CLAS-NOTE 2007-005
Feb 9, 2007

New Start Counter Calibration Procedures

Julian Salamanca, Jörn Langheinrich, Philip Cole, Eugene Pasyuk.

Abstract

As a part of g8b documentation, we shall describe the procedure that we followed in order to obtain the calibration constants for the New Start Counter (ST) at CLAS (CEBAF Large Aceptance Spectrometer).

1 Introduction

In order to find the time calibration delay for each paddle, it is nesesary to find the time of a particle hitting a ST paddle, t, which is defined by [1]:

$$t = c_0 + c_1 T + t_w - t_{pos} \tag{1}$$

where c_0 is the time-delay calibration constant (it is obtained from the difference between RF time and ST time for each paddle), c_1 is a correction to convert T (raw TDC channels) to nanoseconds, t_w is the time walk correction and t_{pos} is the time that the light takes to go through the paddle to the light guide. If a particle hits the leg region (see Fig. 1), t_{pos} will be a linear funtion of the position but it hits the nose, it will have another behavior rather than linear. In general, to know t_{pos}, an empirical funtion was used:

$$t_{pos} = \frac{l_0}{veff_{leg}} + k_0 + k_1 l_2 + k_2 l_2^2 \tag{2}$$

where l_0 is the entire leg length, l_2 is the distance from hit position to the leg-nose juntion and $veff\_leg$, k_0, k_1, k_2 are the calibrations constans to be found. In fact, 5 calibrations constants for each paddle are written into ST software as:

1. delta_T_pd2pd (time off-set between paddles)

2. veff_leg (effective velocity in leg region)

3. veff_nose2 (nose region fit parameter)

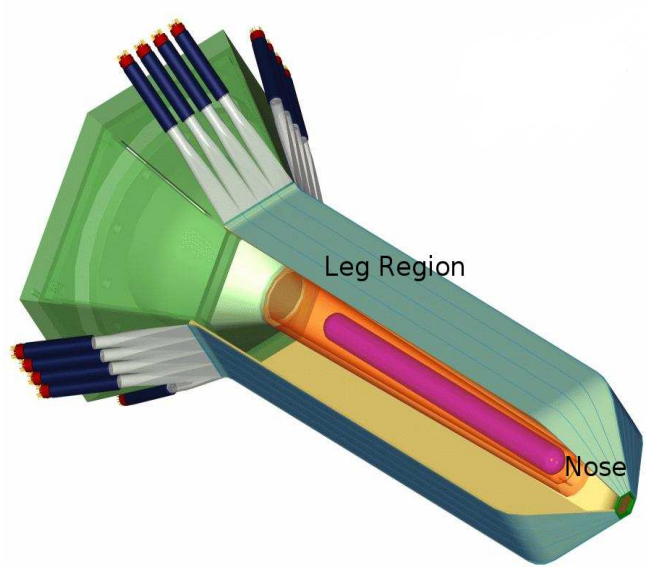

Figure 1: New Start Counter.

4. veff_nose1 (nose region fit parameter)

5. veff_nose (nose region fit parameter)

where $veff_{leg}$, k_0, k_1, k_2 are related to veff_leg, veff_nose1, veff_nose, veff_nose2 respectively. Firstable, the cooked data is the input of *stn_calib.cc* which recontruct ST events and gives a root files as output where a set of histogramas are placed to be analyzed. Two of them, sttag[1] and dt_dist[2] are used by *stn_veff_fit.C* to pull out the constants for each paddle.

2 Setting User Account

Because of the old version of CLAS code was linked to the old Start Counter (six paddles), the New Start Counter package *stn_calib* must be linked to the latest version of the CLAS code. Working on ifarm, the stn_calib compilation and linking was built by using /group/clas/builds/release-4-15. Adding to this, a version of *ST* package is in cvs. It can be placed into user area by:

```
ifarml1.jlab.org> cvs checkout packages/utilities/stn_calib/
```

3 Compiling the *stn_calib*

Once the user account is ready, you need to be in stn_calib directory and make:

[1]Difference bewen ST time and TAG vs paddle number.
[2]Differene between ST time and TAG time vs particle hit position in the paddle.

```
ifarml2.jlab.org> make clean
ifarml2.jlab.org> make all
```

This generates a stn_calib executable file which will appear as:

```
~/bin/LinuxRHEL3/stn_calib
```

into user's area. What it does is just to reconstruct ST events base on banks information and produce several histograms to be analyzed.

3.1 How to Run the stn_calib executable

By doing:

```
ifarml2.jlab.org> stn_calib -h
```

you will get command help. It looks like this:

```
Usage: stn_calib [-opt1 ... -optn] file1 [file2] ... [fileN]

Options:
  -h          Print this message.

  -n[#]       Process only # number of events

  -o[file]    Output <filename>

  -r[dir]     Raw data input from (not BOS) file <dir>/stnraw_<runno>_a<no>

  -t[#.#]     Tag/ToF time match in #.#ns window for single-track events

  -T[#.#]     Tag/ToF time match in #.#ns window for multi-track ambiguities

  -X          X-windows: dont open histogram canvas window
```

Cooked files to run the stn_calib executable will be needed [3]. Each one must have HEAD, CL01, EVNT, SCPB, DCPB, TAGR, TDPL, STN0, and STN1 banks.

As always, a set of several files come from the cooking should be analyzed by stn_calib.cc code (by stn_calib executable) so it will be a good idea to use an script to run stn_calib. An example is shown:

[3] We suggest at least 2 million events.

```
 #!/bin/csh -f
foreach i ( /w/work/clas/disk4/user_name/*.* )
   echo $i
   /u/home/user_name/bin/LinuxRHEL3/stn_calib -X $i -o$i.root
   echo "============================================="
end
$
```

The executable output will be a set of root files which will be located in the same area where cooked BOS files are.//

4 Concatenating ROOT files

There are several cooked files per each run and a root file per each cooked file. In order to concatenate root files, we used "concatenate5.0.C" [4]. This code puts together all the statistics contained in the root files into one root file. The big root file is called "one.root".

concatenate5.0.C works as follow:

1. Put the cooked files in a list (take care about the last line because End of File problems can ocurr). This list will be used as input of concatenate5.0.C code. Both must be in the same directory.

3. do:

```
ifarml2.jlab.org> root -l

root[1] .x concatenate5.0.C
```

4. Make sure the output is "one.root" (check its size).

Note: Everything was made on ROOT and it important to take care about the ROOT version to avoid problems. Actually we used 5.12ROOT version.

4.1 Running stn_veff_fit.cc

This code anlyzes a part of stn_calib.cc output in order to extract the calibration constants. stn_veff_fit.cc can be run by:

```
root [0] TFile f("one.root")
root [1] .x stn_veff_fit.C
```

A window showing a histogram related to an specific CLAS sector and a specific paddle in that sector will be opened (see Fig 2).

[4] This code is part of the *ST* software

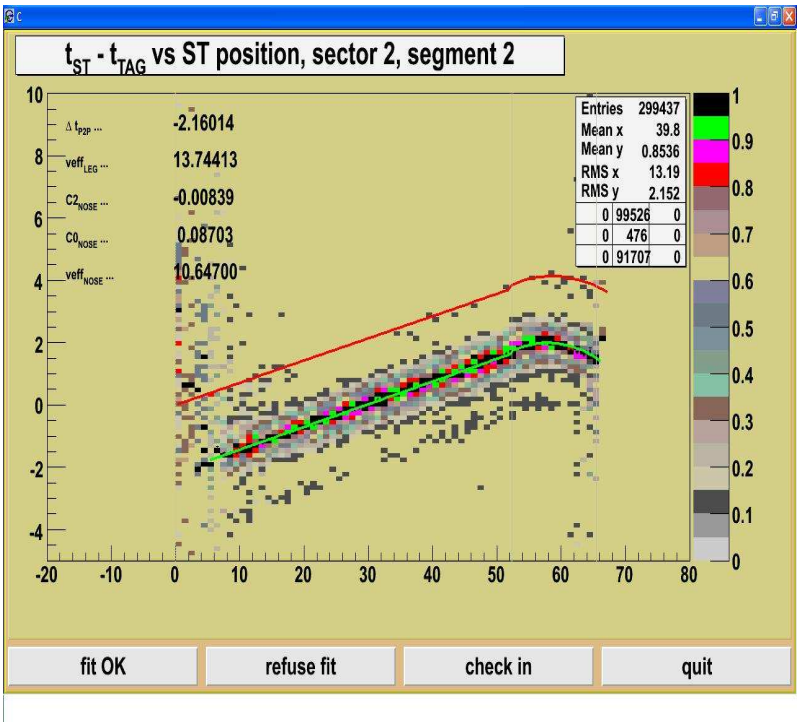

Figure 2: As expected, the fit comes as linear function in the leg, but in the nose , it behaves as quadratic polynomial.

If you are in JLab, you can click into "Checkout" buttom and automaticaly the data base will be changed (make sure this works properly). If don't, you can extract the fit values directly by using *test_stn_veff_fit.C*[5].It generates 5 files called:

1. STN_CALIB_delta_T_pd2pd.dat
2. STN_CALIB_veff_leg.dat
3. STN_CALIB_veff_nose2.dat
4. STN_CALIB_veff_nose1.dat
5. STN_CALIB_veff_nose.dat

By hand you can update the data by using STN_CALIB ... files and doing:

```
/group/clas/tools/caldb/caldb_write_and_link.pl s=STN_CALIB ss=delta_T
i=pd2pd min=47807 max=48699 ci="constants taken from run 47900" f=file_address
STN_CALIB_delta_T_pd2pd.dat it=calib_user.RunIndexg8b
```

where "s", "ss", "i", "it" and other options are explained in [2].

[5]It is a part of ST software.

It is important to check "*it*" option, the run index number table [6], because it is essential to run correctly the whole software related to ST.

5 Offset Correction

Looking at "sttag" histogram produced by running "stn_calib.cc", you will not find pd2pd offset alignment at all (see Fig 3 and Fig 4). 2D sttag histogram gives the difference between ST time and Tagger time. It must be "zero" or "close to".

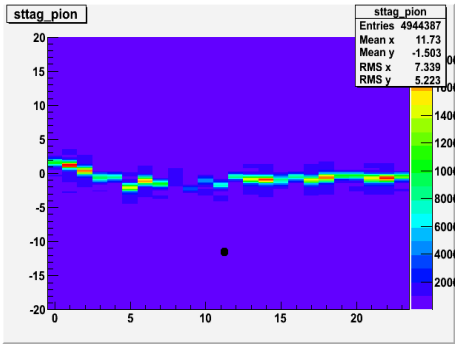

Figure 3: sttag histogram. As shown, offsets are out of alingment.

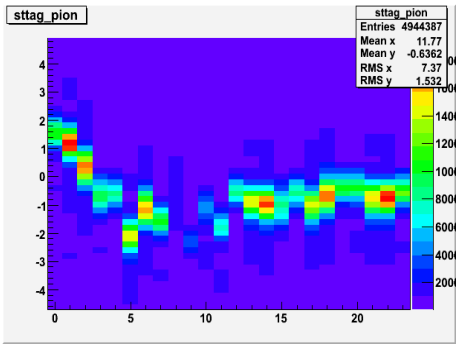

Figure 4: sttag histogram zoom.

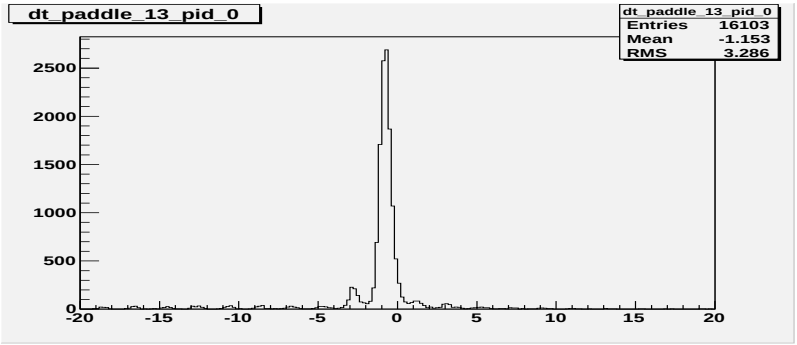

Figure 5: Single projection of sttag histogram . For the 13 paddle, the maximum peak is located arround -0.9ns. It means the offset for this paddle must be correted by 0.9ns.

To adjust the offsets (get the offsets to zero for each paddle) we modified "stn_calb.cc" to get 24 y-projections histograms from "sttag" histogram. From those, you can extract the value where the maximum peak is. In Fig 5 this value is arround -0.9ns and correspond to the value by the offset must be adjusted (for this specific paddle).

[6] Into the software, "*it*" appears as $calib\_user.RunIndex < experiement\_name >$.

The new pd2pd constans will be the result of subtracting those values from the current pd2pd contans. To make sure the alignment is done, the new pd2pd constants must be placed into the data base and used to re-run "stn_calib.cc". What you expect is shown in Figs 6, 7 and 8.

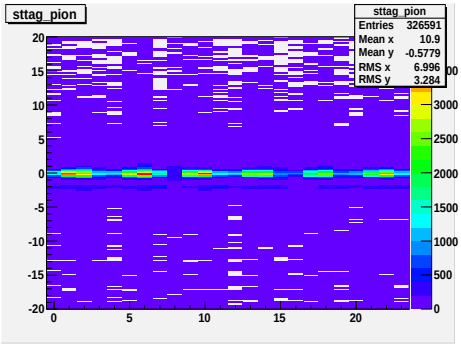

Figure 6: Corrected sttag histogram. As expected, offsets are arround "zero"

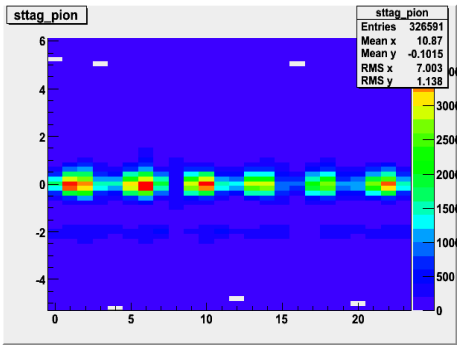

Figure 7: Corrected sttag histogram zoom arround "zero".

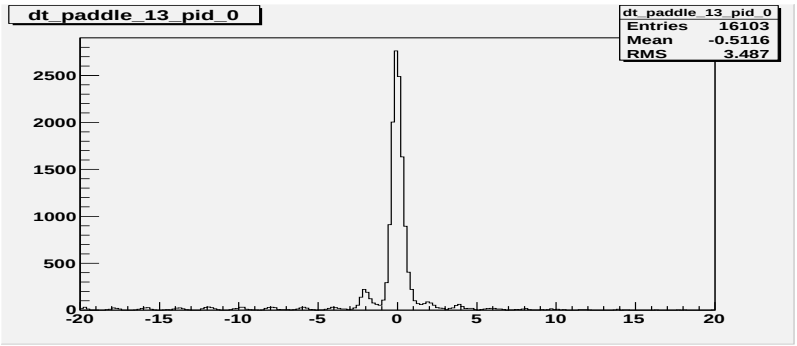

Figure 8: Single projection from corrected sttag histogram. As expected, the maximum is located at "zero"

5.1 Offset Resolution

By using a single run, the calibration constants come out for every run. This is because the Start Counter Resolution is about 0.1ns and the difference between each value of pd2pd from one run to another is less than the resolutions' detector.

It was checked by taking 14 runs and compared them to see the differences in between. The result is shown in Fig 9. Standard deviations are less than 0.1ns as expected.

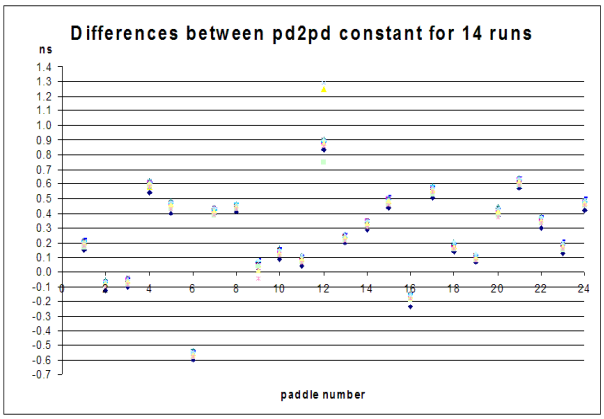

Figure 9: Differences between pd2pd constants ($x-$axis) for 14 runs (paddles $y-$axis). For a single pd2pd constant, 14 points were ploted and those are very close to each other (i.e see paddle 22).

6 Conclusions

This documentation shows in a great detail which the procedures to obtain the New Start Counter calibrations constants are hold. For future calibrations, it will be a helpful documentation, especially for the people who have no expertise in CLAS software.

References

[1] Sharabian, Y. G.; Battaglieri, M.; Burkert, V. D.; Devita, R.; Elouadrhiri, L.; Guo, L.; Kashy, D.; Kubarovsky, V.; Mutchler, G. S.; Ostrick, M.; Ripani, M.; Rossi, P.; Rottura, A.; Pasyuk, E.; Weygand."A new highly segmented start counter for the CLAS detector". Nucl. Phys. **A** 556 **246** (2006).

[2] CLAS-NOTE 2001-003