Table of Contents

TbUT analysis software

This page provides information about the TbUT package. The software is a part of the Kepler project. The main aim of this software is to perform whole processing of the testbeam data using Gaudi framework.

Contact persons

Christopher Betancourt Mail, Iaroslava Bezshyiko Mail, Steven Blusk Mail, Adam Dendek Mail

Installation

- To install (you need to be in the lhcb computing group on lxplus):

```
lb-dev Kepler v3r0
cd ./KeplerDev_v3r0
git clone https://github.com/yabezsh/Tb.git
make -j4 make install
```

- Make sure, that in your config file (in Tb/.git), you have a line that has a URL with your username included:

```
     url = https://sblusk@github.com/yabezsh/Tb.git
```

Setup

- After each time you open a new terminal

```
SetupProject LHCb
cd ~/KeplerDev_v3r0
./run bash
source /afs/cern.ch/project/eos/installation/lhcb/etc/setup.sh
eosmount ~/eos
cd Tb/
```

- To run over the data from May testbeam: For tests, please, go to Analysys.py and Analysis_onlyPlots.py and change the line

outputPath = "$KEPLERROOT/eos_"+str(sigFile.split('-')[3])+"/lhcb/testbeam/ut/TemporaryData/May2016/DQMTest" to the folder where you want to have your outputs.

Running the TbUT software

For tests, please, go to Analysys.py and Analysis_onlyPlots.py and change the line outputPath = "$KEPLERROOT/eos_"+str(sigFile.split('-')[3])+"/lhcb/testbeam/ut/TemporaryData/May2016/DQMTest" to the folder where you want to have your outputs.

After that you can run (Oct 2016 data example):

```
python -u RunAnalysis.py --board A3_Lower  --PA TTV2_01 --DUTRun 2259 --mode local --mask 1
```

Here, a May 2016 data example:

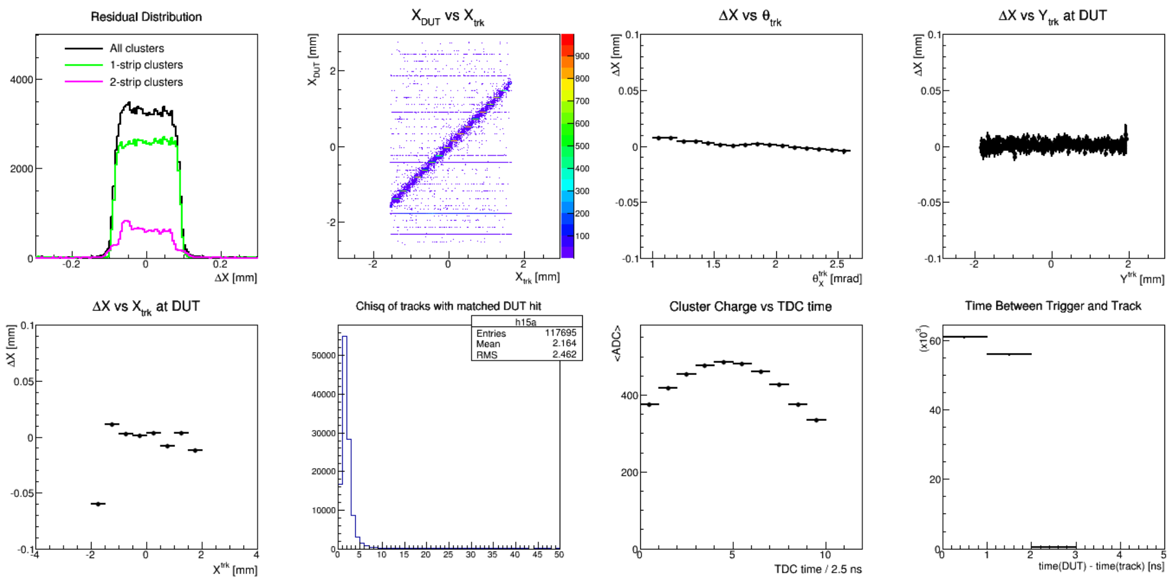
```
python -u RunAnalysis.py --board F1 --PA FanUp --DUTRun 1073 --mode local --evts 100000
```

The latter example will produce all plots in "outputPath"/F1/FanUp/output_1073/Plots/ If you want to update plots without reproducing of all data analyze you can add flag "--onlyPlots 1" If you want to run over all events in run just take away the flag "--evts" (usually 1000000 events; it's better to go over all of them to see nice plots). You can run it in terminal (use "--mode local") or in batch ("--mode batch").

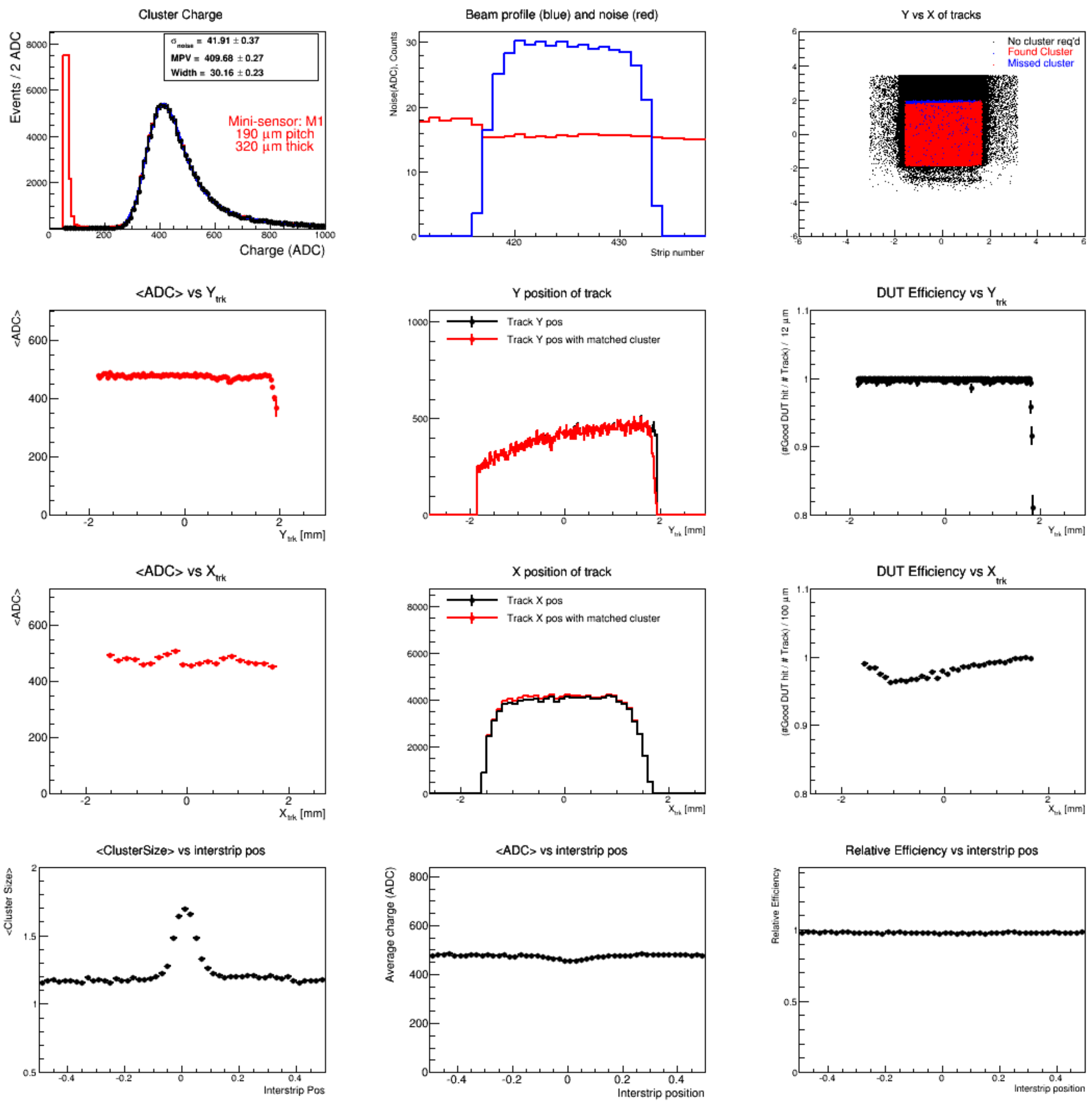
If you will not specify a run number ("--DUTRun 1073") it will start to process all runs from testbeam data sheet (https://docs.google.com/spreadsheets/d/1xYgpb3AxP4JJ7Dkx1YeNV75q0nNxwPrvBhoMgyBD0N4/edit)

Two canvases will be produced, one with Alignment information and one with Performance information. They should be hecked by the user. Below is an example for Run 15046, sensor M1, 300 V bias, 0 degree rotation angle

- **Alignment Plots**



- **Performance Plots**


Cluster Charge
σnoise = 41.91 ± 0.37
MPV = 409.68 ± 0.27
Width = 30.16 ± 0.23
Mini-sensor: M1
190 µm pitch
320 µm thick
Events / 2 ADC
Charge (ADC)
Beam profile (blue) and noise (red)
Noise(ADC), Counts
Strip number
Y vs X of tracks
No cluster req'd
Found Cluster
Missed cluster
<ADC> vs Ytrk
<ADC>
Ytrk [mm]
Y position of track
Track Y pos
Track Y pos with matched cluster
Ytrk [mm]
DUT Efficiency vs Ytrk
(#Good DUT hit / # Track) / 12 µm
Ytrk [mm]
<ADC> vs Xtrk
<ADC>
Xtrk [mm]
X position of track
Track X pos
Track X pos with matched cluster
Xtrk [mm]
DUT Efficiency vs Xtrk
(#Good DUT hit / # Track) / 100 µm
Xtrk [mm]
<ClusterSize> vs interstrip pos
<Cluster Size>
Interstrip Pos
<ADC> vs interstrip pos
Average charge (ADC)
Interstrip position
Relative Efficiency vs interstrip pos
Relative Efficiency
Interstrip position

Known issues

- **The TbUT cannot read eos files. *workaround: copy analyzed file into your work directory or use eosmount as shown above.**
- **TbUT (run mode) need to be executed twice to correct calculate noise and extract clusters.**
- **The tools should inherit from GaudTool class.**
- **The unit test have to be implemented**

Developers section

General overview

Every TbUT component class is a member of `TbUT` namespace.

Algorithms

The TbUT is a Gaudi based application. The application is separated into two sets of Gaudi algorithms. First group is responsible for processing the testbeam data. This algorithms inherit from GaudiAlgorithm class. The names of it contains string `Algorithm` for instanceTbUTRawDataReaderAlgorithm .

To understand TbUT source code, the algorithm classes should be treated as a starting point

The second group was designed to prepare monitoring plots. This classes inherit from base called TbUTDataMonitorAlgorithm (see also implementation). According to the file name convention it's names have to contains string `DataMonitorAlgorithm` in their's file names. E.g. TbUTPedestalSubtractorDataMonitorAlgorithm .

Tools

That kind of classes are responsible of performing the data modification. Each of this inherit from virtual interface called IProcessingEngine, therefore they have to implement `processEvent` method.

Tool's Factories

This kind of classes are responsible for dynamic, depend of options, creation of proper version of tools. This classes based on Factory design pattern (see also the beautiful explanation why is is so useful). See for example Raw Data Reader Factory and it implementation

Other types

- **Data structures (inherits from `GaudiKernel/DataObject`) are used to be manipulated via the tools and be stored in TES. The TbUT package contains Raw Data and Clusters .**
- **Additional data structures. Can be distinguished noise and pedestals. Their are used as a auxiliary structures by the tools.**
- **Additional services. Used for manipulation of the additional data structures. See for instance Noise Calculator. Each of this services has own interface (e.g. Noise calculator are based on INoiseCalculator).**

TbUT high level architecture

Description of the algorithms

Raw Data Reader

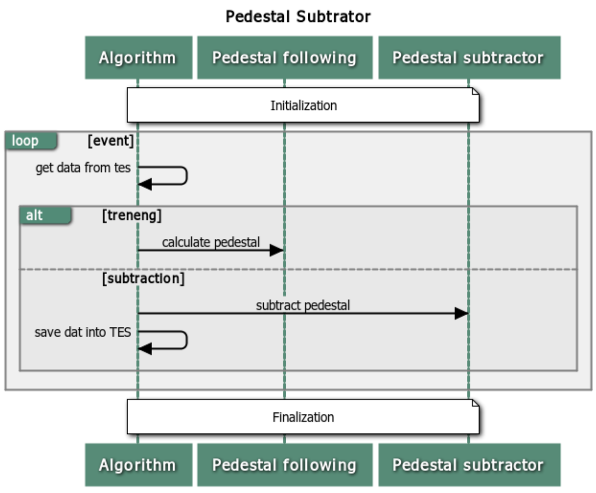
This algorithm is designed to read raw data from specific input file. Currently TbUT can work with files produced by Alibava and Mamba DAQs. The heart of this algorithm is tool inherit from interface IDataReader. This tool acts fascade for specific DAQ reader.

The execution of Raw Data Reader can be modified using options:

option name	brief description
InputDataType	type of string, choose one of DAQs data format (Mamba or Alibava)
inputData	type of string, path to the input data.
standalone	type of boolean, if is true the run is executed with Kepler (haven't been tested yet)

Pedestal Subtractor

The pedestal subtraction algorithm has two phases. In first, optional one, the pedestal values are calculated. This phase is also called training. During the second one the determined pedestal values are subtracted from the raw ADC data. From technical point of view the Pedestal Subtractor Algorithm calls two types of tools. One of them calculates or retrieves pedestals values(this classes inherit from I pedestal following) and the second one remove pedestals values from the raw data. See data flow.



The execution of Pedestal Subtractor Algorithm can be modified using options:

option name	brief description
FollowingOption	type of string, choose type of execution (training or run)
treningEntry	type of integer, number of training entry
ChannelMaskInputLocation	type of string, location of the channel mask
PedestalInputFile	type of string, path to the location where previously calculated pedestals are stored
PedestalOutputFile	type of string, path to the location where calculated pedestal have to be stored
standalone	type of boolean, if is true the run is executed with Kepler (haven't been tested yet)

Pedestal following

From the mathematical point of view the calculation of the pedestal can be described as a running average. In every training event then pedestal sum is updated. This update takes into account the previous value of the pedestal sum and the current ADC count. The pedestal sum is calculated for each channel separately. To be more precisely, the pedestal sum, $p_i(n)$, for channel n and event i can be expressed as follows:

$$P_i^{sum}(n+1) = P_i^{sum}(n) + \frac{\Delta_i(n+1)}{N}$$

Where the $\Delta_i(n+1)$ is a event to pedestal correction. This correction is expressed as:

$$\Delta_i(n+1) = ADC_i(n+1) - P_i^{sum}(n)$$

To increase the suitability of the pedestal the limit for correction is applied. If the condition: $|\Delta_i(n+1)| \leq 15$ is not fulfilled the correction value is set to 15.

To determinate the pedestal values the pedestal sum should be normalized, so: $p_i = \frac{P_i^{sum}}{N}$

Pedestal removing algorithm

The second phase of the pedestal subtraction algorithm is subtraction determined pedestal values from the raw data. This procedure can be expressed as fallows:

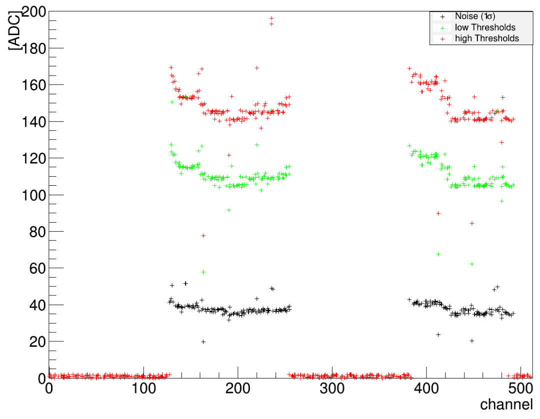
$$ADC_i = ADC_i^{RAW} - p_i$$

where: ADC_i is signal value after pedestal subtraction for event i, ADC_i^{RAW} is a raw data and the p_i is the pedestal value. Each of this quantities are in the unit of ADC counts.

Additional scripts

All scripts are stored in `TbUT/scripts` directory. (Not yet committed)

- **Noise monitor This script takes as an input noise file. The exemplary output:**



Operations with the git repository

- **To update your local copy with the latest version in the git repository (you will need a git account)**

```
git pull
```

- **To make modifications, you will need to be added to the list of users allowed to modify (contact Iaroslava). After you make your changes:**

```
git add
```

* To get a completely fresh working copy, see instructions under Installation, above.

-- AdamMateuszDendek - 2015-08-03

This topic: LHCb > TbUT
Topic revision: r25 - 2016-10-31 - StevenBlusk