11.04.2008
ALICE Offline week

Michal Oledzki
University of Jyväskylä Finland

ALICE T0
Detector Algorithms
Preprocessor

Outline

- Status of DAs
- Preprocessor
- DCS part
- Conclusions

Detector Algorithms

Run types:

- STANDALONE – calibration events
- PHYSICS – physics events

DAs:

- T0Physda – equalizing channels
- T0Laserda – walk correction
- T0Cosmicda – compilation of 2 previous ones

DA

- T0Physda, T0Cosmicda – set and configured on monitoring machine: *aldaqdqm14*
- Takes an input parameters file from DAQDB (going to be changed before next cosmic run)
- Collects histograms which are processed by Preprocessor
- Sends an output file to the FXS with corresponding file id.

Preprocessor

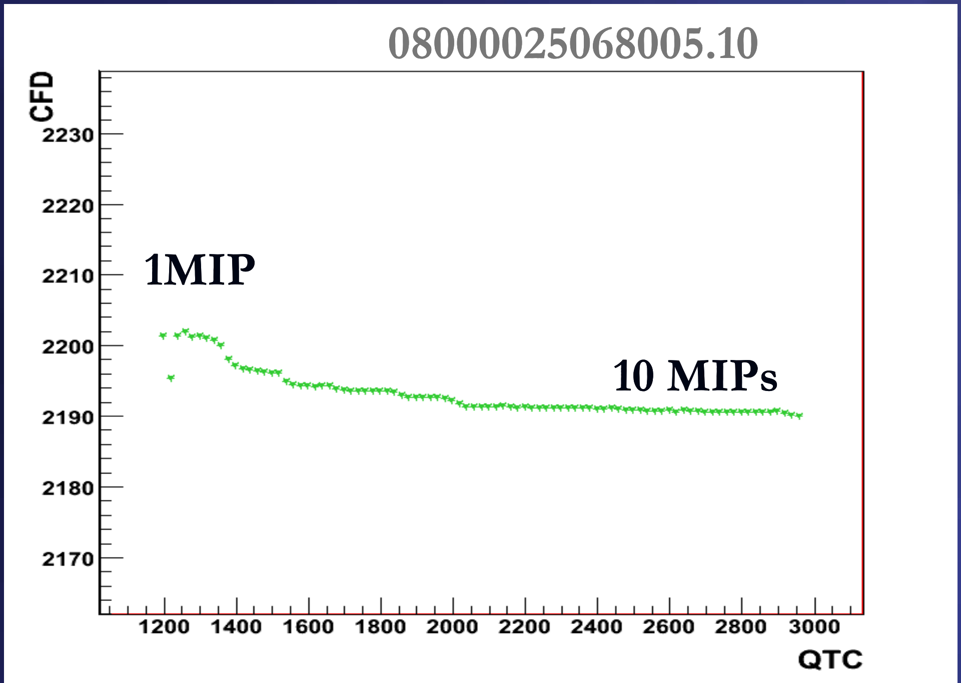
- AliT0PreprocessorCosmic – to be unified with *AliT0Preprocessor*
- Consists of 3 main methods:
  - ProcessDCSDataPoints()
  - ProcessLaser()
  - ProcessPhysics()
- Has been set up and was working together with DAs.
- Modified to work only with Laser Calibration System (also in global run)
- Process data from DAs and stores results in OCDB.
- DCS part still missing, to be done before next cosmic run.

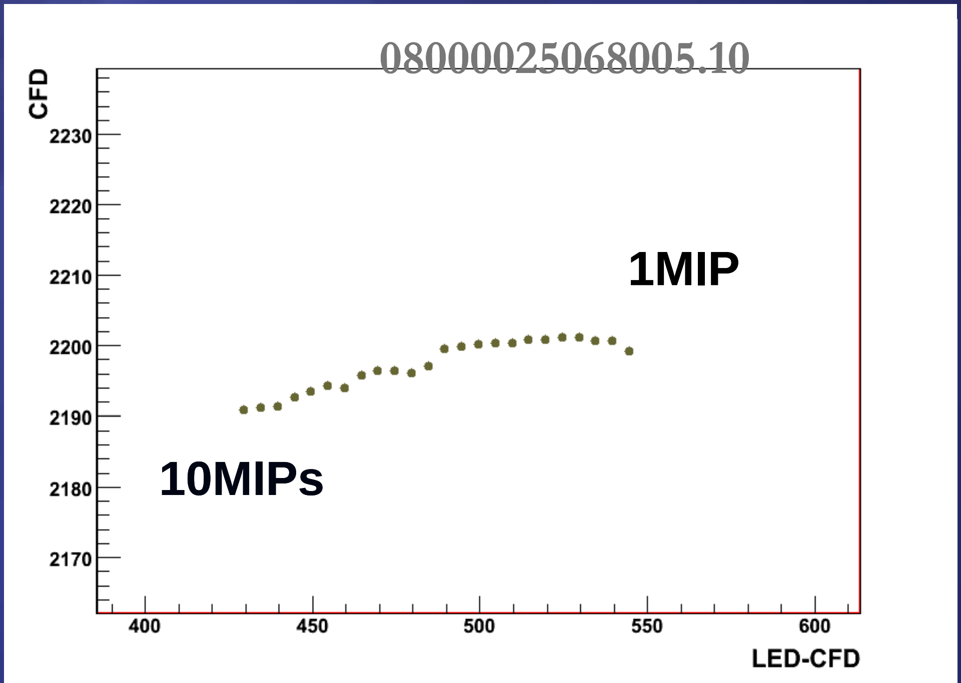
Preprocessor

- ProcessLaser() - works during Laser Calibration Runs.

It makes amplitude-time corrections in two ways:

QTC (charge-time converter) and as the difference of time measurements by Constant Fraction Discriminator and Leading Edge Discriminator (LED-CFD)





Preprocessor

- ProcessPhysics() - currently is also a part of ProcessLaser() method. (The only way to check if it works is using our LCS.)

  It process histograms with CFDtime1-CFDtime(i) on each side collected by T0Physda. The preprocessor gets the mean of these histograms and writes it into OCDB to be used in reconstruction as the value of shift between channels for equalizing.

  The preprocessor should fit histograms and write in the OCDB the mean, sigma and number of entries. But, because of not fully tuned electronics, the fitting method used previously was not fully functional.

T0 DCS Data Points

DCS alias	N of channels	Data type	Unit	Value	% fluctuation	Update Frequency (s)
t00_a_hv_imon_[0..11]	12	float	uA	83	0.9	300
t00_a_hv_vmon_[0..11]	12	float	V	1325	0.85	300
t00_a_lv_imon_[0..1]	2	float	uA	83	0.9	300
t00_a_lv_vmon_[0..1]	2	float	V	6	0.15	300
t00_c_hv_imon_[0..11]	12	float	uA	83	0.9	300
t00_c_hv_vmon_[0..11]	12	float	V	1325	0.85	300
t00_c_lv_imon_[0..1]	2	float	uA	83	0.9	300
t00_c_lv_vmon_[0..1]	2	float	V	6	0.15	300
t00_a_cfd_thre_[0..11]	12	float	V	0.5	10	300
t00_a_cfd_walk_[0..11]	12	float	V	-0.1	10	300
t00_c_cfd_thre_[0..11]	12	float	V	0.5	10	300
t00_c_cfd_walk_[0..11]	12	float	V	-0.1	10	300
t00_ac_scaler_[0..31]	32	float	1/s	3*10^8	50	300
t00_ac_trm_[0..19]	20	float	Cº	35	3	300
t00_ac_drm	1	float	Cº	35	3	300

AliT0Preprocessor: processed with AliT0DataDCS, results stored to Reference DB

DCS part of the preprocessor

- Tested with T0 scaler output being randomly generated distributions

- Computing the needed parameters, e.g. mean, in AliT0DataDCS

```
aliasEntr[j] = aliasArr->GetEntries();
for(int l=0; l<aliasEntr[j]; l++)
{
  AliDCSValue *aValue=dynamic_cast<AliDCSValue*> (aliasArr->At(l));
  t0_scaler[j]+= aValue->GetFloat();
}
fScalerMean[j] = t0_scaler[j] / aliasEntr[j] ;
```

- Storing the results to Reference DB

- To be tested with the data points generated by DCS, when t00_ac_scaler_[0..31] implemented in T0 DCS software

Conclusions

- We are using T0Cosmicda to check that both calibration methods work.
- We need to:
  - modifie calibration method for physics case
  - stop using input files for DA
  - Unifie AliT0Preprocessor
  - DCS part to be done
- All changes should be done a.s.a.p. The latest by the end of April.