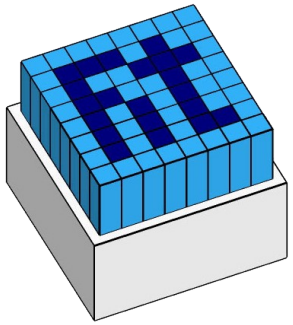


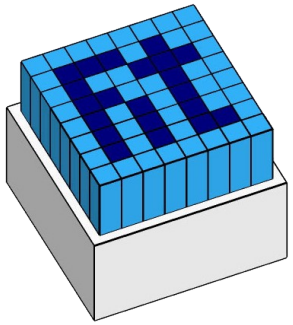
Tutorial (2)

Held before FIT beam test @ PS (06.2016)

30.05.2016, Vidyo

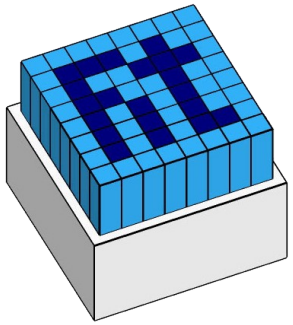
Maciej Slupecki
University of Jyvaskyla, Finland

UNIVERSITY OF JYVÄSKYLÄ



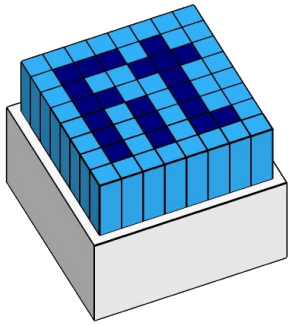
Part 2 – outline

- DAQ scheme
- Data collection
  - DAQ program, binary file format
  - Examples of raw waveforms
  - Which waveform qualities are interesting?
- Data analysis chain
  - Raw waveform analysis – existing solution
    - DigDataProcessing
      - Usage instructions
      - Format of output root file
  - Root file analysis
    - TBrowser (exercise)
    - Simple script: genericTreeReader (exercise)
    - More advanced program: fitTest-2016-06 (work-in-progress)
- DigDataProcessing modification



Part 2 – outline

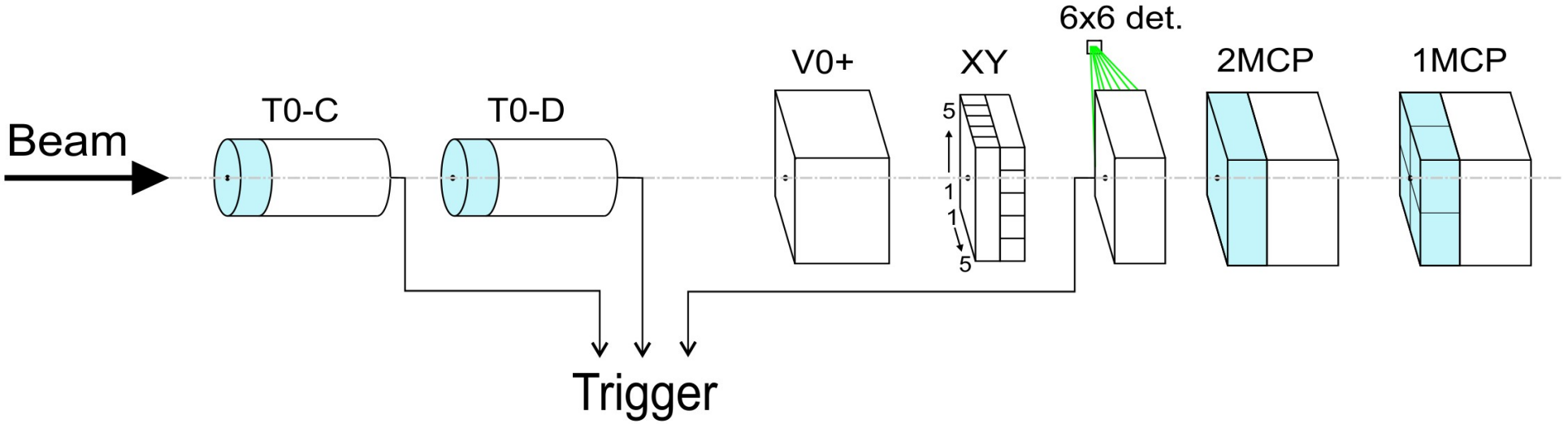
- DAQ scheme
- Data collection
  - DAQ program, binary file format
  - Examples of raw waveforms
  - Which waveform qualities are interesting?
- Data analysis chain
  - Raw waveform analysis – existing solution
    - DigDataProcessing
      - Usage instructions
      - Format of output root file
  - Root file analysis
    - TBrowser (exercise)
    - Simple script: genericTreeReader (exercise)
    - More advanced program: fitTest-2016-06 (work-in-progress)
- DigDataProcessing modification

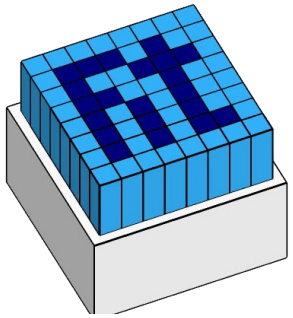


Part 2 – outline

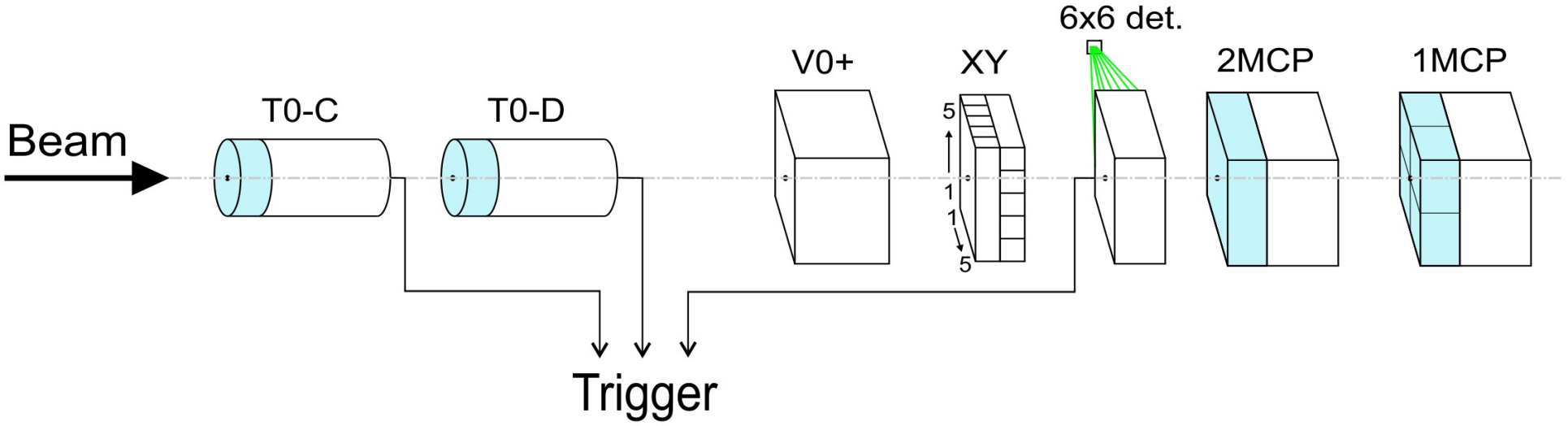
- DAQ scheme
- Data collection
  - DAQ program, binary file format
  - Examples of raw waveforms
  - Which waveform qualities are interesting?
- Data analysis chain
  - Raw waveform analysis – existing solution
    - DigDataProcessing
      - Usage instructions
      - Format of output root file
  - Root file analysis
    - TBrowser (exercise)
    - Simple script: genericTreeReader (exercise)
    - More advanced program: fitTest-2016-06 (work-in-progress)
- DigDataProcessing modification

DAQ scheme





DAQ scheme

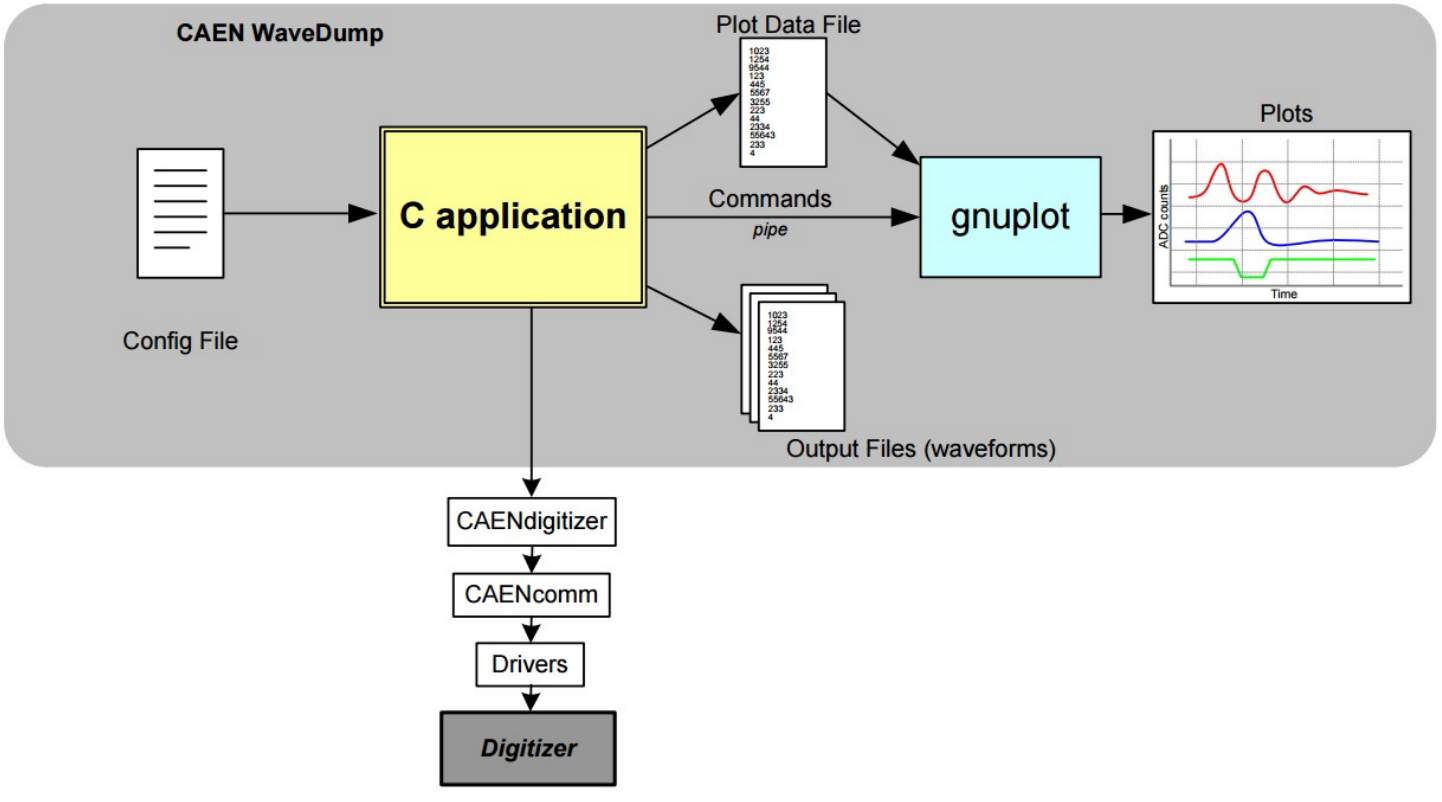


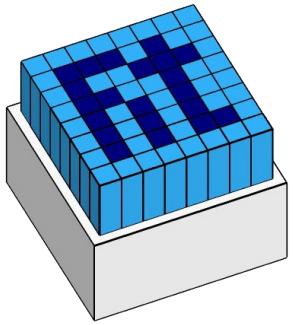
Name	T0-C	T0-D	V0+	XY	6x6 det	2MCP*	1MCP*
Detector type	PMT	PMT	PMT / MCP / SiPM	Custom	Custom	MCP	MCP
Number of signals	1	1	1-16?	2	1	4-5	4-5
Triggering?	yes	yes	no	no	(yes, if present)	no	no

*any of the following is possible: 1MCP, 2MCP, 4MCP, 5MCP

Digitizer & DAQ program

- Digitizer: CAEN DT5742
  - http://www.caen.it/jsp/Template2/CaenProd.jsp?parent=14&idmod=651
- DAQ program: WaveDump
  - http://www.caen.it/jsp/Template2/CaenProd.jsp?idmod=692&parent=38
  - Needs a configuration file (which should be prepared by experts)





WaveDump usage

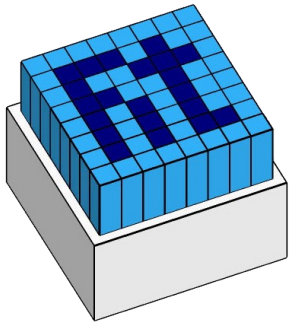
[space]	Display help (full command list)
q	quit WaveDump
s	**start data acquisition***
W [shift+w]	**enable continuous writing to disk**
p	plot a single event
P [shift+p]	enable continuous event plotting
0..7	enable or disable plotting waveform from a given channel
g	switch to the next group of 8 channels**

*** The data is not being saved to the disk at this point**

** The plotter is able to plot 8 channels at maximum, so to plot the next group of 8 channels the 'g' switch is introduced

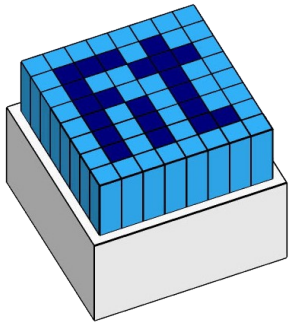
The algorithm to start DAQ is following:

- Start WaveDump and press 's'
- Check from the terminal that some data is being collected and/or start plotting by pressing 'P'
- If all is ok, start writing to disk: 'W'



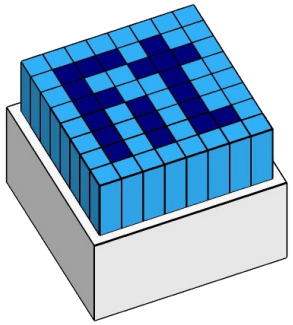
Data analysis chain
Recap

- → Detectors
- → FE electronics
- → CAEN Digitizer
- → DAQ program
- → **Binary files**
- → Waveform analysis
- → Root file
- → Final analysis



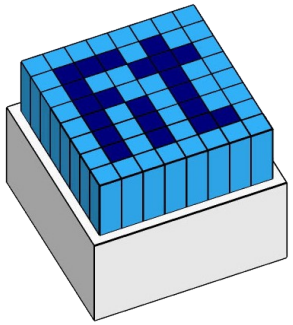
Binary data format

- Data from every channel goes to a separate file → acquisition directory contains the following files:
  - wave_0.dat, wave_1.dat, …, wave_15.dat
- Each file contains just the data, there are no headers, tailers or consistency error bits, for example **wave_0.dat** has the following structure:
  - event 0:
    - starts at 0th byte of the file
    - ends at 4095th byte
    - contains **1024** float numbers (**f**)
    - each **f** is 4 bytes long and represents the voltage (measured in arbitrary units) in channel zero (coming from wave_0.dat file) at the time (200 ps * f_position), where f_position ranges from **0** to **1024**
    - The whole event (set of **1024 f**-numbers) forms a waveform (or oscillogram) with the total acquisition length of 204.8 ns and sampling rate of 5Gs/s
  - event 1:
    - starts at 4096th byte of the file
    - ends at 8191th byte
    - ...



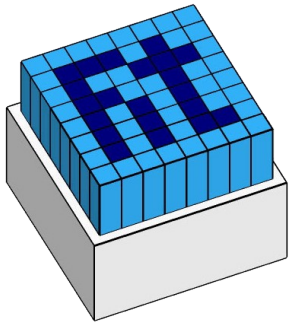
Binary data format

- Data from every channel goes to a separate file → acquisition directory contains the following files:
  - wave_0.dat, wave_1.dat, …, wave_15.dat
- Each file contains just the data, there are no headers, tailers or consistency error bits, for example **wave_0.dat** has the following structure:
  - event 0:
    - starts at 0th byte of the file
    - ends at 4095th byte
    - contains **1024** float numbers (**f**)
    - each **f** is 4 bytes long and represents the voltage (measured in arbitrary units) in channel zero (coming from wave_0.dat file) at the time (200 ps * f_position), where f_position ranges from **0** to **1024**
    - The whole event (set of **1024 f**-numbers) forms a waveform (or oscillogram) with the total acquisition length of 204.8 ns and sampling rate of 5Gs/s
  - event 1:
    - starts at 4096th byte of the file
    - ends at 8191th byte
    - ...



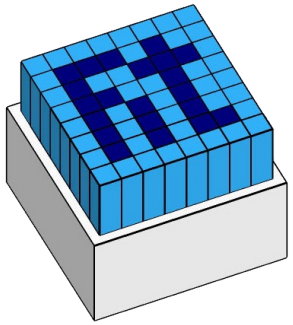
Binary data format

- Data from every channel goes to a separate file → acquisition directory contains the following files:
  - wave_0.dat, wave_1.dat, …, wave_15.dat
- Each file contains just the data, there are no headers, tailers or consistency error bits, for example **wave_0.dat** has the following structure:
  - event 0:
    - starts at 0th byte of the file
    - ends at 4095th byte
    - contains **1024** float numbers (**f**)
    - each **f** is 4 bytes long and represents the voltage (measured in arbitrary units) in channel zero (coming from wave_0.dat file) at the time (200 ps * f_position), where f_position ranges from **0** to **1024**
    - The whole event (set of **1024 f**-numbers) forms a waveform (or oscillogram) with the total acquisition length of 204.8 ns and sampling rate of 5Gs/s
  - event 1:
    - starts at 4096th byte of the file
    - ends at 8191th byte
    - ...



Binary data format

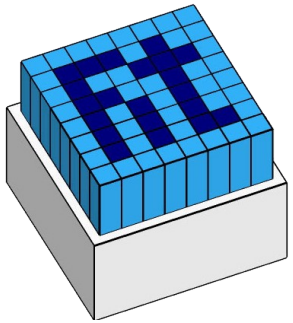
- Data from every channel goes to a separate file → acquisition directory contains the following files:
  - wave_0.dat, wave_1.dat, …, wave_15.dat
- Each file contains just the data, there are no headers, tailers or consistency error bits, for example **wave_0.dat** has the following structure:
  - event 0:
    - starts at 0th byte of the file
    - ends at 4095th byte
    - contains **1024** float numbers (**f**)
    - each **f** is 4 bytes long and represents the voltage (measured in arbitrary units) in channel zero (coming from wave_0.dat file) at the time (200 ps * f_position), where f_position ranges from **0** to **1024**
    - The whole event (set of **1024 f**-numbers) forms a waveform (or oscillogram) with the total acquisition length of 204.8 ns and sampling rate of 5Gs/s
  - event 1:
    - starts at 4096th byte of the file
    - ends at 8191th byte
    - ...



Wake up question

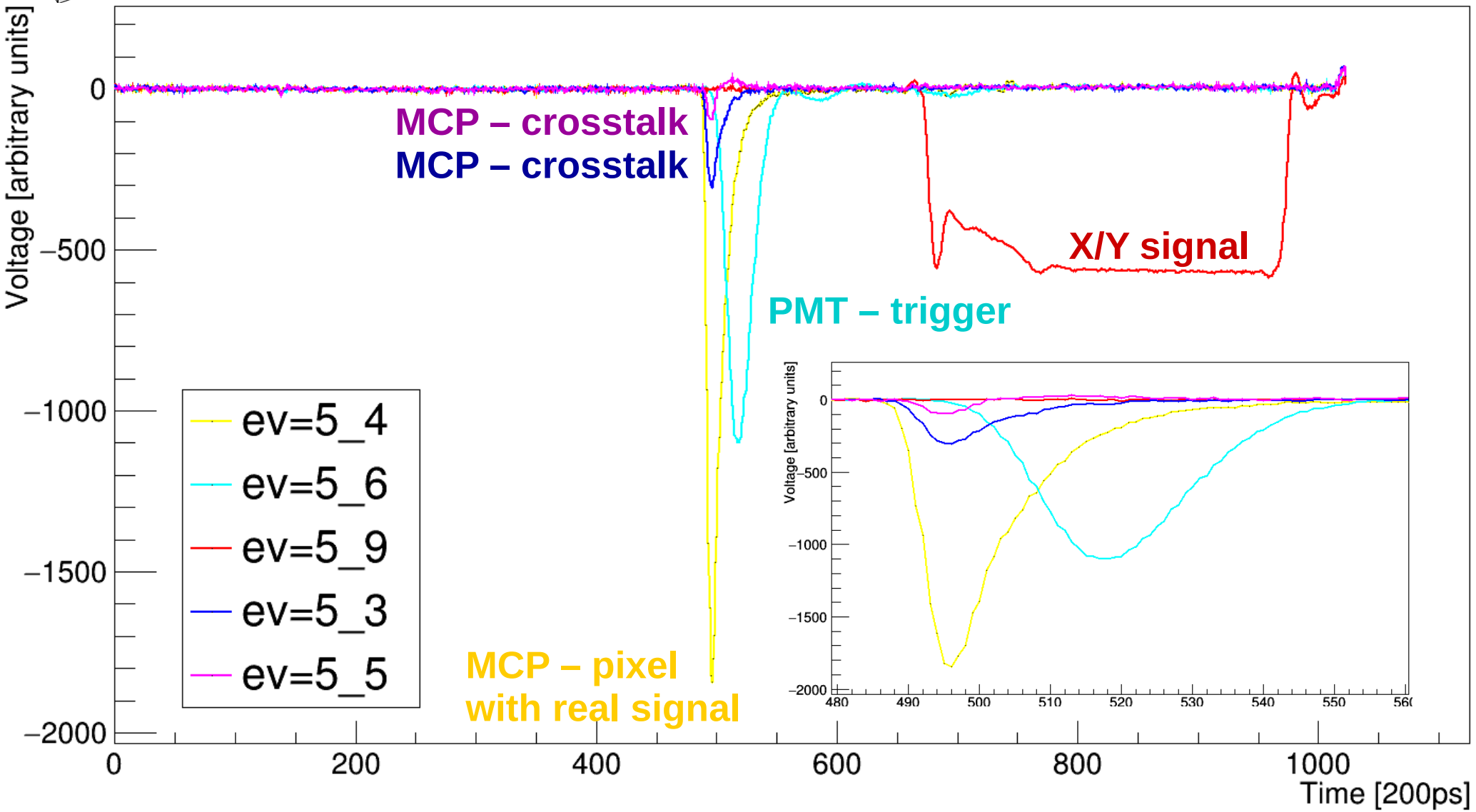
Keeping in mind that the sampling rate of the digitizer is 5 Gs/s → **200 ps.**

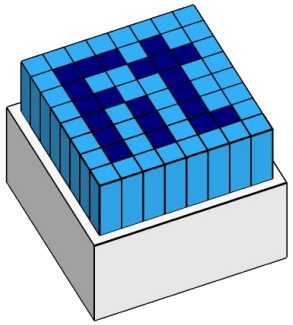
How can we study the detector **time** features (resolution) with **~10 ps accuracy**?



Raw waveforms

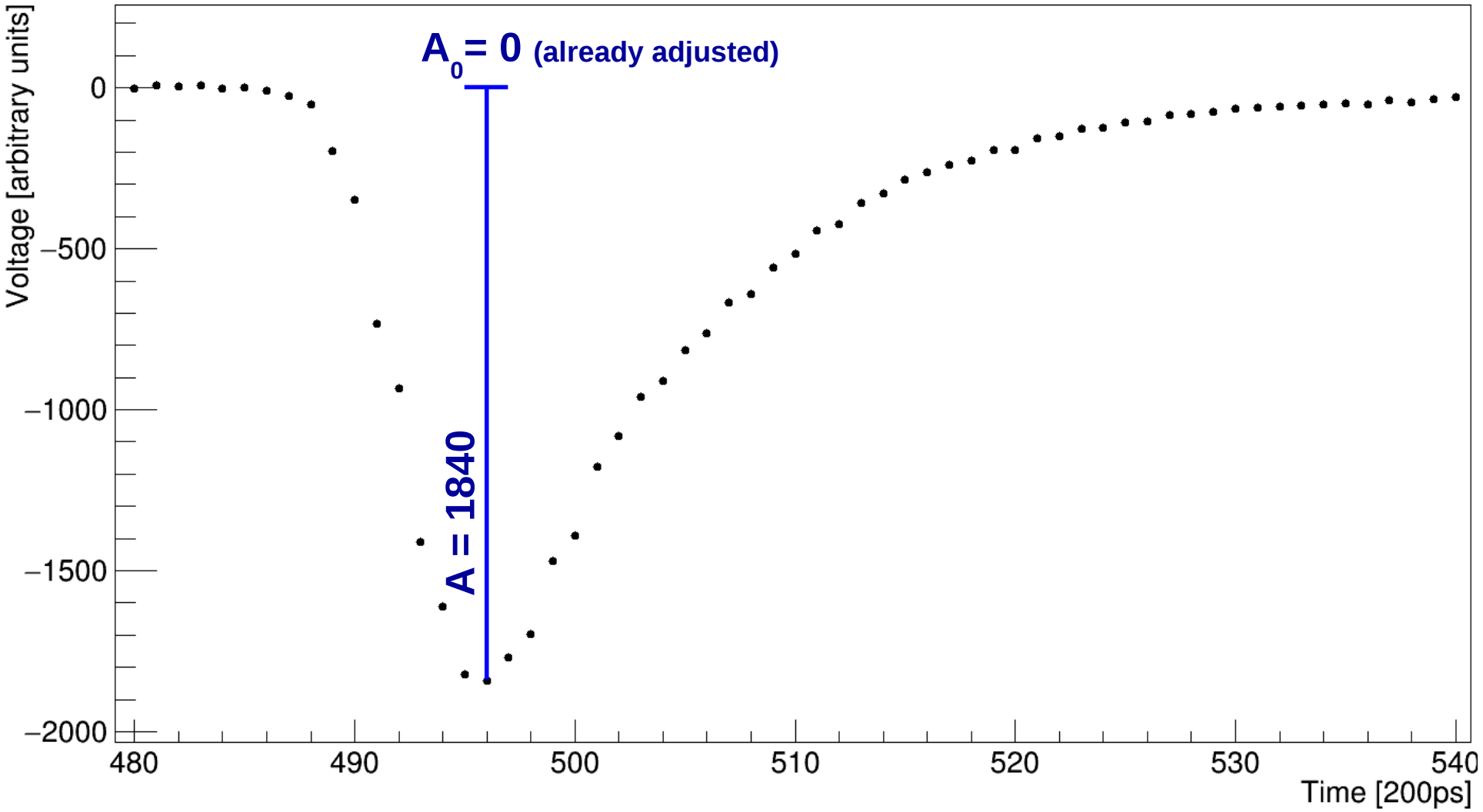
ALICE
A JOURNEY OF DISCOVERY

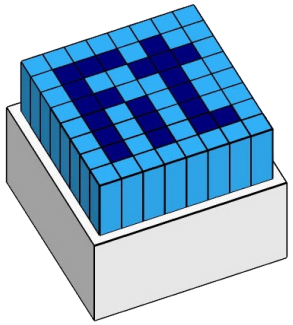




Raw waveforms

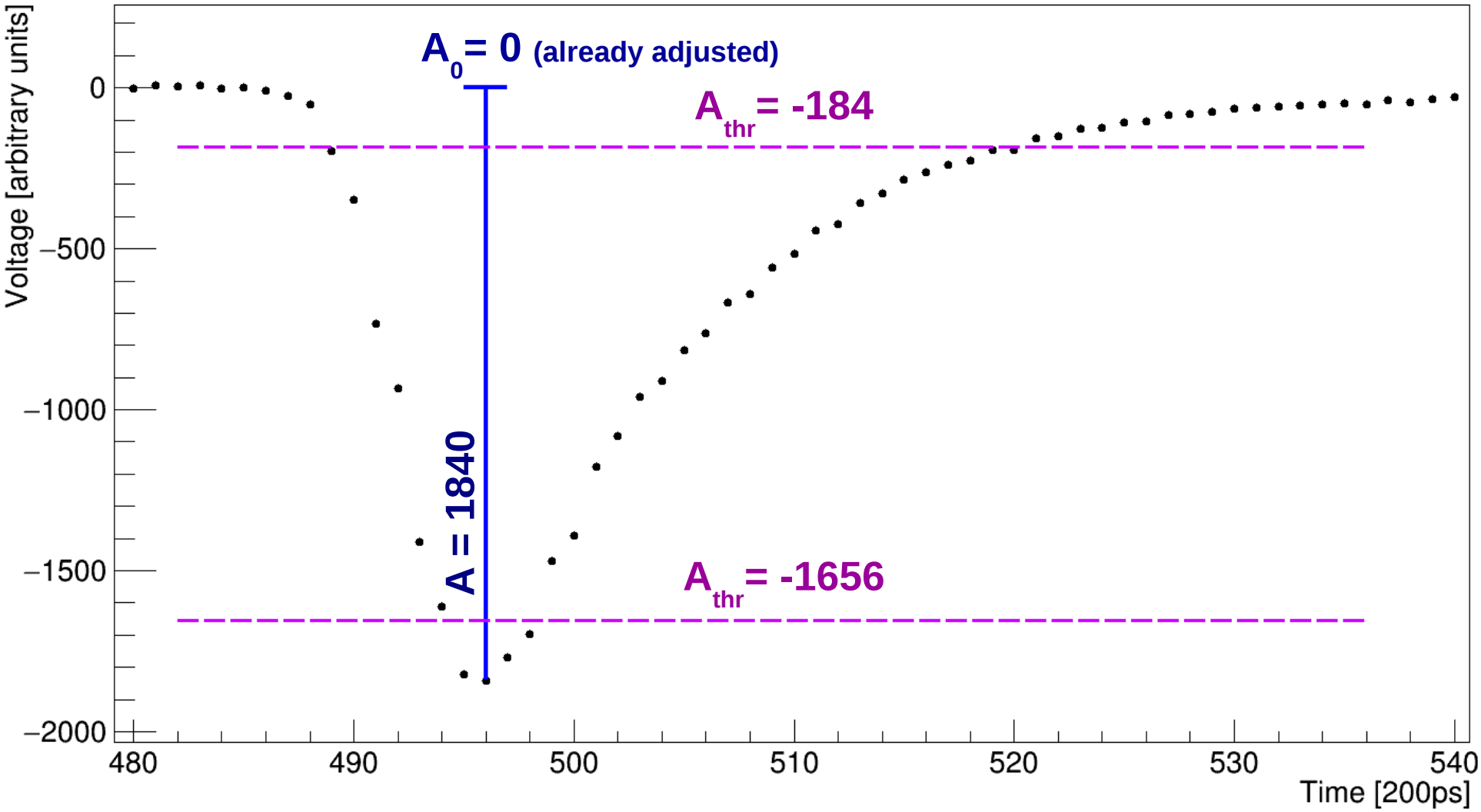
0: Find zero-level
1: Find amplitude





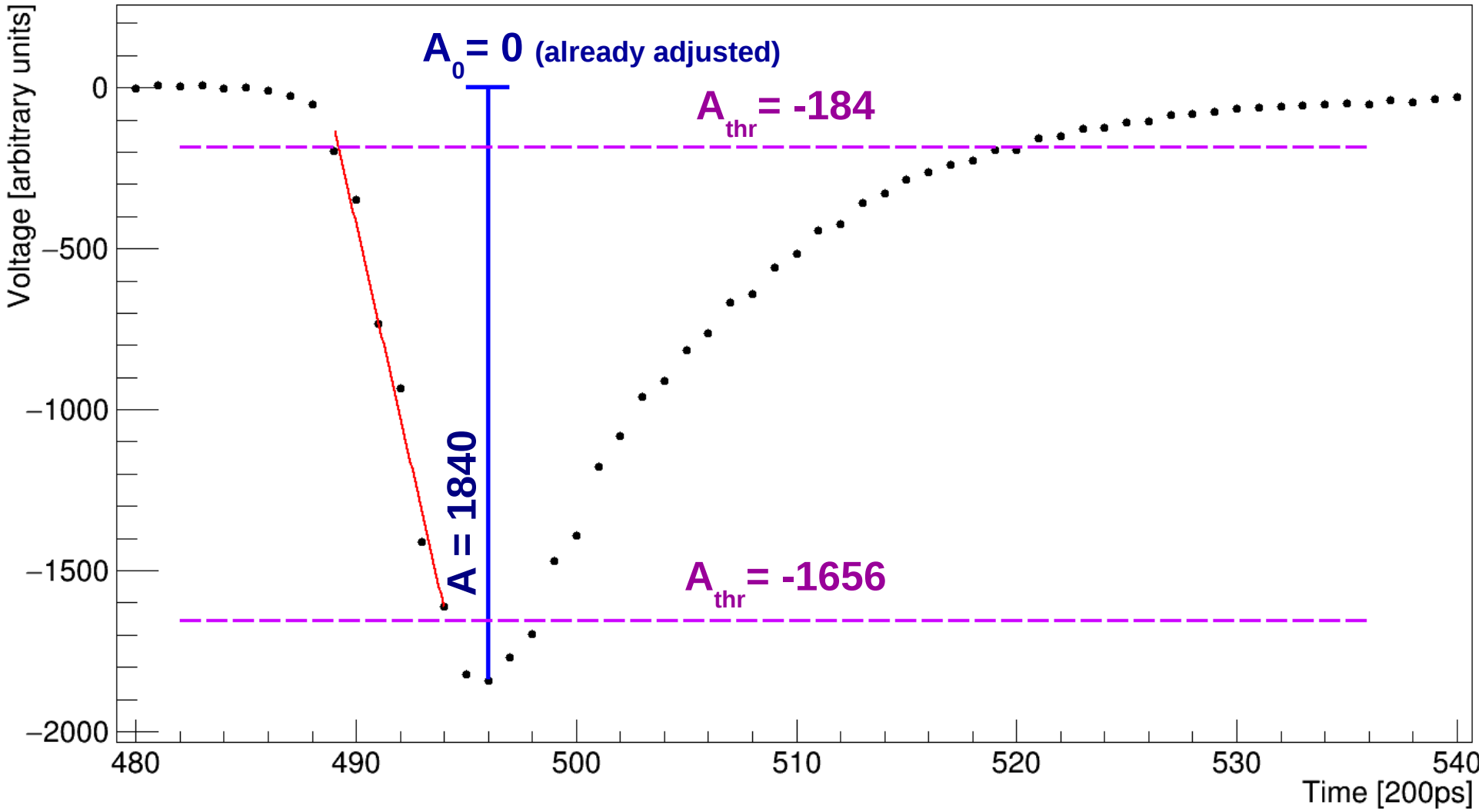
Raw waveforms

2: Set 10% and 90% amplitude thresholds



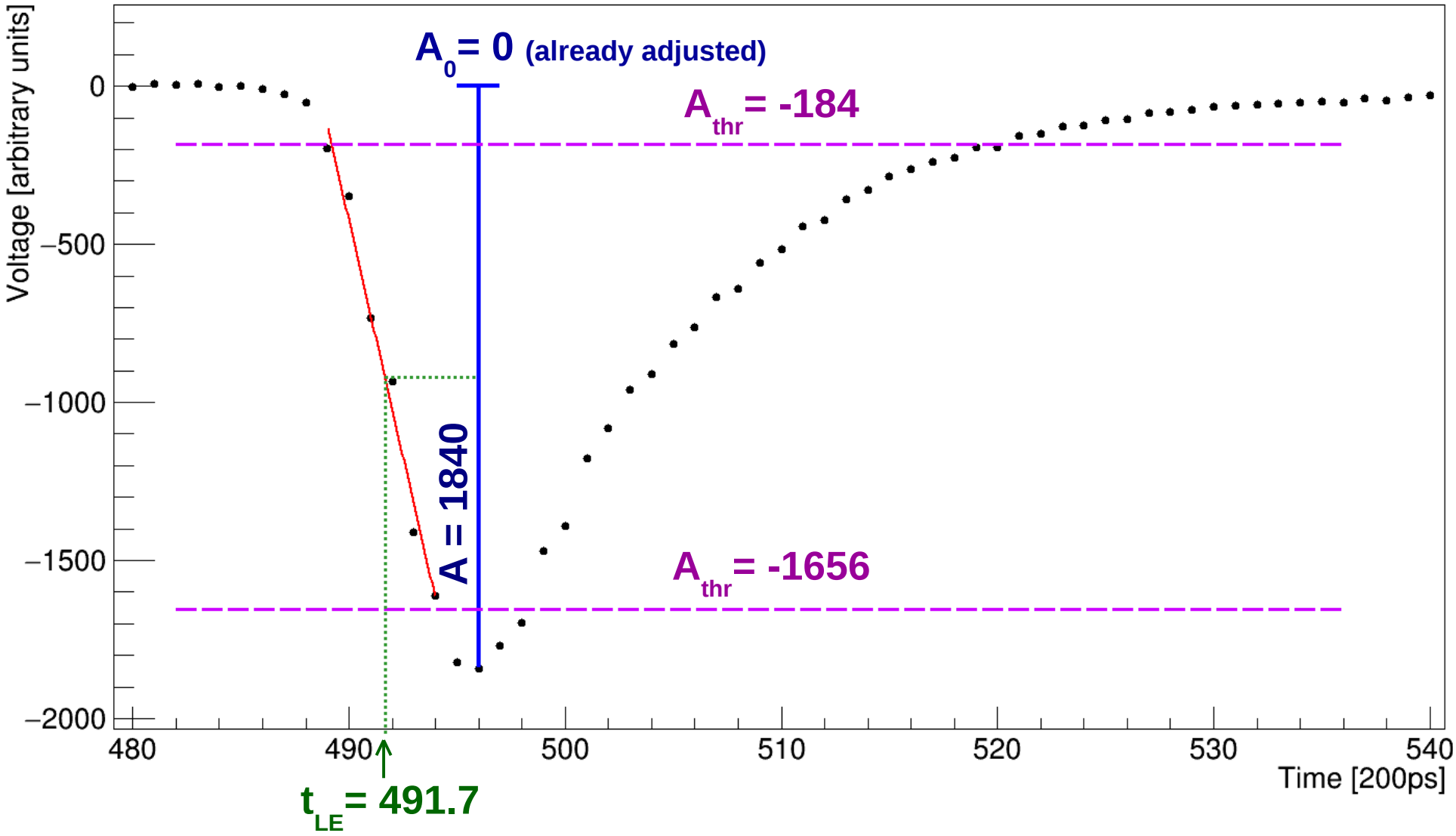
Raw waveforms

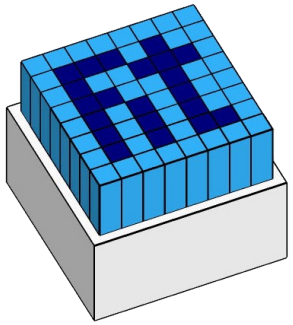
3: Fit the leading edge (LE)



Raw waveforms

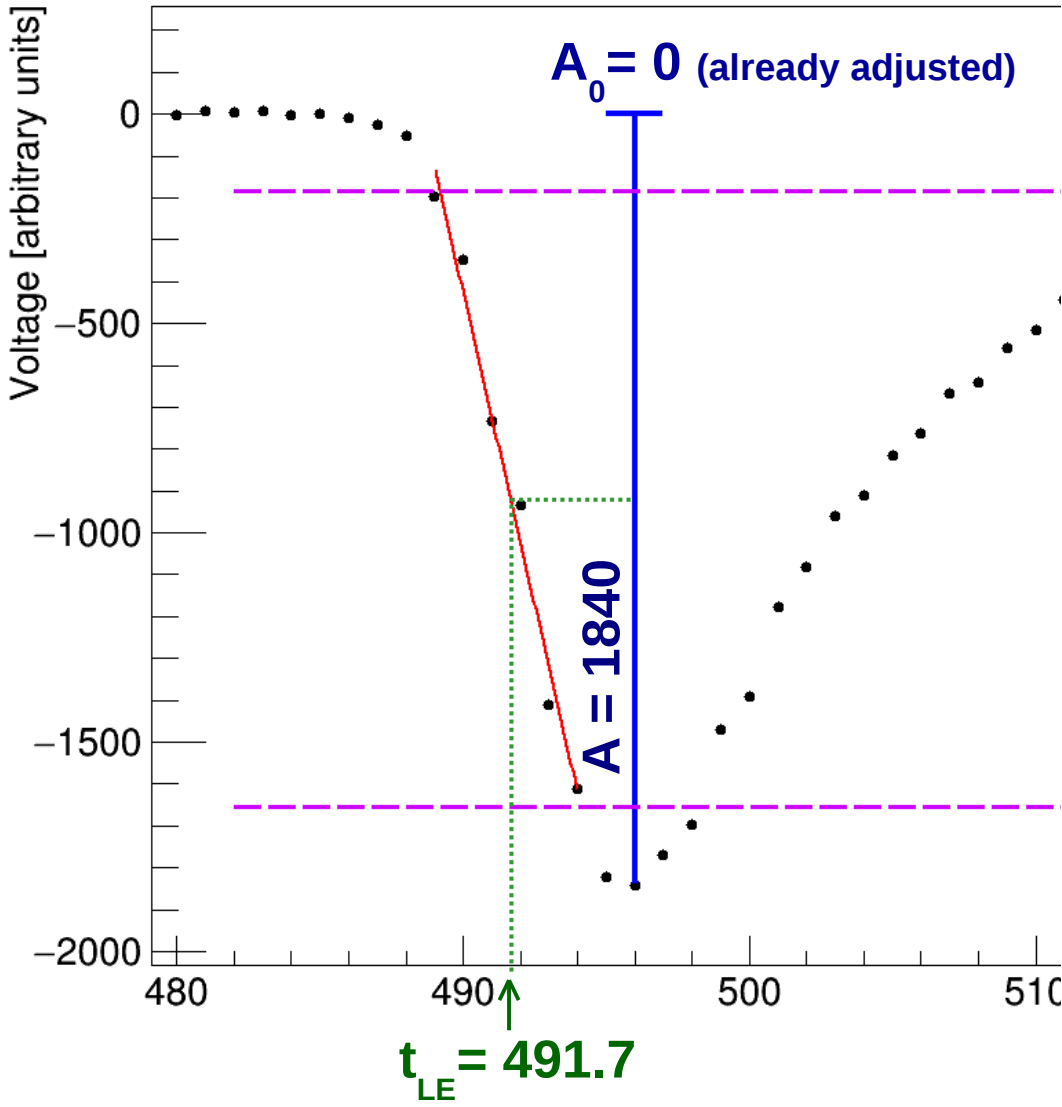
4: Get time of LE at half-amplitude level

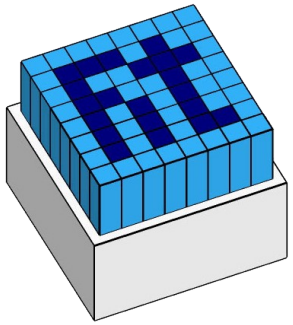




Raw waveforms
Issues to consider

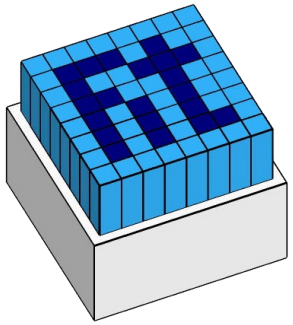
- What if:
  - Amplitude too **big**
  - Amplitude too **small**
  - LE very **steep**
  - Shape **anomalous**
    - a smaller pre-pulse at the LE of main pulse
  - **Several** pulses





Data analysis chain
Recap

- → Detectors
- → FE electronics
- → CAEN Digitizer
- → DAQ program
- → Binary files
- → **Waveform analysis**
- → Root file
- → Final analysis



Waveform analysis
DigDataProcessing - compilation

ALICE
A JOURNEY OF DISCOVERY

- Copy digdataprocessing.zip from indico to your disk and unpack:
  - https://indico.cern.ch/event/532602/attachments/1280341/1902644/digdataprocessing.zip
- In terminal go to the directory where you just unpacked digdataprocessing software

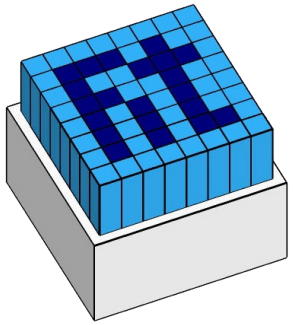
```
cd /somewhere/digdataprocessing
make
./digdataprocessing
```

  - if everything is working fine then the following should be printed

```
<I> === digdataprocessing ===
<I>   - FIT / T10@CERN beam test software.
<I>   - Takes as input raw binary files generated by CAEN Digitizer DT57**
<I>   - Plots misc stuff and/or converts data to root tree

<I> Command line syntax:
<I>   -g           ---> GUI mode
<I>   -c           ---> text mode, convert binary data to root
<I>   -i=in_dir    ---> input directory path where run folders are placed
<I>   -o=out_path ---> output file path where root file is saved [default: in_dir/../rootFiles/in_dir.root]
<I>   -n=n_events ---> number of events to be processed [default => all events]

<I> Example:
<I>   ./digdataprocessing -g
<I>   ./digdataprocessing -c -i=/path_data/
```



Waveform analysis
DigDataProcessing - compilation

- Copy digdataprocessing.zip from indico to your disk and unpack:
  - https://indico.cern.ch/event/532602/attachments/1280341/1902644/digdataprocessing.zip
- In terminal go to the directory where you just unpacked digdataprocessing software

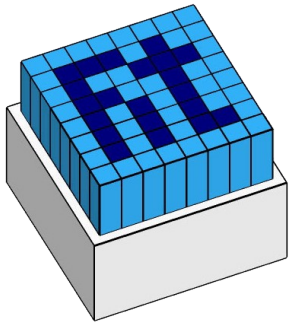
```
cd /somewhere/digdataprocessing
make
./digdataprocessing
```

  - if everything is working fine then the following should be printed

```
<I> === digdataprocessing ===
<I>   - FIT / T10@CERN beam test software.
<I>   - Takes as input raw binary files generated by CAEN Digitizer DT57**
<I>   - Plots misc stuff and/or converts data to root tree

<I> Command line syntax:
<I>   -g           ---> GUI mode
<I>   -c           ---> text mode, convert binary data to root
<I>   -i=in_dir    ---> input directory path where run folders are placed
<I>   -o=out_path ---> output file path where root file is saved [default: in_dir/../rootFiles/in_dir.root]
<I>   -n=n_events ---> number of events to be processed [default => all events]

<I> Example:
<I>   ./digdataprocessing -g
<I>   ./digdataprocessing -c -i=/path_data/
```



Waveform analysis

DigDataProcessing – compilation / troubleshooting

- If 'make' command fails:
  - check that the root related environmental variables are set correctly:

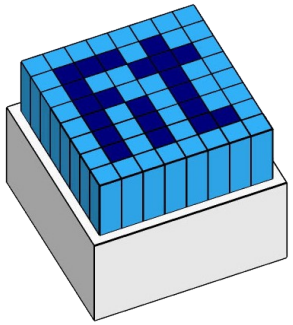
    '**`echo $PATH`**' should contain: *root/bin/*

    '**`echo $LD_LIBRARY_PATH`**' should contain: *root/lib/*

    - If any of them are missing then run:

      '**`source /path_to_your_root/bin/thisroot.sh`**'

      - Hint: This line can be added to your ~/.bashrc file (this file is executed every time a new terminal is opened)
    - re-run make (after navigating back to the digdataprocessing directory)
- If the digdataprocessing executable is not present after running 'make' command → it means '**`make`**' failed
- In some cases when the source code is modified, before running '**`make`**' try '**`make clean`**'



Waveform analysis

DigDataProcessing – compilation / troubleshooting

- If 'make' command fails:
  - check that the root related environmental variables are set correctly:

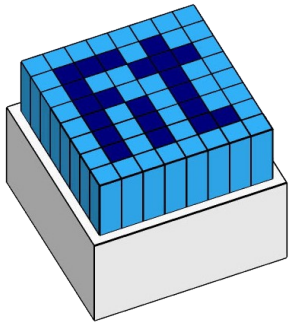
    '**`echo $PATH`**' should contain: *root/bin/*

    '**`echo $LD_LIBRARY_PATH`**' should contain: *root/lib/*

    - If any of them are missing then run:

      '**`source /path_to_your_root/bin/thisroot.sh`**'

      - Hint: This line can be added to your ~/.bashrc file (this file is executed every time a new terminal is opened)
    - re-run make (after navigating back to the digdataprocessing directory)
- If the digdataprocessing executable is not present after running 'make' command → it means '**`make`**' failed
- In some cases when the source code is modified, before running '**`make`**' try '**`make clean`**'



Waveform analysis
DigDataProcessing – usage

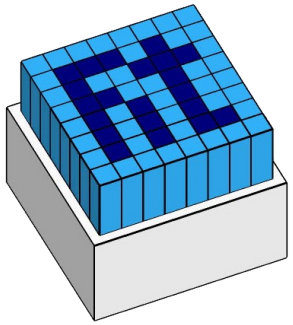
- Download a sample of raw data collected previously and unpack it (1.1 Gb):
  - https://filesender.funet.fi/?vid=297c1cf5-0de9-74c8-a3c5-00004a5a92c1
- Assuming the data has been unpacked into: /alicedata/ run:

```
./digdataprocessing -c -i=/alicedata/08102015_1758_RUN17_
NBI1655frag22_MOD1618frag4_6gevc_collimator_min_3x3
```

  - The program stops immediately and asks that user creates a directory called **rootFiles** in **/alicedata/**, which is where the output files will be stored by default. Do it and **rerun**.
  - The default output location can be overridden by adding to the command line:

```
o=/path_where_the_root_files_go/filename.root
```

    Note that the **full path** and **full name** of the file has to be provided with this option
- The digdataprocessing is running → converting every waveform into event-channel amplitude-time set of numbers
  - (+ some more characteristics, like total charge, time reconstructed using different methods, etc)
- After it is finished, the result is pre-analysed **root file** (2 Gb → 10 Mb)



Waveform analysis
DigDataProcessing – usage

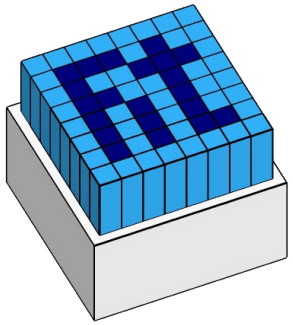
- Download a sample of raw data collected previously and unpack it (1.1 Gb):
  - https://filesender.funet.fi/?vid=297c1cf5-0de9-74c8-a3c5-00004a5a92c1
- Assuming the data has been unpacked into: /alicedata/ run:

```
./digdataprocessing -c -i=/alicedata/08102015_1758_RUN17_
NBI1655frag22_MOD1618frag4_6gevc_collimator_min_3x3
```

  - The program stops immediately and asks that user creates a directory called **rootFiles** in **/alicedata/**, which is where the output files will be stored by default. Do it and **rerun**.
  - The default output location can be overridden by adding to the command line:

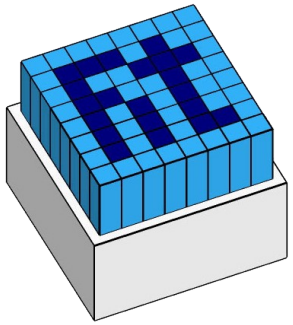
```
o=/path_where_the_root_files_go/filename.root
```

    Note that the **full path** and **full name** of the file has to be provided with this option
- The digdataprocessing is running → converting every waveform into event-channel amplitude-time set of numbers
  - (+ some more characteristics, like total charge, time reconstructed using different methods, etc)
- After it is finished, the result is pre-analysed **root file** (2 Gb → 10 Mb)



Data analysis chain
Recap

- → Detectors
- → FE electronics
- → CAEN Digitizer
- → DAQ program
- → Binary files
- → Waveform analysis
- → **Root file**
- → Final analysis



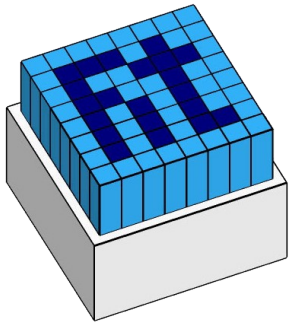
Waveform analysis
DigDataProcessing – output root file format, tree structure

- Root file (**TFile**, “08102015_1758_RUN17...root”)
  - Root tree (**TTree**, “RawDataTree”)
    - Branches: channels related to a specific detector or pixel Examples:
      - **4MCP2** <=> **4MCP** is the one purchased by NBI, **2** is the pixel number)
      - **T0_C1**, **T0_C2** <=> Two signals coming from the same triggering PMT (but not identical*)
      - **T0_D1**, **T0_D2** <=> The other triggering PMT (see slide 5)
      - **X**, **Y** <=> 2 position signals coming from XY detector
    - Two types of branches for each channel:
      - **main_4MCP2 – usable, stable and tested**
      - **4MCP2** – usable only in case of **X** and **Y** branches, otherwise treat as experimental (not fully implemented and tested, in general avoid using it)

*Digitizer has 16 input channels in 2 groups: 8 channels each. Each group has a separate internal clock (no synch between them), so to obtain accurate timing a reference (trigger) signal is provided for each group separately. So when relative timing of given channel is calculated an appropriate pair of trigger signals should be used (either C1&D1 or C2&D2) as reference.

There are two ways to check which signal was in which channel originally:

- Config file: DataConfig.txt – located in the same folder as binary data
- Logbook (link from beam test performed in 10.2015, there will be a new one created this year): https://docs.google.com/document/d/1ntYFb_oJMFxbDO4RM72wBJlVpJs8qH_kdIdd5wCkAzM/edit?usp=sharing



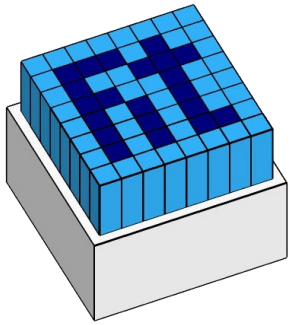
Waveform analysis
DigDataProcessing – output root file format, tree structure

- Root file (**TFile**, “08102015_1758_RUN17...root”)
  - Root tree (**TTree**, “RawDataTree”)
    - Branches: channels related to a specific detector or pixel Examples:
      - **4MCP2** <=> **4MCP** is the one purchased by NBI, **2** is the pixel number)
      - **T0_C1**, **T0_C2** <=> Two signals coming from the same triggering PMT (but not identical*)
      - **T0_D1**, **T0_D2** <=> The other triggering PMT (see slide 5)
      - **X**, **Y** <=> 2 position signals coming from XY detector
    - Two types of branches for each channel:
      - **main_4MCP2 – usable, stable and tested**
      - **4MCP2** – usable only in case of **X** and **Y** branches, otherwise treat as experimental (not fully implemented and tested, in general avoid using it)

*Digitizer has 16 input channels in 2 groups: 8 channels each. Each group has a separate internal clock (no synch between them), so to obtain accurate timing a reference (trigger) signal is provided for each group separately. So when relative timing of given channel is calculated an appropriate pair of trigger signals should be used (either C1&D1 or C2&D2) as reference.

There are two ways to check which signal was in which channel originally:

- Config file: DataConfig.txt – located in the same folder as binary data
- Logbook (link from beam test performed in 10.2015, there will be a new one created this year): https://docs.google.com/document/d/1ntYFb_oJMFxbDO4RM72wBJlVpJs8qH_kdIdd5wCkAzM/edit?usp=sharing



Waveform analysis

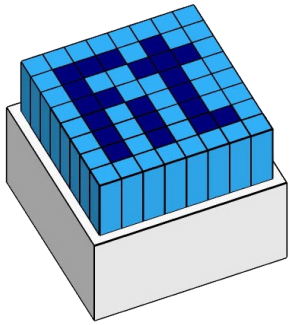
DigDataProcessing – output root file format, tree structure

- Root file (**TFile**, “08102015_1758_RUN17...root”)
  - Root tree (**TTree**, “RawDataTree”)
    - Branches: channels related to a specific detector or pixel Examples:
      - **4MCP2** <=> **4MCP** is the one purchased by NBI, **2** is the pixel number)
      - **T0_C1**, **T0_C2** <=> Two signals coming from the same triggering PMT (but not identical*)
      - **T0_D1**, **T0_D2** <=> The other triggering PMT (see slide 5)
      - **X**, **Y** <=> 2 position signals coming from XY detector
    - Two types of branches for each channel:
      - **main_4MCP2 – usable, stable and tested**
      - **4MCP2** – usable only in case of **X** and **Y** branches, otherwise treat as experimental (not fully implemented and tested, in general avoid using it)

*Digitizer has 16 input channels in 2 groups: 8 channels each. Each group has a separate internal clock (no synch between them), so to obtain accurate timing a reference (trigger) signal is provided for each group separately. So when relative timing of given channel is calculated an appropriate pair of trigger signals should be used (either C1&D1 or C2&D2) as reference.

There are two ways to check which signal was in which channel originally:

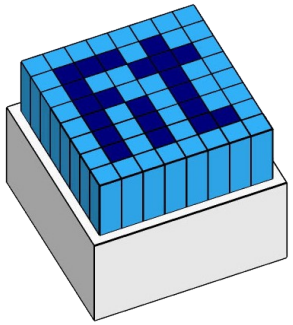
- Config file: DataConfig.txt – located in the same folder as binary data
- Logbook (link from beam test performed in 10.2015, there will be a new one created this year): https://docs.google.com/document/d/1ntYFb_oJMFxbDO4RM72wBJlVpJs8qH_kdIdd5wCkAzM/edit?usp=sharing



Waveform analysis

DigDataProcessing – output root file format, tree structure

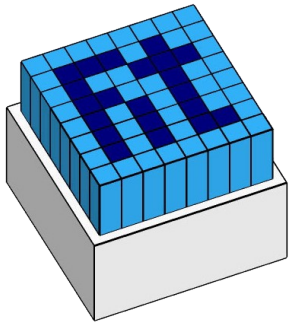
- **Leaves** inside branches
  - each leaf corresponds to a waveform property extracted from the binary data
- inside branch **main_5MCP2**:
  - **Amplitude** – pulse amplitude (slide 15)
  - Time – default method of calculating time (?)
  - **Time_pol1** – leading edge fitted with polynomial of the 1st order (linear fitting) → most accurate
  - Time_pol3 – fitted with polynomial of the 3rd order → less reliable
  - ZeroLevel – (top of slide 15)
  - ZeroLevelRMS – quality check
  - time_peak – for internal use
  - time_begin – for internal use
  - time_end – for internal use
  - time_front_end – for internal use
  - time_back_begin – for internal use
- inside branch **X**:
  - Level – the level of the pulse (~amplitude of the flat part, see slide 14) corresponding to the stripe(s) hit
  - Xch – x-position in channels (1 stripe → 1 channel): 0-5
  - Xmm – x-position in mm



Waveform analysis

DigDataProcessing – output root file format, tree structure

- **Leaves** inside branches
  - each leaf corresponds to a waveform property extracted from the binary data
- inside branch **main_5MCP2**:
  - **Amplitude** – pulse amplitude (slide 15)
  - Time – default method of calculating time (?)
  - **Time_pol1** – leading edge fitted with polynomial of the 1st order (linear fitting) → most accurate
  - Time_pol3 – fitted with polynomial of the 3rd order → less reliable
  - ZeroLevel – (top of slide 15)
  - ZeroLevelRMS – quality check
  - time_peak – for internal use
  - time_begin – for internal use
  - time_end – for internal use
  - time_front_end – for internal use
  - time_back_begin – for internal use
- inside branch **X**:
  - Level – the level of the pulse (~amplitude of the flat part, see slide 14) corresponding to the stripe(s) hit
  - Xch – x-position in channels (1 stripe → 1 channel): 0-5
  - Xmm – x-position in mm



Waveform analysis

DigDataProcessing – output root file format, tree structure

- **Leaves** inside branches
  - each leaf corresponds to a waveform property extracted from the binary data
- inside branch **main_5MCP2**:
  - **Amplitude** – pulse amplitude (slide 15)
  - Time – default method of calculating time (?)
  - **Time_pol1** – leading edge fitted with polynomial of the 1st order (linear fitting) → most accurate
  - Time_pol3 – fitted with polynomial of the 3rd order → less reliable
  - ZeroLevel – (top of slide 15)
  - ZeroLevelRMS – quality check
  - time_peak – for internal use
  - time_begin – for internal use
  - time_end – for internal use
  - time_front_end – for internal use
  - time_back_begin – for internal use
- inside branch **X**:
  - Level – the level of the pulse (~amplitude of the flat part, see slide 14) corresponding to the stripe(s) hit
  - Xch – x-position in channels (1 stripe → 1 channel): 0-5
  - Xmm – x-position in mm

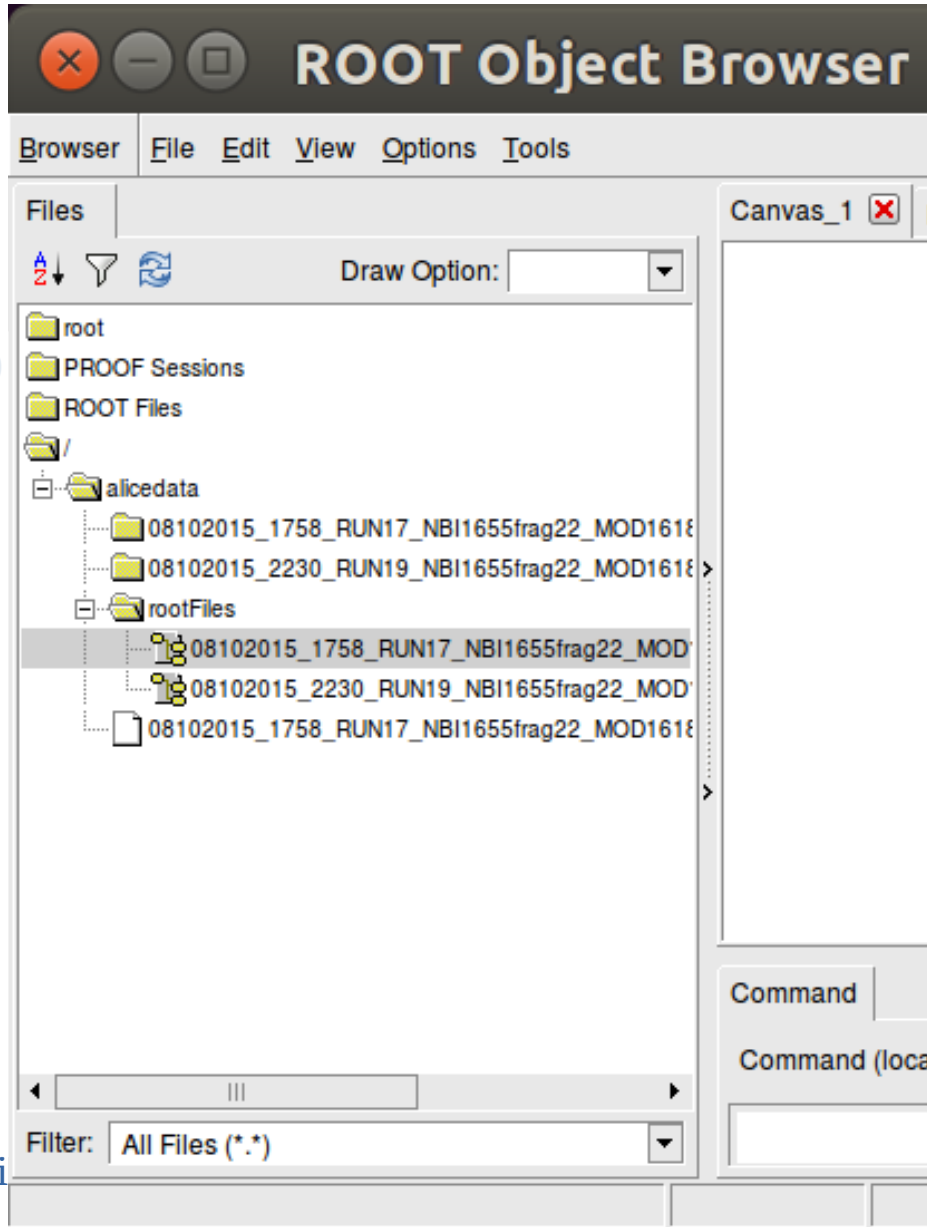
Root file analysis
TBrowser

- Quick browsing of a root file.

  In terminal type:

  ```
  root
  new TBrowser
  ```

- Navigate through the disk using the left panel and double click the root file to open it (right)
- Then open the RawDataTree to see branch structure (down)



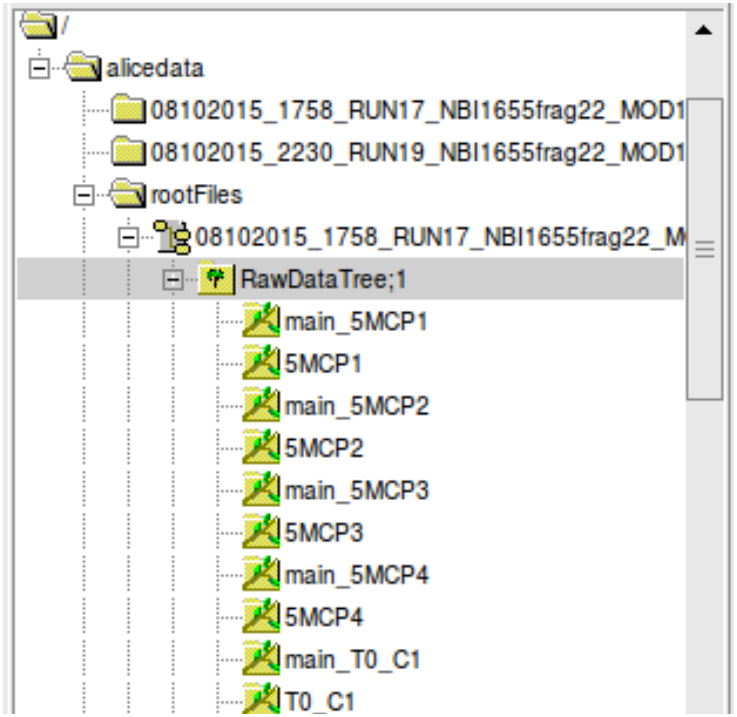
Root file analysis
TBrowser

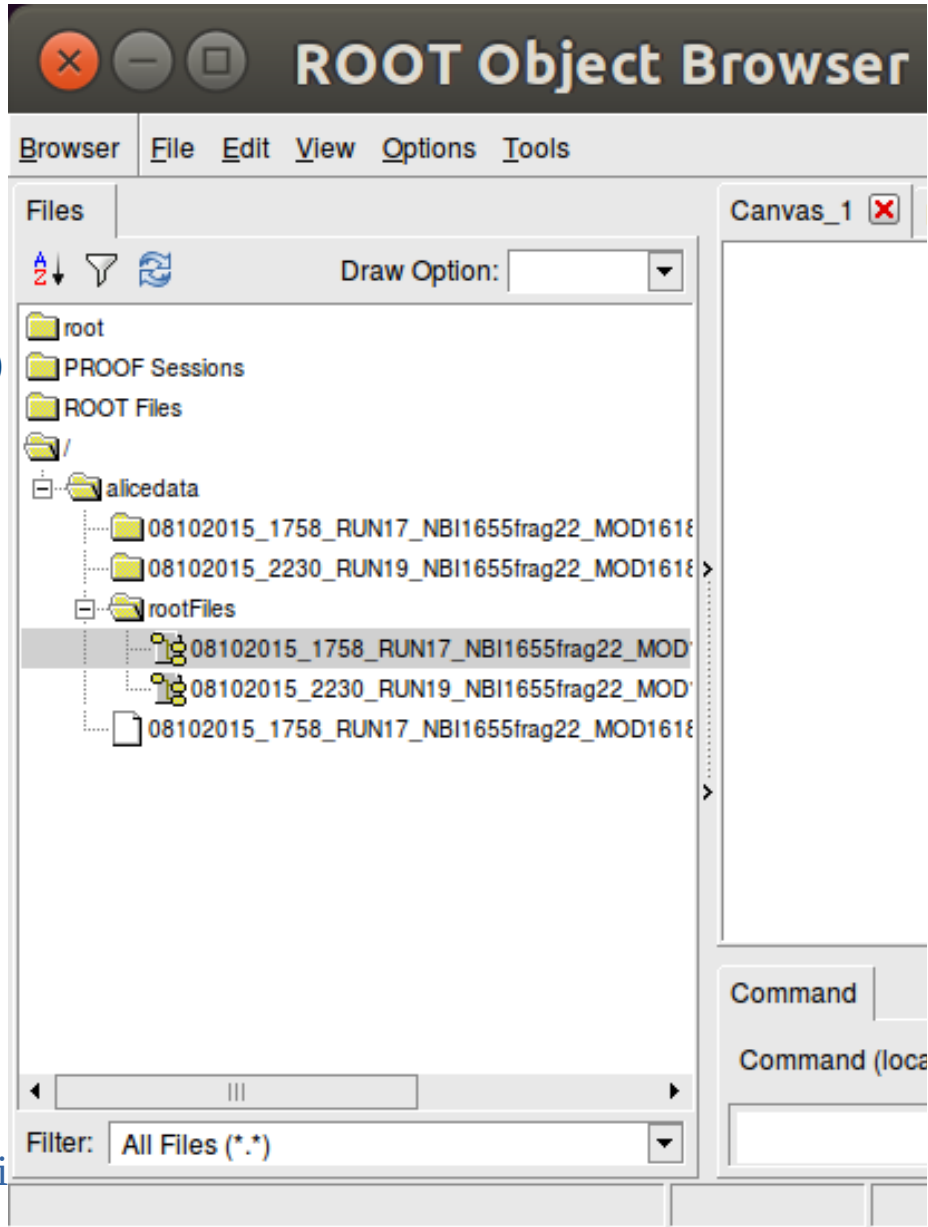
- Quick browsing of a root file.

  In terminal type:

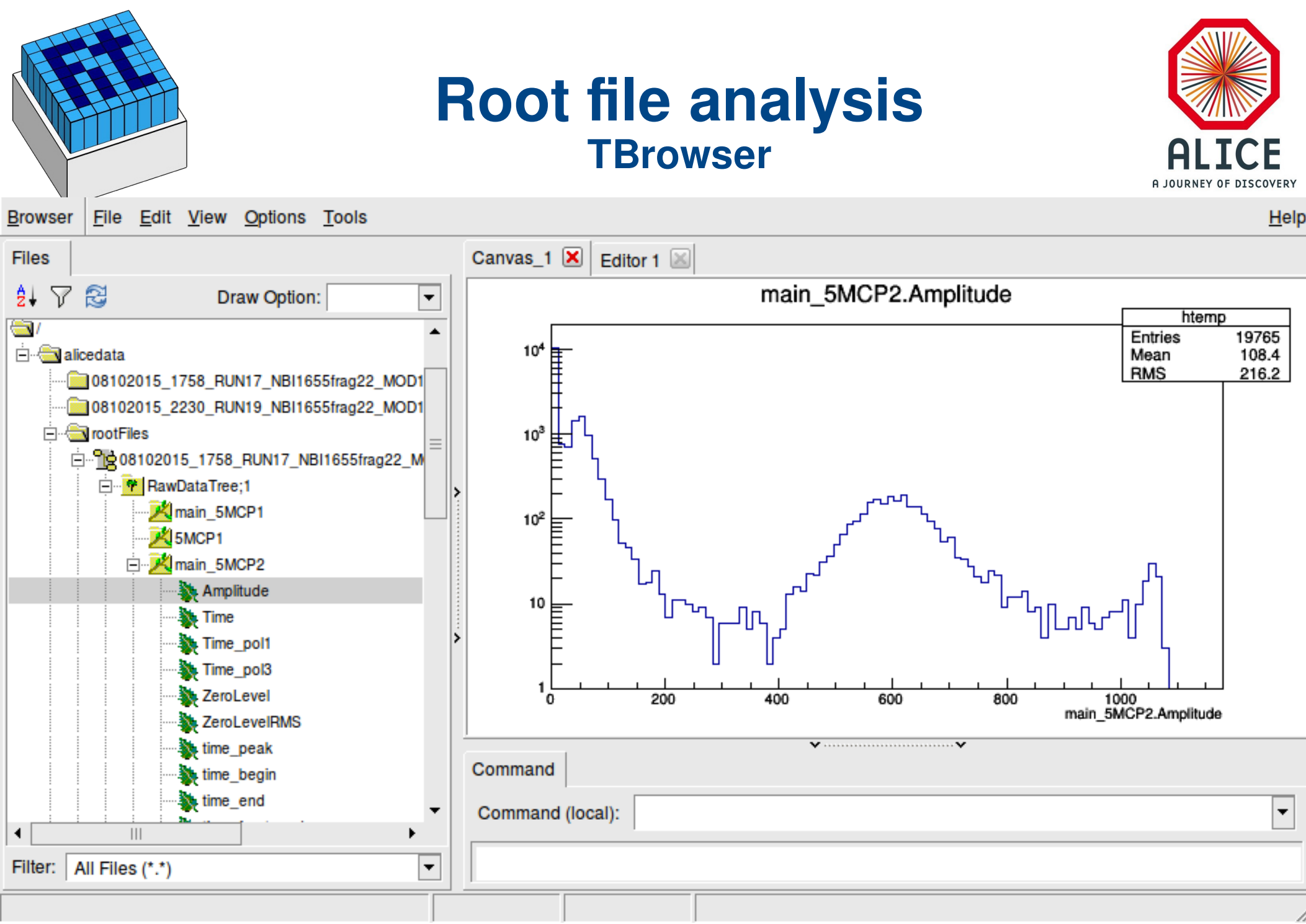
  ```
  root
  new TBrowser
  ```

- Navigate through the disk using the left panel and double click the root file to open it (right)
- Then open the RawDataTree to see branch structure (down)

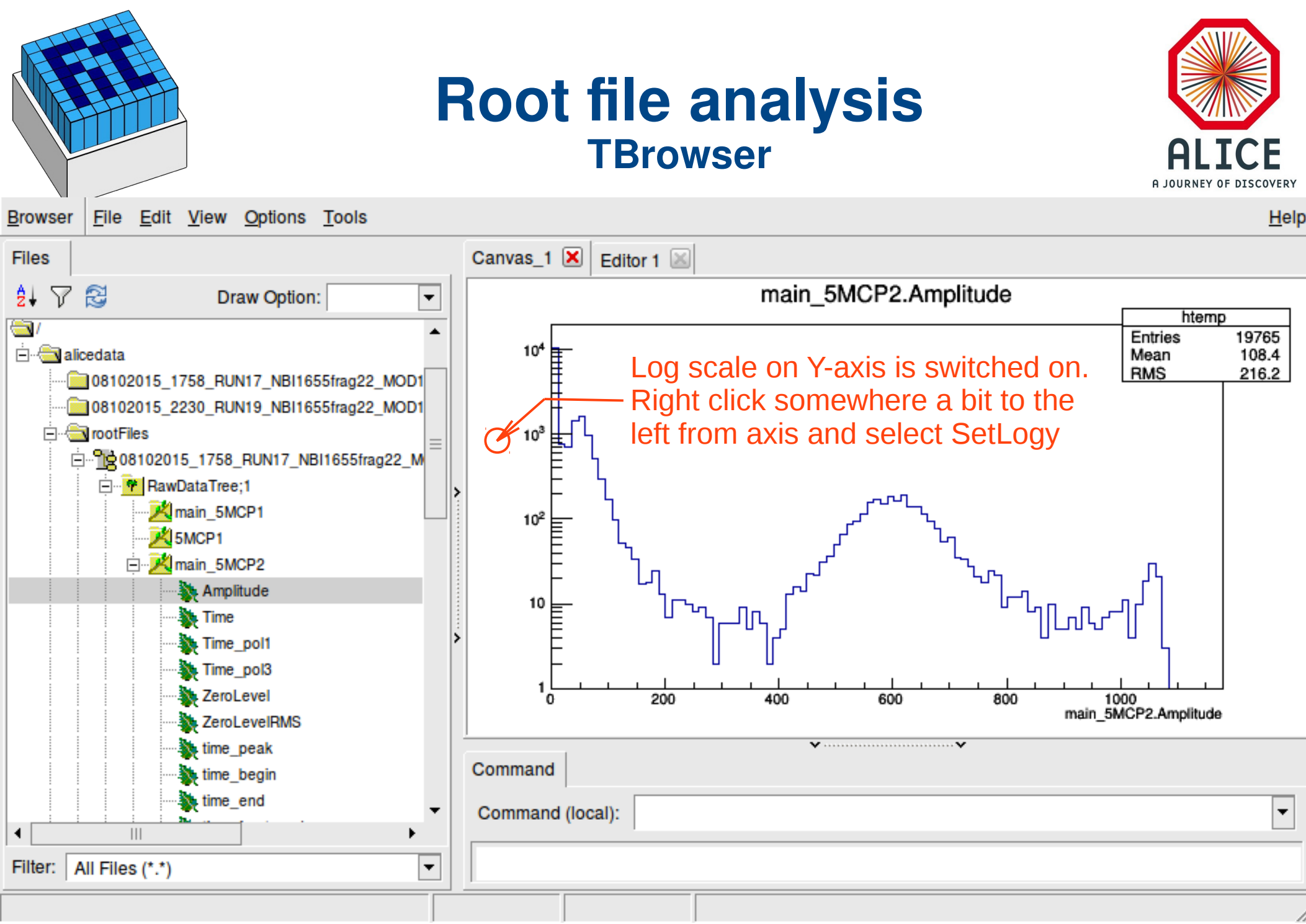


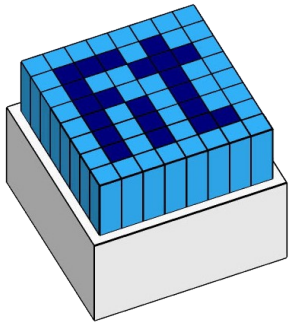


Root file analysis
TBrowser



Root file analysis
TBrowser





Data analysis chain
Recap

→ Detectors

→ FE electronics

→ CAEN Digitizer

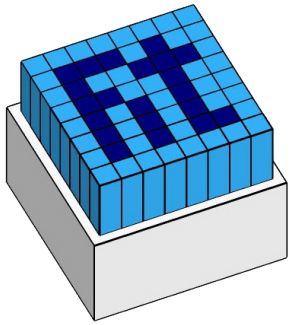
→ DAQ program

→ Binary files

→ Waveform analysis

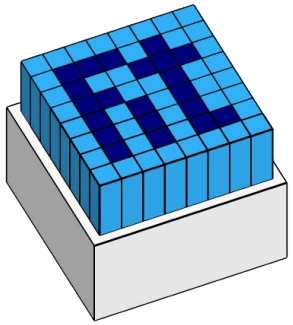
→ Root file

→ **Final analysis**



Root file analysis
genericTreeReader

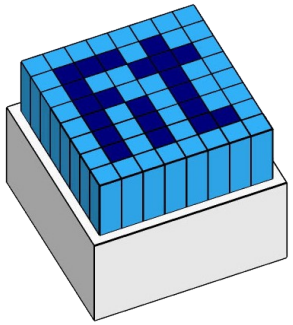
- TBrowser is great to take a quick peek at the data, but usually not sufficient to perform event-by-event analysis of multi-channel data
  - Solution?



Root file analysis
genericTreeReader

- TBrowser is great to take a quick peek at the data, but usually not sufficient to perform event-by-event analysis of multi-channel data
  - Solution?

Write own analysis script / program



Root file analysis
genericTreeReader – compilation and running

- A sample code: genericTreeReader. Download from indico and unpack:
  - https://indico.cern.ch/event/532602/attachments/1280341/1901690/genericTreeReader.zip
  - A simple heavily commented **example**, how to read a tree.
  - Experiment with **main.cpp** file and have fun! (3 exercises inside).
  - Start from **main()** function (line **153**). The whole relevant algorithm is in the **ReadAndDraw()** function.
  - In terminal go to the directory where you just unpacked genericTreeReader script

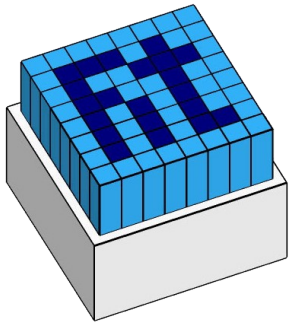
```
cd /somewhere/genericTreeReader
make
./genericTreeReader
```

    - There will be similar message as in the case of **digdataprocessing** program
    - Provide the input file path to the file, which you created at slide 26, with: -i=/path_to_root_file_to_be_read
    - Or use the root file already included in the genericTreeReader directory and leave the '-i=' field empty:

```
./genericTreeReader -i=
```

    - Both of these root files should be equivalent (possibly even identical)
- Alternatively run as root script. The effect will be similar. In terminal type:

```
root
.L main.cpp
ReadAndDraw("/file_path/to_your/data_file.root");
```



Root file analysis
genericTreeReader – compilation and running

- A sample code: genericTreeReader. Download from indico and unpack:
  - https://indico.cern.ch/event/532602/attachments/1280341/1901690/genericTreeReader.zip
  - A simple heavily commented **example**, how to read a tree.
  - Experiment with **main.cpp** file and have fun! (3 exercises inside).
  - Start from **main()** function (line **153**). The whole relevant algorithm is in the **ReadAndDraw()** function.
  - In terminal go to the directory where you just unpacked genericTreeReader script

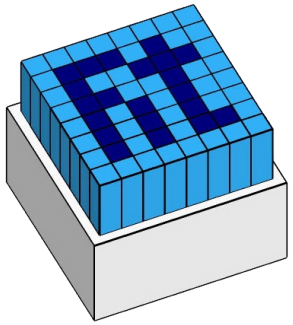
```
cd /somewhere/genericTreeReader
make
./genericTreeReader
```

    - There will be similar message as in the case of **digdataprocessing** program
    - Provide the input file path to the file, which you created at slide 26, with: -i=/path_to_root_file_to_be_read
    - Or use the root file already included in the genericTreeReader directory and leave the '-i=' field empty:

```
./genericTreeReader -i=
```

    - Both of these root files should be equivalent (possibly even identical)
- Alternatively run as root script. The effect will be similar. In terminal type:

```
root
.L main.cpp
ReadAndDraw("/file_path/to_your/data_file.root");
```



Root file analysis
genericTreeReader – compilation and running

- A sample code: genericTreeReader. Download from indico and unpack:
  - https://indico.cern.ch/event/532602/attachments/1280341/1901690/genericTreeReader.zip
  - A simple heavily commented **example**, how to read a tree.
  - Experiment with **main.cpp** file and have fun! (3 exercises inside).
  - Start from **main()** function (line **153**). The whole relevant algorithm is in the **ReadAndDraw()** function.
  - In terminal go to the directory where you just unpacked genericTreeReader script

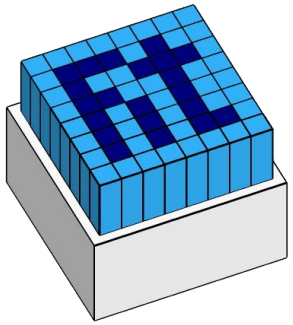
```
cd /somewhere/genericTreeReader
make
./genericTreeReader
```

    - There will be similar message as in the case of **digdataprocessing** program
    - Provide the input file path to the file, which you created at slide 26, with: -i=/path_to_root_file_to_be_read
    - Or use the root file already included in the genericTreeReader directory and leave the '-i=' field empty:

```
./genericTreeReader -i=
```

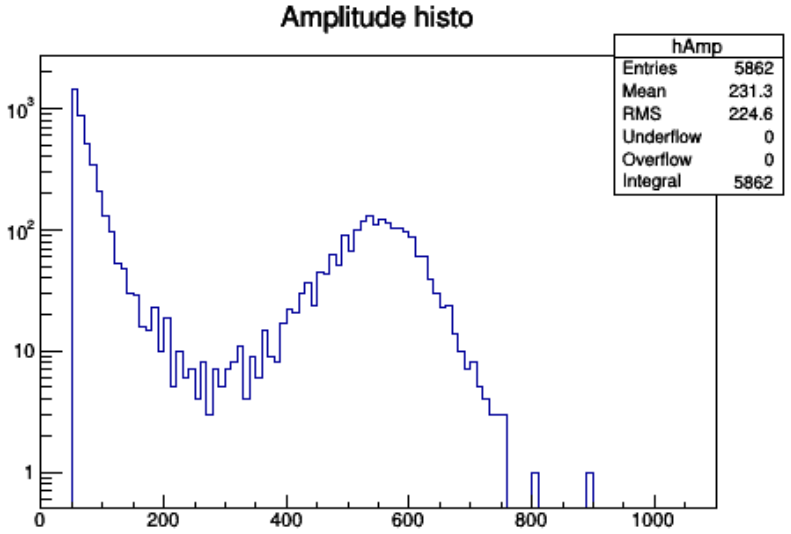
    - Both of these root files should be equivalent (possibly even identical)
- Alternatively run as root script. The effect will be similar. In terminal type:

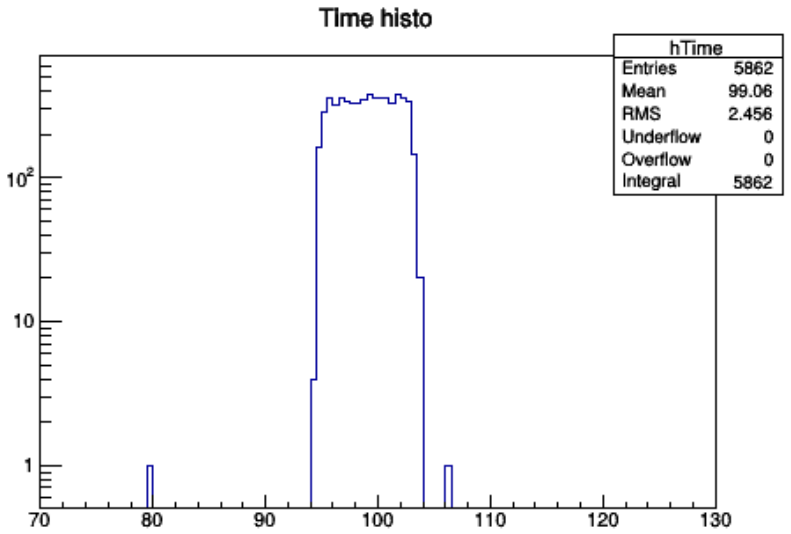
```
root
.L main.cpp
ReadAndDraw("/file_path/to_your/data_file.root");
```

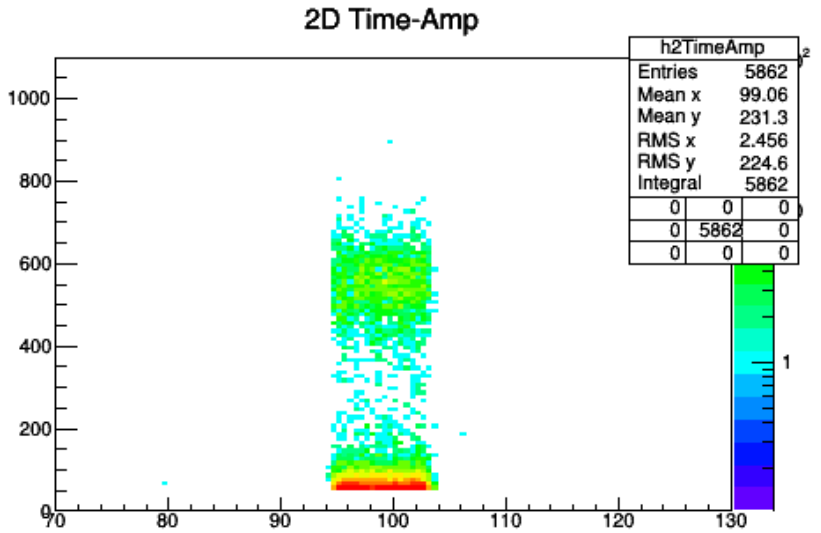


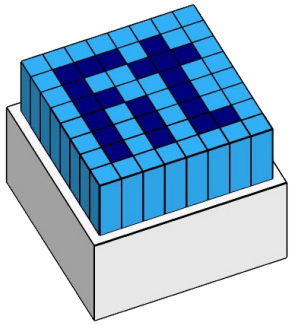
Root file analysis

genericTreeReader – result



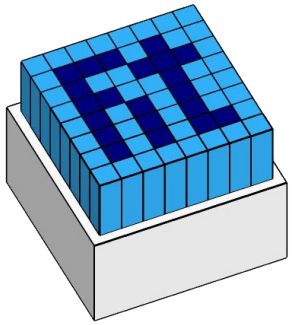






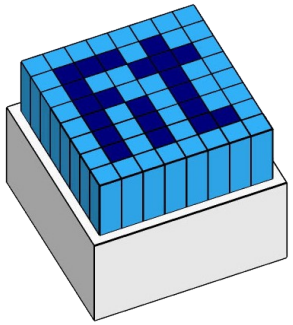
Root file analysis
Next steps

- More advanced examples of tree analysis can be found in digdataprocessing source code:
  - files: DigDataAnalysis.cpp and DigDataAnalysis.h
    - The features from there are not included yet as a switchable option when running the main program
  - the main developer of digdataprocessing and a contact person is:
    - Dima Finogeev (dmitry-finogeev@yandex.ru)
- Extend the genericTreeReader example
  - or just use it to read and select data, save it to a text file and use origin or excel (not recommended) to plot it



Advanced part:
digDataProcessing modification

- Adjustments to meet the specifics of V0:
  - Will start this week (Dima)
  - It is a more complex task, but if there are people willing to contribute, we can arrange sth ...



Summary

→ Detectors	→ no details
→ FE electronics	→ no details
→ CAEN Digitizer	→ technical specs
→ DAQ program	→ how to run
→ Binary files	→ structure
→ Waveform analysis	→ examples, algorithm
→ Root file	→ structure, reading
→ Final analysis	→ example programs

And finally some dates ...
... and other info

- The beam time: 15.06 – 22.06
- I will be at CERN: 07.06 – 23.06
  - You are welcome to contact me at **maciej.slupecki@gmail.com** in case of any problems or questions, including:
    - problems with compilation or understanding any of the examples provided here
    - practical issues
  - Don't forget your power adapters.