Table of Contents

DTCalibration package: a tool for DT calibration and synchronization

Complete: 🟩

By Sara Bolognesi & Gianluca Cerminara & Marina Giunta & Silvia Maselli & Giorgia Mila

Introduction

`CalibMuon/DTCalibration` is a CMSSW package which provides some tool for the computation of calibration constants needed for Rechit reconstruction in the DTs. It allows to compute global ttrig offset from the time box and drift velocities from the menatimer, t0 from the test pulses and noise channels. In the following a brief description of the package is provided toghether with some practical informations about its usage.
The complete and detailed instructions to run the calibration can be found here.

Applications available with the package

The code can be checked out with
`cvs co CalibMuon/DTCalibration`

The classes provided by this package to **perform calibration** are:

- `DTTTrigCalibration`: fills the time-boxes needed for ttrig offset computation, it can already compute the ttrig and write the results into DB or simply write the time box histograms in a root file.
- `DTTTrigWriter`: implements the computation of the ttrig from the time box fit, it needs a root file of time box histograms as input and it writes the results into DB
- `DTVDriftCalibration`: fills the TMax histograms for the drift velocity and hit resolution computation (the meantimers are computed from 4D segments with different formulas for different pattern), it can already compute the vdrift and write the results into DB or simply write the time box histograms in a root file.
- `DTVDriftWriter`: implements the computation of the vdrift and hit resolution from the time box fit, it needs a root file of TMax histograms as input and it writes the results into DB
- `DTT0Calibration`: implements the computation of the t0 from the test pulse data, it write the results into DB and it produces a root file with the test pulse peak for some cells (which you can chose in the cfg)
- `DTNoiseCalibration`: fills two different sets of histograms: the first one contains the number of hits per wire per event while the second one contains the average noise per wire (granularity: per layer).

The classes provided by this package to **analyze the calibration results** are:

- `DTTrigAnalyzer`: reads a ttrig DB and store the results in a root file. It contains a tmean, sigma and ttrig histo for each wheel, for each kind of SL (ph1,ph2,theta) and a tmean, sigma and ttrig distribution for each wheel, station and kind of SL.
- `DTVDriftAnalyzer`: reads a vdrift DB and store the results in a root file. It contains a vdrift and resolution histo for each wheel, for each kind of SL (ph1,ph2,theta) and a vdrift and resolution distribution for each wheel, station and kind of SL.
- `DTT0Analyzer`: reads a t0 DB and store the results in a root file. It contains a tmean and sigma histo for each layer
- `DTNoiseComputation`: produces histograms with the number of cells with average noise under a fixed threshold (granularity: per station) and compute a final histogram with the noise time dependence as a function of the average noise

The **utilities classes** provided by this package are:

- `DTFakeTTrigESProducer`: utility to store tmean and sigma fake values (same values for all the SL, taken from cfg cards) in the EventSetup to have exactly the same reconstruction with or without accessing the DB
- `DTFakeTT0ESProducer`: utility to store t0 and sigmat0 fake values (zero for all the wires) in the EventSetup to have exactly the same reconstruction with or without accessing the DB
- `ProduceFakeDB`: utility to produce DB of ttrig, vdrift or t0 fake values (same values for all the SL or all the wires, taken from cfg cards)
- `DumpDBToFile`: utility to read the constants from the DB and write them into a textual file
- `DumpFileToDB`: utility to read the constants from a textual file and write them into the DB
- `ShiftTTrigDB`: utility to read the ttrig constants from a DB and fill a new DB after having shifted the ttrig of a given list of chambers with a given list of shift values (chamber list and shift values are read fom the cfg file)

Example of configuration file for each class are provided in the test directory of the package. More information about the code can be found in the CMS Software Manual (updated to CMSSW_1_3_0_pre2 version)

Algorithm actually implemented for the calibration

A complete and detailed description of the calibration algorithms can be found in (4).

ttrig

The rising edge of the time box is fitted with the integral of a Gaussian. The program writes into the table the mean value and the sigma of the gaussian and the ttrig is computed as

```
ttrig = <t> + k * sigma
```

where k is a factor tuned to minimize the residuals The fit procedure demonstrated to be quite robust against time-box "patologies" and it can be run in un-attended mode. However to control the fit you can run the **same code** used by the automatic procedure in a root macro for a specific histogram:

```
ROOT>TFile file("MyTimeBoxFile.root")
ROOT>.x loadTBPlotter.C
ROOT>TBplotter->plotTimeBox(wheel, station, sector, sl, "fit")
ROOT>TBplotter->setInteractiveFit(1) //to pass by hand the seed of the fit (useful for distorced
ROOT>TBplotter->setInteractiveFit(0) //to switch again to the completely authomatic fitting proce
```

vdrift and hit resolution

Different TMax formulas (up to 6) are applied accordingly to the track pattern (1). Therefore, for each SL six different histograms are filled. The TMax histograms are fitted with a Gaussian function and the drift velocity and the hit resolution are computed as

```
vdrift = L / (2 * TMax)
```

where L is the cell lenght and TMax is the weighted average of the means of the different TMax histograms. The resolution from each TMax histograms is computed as

```
reso = f * sigma
```

where f is a factor which relates the standard deviation (sigma) of the TMax histogram and the resolution. This factor is different for each different TMax formulas. The final resolution is computed as the weighted average of the resolutions computed from each TMax histogram.

To control the TMax histograms fit you can run the **same code** used by the automatic procedure in a root macro for a specific histogram:

```
ROOT>TFile file("MyMeanTimerFile.root")
ROOT>.x loadPlotter.C
ROOT>MTplotter->plotTimeBox(wheel, station, sector, sl, "fit")  //to plot all the TMax together
ROOT>MTplotter->plotTimeBox(wheel, station, sector, sl, "SingleFormula") //to plot each TMax form
ROOT>MTplotter->plotTimeBox(wheel, station, sector, sl, "fitsame") //you can use more then one ot
ROOT>MTplotter->resetColor() //to reset the histo color
ROOT>MTplotter->setRebinning(2) //to rebin histograms
```

NOTE: `DTVDriftCalibration` in order to recognize the track pattern this application runs the 4D segment reconstruction therefore all your "favourite" cards for this task are required.

NOTE: if you are reading real data you need first to calibrate ttrig and then (optionally) vdrift

t0 from test pulses

The main problems into automatizing the t0 calibration are:

- you cannot have an histogram with the test pulse digis for each wire because it would require too much memory (~200.000 wires)
- you don't know a priori the absolute position of the test pulse peak (it can change when you change your electronic chain or your trigger latency)

These problems have been solved with this method:

- The first n events (where n is set by cfg) are used to discover the raw position of the peak: a histogram per layer with the test pulse digis is made. For each of this histo you can have: the right test pulse peak (RMS<5), the noise peak 400 ns before the right test pulse peak (10<RMS<15), spread noise without peak (high RMS).
- The values taken from each layer are plotted in a single histogram for all the sector and its peak is taken as t0 reference.
- The remaining events are used to compute the t0 for each wire: only the digis close to the t0 reference are used (+/-10 TDC counts) and the RMS is computed "on fly".
- The mean difference between even and odd layers of each SL is corrected, being due to the usage of two different electronic test pulse lines.
- Finally the differences between the t0 reference and the absolute t0 of each wire are stored in a DB

noise calibration

The module works both on noise runs (random trigger) and normal runs (in this case only hits registered before the tTrig are considered).
The module study various aspects of the observed noise:

- **the noise rate**: plots of per wire occupancies normalized to the data taking time
- **the geometrical pattern of noise**: the position of all the wires with enough hits to be called "noisy" is reported in dedicated plots
- **the time dependence of noise**: the distance between a noise event and the following one is used to define an exponential function which describes the noise behaviour VS time. The time constant of the function is finally plotted as a function of the average noise.

Further considerations about calibration

The TMax value depends on the ttrig used (see (2) and (3)). Therefore, the DTCalibration package cannot be used as a black box but great accuracy is needed in the choice of the ttrig.

DB issues

The calibration constants are written into a sqlite DB using the Ronchese utilities (CondFormats/DTObjects). The switch to ORCON or Frontier DB is a matter of configuration in the .cfg file. For the comfort of the user also two application to dump the DB into a textual file and viceversa are provided (DumpDBToFile, DumpFileToDB): with these applications you can manage channels map, noise, t0, ttrig and vdrift databases.

Activities & Results

Real data from DT commissioning
Simulated data from CSA07

TODO

- kfactor optimization

References

1. `CMS-TN/95-01` Appendix A and B
2. `CMS NOTE 2003/007`
3. `CMS IN 2003/020`
4. `CMS NOTE 2007/034`

Review Status

Editor/Reviewer and date	Comments
Main.sbologne - 05 Feb 2007	page author

Responsible: Main.sbologne (Sara Bolognesi)
Last reviewed by: Reviewer

This topic: CMSPublic > SWGuideDTCalib
Topic revision: r21 - 2008-06-04 - GiorgiaMila