CLAS-NOTE 1999-005

CLAS Čerenkov Detector Calibrations

Alexander V. Vlassov, Vladimir S. Serov, Alexey V. Stavinsky

Institute of Theoretical and Experimental Physics
B.Cheremushkinskaya, 25, Moscow, 117259 Russia

Volker D. Burkert

CEBAF, 12000 Jefferson Avenue, Newport News,
Virginia 23606,USA

April 28, 1999

1 Overview.

The Čerenkov detector calibrations are important for determing the detector efficiency and for correct time measurement. To produce the quality and precise knowledge of the amplitude and the time from Čerenkov detector, we need to calibrate some parameters from run to run.

Some calibration parameters make use of beam interactions and can be calibrated by off-line data analysis. This kind of calibrations are usually referred to as **beam** calibrations. The other parameters can be determined without beam, either by using cosmic particles, or intrinsic photo-multiplier noise. Historically this calibration is called **cosmic** calibrations.

2 Calibration data maps and items to calibrate

The results of the Čerenkov calibrations are stored in the **CC_CALIB.map**. the command

> `$CLAS_BIN/scan_map -t $CLAS_PARMS/Maps/CC_CALIB.map`

shows you the current structure and number of entries in the map. Current map structure is :

Map: *CC_CALIB.map*

- Subsystem: HV, nitems: 1
  - Item: value, length: 216, type: float, narray:36
- Subsystem: TDC, nitems: 2
  - Item: T0, length: 216, type: float, narray:0
  - Item: T1, length: 216, type: float, narray:4
- Subsystem: pedestals, nitems: 2
  - Item: mean, length: 216, type: float, narray:8
  - Item: sigma, length: 216, type: float, narray:3
- Subsystem: photoE, nitems: 1
  - Item: amplitude, length: 216, type: float, narray:11

Comments to the map structure :

Names of subsystems : **HV** stands for High Voltage values, **TDC** - Time calibrations, **pedestals** - Amplitude pedestals calibrations, **photoE** - Single photoelectron position calibrations.

There are one or more **items** in each subsystem :

value - the HV values, **T0** - additive constant, the difference between SC and CC time in nsec, **T1** - channels-to-time constant, **mean** - mean value of the amplitude pedestal, **sigma** - it's dispersion, **amplitude** - the amplitude of the single photoelectron signal from PMT, after pedestal subtraction (in channels).

length: - is the array length for every entry, is equal to the total number of PMT in Cerenkov detector (216).

type: - type of the entries - **float** means that the values are real numbers.
narray: - number of calibration sets. Every calibration is associated with some run number.

3 Cosmic calibrations

For Čerenkov detector it is **T1, pedestals and Single photoelectron position** , which are determined without beam interactions.

3.1 T1 calibrations

T1 is the parameter, which converts TDC in channels to nsec. Usual value of **T1** is about 0.048 (nsec/channel). The **principle** of calibration is that a pulser signal is send with different time delays to each TDC channel.

Signals with definite time are repeated several times to increase the accuracy of the measurements. The TDC response is fitted by a linear function and T1 - is a slope parameter for the channel-time dependence.

This calibration procedure is the part of the forward carriage TDC calibration, and after every such run you need to add the entry into CC_CALIB.map. Also this calibration is needed if you change some hardware in the Cerenkov crate (number 12 now).

- First of all you need to start a special run, using usual DAQ procedure. Configuration you need to load : **TDC_FC**. No trigger configuration is needed for this run.
- Number of events needed and/or time : This calibration run usually takes about 40 min, and it requires 40000 events. After getting this number of events the run will stop automatically.
- The result of that run will be the file in

  ```
  /raid/stage_in
  ```

  directory with name looks like :

  ```
  clas_fc_tdc_* : i.e.  clas_fc_tdc_017206.A00
  ```

  Here 017206 - is the run number. As any other file, it will be stored on SILO, and you can restore this file and make changes in **CC_CALIB.map** some other time. Or restore the file , which already exists to make another entry in the map.
- Program to analyze the run is in CLAS standard place under cvs in the **utilities/bosanal** directory. So you need :
  - Somewhere in your directory you type:

    ```
    > cvs checkout include
    > cvs checkout c_bos_io
    > cvs checkout utilities/bosanal
    ```

    Then you'll have all code in your directory.
  - `> cd utilities/bosanal`
  - `> make exe`

    Hopefully you'll have executable to work with this kind of runs after that command. The name of the executable program is **TDCcal** .

- If you type TDCcal -h, you'll see brief instruction for it's using:

```
***** TDCcalibrate 2.2 *****


The program TDCcalibrate analyzes the raw banks (RCxx) of a data file
for a TDC calibration run. A TDC calibration run consists of M sets of
N events (N=20 typical) with a fixed delay T to the TDC for each of
the M sets. T[i+1]=T[i] + dT for each successive run i. Each group of N
events is fit, and the resulting mean values are fit to a 2nd order
polynomial. The constants of this polynomial are then put in the MAP
and written to a file.

USAGE HINTS: If you get many 'data jump' errors, the data may be too
noisy. In such a case you may want to check the data sequence using
'-spy 13.8.0' (or any other channel). Then count by how many channels
the trigger is off. I.e. the peak number goes up one for # channels
before the TDC value jumps up. You then use the -skew # to correct the
trigger point. (and check again.)
  If the data was taken with the scalers on, or if you expect there may
be missing events, you can ask the code to attempt to correct these
problems with -autojump. This tries to detect a premature jump in the
time delay by spying on the TDC values and detecting a jump larger than
30 channels. This does not work with noisy data !

USAGE:  TDCcal <datafile> <options>

OPTIONS SUMMARY:
        -quiet          Become progressively more quiet.
        -l #            Start processing at event number #
        -m #            Process until event number #
        -skew #         'Skew' the trigger, move the trigger point up
                        (down) by #.
        -map            Put entry in MAP for this run.
        -mapfile <file> Use mapfile <file> instead of $CLAS_PARMS/Maps/
        -hist           Create a histogram (ascii) output file.
        -noTT           Do not use Translation Tables.
        -uTT <file>     Use Translation Table from <file>
        -faults #       Tolerate # number of errors reading crates. (10)
        -ncoeff #       Fit the TDCs up to order # (default is 2)
        -nocompare      Do not compare new values against the old ones.
        -autojump       Automatically jump the trigger when code detects
                        a jump in TDC values.
        -notdc          Do not write the TDC_RCxx files.
```

```
-paddle          Show progress with a paddle wheel.
-skip #          Skip processing of crate # (can be repeated.)
-only #          Only process for crate #.
-debug           Become progressively more verbose.
-spy cr.sl.ch    spy on crate.slot.channel
-backup          Make a backup of the map before writing a new one.
                 (default)
-nobackup        DO NOT make a backup of the map before writing
                 a new one.
-help -h         Print this help.

For more information: http://www.physics.unh.edu/Maurik/tof_utils.shtml
or email me at maurik.holtrop@unh.edu.
```

- Next step is to copy **CC_CALIB.map** from **/CLAS_PARMS/Maps** directory to your directory (i.e. **bosanal**).
- Then you start the program. Example of how to use the program :

  ```
  > TDCcal /work/clas/disk2/vlassov/clas_fc_tdc_017206.A00
    -mapfile CC_CALIB.map -notdc -only12 -map -paddle ;
  ```

  Note : Čerenkov detector is in crate 12. Other parameters are explained in the help message above.

  After the end of the process, the program will show you new values of the parameter **T1** with comparing to the old values and will ask if you agree. You must check all values and make sure that the difference is small or you understand it (i.e. in case of new hardware block). The result of this program is the data map modification in your directory.
- The last things you need to do is to copy back **CC_CALIB.map** to the **$CLAS_PARMS/Maps** directory and notify the person, responsible for CC package, that you make new entry.

3.2 Pedestal calibrations

Pedestals must be subtracted for accurate ADC measurement. Usual value of pedestals vary from 300 to 800 channels, the sigma is about $2 \div 3$. This calibration procedure is a part of pedestals calibrations for all detector groups.

- You need to start the pedestal run, using standard DAQ procedure. Configuration you need to load : **PEDS_ALL**. No trigger configuration is needed for this run.
- Number of events needed and/or time : $2000 \div 3000$ event is sufficient for pedestal measurements.
- Program to analyze the run is **PEDMAN**. You have to type

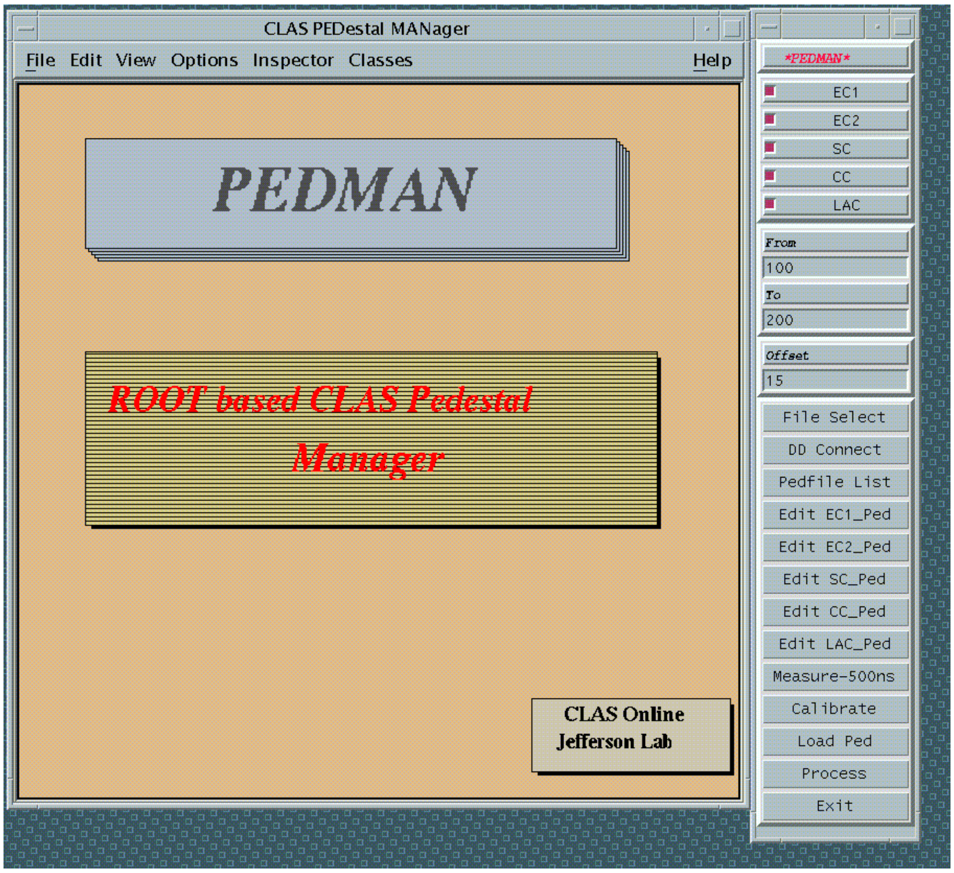

Figure 1: Pedman GUI

```
> pedman
```

on any clon machine (i.e. clon00). you'll see the pedman GUI, shown on Fig.1.

- **IMPORTANT!** For now the program works only with files in /raid/stage_in on clon cluster! Use buttons on the right
  - click "CC" to select it. It is better to proceed with all crates (EC1,EC2,SC,CC,LAC) selected.
  - define the event range , default from event number 100 to 200.
  - offset must be 15
  - click on "Process"
- As a result, two kind of output files - *.ped and *.tped will be created, cc.tped is a convenient file for making new entry in the data map.
- To make the new entry in the *CC_CALIB.map* you need :
  - in your directory on clon machine type:

    ```
    > ls -l $CLON_PARMS/pedman/Tfiles/cc.tped
    ```

    it will show you the run number for latest pedestal calibration.
  - copy cc.tped and *CC_CALIB.map* to your directory:

    ```
    > cp $CLON_PARMS/pedman/Tfiles/cc.tped .
    > cp $CLAS_PARMS/Maps/CC_CALIB.map .
    ```
  - check if *CLAS_BIN* is defined. If not, define it :

    ```
    > setenv CLAS_BIN /group/clas/builds/PRODUCTION/bin/SunOS
    ```
  - Prepair files for map filling:

    ```
    > tail -222 cc.tped | grep -v '*SECTOR' | awk '{print$3}' > mean.txt
    > tail -222 cc.tped | grep -v '*SECTOR' | awk '{print$4}' > sigma.txt
    ```
  - for map filling, say for the run number 17841, you need to type:

    ```
     > $CLAS_BIN/put_map_float -mCC_CALIB.map -spedestals -imean
    -t17841 < mean.txt
     > $CLAS_BIN/put_map_float -mCC_CALIB.map -spedestals -isigma
    -t17841 < sigma.txt
    ```
  - And at last,you must copy back the *CC_CALIB.map* :

    ```
    > cp CC_CALIB.map $CLAS_PARMS/Maps/
    ```

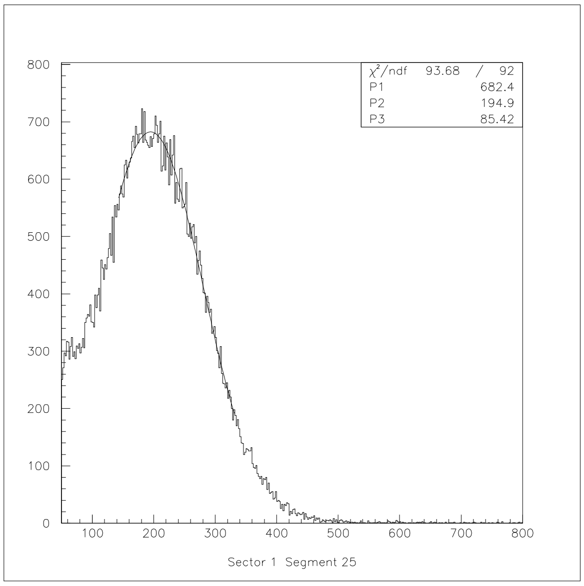

Figure 2: Example of the SPE histogram

3.3 Single photoelectron (SPE) amplitude

Single photoelectron amplitude calibration is a unique procedure, used for CC only. The idea is to use self-triggering of Čerenkov detector to see the noise function and to define the position of ONE photoelectron.

- Configuration you need to load : **CC_CALIB**. Only CC package is in configuration. Trigger you need to load (in TIGRIS) :

  ```
  /home/trigger/TriggersFeb98/beam_triggers/e1_CC.trg
  ```

- Minimum number of events you need 10000000 , this is about one 2Gb file. Usual counting rate about 5000 events/sec, so you need about 40 min for a run. The problem is that the number of entries for different PMT can deviates by factor of 100, so you need that many events to be sure, that you have sufficient information for every PMT calibration. After the end of the run you have to copy the output file to the work disk for the SPE positions analysis.

- To measure the SPE positions we need simply to fill ADC distribution histograms for every PMT. It is possible to do that by any program, say RECSIS, but it takes a lot of time. Simplified code to do that was written and checked under cvs in the **specal** directory.

- To produce your executable somewhere in your home tree you type:

```
> cvs checkout cc_cal
> cd cc_cal
> make
```

- next you edit spe.ini file here, where you indicate the run number and the full path to the file you want to analyze, say :

```
RunNumber=17717
DataFile=/work/clas/disk2/vlassov/clas_cc_017717
NTupleFile=NONE
[End-Of-File]
;
```

- to start the program you type (for SunOS platform):

```
~/bin/SunOS/specal spe.ini
```

- After the job stops you'll see a file with histograms . The file name is **adc_cc.rz** .

- Next step is to fit the ADC histograms to define the single photoelectron position. The simplest way is to fit by Gaussian distribution near the maximum, with larger range to the right. The other way is to use **fitfun.for** function,in the same directory, which takes into account possible noise function in the form of the

$$a * exp(-b * x)$$

Unfortunately, in case of the large noise function, it is difficult to be sure that the SPE position is correct.

For now a Gaussian function was used for SPE position definition. The method of SPE definition is not crucial as soon as we know the efficiency dependence on the number of photoelectrons by direct measurements, based on these SPE values.

You can use **spe.kumac** for fitting the photoelectron peaks.

```
> paw
> exe spe
```

spe.kumac provides also a file with SPE position values -**spe.dat**, which can be used for date map filling. It is important to check the fitted plot for every PMT, because sometimes it fails to fit properly. (You can see all plots in the **tmp.ps** file.) An example of a good **SPE** fitting is shown in figure 2. **P2** here is the single photoelectron position.

As an example of the PMT with the large noise function, the ADC spectra from PMT number 19 in sector 1 is presented in figure 3. Top fitting is for Gaussian function near the peack, bottom is fitting by **fitfun.for** with the noise function.

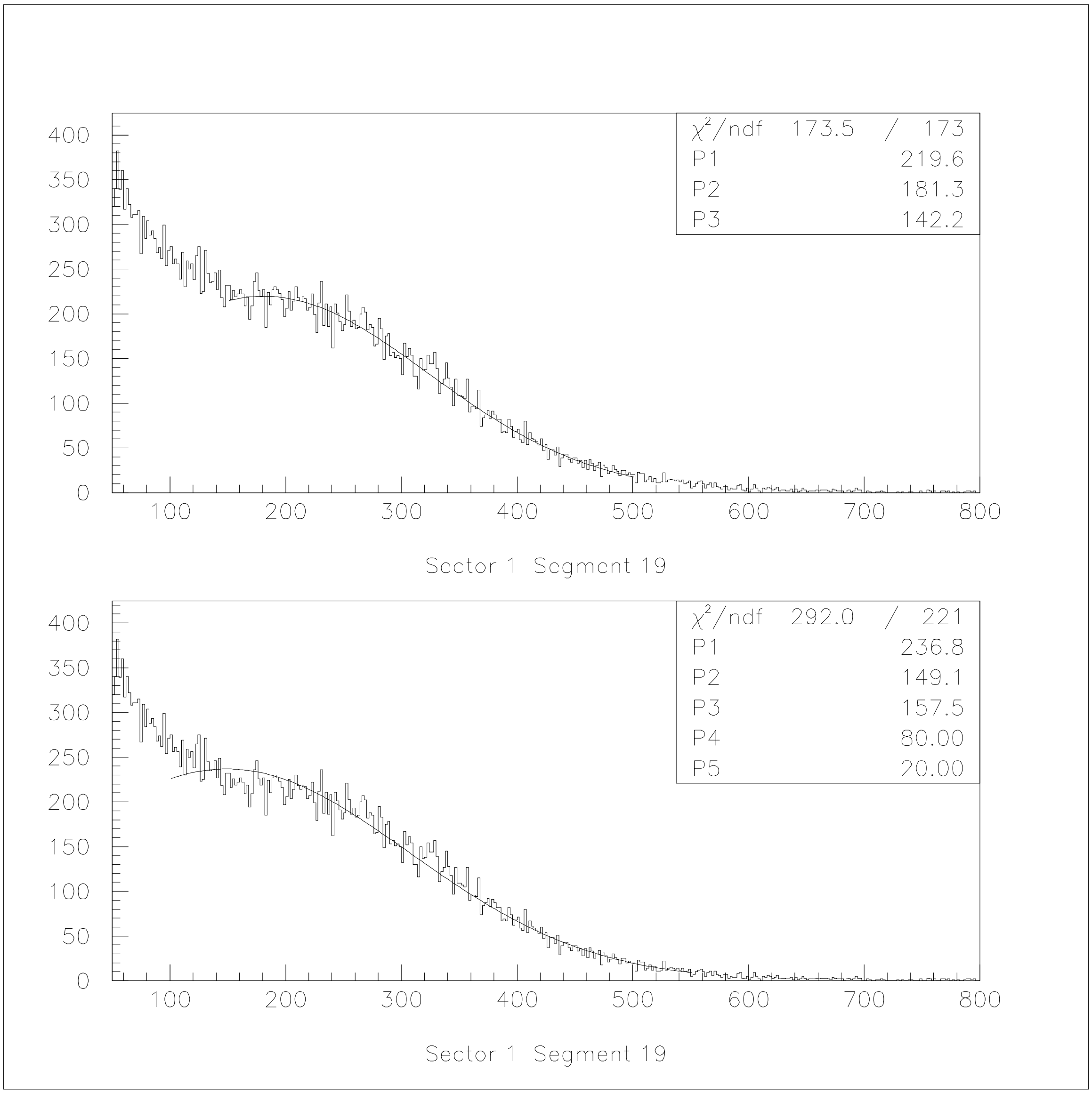

Figure 3: Example of PMT with a large noise function

- If you find that some plot are not fitted properly, you can use **tmp.kumac** file to do that by hand to define correct values of SPE and edit **spe.dat** file to put correct values in the file.

- To fill data map, you can follow similar procedure, as when filling pedestal values :

```
> cp $CLAS_PARMS/Maps/CC_CALIB.map .
> awk '{print $3}' spe.dat > spe.map
> $CLAS_BIN/put_map_float -mCC_CALIB.map -sphotoE -iamplitude
  -t17845 < spe.map
> $CLAS_BIN/scan_map -t CC_CALIB.map
> cp CC_CALIB.map $CLAS_PARMS/Maps/
```

4 Data based calibrations (under development)

4.1 T0 - time offset for Cerenkov detector

Time offset is needed to obtain improved timing from the Čerenkov detector. For now there is no standard procedure for that yet.

This offsets can be easily calculated during data analysis, where you have the time measured both by CC and ToF (SC) and you know the track lengths to the CC plane and SC plane.

Figure 4 shows the difference between time , measured by Čerenkov and ToF counters, corrected for the track length difference. No offset applied. One can see that the time difference is about 20 nsec; left and right PMT (odd and even) has different mean time values. Some number of events at about 70 nsec probably are non-stop events.

One can use RECSIS to plot histograms like Fig.4 or use ntuples . The mean values can be calculated, for example, by

- booking profile histograms by :

```
> call hbprof(NHIS,' CC - SC time, corrected for track length',
>&  36,0.5,36.5, 0.,40.,' ')
```

  Here 36 stands for the number of PMT in one sector, and $0. \div 40.$ - the range in which the mean value is calculated. For more details see HBOOK description.

- filling it by

```
> call hfill(NHIS,PMT_number, time_diff, 1.)
```

- and in a result the histogram contents will contain the mean values of time_difference as a function of PMT number.

The example of profile histogram with mean values of **T0** is shown on figure 5.

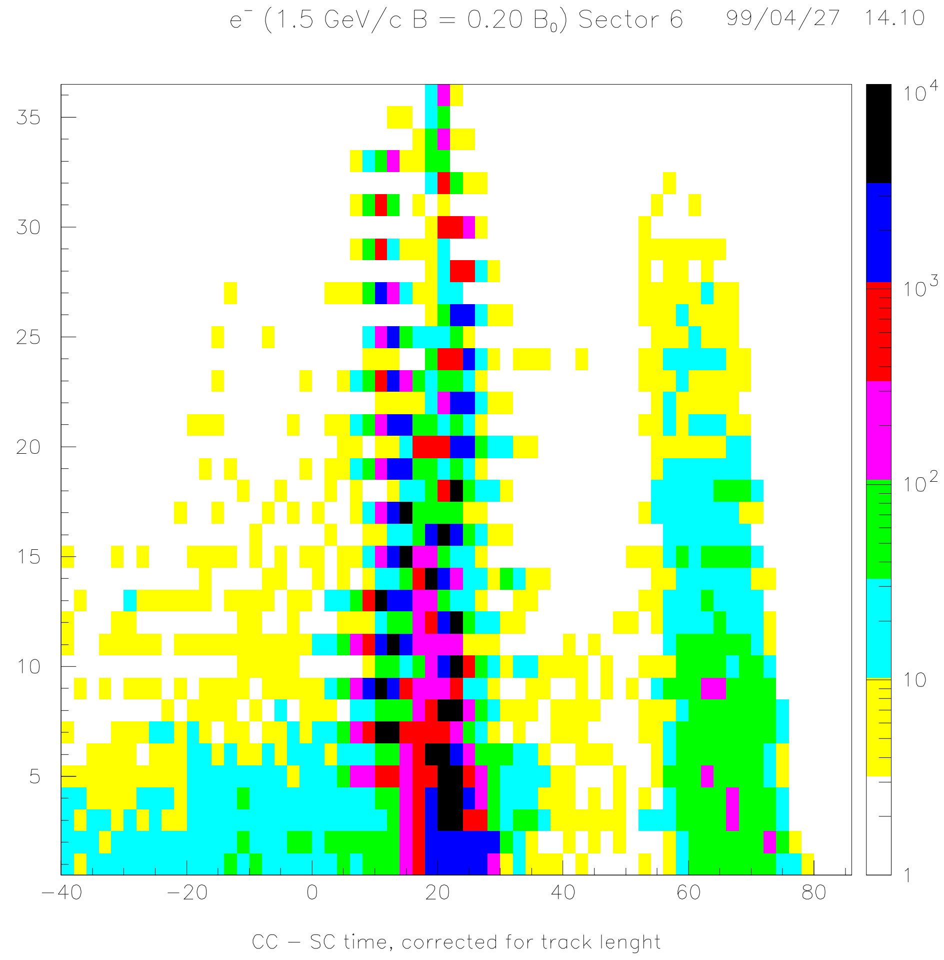

Figure 4: Time shifts for different PMT

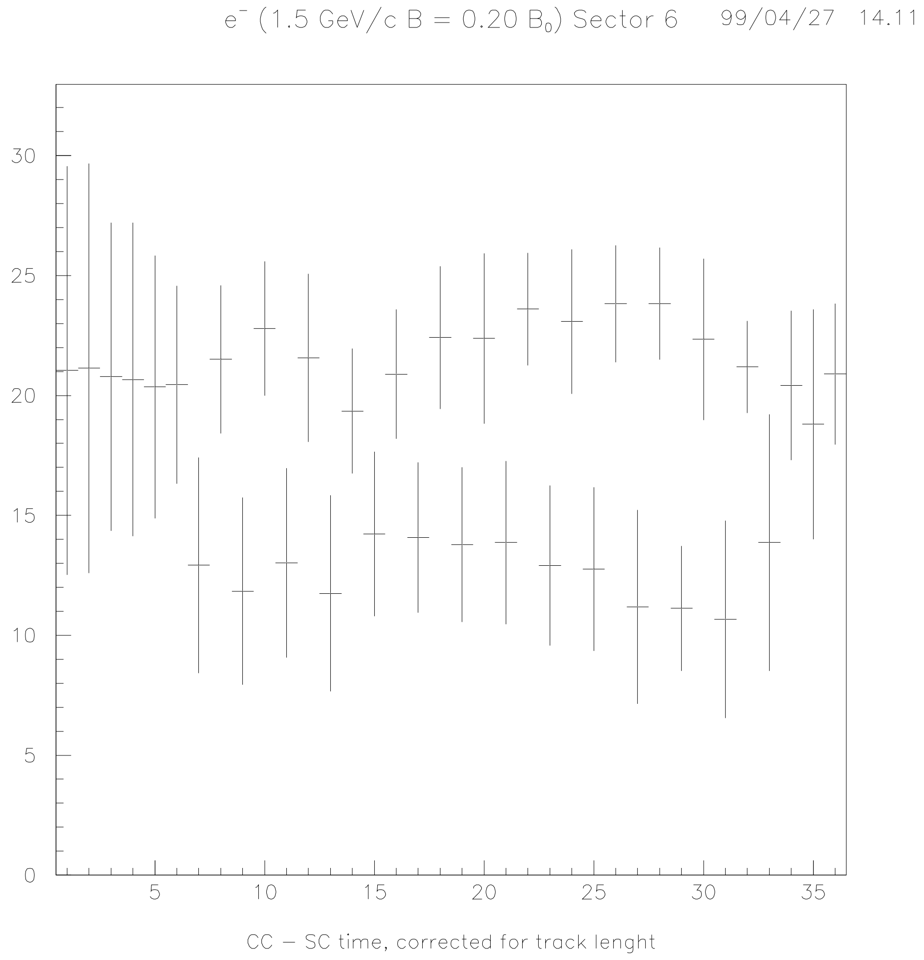

Figure 5: Mean value of T0 as a function of PMT number