Status of 2021 Test Beam(s) SW

Lorenzo Pezzotti, CERN

on behalf of the 2021 Dual-Readout Test Beam Group

Dual-Readout Calorimetry bi-weekly meeting
13/10/2021

Two beam tests in 2021

- June @Desy

  - first report from Romualdo [link]
  - e^{-} with energies from 1 to 6 GeV

Two beam tests in 2021

- June @Desy

  - first report from Romualdo [link]
  - e^- with energies from 1 to 6 GeV

- August @CERN SPS H8 beam line

  - first report from Gabriella [link]
  - e^- with energies from 6 to 125 GeV

Two beam tests in 2021

- June @Desy

  - first report from Romualdo [link]
  - e^- with energies from 1 to 6 GeV

- August @CERN SPS H8 beam line

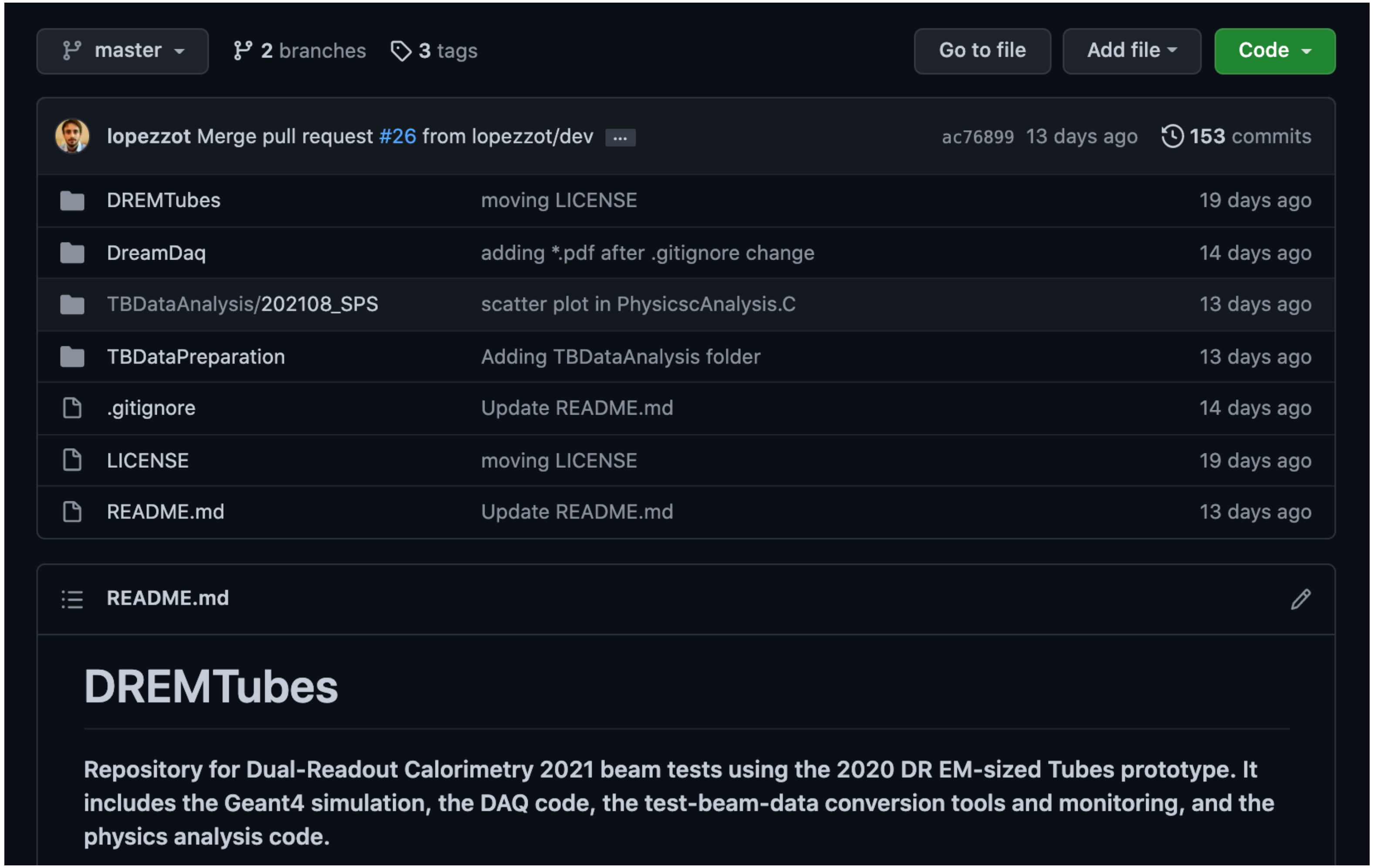
  - first report from Gabriella [link]
  - e^- with energies from 6 to 125 GeV

- From now on it is all about SW (data preparation, data analysis and simulation).

Dual-Readout 2021 Test-Beam SW

The 2021 Dual-Readout Calorimetry SW is located in a new GitHub repository.

- GitHub link
- v1.2 released at the end of the latest test-beam. At the moment, master branch aligned to v1.2.
- All releases are created from master branch tags. Always refer to the master.



Dual-Readout 2021 Test-Beam SW

The 2021 Dual-Readout Calorimetry SW is located in a new GitHub repository.

- Mostly coded in July-August with ~ 150 commits from ~ 10 contributors (7 forks).

Dual-Readout 2021 Test-Beam SW

The 2021 Dual-Readout Calorimetry SW is located in a new GitHub repository

- At present the repository is divided in four sub-repos:

  - DREAMDaq/ — *not discussed today* → Test beam data acquisition code. *Contact: Roberto Ferrari*

  - TBDataPreparation/ — *Today's topic* → Converters from raw data to analysis ntuples and raw data monitoring. *Contacts: Lorenzo Pezzotti, Iacopo Vivarelli, Andrea Negri Edoardo Proserpio and Romualdo Santoro.*

  - TBDataAnalysis/ — *Introduced today* → Macros for high-level-objects analysis. *Contacts: Lorenzo Pezzotti, Iacopo Vivarelli, Edoardo Proserpio and Romualdo Santoro.*

  - DREMTubes/ — *Introduced in July 2021, recap today* → Geant4 test-beam simulation. *Contact: Lorenzo Pezzotti*

Geant4 simulation

./DREMTubes

A Geant4 simulation of the 2020 Dual-Readout em-sized tubes prototype beam tests

- Documentation available in README.md [link]
- Tested with no crashes and no warnings for multi-threaded data production on Mac, lxplus and lxplus+HTCondor.
- Code already described in June [presentation].
- On branch dev/ some developments are going on related to geometry and signal simulation.
  Contact me if you want to contribute.

DREMTubes

A Geant4 simulation of the 2020 Dual-Readout em-sized tubes prototype beam tests.

▼ Table of Contents

1. Project description
2. Authors and contacts
3. Documentation and results
   - Selected presentations
4. How to
   - Build, compile and execute on Mac/Linux
   - Build, compile and execute on lxplus
   - Submit a job with HTCondor on lxplus
5. My quick Geant4 installation

How to use

- **Build and compile just by sourcing the Geant4 env, as explained [here].**
- **Execute with no GUI:**

```
./DREMTubes -m DREMTubes_run.mac -t 2 -pl FTFP_BERT -opt false
```

Select the PhysicsList
- Default FTFP_BERT

Executable created with CMake

Select a macro card for beam parametrization

- DREMTubes_run.mac automatically copied in build directory
- A very simple macro card [here]

Select # threads
- Default t=2
- Make sure you enabled multi-threading with Geant4.

- If true optical photons are propagated inside fibers.
- Otherwise a fast signal computation is performed.
- Default *false*.

Optical photons

-opt false

- Scintillation signal is parameterized starting from the ionizing energy deposited in S fibers.
  Photo-statistical fluctuations included.

- Cherenkov signal is taken from the Chrenkov photons trapped (and KILLED!) in C fibers.
  Photo-statistical fluctuations included.

- FastSteppingAction methods computes both signals.

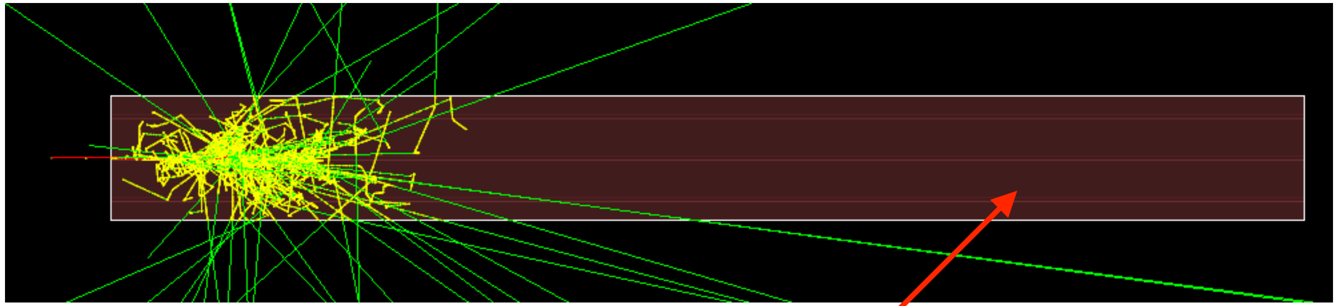
-opt true

- Optical photons are killed at their first step with a Poissonian probability tuned on the experimental light yields.

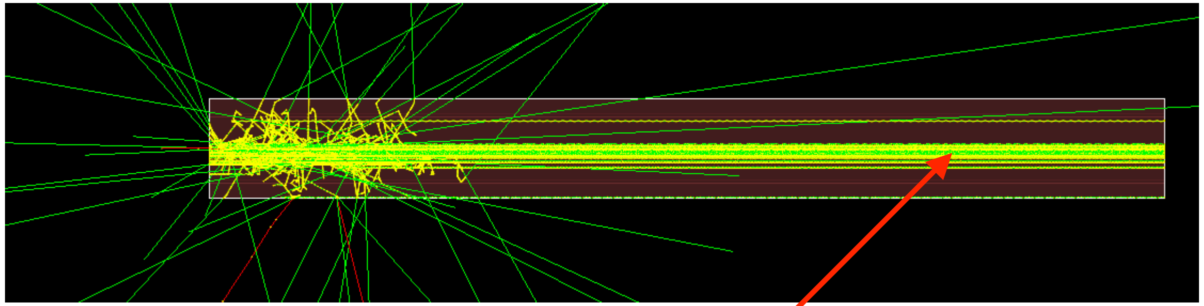
- Signals come from the SURVIVED optical photons, propagated within fibers and detected at the SiPMs surface.

- Suitable for studies on light absorption, light cross talk, optical fibers properties, …

- Slow SteppingAction method computes both signals.



No optical photons tracked inside fibers



Optical photons tracked inside fibers

Raw test-beam data handling

Raw data handling workflow

As two DAQ systems are involved for PMTs and SiPMs, the workflow looks like this:

Raw Data Handling

Raw ASCII Data as DAQ Output for PMTs and auxiliaries

↓

Rootified raw data output for PMTs and auxiliaries

Rootified raw data output for SiPMs

↑

.dat files as raw data output for SiPMS

Data Merging

Stored events from the two DAQs are synchronized by taking into account the random pedestal events and stored in a dedicated root merged file.

→

Calibration and High-Level Abstraction

Calibration constants are applied for ADC to GeV conversion, final information is saved in a high-level-object (class) ready for analysis.

Raw data

Raw data (PMTs + auxiliaries) are stored as ASCII files. An event looks like this


```
 ev # 105265 tow cts 95696 9570 103507 trigger mask 1 values: 0 47 16 d3 1 4b 17 e8 2 3b 18 de 3 48 19 da 4 39 20 95 5 10 21 b2 6 3f 22 f4 7
4 23 d7 8 c9 24 de 9 b0 25 d9 10 ed 26 ea 11 d7 27 dc 12 dd 28 ef 13 99 29 cb 14 bf 30 4a 15 c7 31 69 32 fd 48 10b 33 fe 49 118 34 10d 50 114
 35 132 51 10d 36 f3 52 11d 37 13e 53 113 38 118 54 11a 39 11e 55 101 40 130 56 100 41 109 57 fc 42 fb 58 12d 43 137 59 113 44 103 60 108 45
118 61 e5 46 f7 62 e7 47 10e 63 115 64 a0 80 e0 65 67 81 c8 66 c2 82 d1 67 a9 83 c5 68 cc 84 f3 69 ae 85 d7 70 bf 86 bb 71 d7 87 eb 72 e4 88
e6 73 d9 89 c1 74 e7 90 c1 75 101 91 e2 76 db 92 bd 77 ef 93 c7 78 d2 94 e1 79 ea 95 f0 TDC size 0 val.s
```

Raw data

Raw data (PMTs + auxiliaries) are stored as ASCII files. An event looks like this

```
ev # 105265 tow cts 95696 9570 103507 trigger mask 1 values: 0 47 16 d3 1 4b 17 e8 2 3b 18 de 3 48 19 da 4 39 20 95 5 10 21 b2 6 3f 22 f4 7
4 23 d7 8 c9 24 de 9 b0 25 d9 10 ed 26 ea 11 d7 27 dc 12 dd 28 ef 13 99 29 cb 14 bf 30 4a 15 c7 31 69 32 fd 48 10b 33 fe 49 118 34 10d 50 114
35 132 51 10d 36 f3 52 11d 37 13e 53 113 38 118 54 11a 39 11e 55 101 40 130 56 100 41 109 57 fc 42 fb 58 12d 43 137 59 113 44 103 60 108 45
118 61 e5 46 f7 62 e7 47 10e 63 115 64 a0 80 e0 65 67 81 c8 66 c2 82 d1 67 a9 83 c5 68 cc 84 f3 69 ae 85 d7 70 bf 86 bb 71 d7 87 eb 72 e4 88
e6 73 d9 89 c1 74 e7 90 c1 75 101 91 e2 76 db 92 bd 77 ef 93 c7 78 d2 94 e1 79 ea 95 f0 TDC size 0 val.s
```

Decoding/rootification is done with [DRrootify.py] and [DREvent.py]

```python
class DRrootify:
    '''Class to rootify raw ASCII files'''

    def __init__(self, fname):
        '''Class Constructor skipped here’''

    def ReadandRoot(self):
        '''Read ASCII files line by line and rootify'''
        print "--->Start rootification of "+self.drfname
        for i, line in enumerate(open(self.drfname)):
            if i%5000 == 0 : print "------>At line "+str(i)+" of "+str(self.drfname)
            evt = DREvent.DRdecode(line)
            self.EventNumber[0] = evt.EventNumber
            self.NumOfPhysEv[0] = evt.NumOfPhysEv
            self.NumOfPedeEv[0] = evt.NumOfPedeEv
            self.NumOfSpilEv[0] = evt.NumOfSpilEv
            self.TriggerMask[0] = evt.TriggerMask
            for counter, l in enumerate(evt.ADCs.items()):
                self.ADCs[counter] = l[1]
            for counter, l in enumerate(evt.TDCs.items()):
                self.TDCsval[counter] = l[1][0]
                self.TDCscheck[counter] = l[1][1]
            self.tbtree.Fill()
        print "--->End rootification of "+self.drfname
```

Loop through events/lines and apply the DREvent.DRdecode() method.

Assign DRrootify data members

Fill a ROOT Tree with ADCs (int and array<int>)

Raw data

Raw data (PMTs + auxiliaries) are stored as ASCII files. An event looks like this

```
 ev # 105265 tow cts 95696 9570 103507 trigger mask 1 values: 0 47 16 d3 1 4b 17 e8 2 3b 18 de 3 48 19 da 4 39 20 95 5 10 21 b2 6 3f 22 f4 7
4 23 d7 8 c9 24 de 9 b0 25 d9 10 ed 26 ea 11 d7 27 dc 12 dd 28 ef 13 99 29 cb 14 bf 30 4a 15 c7 31 69 32 fd 48 10b 33 fe 49 118 34 10d 50 114
 35 132 51 10d 36 f3 52 11d 37 13e 53 113 38 118 54 11a 39 11e 55 101 40 130 56 100 41 109 57 fc 42 fb 58 12d 43 137 59 113 44 103 60 108 45
118 61 e5 46 f7 62 e7 47 10e 63 115 64 a0 80 e0 65 67 81 c8 66 c2 82 d1 67 a9 83 c5 68 cc 84 f3 69 ae 85 d7 70 bf 86 bb 71 d7 87 eb 72 e4 88
e6 73 d9 89 c1 74 e7 90 c1 75 101 91 e2 76 db 92 bd 77 ef 93 c7 78 d2 94 e1 79 ea 95 f0 TDC size 0 val.s
```

After rootification data look, referred as *rawNtuples*, looks like →

```
******************************************************************************
*Tree    :CERNSPS2021: CERNSPS2021                                            *
*Entries :    22468 : Total =        17854407 bytes  File  Size =    3731278 *
*        :          : Tree compression factor =   4.79                       *
******************************************************************************
*Br    0 :EventNumber : EventNumber/I                                        *
*Entries :    22468 : Total  Size=      90636 bytes  File Size  =      31672 *
*Baskets :        3 : Basket Size=      32000 bytes  Compression=   2.85     *
*............................................................................*
*Br    1 :NumOfPhysEv : NumOfPhysEv/I                                        *
*Entries :    22468 : Total  Size=      90636 bytes  File Size  =      30772 *
*Baskets :        3 : Basket Size=      32000 bytes  Compression=   2.93     *
*............................................................................*
*Br    2 :NumOfPedeEv : NumOfPedeEv/I                                        *
*Entries :    22468 : Total  Size=      90636 bytes  File Size  =       6365 *
*Baskets :        3 : Basket Size=      32000 bytes  Compression=  14.16     *
*............................................................................*
*Br    3 :NumOfSpilEv : NumOfSpilEv/I                                        *
*Entries :    22468 : Total  Size=      90636 bytes  File Size  =      31670 *
*Baskets :        3 : Basket Size=      32000 bytes  Compression=   2.85     *
*............................................................................*
*Br    4 :TriggerMask : TriggerMask/L                                        *
*Entries :    22468 : Total  Size=     180771 bytes  File Size  =       2850 *
*Baskets :        6 : Basket Size=      32000 bytes  Compression=  63.25     *
*............................................................................*
*Br    5 :ADCs      : ADCs[96]/I                                             *
*Entries :    22468 : Total  Size=    8654579 bytes  File Size  =    2955337 *
*Baskets :      271 : Basket Size=      32000 bytes  Compression=   2.93     *
*............................................................................*
*Br    6 :TDCsval   : TDCsval[48]/I                                          *
*Entries :    22468 : Total  Size=    4327913 bytes  File Size  =     599517 *
*Baskets :      136 : Basket Size=      32000 bytes  Compression=   7.21     *
*............................................................................*
*Br    7 :TDCscheck : TDCscheck[48]/I                                        *
*Entries :    22468 : Total  Size=    4328193 bytes  File Size  =      67644 *
*Baskets :      136 : Basket Size=      32000 bytes  Compression=  63.94     *
*............................................................................*
```

Note that we are saving integers as, for instance ADCs[96], no mapping with detector elements is done so far.

Raw data - SiPMs

Raw data from SiPMs boards are stored as .dat files.

Similarly to the previous case, they are decoded and rootified in rawNtupleSiPMs
and look like

```
******************************************************************************
*Tree    :SiPMData  : Data from SiPM                                         *
*Entries :   164471 : Total =        44940556 bytes  File  Size =   24567624 *
*        :          : Tree compression factor =   1.83                       *
******************************************************************************
*Br    0 :HighGainADC : HighGainADC[64]/s                                    *
*Entries :   164471 : Total  Size=   21070185 bytes  File Size  =   13798493 *
*Baskets :      164 : Basket Size=     128000 bytes  Compression=   1.52     *
*............................................................................*
*Br    1 :LowGainADC : LowGainADC[64]/s                                      *
*Entries :   164471 : Total  Size=   21070015 bytes  File Size  =    9522777 *
*Baskets :      164 : Basket Size=     128000 bytes  Compression=   2.20     *
*............................................................................*
*Br    2 :TriggerId : Triggerid/l                                            *
*Entries :   164471 : Total  Size=    1317313 bytes  File Size  =     150517 *
*Baskets :       10 : Basket Size=     128000 bytes  Compression=   8.50     *
*............................................................................*
*Br    3 :TriggerTimeStampUs : TriggerTimeStampUs/D                          *
*Entries :   164471 : Total  Size=    1317457 bytes  File Size  =     908085 *
*Baskets :       10 : Basket Size=     128000 bytes  Compression=   1.41     *
*............................................................................*
*Br    4 :BoardId   : BoardId/b                                              *
*Entries :   164471 : Total  Size=     165212 bytes  File Size  =      45380 *
*Baskets :        1 : Basket Size=     128000 bytes  Compression=   2.82     *
*............................................................................*
```

- Note that EventID in these files does not correspond to the PMTs EventID as per each "real event" each triggered board is written as a new entry. This means that to a given "real event" more than one tree entry correspond to it.
- In addition there is a constant offset between the TriggerIDs of the two DAQs.
- Need to align and merge the two root files.

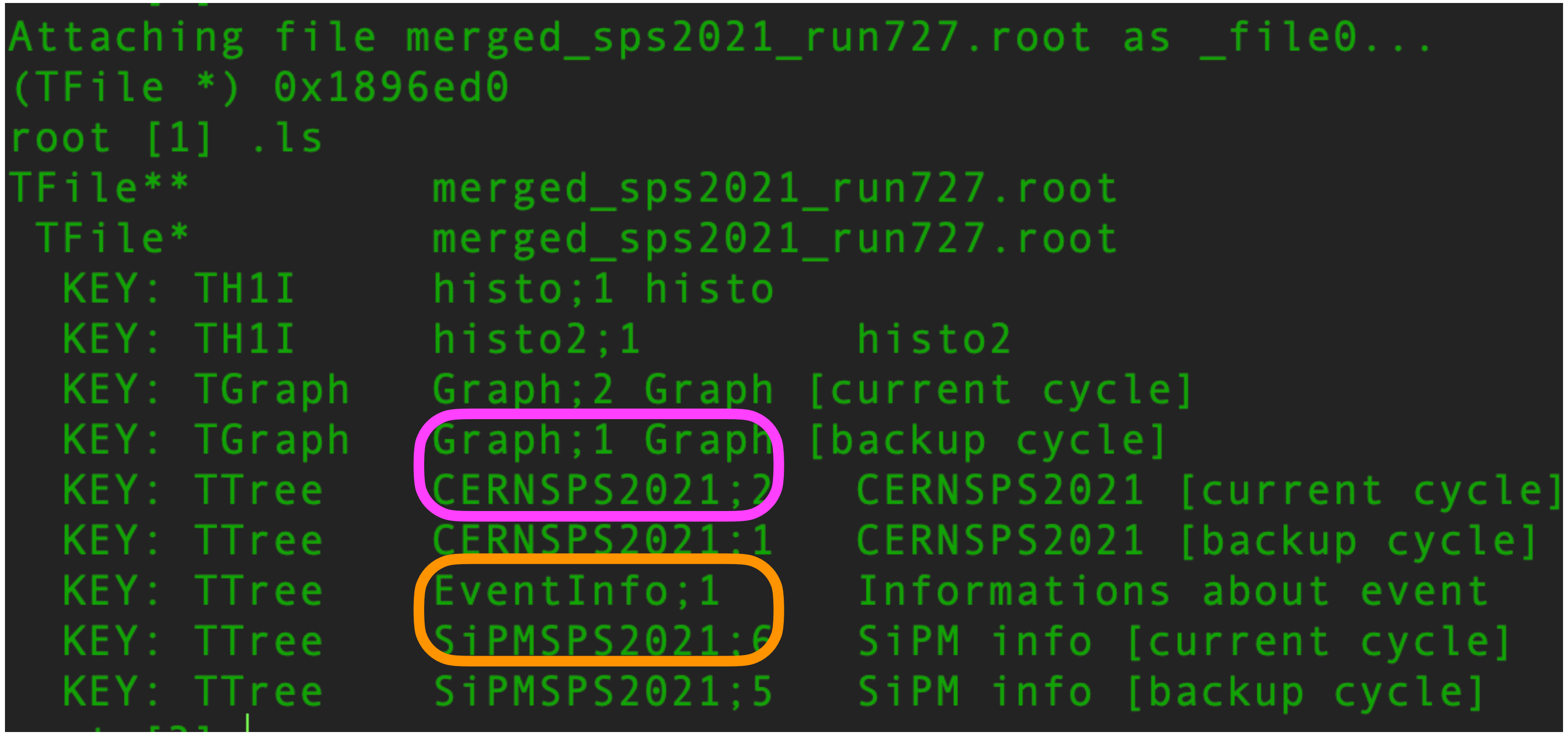
Merged data - PMTs + SiPMS

The alignment and merging of the two trees is done with [DRBlendedDaq2Root.py].

mergedNtuples are the first files that contain all the info on an event-by-event basis.
They contain two aligned trees,

- CERNSPS2021
- SiPMS2021

All merged_sps2021_run*.root are already produced and (hopefully) will be frozen all the analysis long.



Merged data - PMTs + SiPMS

The alignment and merging of the two trees is done with [DRBlendedDaq2Root.py].

mergedNtuples are the first files that contain all the info on an event-by-event basis.
They contain two aligned trees,

- CERNSPS2021
- SiPMS2021

All merged_sps2021_run*.root are already produced and (hopefully) will be frozen all the analysis long.

We decided to implement a further conversion and saving high-level-objects (classes) in new root files and distribute these files for the analysis….

```
Attaching file merged_sps2021_run727.root as _file0...
(TFile *) 0x1896ed0
root [1] .ls
TFile**         merged_sps2021_run727.root
 TFile*         merged_sps2021_run727.root
  KEY: TH1I     histo;1 histo
  KEY: TH1I     histo2;1        histo2
  KEY: TGraph   Graph;2 Graph [current cycle]
  KEY: TGraph   Graph;1 Graph [backup cycle]
  KEY: TTree    CERNSPS2021;2   CERNSPS2021 [current cycle]
  KEY: TTree    CERNSPS2021;1   CERNSPS2021 [backup cycle]
  KEY: TTree    EventInfo;1     Informations about event
  KEY: TTree    SiPMSPS2021;6   SiPM info [current cycle]
  KEY: TTree    SiPMSPS2021;5   SiPM info [backup cycle]
```

From mergedNtuple to physicsNtuple

We want to store high-level-objects as input for the analysis so that users:

- have access to energies instead of ADCs
- have access to detector components instead of vector<int>
- methods for the analysis can be made part of the class and automatically shared.

Conversion is done with [PhysicsConverter.C] and a [RunXXX.json] with calibration constants.


```
{
  "Calibrations" :
  {
    "SiPM" :
    {
      "highGainPedestal" : [64.61,59.20,59.40,60.16,60.09,61.02,60.45,59.72,60.26,61.24,60.2
      "lowGainPedestal" : [60.74,59.62,59.23,60.04,58.39,60.25,59.94,59.80,58.71,60.57,59.5
      "highGainDpp" : [24.63,26.31,25.78,24.86,23.84,23.56,24.54,24.53,28.43,24.19,24.68,25.
      "PhetoGeVS" : [200],
      "PhetoGeVC" : [82]
    },
    "PMT" :
    {
      "PMTS_pd" : [196,180,236,212,220,156,188,196],
      "PMTS_pk" : [657,650,781,726,642,763,623,808],
      "PMTC_pd" : [235,235,220,228,196,172,220,164],
      "PMTC_pk" : [800,754,816,744,733,835,725,782]
    }
  }
}
```


Be careful numbers must be updated!

Merged data - PMTs + SiPMS

Conversion is done with [PhysicsConverter.C] and a [RunXXX.json] with calibration constants.

```
using json = nlohmann::json;

ClassImp(EventOut)

void PhysicsConverter(const string run){
  // …
  //Open merge ntuples
  //
  auto *PMTtree = (TTree*) Mergfile->Get("CERNSPS2021");
  auto *SiPMtree = (TTree*) Mergfile->Get("SiPMSPS2021");
  //Create new tree and Event object
  //
  auto Outfile = new TFile(coutfile,"RECREATE");
  auto ftree = new TTree("Ftree","Ftree");
  ftree->SetDirectory(Outfile);
  auto ev = new Event();
  auto evout = new EventOut();
  ftree->Branch("Events",evout);
  //Create calibration objects
  //
  SiPMCalibration sipmCalibration("RunXXX.json");
  PMTCalibration pmtCalibration("RunXXX.json");
```

1

#include json libraries and EventOut class definition

Open trees from merged ntuple

Create Event and EventOut object and link EventOut to final root file

Retrieve calibration constants from json files

Merged data - PMTs + SiPMS

```
void PhysicsConverter(const string run){

  //Allocate branch pointers
  //
  int EventID;
   PMTtree->SetBranchAddress("EventNumber",&EventID);
  int ADCs[96];
  PMTtree->SetBranchAddress("ADCs",ADCs);
  SiPMtree->SetBranchAddress("HG_Board0",&ev->SiPMHighGain[0]);
  SiPMtree->SetBranchAddress("HG_Board1",&ev->SiPMHighGain[64]);
  SiPMtree->SetBranchAddress("HG_Board2",&ev->SiPMHighGain[128]);
  SiPMtree->SetBranchAddress("HG_Board3",&ev->SiPMHighGain[192]);
  SiPMtree->SetBranchAddress("HG_Board4",&ev->SiPMHighGain[256]);
  SiPMtree->SetBranchAddress("LG_Board0",&ev->SiPMLowGain[0]);
  SiPMtree->SetBranchAddress("LG_Board1",&ev->SiPMLowGain[64]);
  SiPMtree->SetBranchAddress("LG_Board2",&ev->SiPMLowGain[128]);
  SiPMtree->SetBranchAddress("LG_Board3",&ev->SiPMLowGain[192]);
  SiPMtree->SetBranchAddress("LG_Board4",&ev->SiPMLowGain[256]);
```

2

Link HG/LG_Board* to Event data members SIPMLowGain and SiPMHighGain

Map ADCs to Event data memebers (SPMT1, CPMT1, …)
ADC mapping happens here, good point to look for bugs

Calibrate and store this info in EventOut object.
EventOut is out final object ready for analysis.

```
  //Loop over events
  //
  for( unsigned int i=0; i<PMTtree->GetEntries(); i++){
    PMTtree->GetEntry(i);
    SiPMtree->GetEntry(i);
    evout->EventID = EventID;

    //Fill ev data members
    //
    ev->SPMT1 = ADCs[8];
    ev->SPMT2 = ADCs[9];
    ev->SPMT3 = ADCs[10];
    ev->SPMT4 = ADCs[11];
    ev->SPMT5 = ADCs[12];
    ev->SPMT6 = ADCs[13];
    ev->SPMT7 = ADCs[14];
    ev->SPMT8 = ADCs[15];
    ev->CPMT1 = ADCs[0];
    ev->CPMT2 = ADCs[1];
    ev->CPMT3 = ADCs[2];
    ev->CPMT4 = ADCs[3];
    ev->CPMT5 = ADCs[4];
    ev->CPMT6 = ADCs[5];
    ev->CPMT7 = ADCs[6];
    ev->CPMT8 = ADCs[7];
    evout->PShower = ADCs[16];
    evout->MCounter = ADCs[32];
    evout->C1 = ADCs[64];
    evout->C2 = ADCs[65];
    //Calibrate SiPMs and PMTs
    //
    ev->calibrate(sipmCalibration, evout);
    ev->calibratePMT(pmtCalibration, evout);
  }
```

3

Processed data analysis

A toy analysis

Once data are converted to EventOut objects, any analysis is much simpler.
A template [PhysicsAnalysis.C] to be used on recoNtuples/

1. #include custom libraries

2. Link EventOut object to root tree branch

3. Loop over Events

4. Retrieve Events info as PMT energies, SiPM energies, Auxiliary ADCs…

```
#include <TTree.h>
#include <TFile.h>
#include <TH2F.h>
#include <iostream>
#include <stdint.h>
#include <string>
#include <fstream>
#include "../../TBDataPreparation/202108_SPS/scripts/PhysicsEvent.h"

ClassImp(EventOut)

void PhysicsAnalysis(const string run){

  string infile = "/eos/user/i/ideadr/TB2021_H8/recoNtuple/physics_sps2021_run"+run+".root";
  std::cout<<"Using file: "<<infile<<std::endl;
  char cinfile[infile.size() + 1];
  strcpy(cinfile, infile.c_str());

  auto file = new TFile(cinfile);
  auto *tree = (TTree*) file->Get("Ftree");
  auto evt = new EventOut();
  tree->SetBranchAddress("Events",&evt);

  float energyS = 0;
  float energyC = 0;
  auto enesplot = new TH2F("splot", "splot", 100, 0., 100., 100, 0., 100.);

  for (unsigned int i=0; i<tree->GetEntries(); i++){
      tree->GetEntry(i);
      if (evt->PShower>500){                                  //PShower unit is ADC
              energyS = evt->totSiPMSene + evt->SPMTenergy;    //Energy unit is GeV
              energyC = evt->totSiPMCene + evt->CPMTenergy;
              enesplot->Fill(energyS, energyC);
      }
  }
  enesplot->GetXaxis()->SetTitle("Scintillation (SiPM+PMT) - Energy (GeV)");
  enesplot->GetYaxis()->SetTitle("Cherenkov (SiPM+PMT) - Energy (GeV)");
  enesplot->Draw("COLZ");
}
```

Making of the EventOut class

- The [EventOut] class is trivial, however….

- Much of the analysis code will be implemented directly inside the EventOut class as class methods. Some examples:

  - e^- selection method
  - μ^- selection method
  - calculate impact position at the calorimeter front face
  - Event display methods
  - …

  you can start working directly there.

```cpp
class EventOut{
  public:
     EventOut(){};
     ~EventOut(){};
     uint32_t EventID;
     float SPMT1, SPMT2, SPMT3, SPMT4, SPMT5, SPMT6, SPMT7, SPMT8;
     float CPMT1, CPMT2, CPMT3, CPMT4, CPMT5, CPMT6, CPMT7, CPMT8;
     float SiPMPheC[160] = {0};
     float SiPMPheS[160] = {0};
     float totSiPMCene = 0.;
     float totSiPMSene = 0.;
     float SPMTenergy = 0.;
     float CPMTenergy = 0.;
     int PShower, MCounter, C1, C2;

    void CompSPMTene()
    {SPMTenergy = SPMT1+SPMT2+SPMT3+SPMT4+SPMT5+SPMT6+SPMT7+SPMT8;}
    void CompCPMTene()
    {CPMTenergy = CPMT1+CPMT2+CPMT3+CPMT4+CPMT5+CPMT6+CPMT7+CPMT8;}
};
```

Action items

- Analysis preparation
  Test-beam: Add methods for particle selection using information from DWC, Cherenkov counters, PreShower detector, Muon counter and calorimeter.
  Simulation: Tune the light yield to the CERN test-beam data and configure the beam setup (moving platform, auxiliaries and beam line configuration).

- Analysis
  Extract new equalization and calibration constants using 20 GeV e^- runs in each tower.
  Cross checking the results with μ^- runs should be possible as well…
  Extract energy resolution and linearity plots and compare the first two points with the simulation.
  Study the prototype imaging capability and em-showers profiles.

- Documentation
  Each of the four project must the described in its README.md, as done for the simulation [here], any help is much appreciated.