Introduction to working with Trigger Data

-- by M. Schimassek (martin.schimassek@ijclab.in2p3.fr), 30.05.2024 --

Repositories: t2raw-example, t2dumps

Data: link

What is covered:

- how to read t2raw files
- how to read t2-dumps

Introduction

What we can cover in this example is a first glimpse into using the low level trigger information to learn about the status of the detector. This introduction will cover how to read and interpret the so-called t2raw-files and T2Dumps. We do not want to encourage doubling existing analysis (e.g. exposure) with tools given in this example.

Due to the size of the files, we won't be able to easily discuss reading the files in ipython notebooks.

T2-raw Data

In this section, we will explore reading t2raw data. It is information on how many T2s a station send for each GPS second. We use this data to calculate the exposure and determine whether a station is alive or not.

Getting the data

To obtain the data, you can use irods and get the data with e.g.

```
iget /pauger/Malargue/Raid/monit/Sd/2024/05/t2raw_2024_05_10_00h00.dat.bz2 .
```

Where ideally you replace the "." with a folder that you want to use to keep the data sorted.

To get the example codes, run

```
git clone git@gitlab.iap.kit.edu:auger-observatory/sandboxes/schimassek-m/
introduction-t2-data.git
```

With this you should have an ipython-notebook called "T2Raw-Introduction.ipynb". Open your

jupyter-notebook from a command line in this directory and open the notebook.

In the notebook you should have instructions on how to read the data and what they are.

For completeness, there is also a python script that does basically the same as the notebook. As it is more difficult to understand, we advise starting with the notebook. If you want to run this example script, simply run

```
./t2raw_reader.py -i t2raw_2024_05_10_00h00.dat.bz2 -o t2raw_2024_05_10 -u
example.uub
```

In this case, the example.uub file contains three station IDs to generate the plots with. You can edit this file or use any other plain text file with station IDs to extract information of other stations. We don't advise running this script with many stations (use < 10 as order of magnitude).

Further analysis

If you are interested in how to read this data in a more rigorous way, to actually perform analysis, you can have a look into the following git-repository: t2raw-utls.

If you want to use this repository, clone it with submodules:

```
git clone --recurse-submodules git@gitlab.iap.kit.edu:auger-observatory/
sandboxes/schimassek-m/t2raw-utls.git
```

The repository is not well documented or written, but you can use "t2raw_checks.cxx" as an example on how to read the data in C++. To compile the code, you need root-6 and boost installed.

With root sourced, you should be able to compile the code by typing "make".

Then you can run it with

```
./t2raw_checks -i t2raw_2024_05_10_00h00.dat.bz2 -o t2raw_2024_05_10 --start
1399334400
```

The --start option specifies the starting GPS-second which is necessary to correctly separate UB/UUB based on the software versions provided (in t2dumpcore). If you don't specify the GPS second the output might not make a lot of sense.

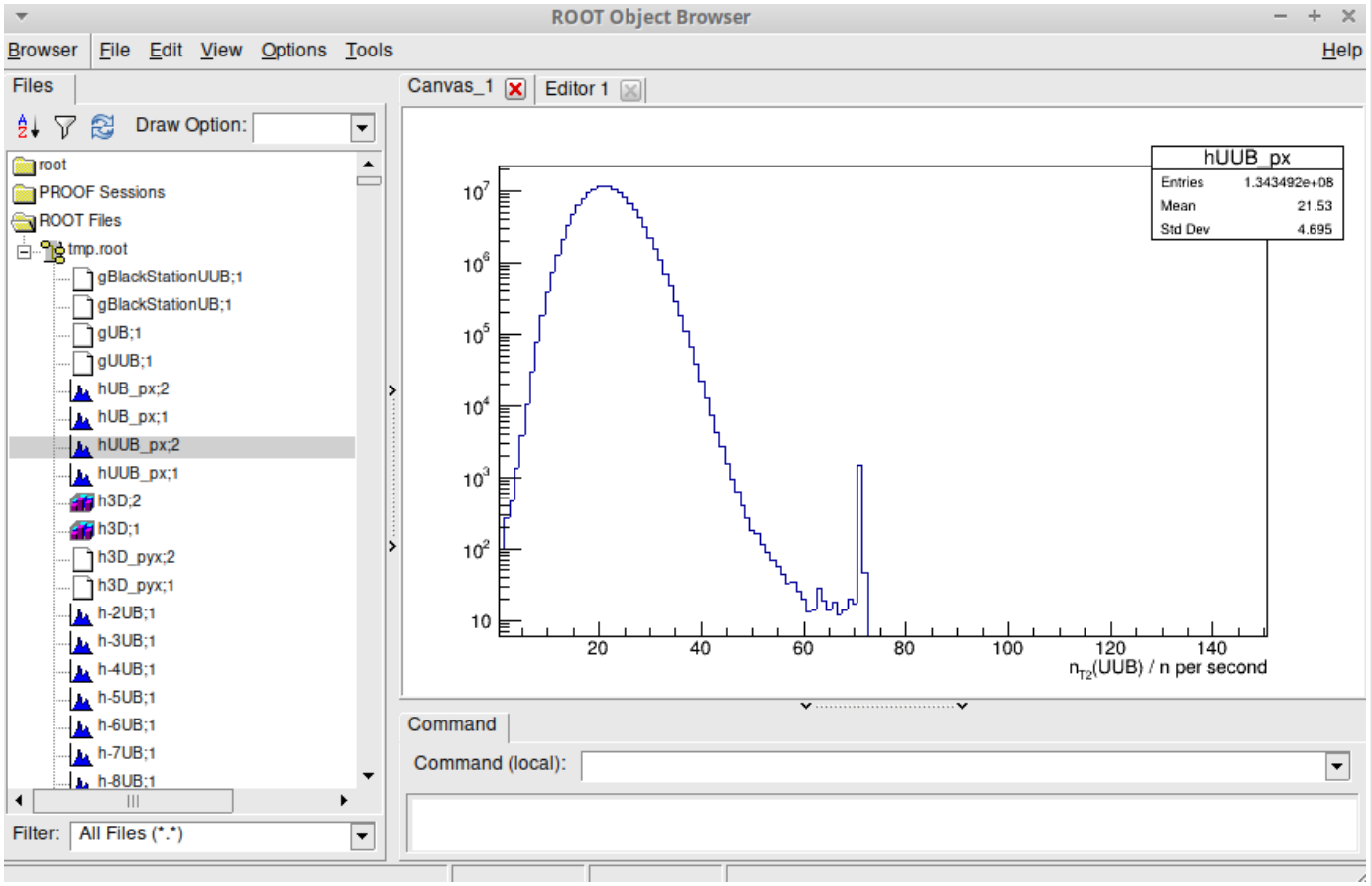
You will see a number of lines on your console that are warnings about UUB-software versions, e.g.

```
unknown version V32R49B50P48
V128R0B0P5 UUB version -> defaulting to UUBDefault
V128R0B0P7 UUB version -> defaulting to UUBDefault
```

You can ignore these for now, as they are not affecting the data we are interested in (for more detail on why these versions can be important see the T2Dump reading part).

The output file will be called t2raw_2024_05_10.root and contain a number of potentially interesting plots. You can access them with the root TBrowser easily.

```
root t2raw_2024_0510.root
[0] new TBrowser()
```



The screenshot above shows e.g. the total distribution of n(T2) / s for all UUB-stations [technicality: at the time of writing the necessary "data base" of station software versions is not updated to May 2024 yet, so it can have inaccuracies in determining UUB/UB].

The other plots "gBlackStationUB/UUB" are a plot of the fraction of black UB/UUB stations over time, "gUB/UUB" is the average T2-rate for the two sets. There are histograms for each station, where the suffix tell you if the code has it classified as UB or UUB.

T2Dumps

A second data type in use for information on the SD-detector are the so-called T2Dumps. This

data is the collection of all T2s send to the central CDAS and what is used to form event level T3-triggers. Since 2016, we also have all this data "on tape" for offline analysis. But be aware, these are by far the largest files collected by the SD! We are talking about 500 MB per hour, i.e. 12 GB / day, compared to roughly 30 GB / month for the event data. As such, it is very challenging to do comprehensive analysis with simple tools.

In this "tutorial", we will explore existing code to read these flies, and do very simple analyses to check the data we see there.

Obtaining the data

As in all other cases, we first need to get the data we want to analyze. For the purpose of this example a single hour of data is sufficient. Through irods, you can download it with

```
iget /pauger/T2Dump/2024/02/t2dump_2024_02_21_22h00.dat.bz2 / ... /T2Dumps/
```

where as usual you use a directory suitable for storing the data on your machine.

Should you be unable to download the data through irods, you can find them as well in the google-drive folder for this folder: link. We chose a file from February explicitly because it contains thunderstorm periods. For fast downloads (file caching in Lyon), you can also download any recent file. You can see the latest with

```
ils /pauger/T2Dump/2024/06/
```

The Source code

The code we will use for this example is the "t2dump-misc" repository as it contains the easiest applications of T2Dump-reading. Note that it is not developed and cleaned for this tutorial, so expect some features and application to be left unexplained here.

You can clone the repository with

```
git clone --recurse-submodules git@gitlab.iap.kit.edu:auger-observatory/
sandboxes/schimassek-m/t2dump-misc.git
```

For compilation, you need boost (command line options) and root-6. With root-6 sourced, you should be able to compile the code simply by typing "make".

Extracting T2dump data

We now want to use a specific program in this repository to extract the T2 data for a single station. Without any fundamental reason, we use station J.W.Cronin (1097) as example.

Run

```
./dump_extended_station_triggers -i /.../T2Dumps/
t2dump_2024_02_21_22h00.dat.bz2 -o t2s_2024_02_21_22h00_1079 --station 1097
```

The expected run-time is about 30s to one minute and you will obtain an output-file called "t2s_2024_02_21_22h00_1097.dump". Also here you will see a number of warnings printed for station versions. You can ignore these.

If you are unable to compile or run these programs, you can find the output files in the google-drive folder of this tutorial.

Understanding the output

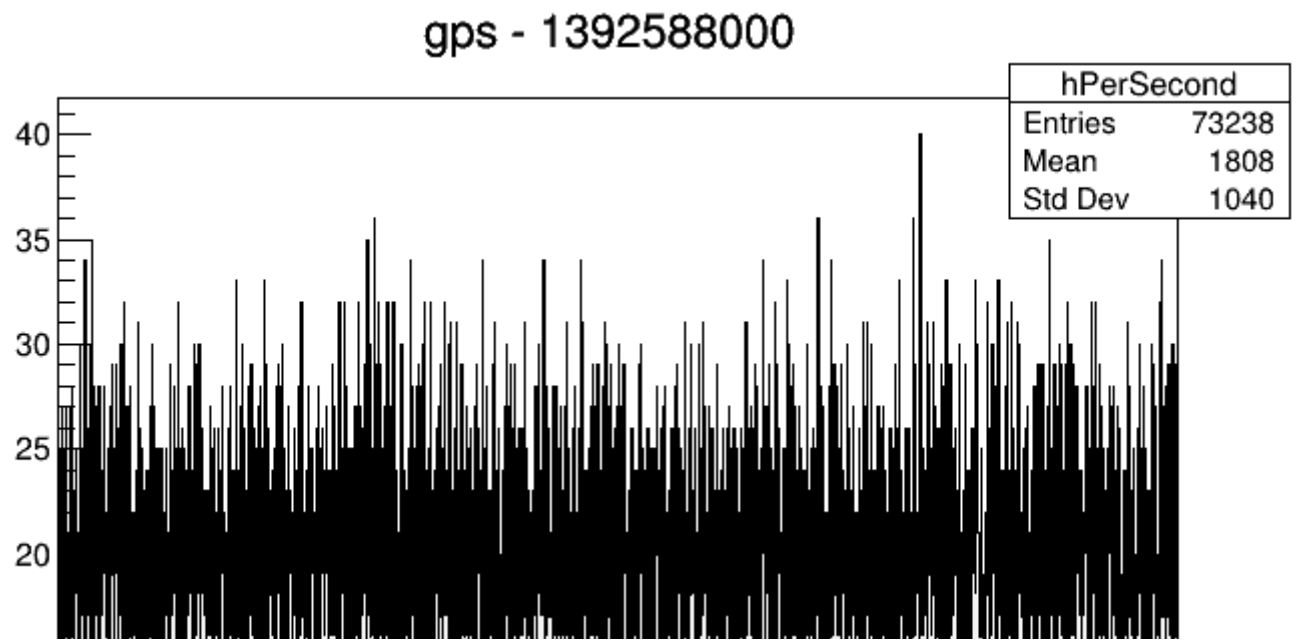
You can open the output file and see that in principle it is human readable.

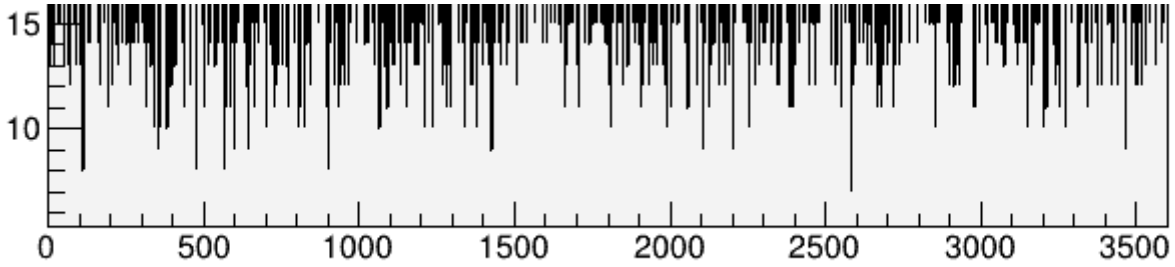
```
gps/I:us:type:isvalid:uub:istot:istotd:ismops
1392588000 21643 1 1 1 0 0 0
1392588000 29742 1 1 1 0 0 0
1392588000 56707 1 1 1 0 0 0
1392588000 95116 1 1 1 0 0 0
1392588000 104705 1 1 1 0 0 0
```

The column descriptor of this file is made for root's TTree::ReadFile(). This mean, you can read this data easily with the root command line.

```
root [0] TTree t;
root [1] t.ReadFile("t2s_2024_02_21_22h00_101.dump")
(long long) 73238
root [2] t.Draw("gps - 1392588000 >> hPerSecond(3600, 0, 3600)")
Info in <TCanvas::MakeDefCanvas>:  created default TCanvas with name c1
root [3]
```

Would give you a plot of the T2-rate of this station per second in the hour analyzed:





As the TTree::Draw interface is somewhat obscure, we will analyze the data also in a pandas data frame in a ipython notebook. You can find this notebook here: notebook.

Additional Information

The key in understanding the trigger flags is the encoding done in the station. This encoding is changing with software-versions and as such, the code reading the data has to be adaptable.

In this repository, we have the "t2dumpcore" submodule. It includes utilities that try to make it easily understandable what happens, while the ugly switching of interpretation is hidden for the analysis use.

You can have a look at "VersionProvider.h" for the interface and VersionProvider.cc for the (ugly) decoding implementation.

A list of different software versions is available in Constants.h

We strongly recommend to use this software as starting point for any analysis of the T2Dumps. There are cases of "semi-broken" files that can be recovered with some rather ad-hoc solution. It is already included here.