity, we use cooked data since fairly good quality tracking is required. The software for building and fitting the histograms needed to measure the average effective velocity for each paddle is in the cvs repository at:

/packages/utilities/sc_calib/veff

The -h option prints out the following help message:

```
Usage: veff_calib [-n] [-o] file1 [file2] etc....
 Options:
         -n[#]           Process only # number of events
         -o[filename]    Histogram output file
         -s[#]           Sector number (1-6)
         -i              Batch mode, no counter
         -h              Print this message.
```

To fit the histograms produced by this program, the hscan_veff program needs to be run to create a text input file readable by minuit. The program min_veff performs the fits and outputs a parameter file with six columns: histogram id, time offset, effective velocity (cm/ns), error on the time offset, error on the effective velocity, χ^2/ndf of the fit, and minuit fit status. The effective velocities can be placed in the map using the awk unix command.

If the fitting procedure is biased due to background hits in the distribution, then the variable *wmin* in the program hscan_veff.f can be increased (from its default value of 2) to optimize background rejection.

4.8 RF offset calibration

To have an acceptable TOF calibration for a long run and a large data sample, the RF offset must be adjusted throughout the run period. At the beginning of a new run period and beam energy, the RF phase changes so new offsets must be determined.

We first describe the parameters for the RF calibration used for all run periods. To be definite, we use the electron-beam program rf_mon. However, the procedures to obtain the parameters rely on different programs for electron and photon beams, which are described in sections a) an b) following. The RF calibration subdivides the RF time into four non-overlapping intervals. The RF correction is then defined by a third degree polynomial over each interval (F1, F2, F3 and F4). The "good" RF for a given run uses either RF1 or RF2, but not the other. The choice of RF1 or RF2 is selected using the calibration status word. All calibration constants are stored in RF_OFFSETS.map in CLAS_PARMS/Map area.
There are two steps in RF offset calibration: building "good" RF from RF1 or RF2, and the measurement of the RF offset relative to the "good" RF.

- Building the "good" RF:

- set the limits for all functions (low and high) arbitrary, but they should cover all RF time range (100 ns) and not overlap;
- set all parameters of F1, F2, F3, F4 to zero (p0, p1, p2 and p3), choose RF1 or RF2 filling the "status" ("1" for RF1, "2" for RF2). Goal of this is to obtain RF time without any correction ;
- run rf_mon and determine regions for every function (non-overlapping), set functions limits properly in the map, fit the distributions for every function and put the value of parameters in the map;

• Measurement of RF offset. This step will need to be repeated after the counter-to-counter calibration is performed (next step):
- run rf_mon, verify that "good" RF is built properly, measure RF offset.

a. Electron beam data code. The code used for the RF offset calibration is "rf_mon.c". It is under CVS version control in : /packages/utilities/rf_mon/rf_mon.c

The program calculates the RF correction and the target RF correction on an event-by-event basis, and creates two output files: HBOOK and text. The HBOOK file includes ten histograms. The most important ones are the RF correction for the TOF vertex time together with the correction on target length (ID 102) and RF correction versus "good" RF (ID 103), Fig. 7. The RF correction should be distributed around zero and there should be no systematic dependence on the RF time. The program fits the RF correction histogram with a Gaussian. The mean value of the distribution is RF offset. The text output file contains the run number and the RF offset for the run. The usage of the program can be obtained by using command line switch "-h":

```
Usage: rf_mon [-n#] [-s] [-o<outputfile>] inputfile
   Options:
      -n[#]          Process only # of events
      -s             Silent mode
      -o[filename]   HBOOK output file name [default: run#_rf.hbk]
                     [text output is setup by default: run#_rf.txt]
      -a             Append text output to rf_mon.txt file
      -R             Rebuild PID banks;
      -h             Print this message.
```

The new calculated RF offset should be added to the previous map entry, which was effectively in the map during running the code, and the resulting offset should be written in the RF_OFFSETS.map:

```
put_map_float -mRF_OFFSETS.map -soffset -ivalue -t1000 < offset.dat
```

b. Photon beam data code. This step is performed during the tagger calibration before the TOF calibration is started.

4.9 Counter-to-counter delay calibration

4.9.1 Electron-beam running

Calibration code: This program should be used when TOF counter are calibrated from scratch. In order to calibrate the time offsets for different paddles one needs to run the program ***p2p_delay_el*** in ***packages/utilities/sc_calib/p2p_delay_el/*** on CVS. In order to get a copy of the program one must simply type:

cvs checkout p2p_delay_el

then change directory to p2p_delay_el by ***cd p2p_delay_el*** and type in:

setenv PHY_DIR 'pwd'

setup root/2.23 (there is no need for this if it has already be set up)

make

which will create an executable ***p2p_delay_el*** in the ***bin/SunOS*** subdirectory.

The calibration program runs with a configuration file. An example of that file calib.conf comes with the source in the main directory of the the package. The parameters that should be set in the config file are the following:

set inputfile = FileName

(name of the file containing banks needed for calibrations)

set rootfile = FileName (name of the histogram file in Root format)

set events = Number1 (number of events from each input file)

set update = Number2 (number of events after which the program reports number of events processed).

There may be more than one cooked file. The program will process **Number1** events from each file.

To run the program one should go to bin/SunOS subdirectory and type:

p2p_delay_el -cConfigFileName

Notice that there is no space between -c and ConfigFileName, where ConfigFileName is the name of the config file mentioned above. The offsets are recorded in the constants.dat file in plain text format. The constants that should go into the map are in column "Delay4Map". One must make sure that constants in the map has not been changed since the time when the raw data was processed for calibration. Also it is very important to have CLAS_PARMS variable set to the same value as when cooking the files for calibration.

Monitoring code: This code was created for fast monitoring of the most important results of TOF calibration and for being executed during data "cooking" (for detailed monitoring see Section 7). The code is located under CVS in:

/packages/utilities/sc_calib/sc_delay_el/sc_delay_el.c

4.9.2 Photon-beam running

Because the source for the start time for the event is different for photon beam running than for electron beam running, the calibration software is different. For photon running the counter-to-counter calibration software is in /packages/utilities/sc_calib/p2p_delay_ph/ Two C programs are involved in the calibration procedure: hist_offsets makes the histograms needed for determining the time offsets and fit_offsets fits these histograms. Command line options for hist_offsets are listed below. You can obtain this help information with the -h option.

```
Usage: hist_offsets [-M] file1 [file2] etc....
 Options:
        -M[#]           Process only # number of events
        -o[hbook_file]  HBOOK file name
        -R              Regenerate the SC, CC and TBID banks
        -h              Print this messages.
```

Command line options for fit_offsets are listed below.

```
Usage: fit_offsets -p[parmfile] -r[runno] hbookfile
 Options:
        -p[parmfile]    Parameter file name
        -r[runno]       Current run number
        -h              Print this message.
```

The -r command line argument is required because the offsets calculated by the fitting routines are relative to what is already present in the map. The program creates a parameter file *parmfile* containing the sector, the scintillator number, the mean time and error on the mean, the sigma and error on the sigma, the new offset and its error, and the χ^2 of the fit. Similar information is provided for the six start counter tubes, but usage of these offsets is not currently recommended. In order to fascilitate putting the numbers in the map, the program also creates two other files called *parmfile*.tof and *parmfile*.start. The file *parmfile*.tof consists of two columns of 288 numbers: the first column is the set of counter-to-counter constants that should be put in subsystem delta_T item paddle2paddle in the map, and the second column consists of the errors (as determined by the fits) on these numbers to be put in subsystem delta_Tu item paddle2paddle. The fit_offsets code forces the average of all the counter-to-counter offsets to be zero. This means that the relative timing between the tagger and the time-of-flight needs to be checked; the tag2tof calibration constant relating the two detector systems may need to be changed. The tag2tof number is stored in the TAG_CALIB map. The program photon_mon (described below) provides histogram 109 for checking this number.

5 Geometry constants

The survey data of the carriages and TOF counters is used to determine the positions of each TOF scintillator relative every other counter and to the CLAS target. A detailed description of how to manipulate the survey data into the geometry map constants can be found in Ref. [5].

6 Monitoring the quality of calibration

The monitoring of the TOF scintillators and raw data is accomplished using the program "sc_mon," which produces an HBOOK output file with many diagnostic histograms. For monitoring the quality of the TOF calibration, we use "pid_mon." It requires reconstructed time-based tracks. To obtain help on using "pid_mon" and "sc_mon" programs use the "-h" argument.

The program "photon_mon" is intended for monitoring the quality of the reconstruction for photon-beam running. The source code is in the cvs repository at:

/packages/utilities/photon_mon/

Further monitoring of the data with electron beams can be accomplished with "sc_delay_el." The source for "sc_delay_el" is in the cvs repository at:

/packages/utilities/sc_calib/sc_delay_el/

7 Acknowledgments.

Many thanks to everyone that has contributed in development of TOF calibration procedure and got it to this point.

A Time-of-flight reconstruction

While the Time-of-flight reconstruction is not the central to the issue of calibration, these procedures are the ones that utilize the constants and as such are of interest to show how the constants are used. For this reason, we include here a few relevant sections of Simon Taylor's thesis for reference.

The Time-of-flight reconstruction code reports the time and energy deposition for each hit in the scintillator array. The raw ADC and TDC information for each struck scintillator in each event is stored in the SC bank. Each "hit" consists of a number identifying the scintillator and the ADC and TDC information for both the left side and the right side of the paddle. The reconstruction algorithm for the time-of-flight comprises four main steps.

A.1 Initialization

At the beginning of the run the calibration constants are read from the Map database and stored in global memory. The map file is SC_CALIBRATIONS.map and is located in $CLAS_PARMS/Maps.

A.2 Construction of the SC1 banks

The SC1 bank contains those hits in the SC bank that have valid timing information. Those hits that have no useful TDC values on either side of the paddle are discarded.

The data remains separated between left and right tubes for each hit but the TDC values are converted to time units (ns) and the ADC values are converted to energy units (MeV). The conversion for the ADC values is

$$E = \frac{10(A-P)}{A_{NMIP}}\text{MeV}, \tag{18}$$

where A is the ADC value of the hit in channels, P is the pedestal, and A_{NMIP} is the pedestal-subtracted ADC value corresponding to a normally incident minimum ionizing particle (NMIP), which deposits 10 MeV in the scintillator material. The TDC values are converted to nanoseconds and corrected for slewing according to

$$T = c_0 + c_1 t + c_2 t^2 + t_w(A, P), \tag{19}$$

where t is the TDC value in channels, c_0 is a constant that relates the timing of each of the time-of-flight channels to each other, c_1 is the conversion factor between TDC channels and nanoseconds, c_2 is a small constant that accounts for the nonlinearity of the TDC and t_w is a correction dependent on the pulse height:

$$t_w = f_w\left(\frac{600}{V_T}\right) - f_w\left(\frac{A-P}{V_T}\right); \tag{20}$$

$$f_w(x) = \begin{cases} \frac{w_2}{x^{w_3}} & \text{if } x < w_o \\ \frac{w_2}{{w_o}^{w_3}}\left(1+w_3\right) - \frac{w_2 w_3}{{w_o}^{w_3+1}} & \text{if } x > w_o \end{cases}$$

V_T is the discriminator threshold in ADC channels. The time-walk correction is set to zero for ADC zeroes or overflows or TDC zeroes or overflows. The time T is set to the error value of 100000.0 for TDC overflows or -100000.0 for TDC underflows.

A.3 Construction of the SCR banks

During the construction of the SCR bank, information from both sides of each paddle with an entry in the SC1 bank is combined to form the best timing, energy and position information for the hit. If good timing values for both sides of a given struck scintillator are found, the time of the hit is determined by

$$T = \frac{T_L + T_R}{2} - y\frac{v_R - v_L}{2v_L v_R}, \tag{21}$$

where T_L (T_R) is the walk-corrected time in ns for the left (right) tube and v_L (v_R) is the speed of light propagation toward the left (right) tube. The position of the hit along the scintillator relative to the center of the paddle is given by

$$y = \frac{v_L v_R (T_L - T_R)}{v_L + v_R}. \tag{22}$$

The propagation speeds v_L and v_R could in principle be different, in which case the time of the hit would depend on the position of the hit along the bar, but in practice a single average value is measured and the position dependence cancels. If both left and right ADCs are present and within range for a given paddle, the energy deposition is determined by

$$E = \sqrt{E_L E_R e^{y \frac{\lambda_R - \lambda_L}{\lambda_L \lambda_R}}}, \tag{23}$$

where E_L and E_R are the pedestal-corrected ADC values in units of MeV from the SC1 bank, y is the position along the length of the paddle relative to the center of the paddle, and λ_R (λ_L) is the attenuation length for light propagation to the right (left) tube. In the case where only the left ADC is present and valid, the energy deposition is given by

$$E = E_L e^{-y/\lambda_L}. \tag{24}$$

In the case where only the left TDC is valid, the time is given by

$$T = T_L - \frac{y}{v_R}. \tag{25}$$

Analogous equations hold for the case where only the right ADC or TDC is valid. By default, the code tries to use tracking information to find y. If this fails, and both ADCs are present for the given hit, then the position can be calculated rather crudely according to:

$$y = \frac{\lambda_L \lambda_R}{\lambda_L + \lambda_R} \ln\left(\frac{E_L}{E_R}\right). \tag{26}$$

Errors are computed for the time, the energy deposition, and the y-position of each hit.

A status word is set according to table 5 depending on the quality of the hit, 15 being best, 1 and 4 being worst.

A.4 Construction of the SCRC banks

The SCRC bank takes individual hits in the SCR bank and looks for clusters in which adjacent hits arise from a single particle depositing energy in two scintillators. The clusters are chosen based on overlap in space and time. If the difference in times or positions is less than three times the uncertainties in the differences, then the two hits in the adjacent counters are considered to be one cluster. If three consecutive counters fire in a given event, then the adjacent pair that are better matched in space and time are clusterized; the other scintillator information enters the SCRC bank as a single hit. If a cluster is found, the

Value	Meaning
1	Left TDC present; no right tube data.
2	Left ADC present; hit discarded.
3	Left ADC and TDC present; no right tube data.
4	Right TDC present; no left tube data.
5	Both TDCs present, no ADC information
6	Left ADC, right TDC present
7	Left ADC and TDC, right TDC present but no right ADC
8	Right ADC present; hit discarded.
9	Left TDC, right ADC present
10	Both ADCs present; no TDC information; hit discarded.
11	Both ADCs present, left TDC present, no right TDC
12	Right TDC and ADC only; no left tube data.
13	Both TDCs present; right ADC present; left ADC missing.
14	Both ADCs present, right TDC present; left TDC missing.
15	Complete set: left TDC, ADC; right TDC, ADC

Table 5: SCR status word definition.

energies of the two hits are added together to form the "cluster energy" and the time is computed from an energy-weighted average of the two hits forming the cluster. The x- and z-positions are simple averages of the individual hit positions. The y-position (along the length of the bar) is an energy-weighted average. If no cluster is found, the SCRC status word is the same as the SCR status word and the i.d. is the same as the SCR i.d.; otherwise, the SCRC status word is the sum of the status word for the first hit in the cluster and 100 times the status word for the second hit in the cluster. The i.d. is the SCR i.d. for the first hit in the cluster.

B The SC calibration map

```
Subsystem: NMIP_ADC,    nitems: 2
    Item: left,     length: 288,    type: float,    narray:22
    Item: right,    length: 288,    type: float,    narray:22
Subsystem: NMIP_ADCu,   nitems: 2
    Item: left,     length: 288,    type: float,    narray:18
    Item: right,    length: 288,    type: float,    narray:18
Subsystem: T0_TDC,      nitems: 2
    Item: left,     length: 288,    type: float,    narray:15
    Item: right,    length: 288,    type: float,    narray:15
Subsystem: T0_TDCu,     nitems: 2
```

```
    Item: left,     length: 288,    type: float,    narray:14
    Item: right,    length: 288,    type: float,    narray:14
Subsystem: T1,  nitems: 2
    Item: left,     length: 288,    type: float,    narray:14
    Item: right,    length: 288,    type: float,    narray:14
Subsystem: T1u, nitems: 2
    Item: left,     length: 288,    type: float,    narray:14
    Item: right,    length: 288,    type: float,    narray:14
Subsystem: T2,  nitems: 2
    Item: left,     length: 288,    type: float,    narray:14
    Item: right,    length: 288,    type: float,    narray:14
Subsystem: T2u, nitems: 2
    Item: left,     length: 288,    type: float,    narray:14
    Item: right,    length: 288,    type: float,    narray:14
Subsystem: T_sigma,     nitems: 2
    Item: first_param,      length: 288,    type: float,    narray:0
    Item: second_param,     length: 288,    type: float,    narray:0
Subsystem: WALK0u,      nitems: 2
    Item: left,     length: 288,    type: float,    narray:7
    Item: right,    length: 288,    type: float,    narray:7
Subsystem: WALK1,       nitems: 2
    Item: left,     length: 288,    type: float,    narray:9
    Item: right,    length: 288,    type: float,    narray:9
Subsystem: WALK1u,      nitems: 2
    Item: left,     length: 288,    type: float,    narray:7
    Item: right,    length: 288,    type: float,    narray:7
Subsystem: WALK2,       nitems: 2
    Item: left,     length: 288,    type: float,    narray:9
    Item: right,    length: 288,    type: float,    narray:9
Subsystem: WALK2u,      nitems: 2
    Item: left,     length: 288,    type: float,    narray:7
    Item: right,    length: 288,    type: float,    narray:7
Subsystem: WALK_A0,     nitems: 2
    Item: left,     length: 288,    type: float,    narray:9
    Item: right,    length: 288,    type: float,    narray:9
Subsystem: atten_length,        nitems: 2
    Item: left,     length: 288,    type: float,    narray:19
    Item: right,    length: 288,    type: float,    narray:19
Subsystem: atten_u,     nitems: 2
    Item: left,     length: 288,    type: float,    narray:19
    Item: right,    length: 288,    type: float,    narray:19
Subsystem: delta_T,     nitems: 4
```

```
    Item: dc2sc,    length: 24,     type: float,    narray:1
    Item: left_right,       length: 288,    type: float,    narray:31
    Item: paddle2paddle,    length: 288,    type: float,    narray:72
    Item: panel2panel,      length: 24,     type: float,    narray:8
Subsystem: delta_Tu,    nitems: 3
    Item: left_right,       length: 288,    type: float,    narray:4
    Item: paddle2paddle,    length: 288,    type: float,    narray:72
    Item: panel2panel,      length: 24,     type: float,    narray:8
Subsystem: pedestals,   nitems: 2
    Item: left,     length: 288,    type: float,    narray:34
    Item: right,    length: 288,    type: float,    narray:34
Subsystem: pedu,        nitems: 2
    Item: left,     length: 288,    type: float,    narray:25
    Item: right,    length: 288,    type: float,    narray:25
Subsystem: pulser,      nitems: 1
    Item: normalisation,    length: 1,      type: float,    narray:3
Subsystem: status,      nitems: 2
    Item: left,     length: 288,    type: int,      narray:12
    Item: right,    length: 288,    type: int,      narray:12
Subsystem: time_walk,   nitems: 1
    Item: ref_adc,  length: 1,      type: float,    narray:1
Subsystem: veff,        nitems: 2
    Item: left,     length: 288,    type: float,    narray:4
    Item: right,    length: 288,    type: float,    narray:4
Subsystem: veffu,       nitems: 2
    Item: left,     length: 288,    type: float,    narray:3
    Item: right,    length: 288,    type: float,    narray:3
```

C The SC bank

The DDL definition of the SC bank is

```
!----------------------------------------------------------------------
!       BANKname BANKtype      ! Comments
 TABLE  SC       B16   ! create write display delete
! Scintillation counter event bank
!
!   ATTributes:
!   -----------
!COL ATT-name FMT Min   Max  ! Comments
!
  1  ID       I     1     48 ! the address of the hit detector element
  2  TDCL     I     0  65536 ! tdc information (channels)
  3  ADCL     I     0  65536 ! adc information (channels)
  4  TDCR     I     0  65536 ! tdc information (channels)
  5  ADCR     I     0  65536 ! adc information (channels)
!
!    RELations:
!    ----------
!COL RELname  RELtype INTbank  ! Comments
!                     (COL)
!
 END TABLE
!
```

The corresponding C data structure is

```
typedef struct {
        uint16 id;    /*  the address of the hit detector element */
        uint16 tdcl;    /*  tdc information (channels) */
        uint16 adcl;    /*  adc information (channels) */
        uint16 tdcr;    /*  tdc information (channels) */
        uint16 adcr;    /*  adc information (channels) */
} sc_t;

typedef struct {
        bankHeader_t bank;
        sc_t sc[1];
} clasSC_t;
```

D   The SC1 bank

The DDL definition of the SC1 bank is

```
!----------------------------------------------------------------------
!       BANKname BANKtype      ! Comments
 TABLE  SC1      ! create write display delete
! Scintillation counter hits bank
!
!   ATTributes:
!   -----------
!COL ATT-name FMT Min   Max ! Comments
!
1   ID         I   1     48 ! the address of the hit detector element
2   time_l     F   0 100000 ! time for left paddle(ns)
```

```
3   energy_l   F   0  65536 ! energy in left paddle(MeV)
4   time_r     F   0  65536 ! time for right paddle(ns)
5   energy_r   F   0  65536 ! energy in right paddle(MeV)
6   dtime_l    F   0  65536 ! uncertainty in time for left paddle(ns)
7   denergy_l  F   0  65536 ! energy uncertainty for left paddle(MeV)
8   dtime_r    F   0  65536 ! uncertainty in time for right paddle(ns)
9   denergy_r  F   0  65536 ! energy uncertainty for right paddle(MeV)
!
!    RELations:
!    ----------
!COL RELname  RELtype INTbank  ! Comments
!                     (COL)
!
 END TABLE
!
```

The corresponding C data structure is

```
typedef struct {
  int id;           /*  the address of the hit detector element */
  float time_l;     /*  time for left paddle(ns)  */
  float energy_l;   /*  energy in left paddle(MeV) */
  float time_r;     /*  time for right paddle(ns) */
  float energy_r;   /*  energy in right paddle(MeV) */
  float dtime_l;    /*  uncertainty in time for left paddle(ns) */
  float denergy_l;  /*  uncertainty in energy in left paddle(MeV) */
  float dtime_r;    /*  uncertainty in time for right paddle(ns) */
  float denergy_r;  /*  uncertainty in energy in right paddle(MeV) */
} sc1_t;
typedef struct {
  bankHeader_t bank;
  sc1_t sc1[1];
} clasSC1_t;
```

E The SCR bank

The DDL definition of the SCR bank is

```
!-------------------------------------------------------------------
!       BANKname BANKtype      ! Comments
 TABLE  SCR          ! create write display delete
! Sc Scintillator reconstruction hit bank
!
```

```
!   ATTributes:
!   -----------
!COL ATT-name FMT Min   Max        ! Comments
1    id        I   0        48     ! paddle id
2    energy    F   0  100000.0     ! Energy (MeV)
3    time      F   0  100000.0     ! time(ns)
4    x       F -999.9  999.9 ! x position in sector coordinate system
5    y       F -999.9  999.9 ! y position in sector coordinate system
6    z       F -999.9  999.9 ! z position in sector coordinate system
7    dx      F -999.9  999.9 ! x error in sector coordinate system
8    dy      F -999.9  999.9 ! y error in sector coordinate system
9    dz      F -999.9  999.9 ! z error in sector coordinate system
10   status  I  -999999    999999  ! status word defined in sc.h
11   denergy   F   0  100000.0     ! uncertainty in Energy (MeV)
12   dtime     F   0  100000.0     ! uncertainty in time (ns)
!
!    RELations:
!    ----------
!COL RELname  RELtype INTbank  ! Comments
!                     (COL)
!
 END TABLE
!
```

The corresponding C data structure is

```
typedef struct {
  int id;    /*  paddle id   */
  float energy;    /*  Energy (MeV)  */
  float time;    /*  time(ns) */
  vector3_t pos;    /*  position in sector coodinate system  */
  vector3_t err;    /*  error in sector coodinate system  */
  int status;    /*  status word defined in sc.h */
  float denergy;    /*  uncertainty in Energy (MeV)  */
  float dtime;    /*  uncertainty in time (ns)  */
} scr_t;
typedef struct {
  bankHeader_t bank;
  scr_t scr[1];
} clasSCR_t;
```

F The SCRC bank

The DDL definition of the SCRC bank is

```
!-----------------------------------------------------------------------
! Time-of-flight Cluster (final result) bank.
!-----------------------------------------------------------------------
!       BANKname BANKtype      ! Comments
 TABLE  SCRC          ! create write display delete
!Sc Scintillator reconstruction hit bank
!
!   ATTributes:
!   -----------
!COL ATT-name FMT Min    Max   ! Comments
1    id        I   0        48 ! cluster id
2    energy    F   0  100000.0 ! cluster Energy (MeV)
3    denergy   F   0  100000.0 ! error in cluster energy (ns)
4    time      F   0  100000.0 ! cluster (energy-weighted) time(ns)
5    dtime     F   0  100000.0 ! error in cluster time (ns)
6    x       F -999.9  999.9 ! x position in sector coordinate system
7    y       F -999.9  999.9 ! y position in sector coordinate system
8    z       F -999.9  999.9 ! z position in sector coordinate system
9    dx      F -999.9  999.9 ! x error in sector coordinate system
10   dy      F -999.9  999.9 ! y error in sector coordinate system
11   dz      F -999.9  999.9 ! z error in sector coordinate system
12   status    I  -999999    999999  ! status word defined in sc.h
!
!    RELations:
!    ----------
!COL RELname  RELtype INTbank  ! Comments
!                     (COL)
!
 END TABLE
!
```

The corresponding C data structure is

```
typedef struct {
  int id;    /*  cluster id    */
  float energy;    /*  cluster Energy (MeV)  */
  float denergy;    /*  error in cluster energy (ns) */
  float time;    /*  cluster (energy-weighted) time(ns)  */
  float dtime;    /*  error in cluster time (ns) */
  vector3_t pos;    /*  position in sector coordinate system  */
  vector3_t err;    /*  {x,y,z} error in sector coordinate system  */
  int status;    /*  status word defined in sc.h */
} scrc_t;
typedef struct {
  bankHeader_t bank;
  scrc_t scrc[1];
} clasSCRC_t;
```

References

[1] E.S. Smith *et al*, “The Time-of-flight system for CLAS”, Nucl. Inst. and Meth. A432, 265 (1999).

[2] L. Elouadrhiri *et al*, ”Charged Particle Identification in CLAS”,
CLAS NOTE 98-004, February 1998.

[3] J. Aftosmis, J. Ficenec, D. Jenkins and E. Smith, ”Computer Modeling of Time-of-Flight Scintillator Pulses”, CLAS-NOTE 96-017, August 24, 1996.

[4] On-line DAQ Manual.

[5] G. Mutcler, S. Taylor and E. Smith, “CLAS TOF Scintillator Positions,” CLAS-NOTE 98-008, June 2, 1998.

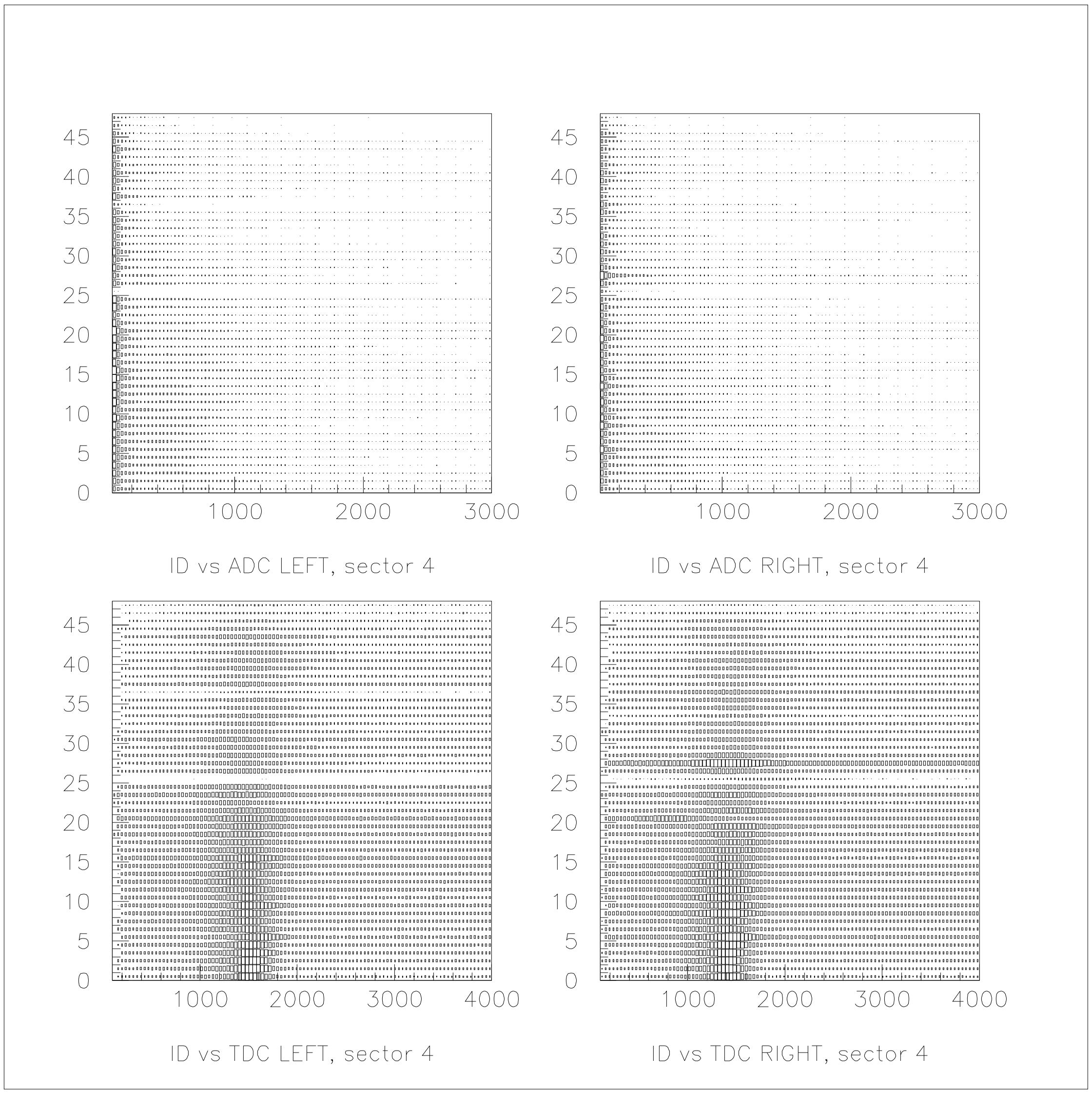

Figure 1: Histograms used in checking the status of the ADC and TDC signals left and right in the TOF calibration. As one can see the counter 26 is dead. There is a clear weakness in the TDC left counter 36. For the counters 40-48 higher statistics might be necessary to make a conclusion about their status.

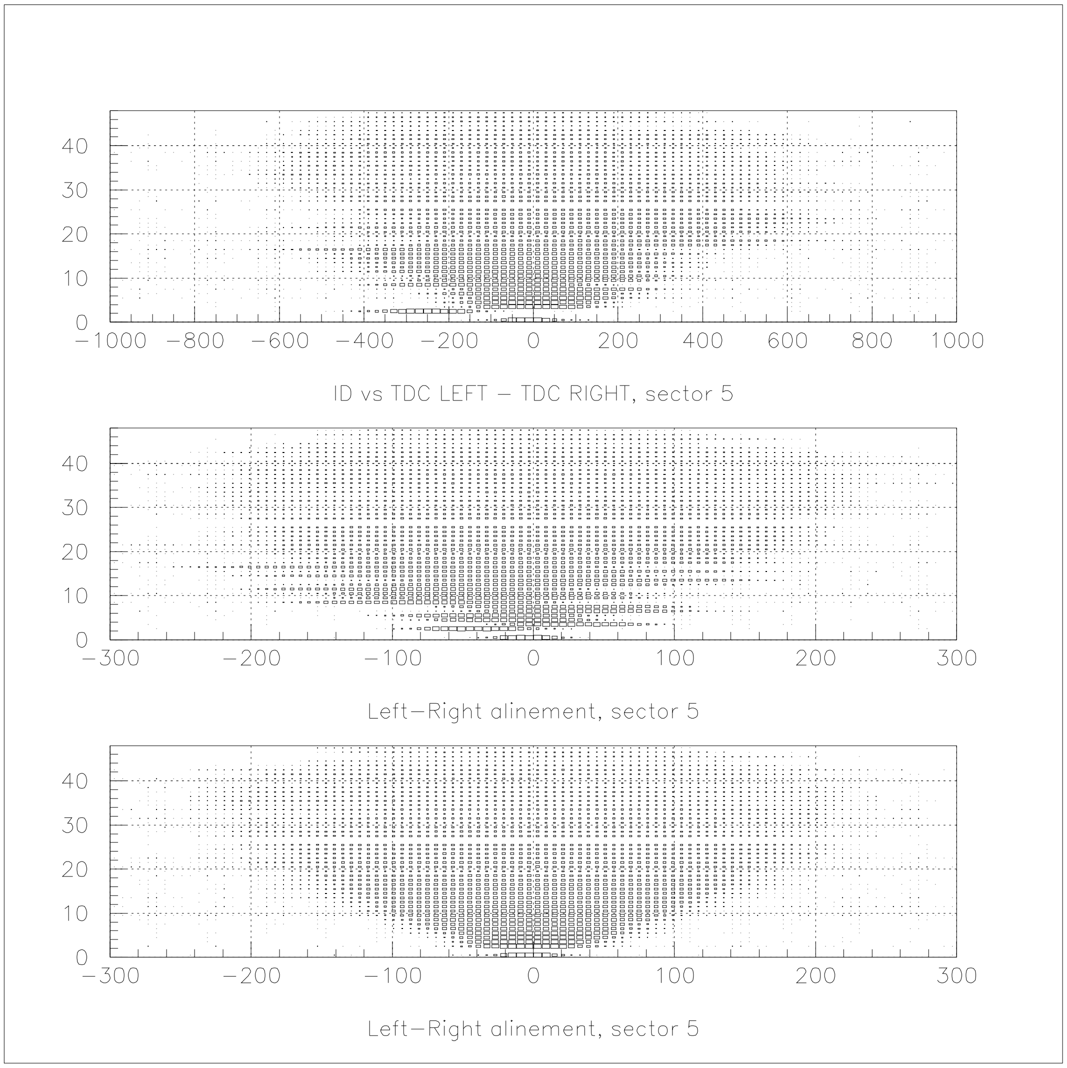

Figure 2: An example of left-right aligned counters in sector 5.

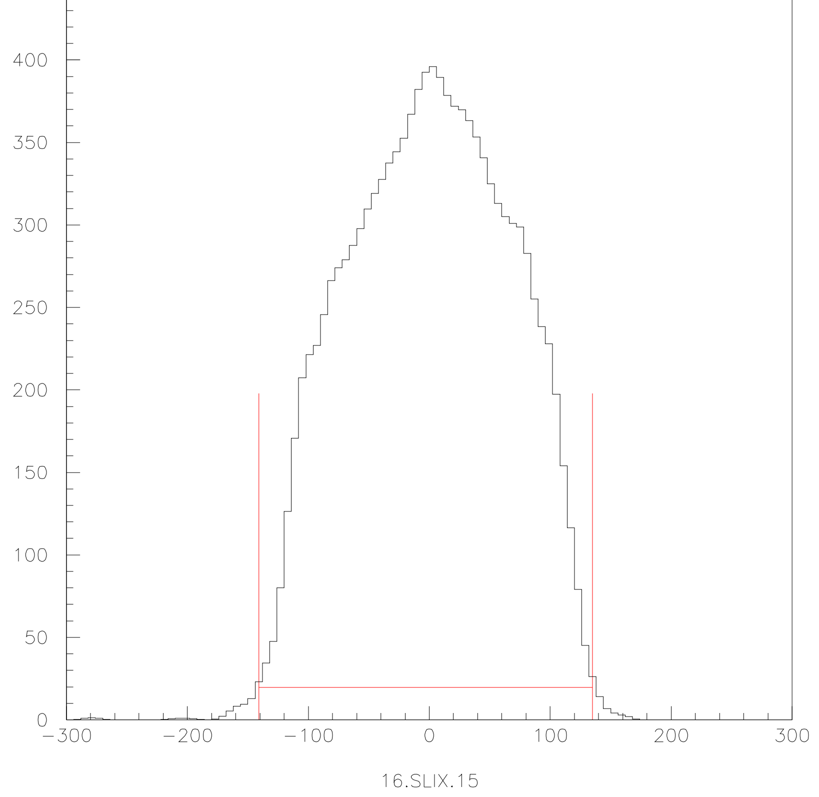

Figure 3: An example of finding left and right edges from the difference distribution of one counter. Once adjusted, the average of the left and right edges should be zero. (Eq. 17).

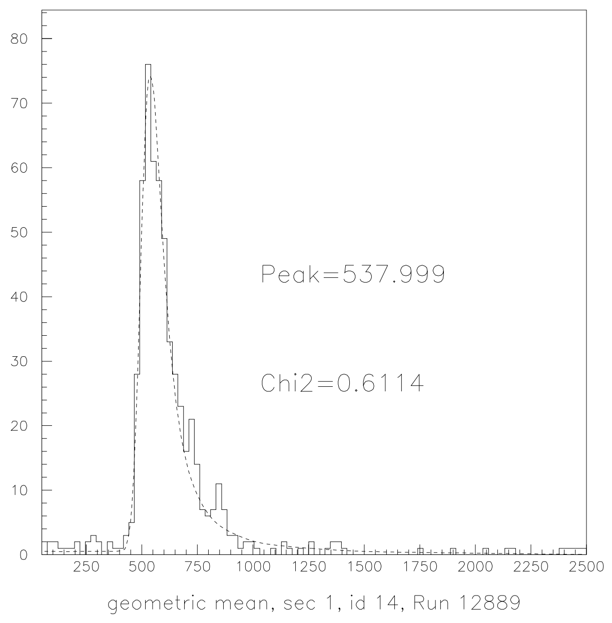

Figure 4: An example of a fit to the geometric mean of the energy loss of minimum-ionizing particles in the counter.

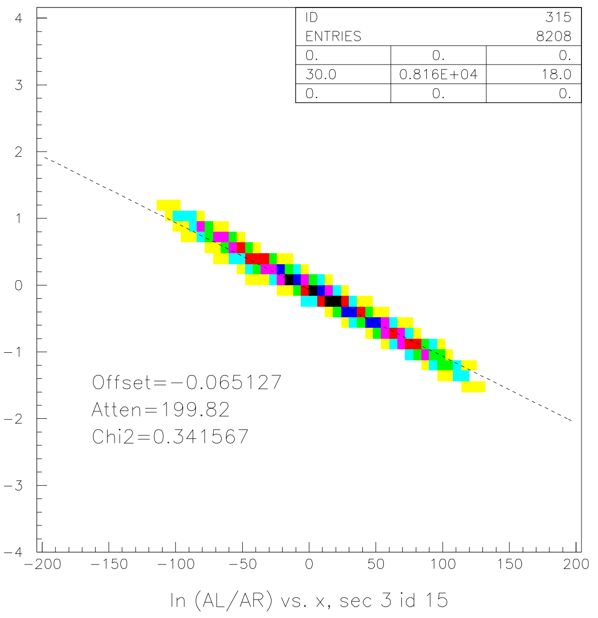

Figure 5: An example of correctly calibrated attenuation length and energy loss.

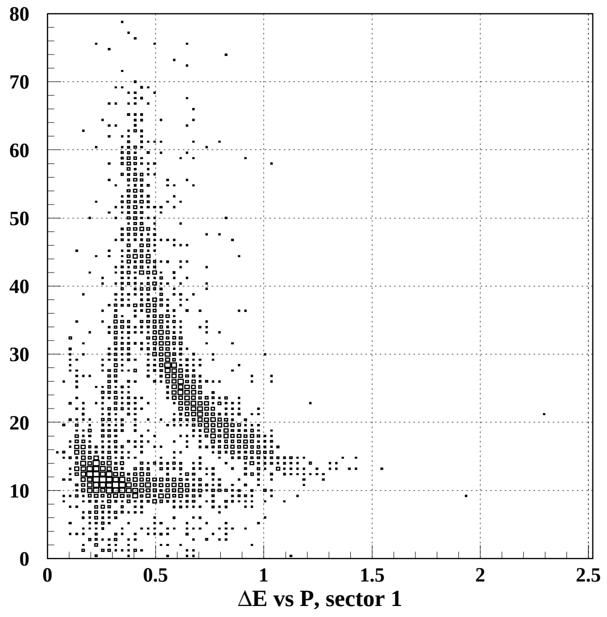

Figure 6: Energy loss in scintillator material versus momenta of the particles. Proton and pion bands are clearly distinguished.

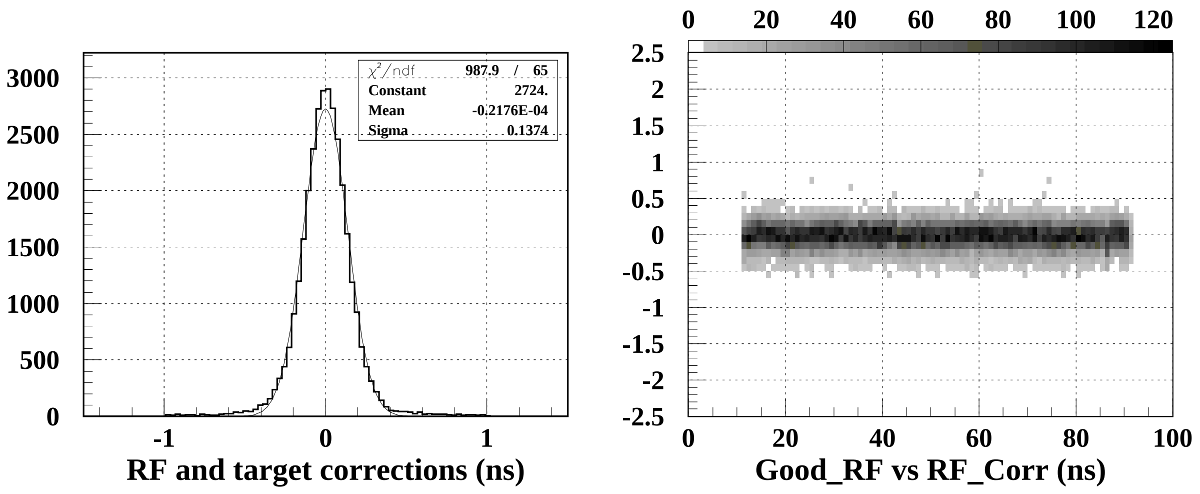

Figure 7: RF correction and "good" RF versus RF correction after calibration.