Status: Implementation of 3x1x1 detector in LArSoft

Kevin Fusshoeller
HEP Masters - ETH Zurich/Université Paris-Saclay

Cern, April 5th 2017
ProtoDUNE‘s Sim/Reco meeting.

Outline

1. Introduction.
2. Implementing the 3x1x1 Geometry.
3. Importing Data from QScan to LArSoft.

1. Introduction

Status and Goals

Current status:

- For protodune only the 6x6x6 geometry is implemented in LArSoft both as single phase and double phase.
- For now the double phase geometry is set with the drift direction in x-direction. (Work in progress to change the drift direction - Balint).

Goal: Implement 3x1x1 detector in LArSoft to test its performance using simulations and data analysis.

Tasks: → Add the 3x1x1 geometry to LArSoft.
→ Allow LArSoft to work with rotated geometries (drift direction in y).
→ Import raw data to LArSoft.

Status and Goals

Current status:

- For protodune only the 6x6x6 geometry is implemented in LArSoft both as single phase and double phase.
- For now the double phase geometry is set with the drift direction in x-direction. (Work in progress to change the drift direction - Balint).

Goal: Implement 3x1x1 detector in LArSoft to test its performance using simulations and data analysis.

Tasks: → Add the 3x1x1 geometry to LArSoft.
→ Allow LArSoft to work with rotated geometries (drift direction in y).
→ Import raw data to LArSoft.

2. Implementing the 3x1x1 geometry

3x1x1 Geometry

How to create a new geometry:

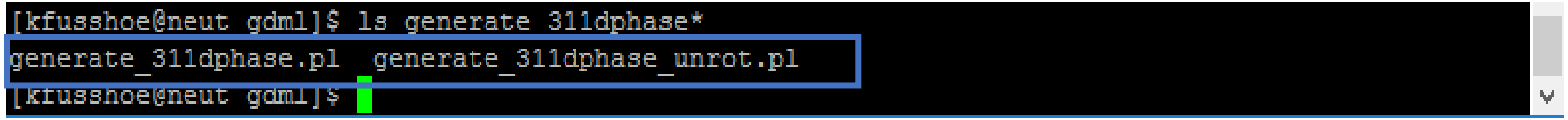
-) Write a perl script to generate the gdml files containing all the relevant parameters of the geometry (detector/cryostat dimensions, CRM parameters etc).

-) Once created, the gdml file can be loaded and used by LArSoft.

Before only the 6x6x6 geometry was implemented.

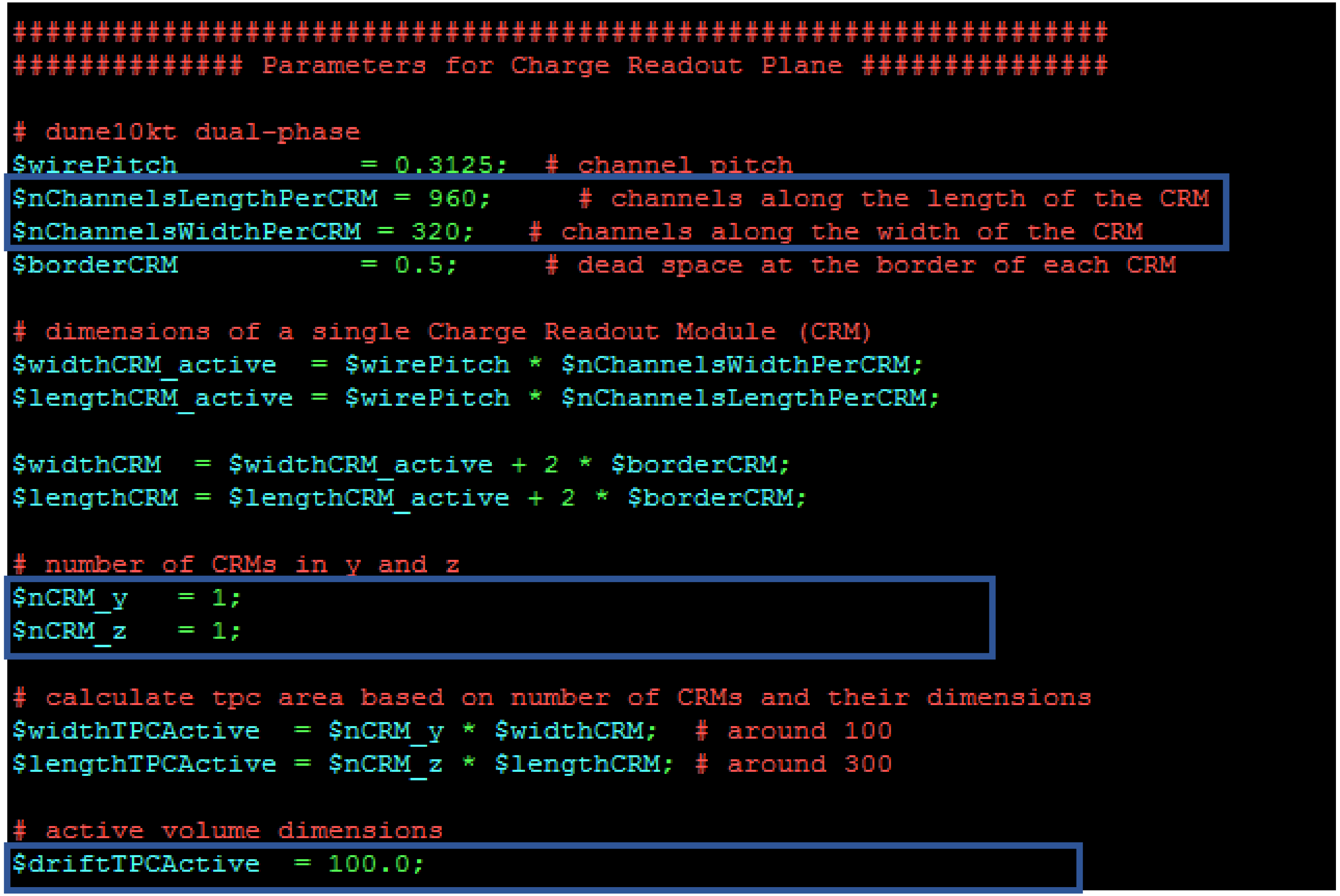
→ Adapt the perl script generating Protodune's geometry.

→ Two new geometries: -) unrotated 3x1x1 geometry
-) rotated 3x1x1 geometry



```
[kfusshoe@neut gdml]$ ls generate_311dphase*
generate_311dphase.pl  generate_311dphase_unrot.pl
[kfusshoe@neut gdml]$
```

Example perl script for unrotated geometry:

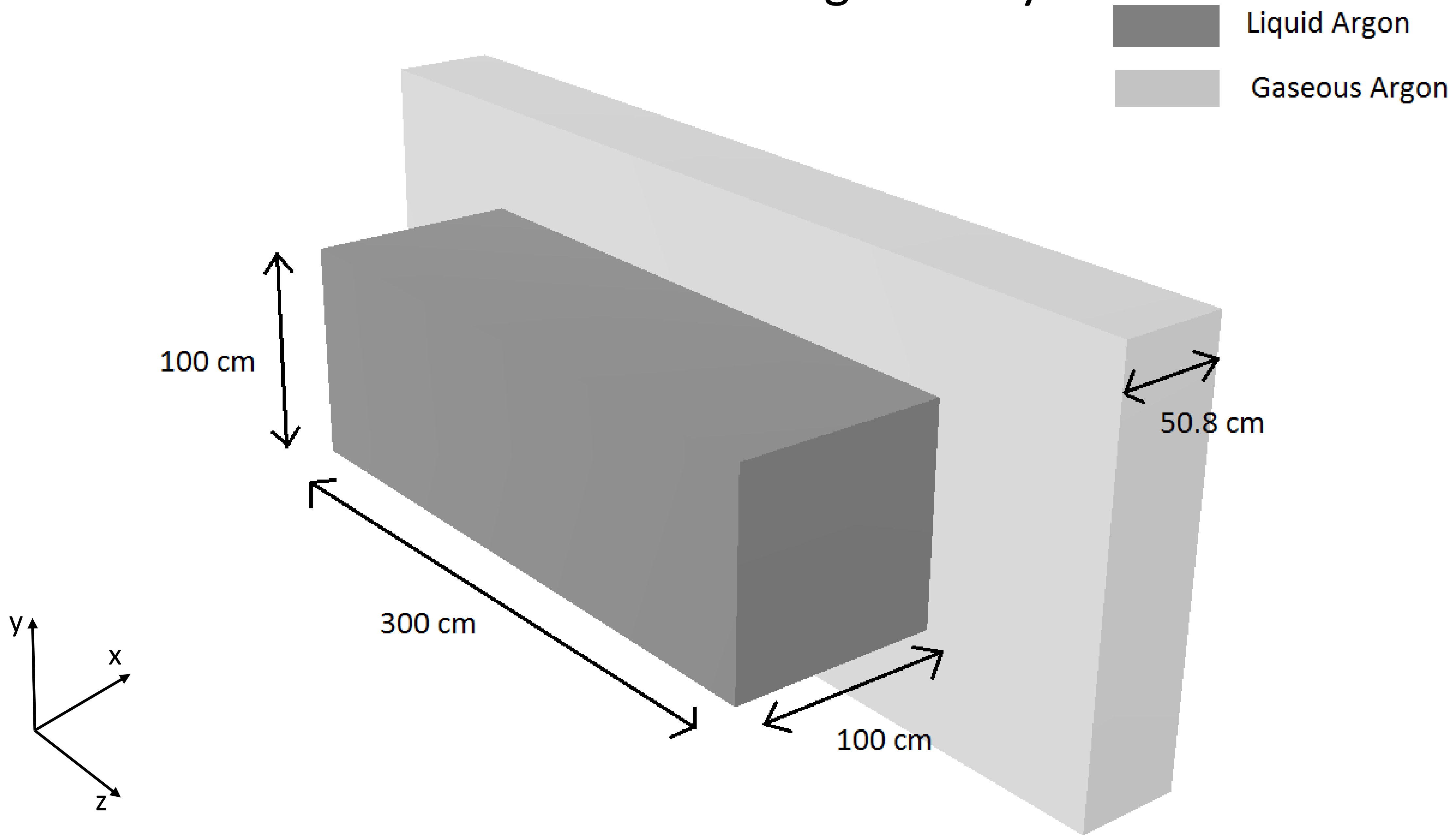


```
##################################################################
############## Parameters for Charge Readout Plane ###############

# dune10kt dual-phase
$wirePitch           = 0.3125;  # channel pitch
$nChannelsLengthPerCRM = 960;     # channels along the length of the CRM
$nChannelsWidthPerCRM = 320;   # channels along the width of the CRM
$borderCRM           = 0.5;     # dead space at the border of each CRM

# dimensions of a single Charge Readout Module (CRM)
$widthCRM_active  = $wirePitch * $nChannelsWidthPerCRM;
$lengthCRM_active = $wirePitch * $nChannelsLengthPerCRM;

$widthCRM  = $widthCRM_active + 2 * $borderCRM;
$lengthCRM = $lengthCRM_active + 2 * $borderCRM;

# number of CRMs in y and z
$nCRM_y   = 1;
$nCRM_z   = 1;

# calculate tpc area based on number of CRMs and their dimensions
$widthTPCActive  = $nCRM_y * $widthCRM;  # around 100
$lengthTPCActive = $nCRM_z * $lengthCRM; # around 300

# active volume dimensions
$driftTPCActive  = 100.0;
```

-) Add new parameter to account for rectangular CRM.
-) Change drift space + cryostat dimensions.
-) Currently drift space still in x-direction.

Geometries available, as soon as I get my FermiLab account.

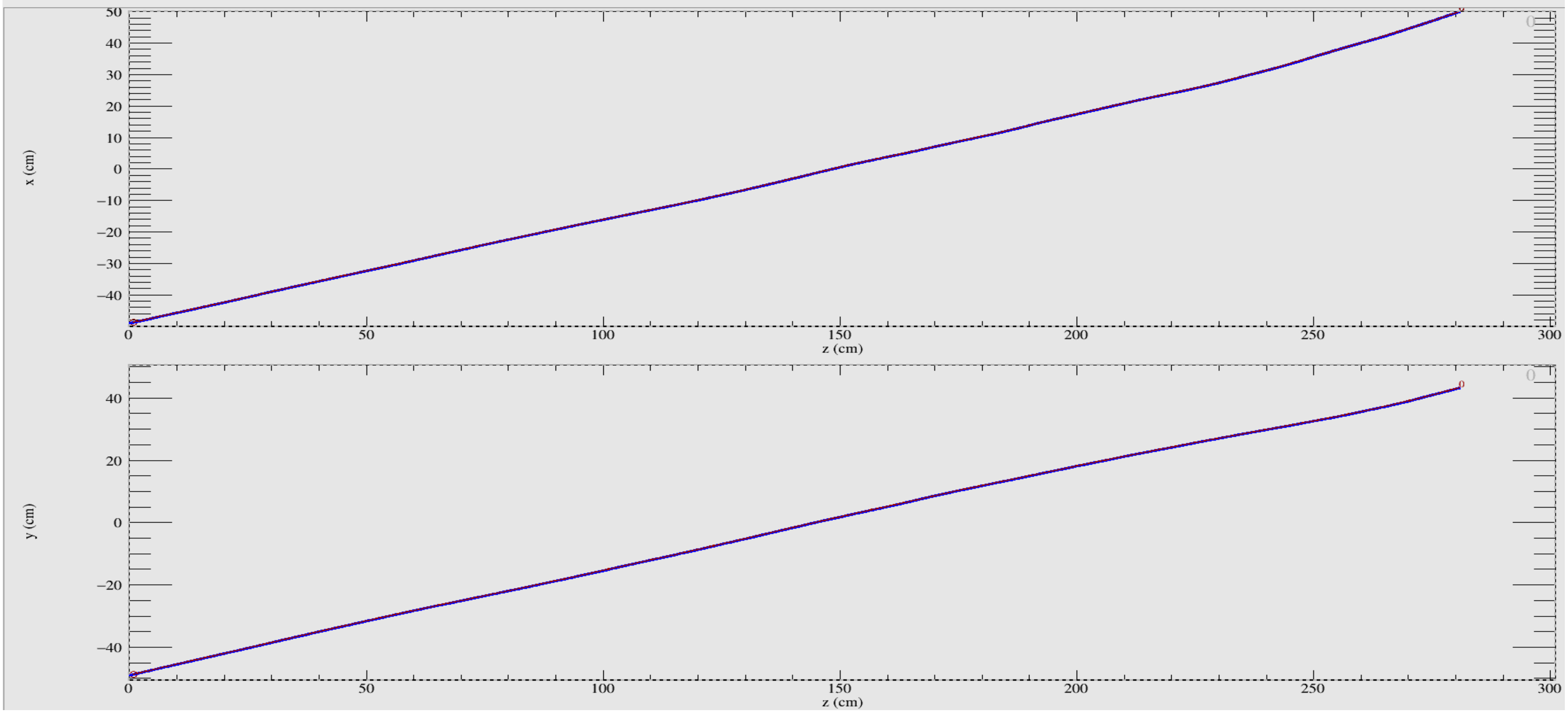
3x1x1 unrotated geometry

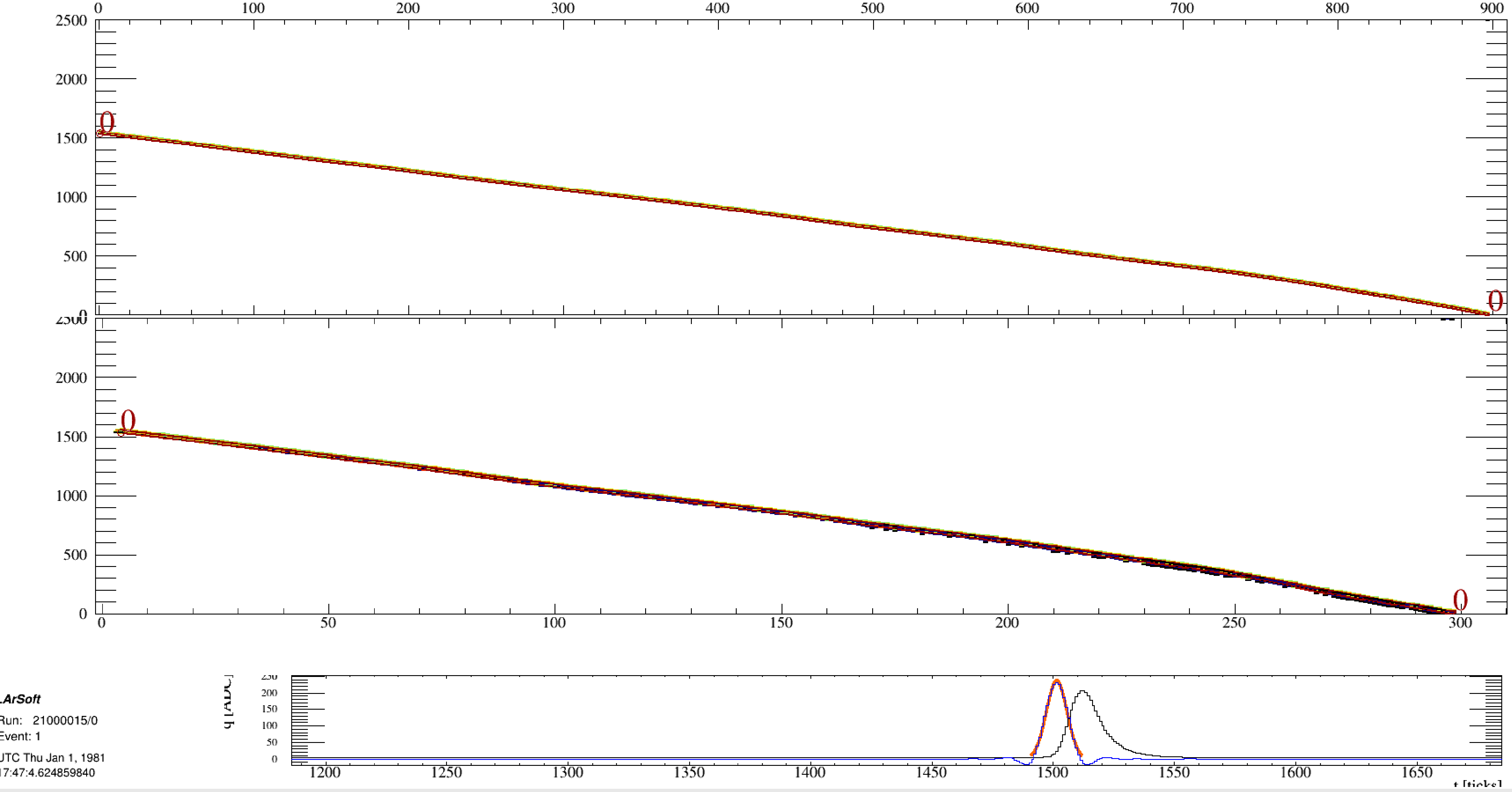


3x1x1 Geometry

Test: simulate a muon at 1.0 MeV.

→ geometry accepted by LArSoft and everything seems to run smoothly.





ArSoft

Run: 21000015/0

Event: 1

UTC Thu Jan 1, 1981

17:47:4.624859840

3. Importing data from QScan to LArSoft

3x1x1 Raw Data Structure

3x1x1 measurements are accessible at eos. Data structure as implemented by Elisabetta and Slavic:

- RawData (raw or manipulated) is stored as binary file.
- Each file contains up to 335 events.
- Per event the data is stored as a single vector holding the ADC counts of all the channels.

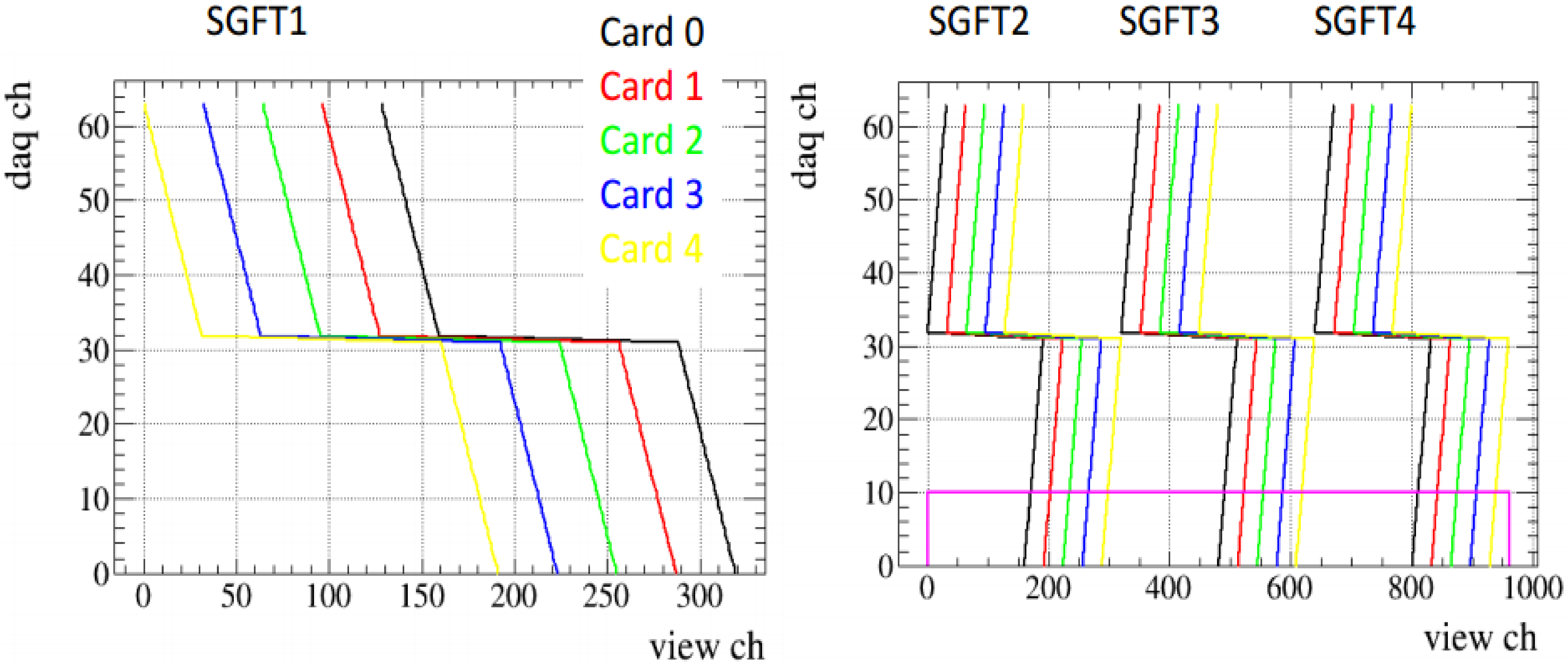
Example: 633-0.dat:

- First 5 bytes: run header: contains the run number (4 bytes) and a flag (1 byte).
- Last 4 bytes: footer: contains keys for internal checks (2 bytes) and the number of events recorded in the file (2 bytes).
- Per event: -) Event header (35 bytes): contains keys for internal checks (2 bytes), trigger info (24 bytes), data quality flag (1 byte), event number (4 bytes) and event size (4 bytes).

  -) Then come the ADC counts: read in card by card, channel by channel.

  -) The data is stored in 12 bit format.

Order of Channels (Graphic taken from Slavic‘s presentation at the general collaboration meeting)



Data Import from QScan

What does LArSoft want: root file containing:

→ art::event containing a collection of raw::Digit

→ raw::Digit is a class with member elements:

-) Channel number.

-) Number of ticks for this channel.

-) ADC vector for this channel.

-) Information about the used compression.

How to import the data?

Steps: -) Read in data (Use Slavic‘s code for this).

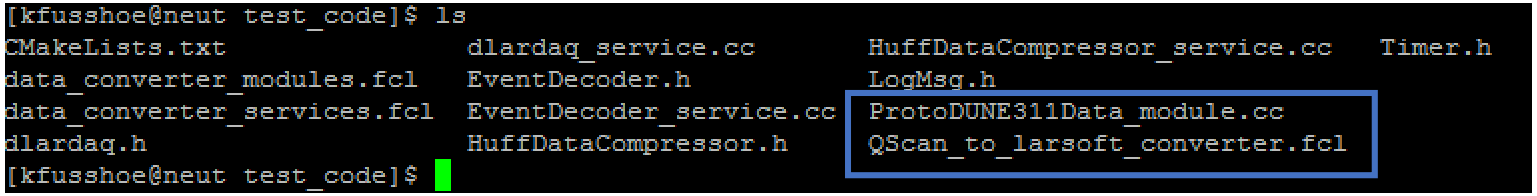
-) Choose the „view Channel“ and find the corresponding „daq Channel“.

-) Extract the ADC counts for that channel and store them in a new vector.

-) Create a raw::Digit for the corresponding channel.

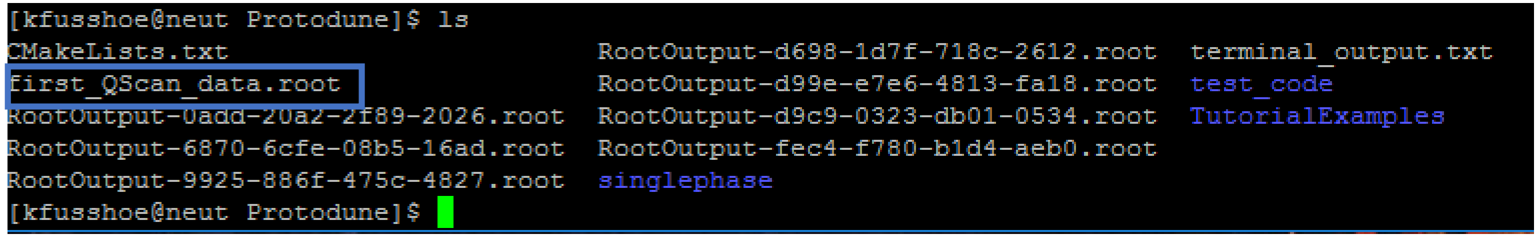
Data Import from QScan

New module create: ProtoDUNE311Data to read and manipulate the data from QScan.



```
[kfusshoe@neut test_code]$ ls
CMakeLists.txt               dlardaq_service.cc       HuffDataCompressor_service.cc   Timer.h
data_converter_modules.fcl   EventDecoder.h           LogMsg.h
data_converter_services.fcl  EventDecoder_service.cc  ProtoDUNE311Data_module.cc
dlardaq.h                    HuffDataCompressor.h     QScan_to_larsoft_converter.fcl
[kfusshoe@neut test_code]$
```

And first data converted:



```
[kfusshoe@neut Protodune]$ ls
CMakeLists.txt                       RootOutput-d698-1d7f-718c-2612.root  terminal_output.txt
first_QScan_data.root                RootOutput-d99e-e7e6-4813-fa18.root  test_code
RootOutput-0add-20a2-2f89-2026.root  RootOutput-d9c9-0323-db01-0534.root  TutorialExamples
RootOutput-6870-6cfe-08b5-16ad.root  RootOutput-fec4-f780-b1d4-aeb0.root
RootOutput-9925-886f-475c-4827.root  singlephase
[kfusshoe@neut Protodune]$
```

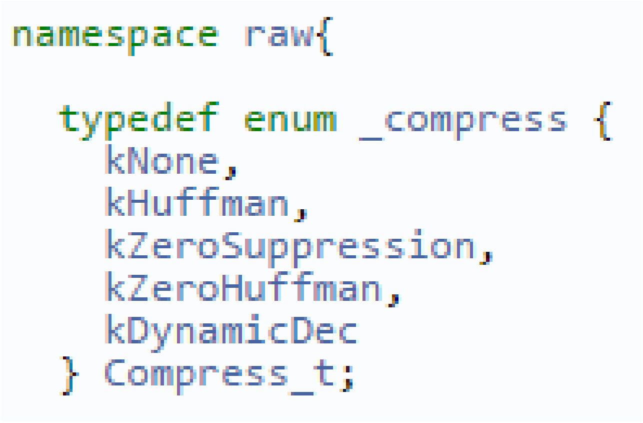
Next up: read the data with the new 3x1x1 geometry (unrotated or rotated?) to check everything went well.

Thanks a lot to Christoph, Robert and Dorota for their help!

Thank you for your attention!

Backup slide

Possible compressions: (access via name or enumerate)



```
namespace raw{

  typedef enum _compress {
    kNone,
    kHuffman,
    kZeroSuppression,
    kZeroHuffman,
    kDynamicDec
  } Compress_t;
```