Physics Analysis User Guide

for the SLAC 2016 testbeam

Kim Siang Khaw, University of Washington
Jason Hempstead, University of Washington
Thomas Stuttard, University College London
Anna Driutti, INFN, Sezione di Trieste e G.C. di Udine and Università di Udine
Antonio Gioiosa, INFN, Sezione di Lecce and Università, Università del Molise
Octavio Escalante, Università di Napoli

Abstract

This is a documentation for the SLAC2016 data analysis. It is intended to guide the users through the analysis framework based on *art* and ROOT. The purpose of SLAC test beam and the test beam program will be summarized in the first chapters, the data analysis framework specific for this test beam will be elaborated subsequently, and the DAQ data structure will be described at the end.

Contents

List of Figures

List of Tables

Acronyms

MIDAS	Maximum Integrated Data Acquisition System
TRUIMF	Canada's national laboratory for particle and nuclear physics and accelerator-based science
μTCA	Micro Telecommunications Computing Architecture
PSI	Paul Scherrer Institute

Chapter 1

Introduction

Prerequisites for the data analysis for SLAC test beam 2016 are a basic understanding of what the Muon g-2 experiment and PbF_2 calorimeters [1, 2] are, some knowledge about the electromagnetic (EM) shower and the ROOT data analysis framework [3]. You can follow the exercise sheet step by step which guides you to the data analysis using the reconstructed electron EM shower clusters. Some analyses may require the use of reconstructed crystal hit, which is the basic object of forming a cluster. Advanced users are encouraged to use the FNAL's *art* framework for the data analysis.

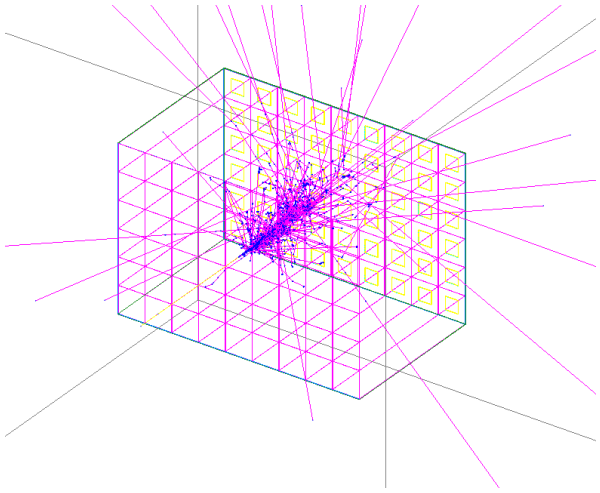

Figure 1.1: *Illustration of the electromagnetic shower of an electron injected from the left to the right, in a* 9×6 *array* PbF_2 *calorimeter. Cherenkov lights created by the charged shower particles are collected by the Silicon Photomultipliers (SiPMs) glued to the end of the* PbF_2 *crystals.*

Data analysis of this test beam has several components that are common with the Muon g-2 experiment data analysis framework [4]. The physics objects that will be used for most of the analyses are the crystal hits and the hit clusters. These objects are reconstructed

from the digitized waveform, after going through steps like pulsing fitting, energy calibration, gain correction, time correction and hit clustering. During the first week of the test beam, all these reconstruction steps will be refined based on the data taken and these will be taken care of by those who are familiar with the *art* framework. The main focus of this documentation is on the physics analysis using high level physics objects like the crystal hit and the cluster, using a C++/ROOT standalone framework. The user can proceed to more sophisticated analysis once he/she is familiar with the tools. The standalone framework can also be used as a stepping stone to develop analysis algorithm before migrating it to the *art* framework.

Several studies that will be covered for the data analysis (non-exhaustive list) are

- energy resolution of the calorimeter
- position and angular resolution of the calorimeter
- degeneracy of the position and angular information of the calorimeter
- stability of gain monitoring system
- pile up separation for multi-electron events

Chapter 2

T-536 program

This chapter summarizes the program we have planned during the SLAC run. For full program, please refer to Appendix A.

- Vertical Sweep
- Calibration Sequnce
- Template xtal 24
- Test PIX image of beam
- crystal centers in the cluster of 4x4 crystals
- high statistics seg 24
- Fine position scan of segment 24
- Cross of 4 segments
- Edge scan outward of segment 26
- Crack scan
- Energy scan at 3 points for each 2.5, 3.5, 4.0, 4.5, 5.0 GeV
- Edge scan outward of segment 26
- Angle Scan at 5, 10, 15 and 25 degrees
- Moving Flight Simulator Tests
- Fine vertical scan seg 24

- Horizontal scan seg 24
- Front face scan
- Undo wide islands for templates back to 8 pre-trigger, and 24 post-trigger
- Flight sim test, FW#1, 96 shots per fill
- Low QE run
- Long run
- Horizontal scan of fiber harp
- Horizontal scan of fiber harp in calibration position, vertically centered in beam
- Scan of intensity and bias voltage
- Horizontal scan of fiber harp in calibration position with narrow beam
- Bias voltage scan
- Horizontal scan of fiber harp in calibration position, new beam width
- Scan of intensity and bias voltage at ideal position along fibers
- Long run at ideal position along fibers
- Long run with white paint on fiber ends
- Horizontal scan of fiber harp in calibration position with white paint
- Vertical scan of T0 counter
- Laser before beam at different time separations

Chapter 3

Data period and logs

Data period is defined according to the filter wheel calibration period. We have mainly 7 periods and they are labeled from A to G. The number might changed once we get more insights from the data analysis. This chapter also summarizes the run log in the mysql database written out by the Maximum Integrated Data Acquisition System (MIDAS) DAQ and the `ELOG` for the SLAC run.

3.1 Data period

Table 3.1: *Summary of the data period. Included are the range of the run numbers and the date-time.*

Data period	Run numbers	Datetime range
A	3-1450	2016/05/31 23:38:30 - 2016/06/02 19:10:56
B	1451-1750	2016/06/02 19:11:43 - 2016/06/03 18:39:08
C	1751-2083	2016/06/03 18:39:40 - 2016/06/05 10:23:53
D	2084-2100	2016/06/05 10:29:14 - 2016/06/05 12:23:57
E	2101-2132	2016/06/05 12:24:05 - 2016/06/05 15:15:11
F	2133-2893	2016/06/05 15:15:38 - 2016/06/09 16:28:16
G	2894-3634	2016/06/09 16:28:29 - 2016/06/15 02:32:28

Each of these data periods has its own calibration constants and SiPM and PIN laser response baselines. (See Chapter 4 for more details.)

3.2 Runlog

The full runlog is documented in the `docdb` under http://gm2-docdb.fnal.gov:8080/cgi-bin/ShowDocument?docid=3964. A better organized runlog can be found under http://gm2-docdb.fnal.gov:8080/cgi-bin/RetrieveFile?docid=3964&filename=CondensedRunLog.xlsx&version=5.

Variables defined for each run are

- `runNum` : number of this run
- `startTime` : start time of the run
- `comment` : comment about this run
- `quality` : quality of this run (N, T, Y, C)
- `crew` : crew(s) on shift
- `beamE` : electron beam energy
- `tableX` : table x-coordinate
- `tableY` : table y-coordinate
- `angle` : angle of the calorimeter w.r.t. the normal placement
- `filterWheel` : filter wheel setting
- `bv1` : bias voltage 1
- `bv2` : bias voltage 2
- `bv3` : bias voltage 3
- `bv4` : bias voltage 4
- `stopTime` : stop time of the run
- `nEvents` : number of events in this run
- `fileSize` : midas file size
- `rate` : event rate

3.3 Elog

The full elog is hosted at https://muon.npl.washington.edu/elog/g2/SLAC+Test+Beam+2016/. Some of the milestones achieved during the SLAC run and interesting plots are summarized here.

Elog number	Comments	Run number (if applicable)
16	first filter wheel calibration	
17	p.e vs xtalNum, gain vs xtalNum and p.e. vs gain	900
26	calibration results after trying to equalize gains	1451-
27	xtal hit map (E-weighted)	1398
34	rider odd-even sample difference	
35	laser template variation	
37	first double pulse spotted	1676 (xtal24, event 898, islandNum 4)
38	black wrapping e^- beam template vs laser template	1673-1680
45	number of p.e. versus position	
56	deltaT from laser pulses	
58	intrinsic noise of the whole calo chain and pedestal	1800
59	xtal14 temperature over 30 hours	
82	laser calibration at 100 kHz	2133-
87	timing resolution versus laser pulse amplitude	2133-2139
88	Napoli DAQ analysis	3-5 Jun
97	trends for Jun 5 position and edge scan cluster energy, laser energy, avg. SiPM temp.	2122-2171
99	exponential plot of the flight simulator, 64 laser per fill	2187
101	linearity of the calorimeter (p.e. versus beam energy)	2352,2270,2282, 2308,2320,2344
103	fine scan in x position	
104	odd/even pedestal vs run number	
105	laser monitor stability (PMT, pin1/pin2 vs FC7 time)	
110	accelerator in 2-bunch mode ($\Delta T = 4.55$ ns)	2411-2425
111	template fit 2-bunch mode pulses ($\Delta T = 4.55$ ns)	2412
112	template fit 2-bunch mode pulses ($\Delta T = 9.8$ ns)	2436
117	deltaT from electron beam and laser pulses	
119	comparison of calibration constants (Jun 5 vs Jun 9)	
130	fitting 15 deg pulse with 0 deg pulse template	3034 and 3036

131	flight simulator runs, pedestal versus time	3109 and 3111
136	position reconstruction (logarithmic weight, W = 3.5)	1930-1936
137	laser response (loaded vs unloaded), FW = 5	3175-3191
140	laser response (loaded vs unloaded), FW = 1	3202
150	beam template overlay xtal14 and xtal24	
163	beam energy versus time (decreases at calo sides)	
166	energy spread across crystals	
169	low QE mode (electron comb)	3379-3380
172	per fill gain correction	
187	hit time distribution of neighboring xtals	1929-1936
217	laser energy versus pulse number	3003-3014 and 2401-2412
240	laser energy versus pin1+pin2	
244	applying gain corrections to runs from which the gain correction baselines are extracted	3308 and 3315

Chapter 4

Data analysis framework

4.1 MIDAS DAQ output in a nutshell

The main DAQ framework for the Muon g-2 experiment is based on MIDAS [5]. The Midas DAQ system is developed at Paul Scherrer Institute (PSI) and Canada's national laboratory for particle and nuclear physics and accelerator-based science (TRUIMF). MIDAS event structure is as depicted in Fig. 4.1.

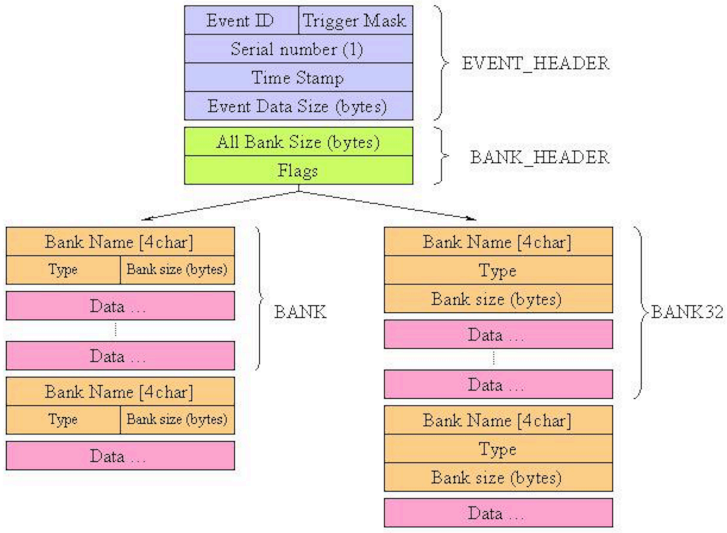

Figure 4.1: *MIDAS event structure. Each event has its header that is followed by the bank header. Then all the banks will appear according the defined order.*

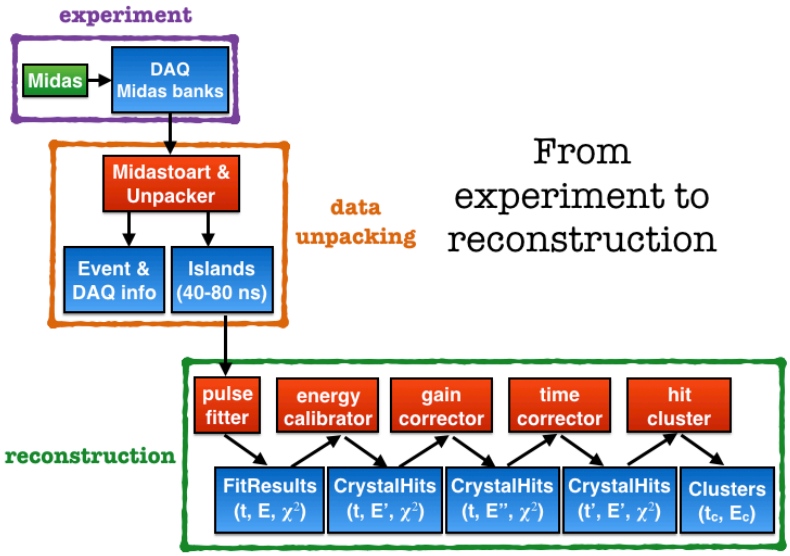

Figure 4.2: *An overview of the Muon g-2 offline framework.*

4.2 Offline framework for the SLAC test beam

As shown in Figure 4.2, data analysis for this test beam has several components. First we need to convert the raw data stored in a MIDAS file (`.mid` or `.mid.gz`) to *art* data products stored in an *art* file. This is handled using *art* framework's modules and is doing nothing more than storing `16-bit` or `32-bit` word into `vectors`. Next we unpack these `vectors` and give them contexts based on the header information stored within the `vectors`. At this step, all the information are stored as data products you are probably familiar with: `RiderArtRecord`, `IslandArtRecord`, etc. Then reconstruction algorithms are ran through these data products and at the end of the chain each physics objects are reconstructed as clusters. The final *art* framework used at SLAC is shown in Figure 4.3.

4.2.1 Data Acquisition

Explain the DAQ flow here roughly (from machine trigger to FC7, from FC7 fanout to AMC13 of all the Micro Telecommunications Computing Architecture (μTCA) crates, then from the AMC13 to AMCs(WFD5s) in a crate).

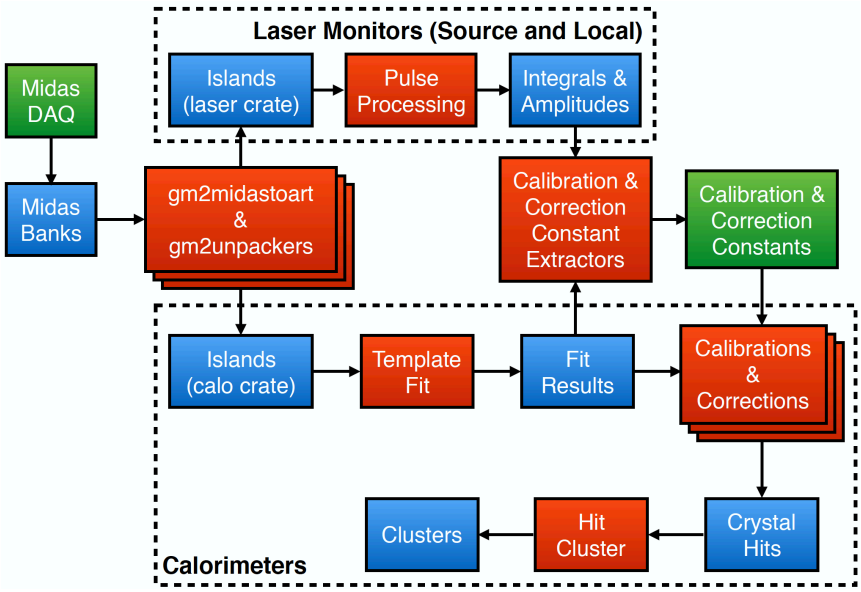

Figure 4.3: *Offline art framework for the SLAC experimental data.*

4.2.2 Data Unpacking

The task of unpacking the raw midas data is divided into several *art* modules. First, the Midas banks are converted into TBranches as vectors in the source input module under the repository gm2midastoart. Then the header information stored in the `CB` banks is unpacked by the module `HeaderUnpacker`, the raw waveforms stored in the `CR` banks in the calorimeter (fc7, laser) crate is unpacked by the RawUnpacker (LaserRawUnpacker, FC7Unpacker), the chopped islands stored in the `CT` banks is unpacked by the IslandUnpacker (LaserUnpacker) and so on. Fhicl file configuration will be explained in the next section.

Unpacking the slow control data like the temperature readout from SCS3000 requires some modification to the gm2midastoart's `MidasBankInput` module. The `event id` for the fast control events is assigned as 1 by the MIDAS DAQ whereas for the slow control it is 11 (`0x000b` in hex representation). The bank name for the SLAC2016 setup is `INPT` and the bank data type is `float`.

4.2.3 Reconstruction

The data reconstruction chain is consisted of 5 modules in series: pulse fitter, energy calibrator, gain corrector, time corrector and hit cluster. Fhicl file configuration will be explained in the next section.

Chapter 5

Getting started

5.1 Installation

This section is based on the official Offline Computing and Software Manual http://gm2-docdb.fnal.gov:8080/cgi-bin/ShowDocument?docid=1825. Only important steps are reproduced here.

5.1.1 Setting up the environment

First of all you need to login to the Fermilab virtual machine (`gm2gpvm02-04`). Once you have logged in, you need to choose a release for the Muon g-2 softwares (libraries, executables) every time you login or you can put it in your `.profile` on `gm2gpvm`. Do the following:

```
source /grid/fermiapp/gm2/setup
```

Alternatively if you want to run codes on your laptop, you can use cvmfs as mentioned in the documentation.

```
source /cvmfs/gm2.opensciencegrid.org/prod/g-2/setup
```

If successful, you will get

```
Using g-2 release repository at /cvmfs/gm2.opensciencegrid.org
g-2 software

--> To list gm2 releases, type
ups list -aK+ gm2
```

```
--> To use the latest release, do
setup gm2 v6_04_00 -q prof

For more information, see
https://cdcvs.fnal.gov/redmine/projects/g-2/wiki/ReleaseInformation.
```

Then follow the instruction and use the latest release of `gm2` (`v6_04_00` at the moment of writing this user guide). Next to do is to go to a folder of your choice (usually `~/work`) and then create a new development place.

```
cd ~/work
mkdir SLACDev
```

Now go into the newly created folder `SLACDev` and initialize it as a new development area

```
cd SLACDev/
mrb newDev
```

Follow the instruction to source the localproducts settings

```
source localProducts_gm2_v6_04_00_prof/setup
```

Now we are ready to install new packages.

5.1.2 Checking out unpacking and reconstruction packages

First we need to go into the `srcs/` folder to checkout the unpacking and reconstruction packages. Packages related to the data unpacking are `gm2midas`, `gm2midastoart` and `gm2unpackers` whereas those related to the reconstruction are `gmcalo`. `gm2dataproducts` holds all the data structures for both.

Checkout (`git`) `gm2calo` package using simplified command `mrb g`.

```
cd srcs/
mrb g gm2midas; mrb g gm2midastoart; mrb g gm2unpackers;
mrb g gm2calo; mrb g gm2dataproducts
```

For the package `gm2midas`, you need to use the master branch,

```
cd gm2midas
git checkout master
```

and build both `midas` and `rome`:

```
cd midas
export MIDASSYS=‘pwd‘
echo "export MIDASSYS=‘pwd‘" >> ~/.bash_profile
make NO_MYSQL=1
cd ../rome
export ROMESYS=‘pwd‘
echo "export ROMESYS=‘pwd‘" >> ~/.bash_profile
make clean; make
```

Please refer to https://cdcvs.fnal.gov/redmine/projects/gm2midas/wiki for more detail. Upon installing `midas` and `rome`, go to `gm2midastoart` to build the midas file library provided by ROME,

```
cd ../../gm2midastoart
git flow feature track SLAC2016
cd rome
make
```

For the packages `gm2unpackers`, `gm2calo` and `gm2dataproducts` you need to use the SLAC2016 feature branch (you will be checking out develop branch if you do nothing)

```
cd ../../gm2calo (gm2unpackers, gm2dataproducts)
git flow feature track SLAC2016
cd ..
```

Execute `'mrb uc'` to update the CMakeLists to include all the packages:

```
mrb uc
```

Setup `ninja` (a small build system with focus on speed) with

```
setup ninja v1_5_3a
```

Read more about it at https://ninja-build.org/. Then start to build the packages with

```
. mrb s
mrb b --generator ninja
```

If the build was successful (it takes about 5 minutes), you will see the following:

```
-----------------------------------
INFO: Stage build successful.
-----------------------------------
```

Now set an alias for the ninja (because it is fast! except the first build though) build and put it in your `.bash_profile`

```
alias ninja='pushd $MRB_BUILDDIR; ninja; popd'
```

Now you are ready to unpack the data, reconstruct the data and analyze the data.

5.1.3 Data unpacking and reconstruction

The `fcl` files for the data unpacking are location in the `gm2unpackers/fcl/` folder, whereas the `fcl` files for the data reconstruction are location in the `gm2calo/fcl/` folder. The script running both data unpacking and reconstruction are combined into a main `fcl` file named `unpackFitConstants.fcl`. Content of this fhicl file is shown below.

```
physics : {
    producers: {

    headerUnpacker : {
      module_type           : HeaderUnpacker
      unpackerInstanceName  : unpacker
      inputModuleLabel      : MidasBankInput
      wordToSkip            : 4096
      nCalos                : 2
      verboseLevel          : 0
        }

    fc7Unpacker : {
      module_type           : FC7Unpacker
      unpackerInstanceName  : unpacker
      inputModuleLabel      : MidasBankInput
      fc7CrateNum           : 2
      verboseLevel          : 0
        }

    rawUnpacker : {
```

```
  module_type           : RawUnpacker
  unpackerInstanceName  : unpacker
  inputModuleLabel      : MidasBankInput
  nCalos                : 1
  pedestalLevel         : 0
  verboseLevel          : 0
    }

islandUnpacker : {
  module_type           : IslandUnpacker
  unpackerInstanceName  : unpacker
  inputModuleLabel      : MidasBankInput
  nCalos                : 1
  nSegments             : 54
  verboseLevel          : 0
    }


islandFitter : {
  module_type           : TemplateFit
  fitterInstanceLabel   : fitter
  unpackerModuleLabel   : islandUnpacker
  unpackerInstanceLabel : unpacker
  peakCutoff            : 1850
  residualCutoff        : 50
  negativePolarity      : true
    }

energyCalibrator : {
  module_type            : EnergyCalibrator
  correctorInstanceLabel : calibrator
  fitterModuleLabel      : islandFitter
  fitterInstanceLabel    : fitter
  readConstants          : true
  constants              : @local::xtal_constants
    }

gainCorrector : {
  module_type            : GainCorrector
```

```
    correctorInstanceLabel : corrector
    calibratorerModuleLabel: energyCalibrator
    calibratorInstanceLabel: calibrator
      }

  hitCluster : {
    module_type            : HitCluster
    clusterInstanceLabel   : cluster
    correctorModuleLabel   : gainCorrector
    correctorInstanceLabel : corrector
    minEnergy              : 1000000
    timeCutoffLow          : 10
    timeCutoffHigh         : 20
      }
```

Since we have several run periods based on their own calibration constants, choose the nearest constant file which is below the run number of the file you want to analyze. For example, for run 1800, the nearest constant file is `constants1751.fcl`. However, `unpackFitConstants1751.fcl` is available so you do not have to worry about creating one yourself. To analyze any midas file you like, simply

```
gm2 -c unpackFitConstants1751.fcl -s run01800.mid -o gm2slac_run01800.art
-T gm2slac_run01800.root
```

where `-o` is followed by the location of the *art* file and `-T` is followed by the location of the root file you want store. Even better, an automation script, `auto_unpack_fit.sh`, where the calibration period is taken care of, is also available. Simply run this script with the following command:

`./auto_unpack_fit.sh startRunNum endRunNum`

provided you have set the paths in the bash script like `midas_dir`, `art_dir` and `root_dir` correctly.

5.1.4 Running the simulation

If you are intereted in running the simulation for the SLAC test run, you can download the package `gm2ringsim` and checkout the SLAC2016 feature branch.

```
cd srcs/
mrb g gm2ringsim
```

```
cd gm2ringsim
git flow feature track SLAC2016
```

To run the simulation, run the following fhicl file in the gm2ringsim/fcl folder:

```
gm2 -c slacTestBeam.fcl -o slacTestBeam.root
```

To reconstruct and analyze the simulated data, run the following fhicl file in the gm2calo/fcl folder:

```
gm2 -c slacTestBeamRecon.fcl -s slacTestBeam.root \\
-o slacTestBeam_recon.root -T slacTestBeam_ana.root
```

You can specify the name of the art output (after -o option) and TFile root output (after -T option) by yourself.

Chapter 6

Standalone C++ Analysis framework

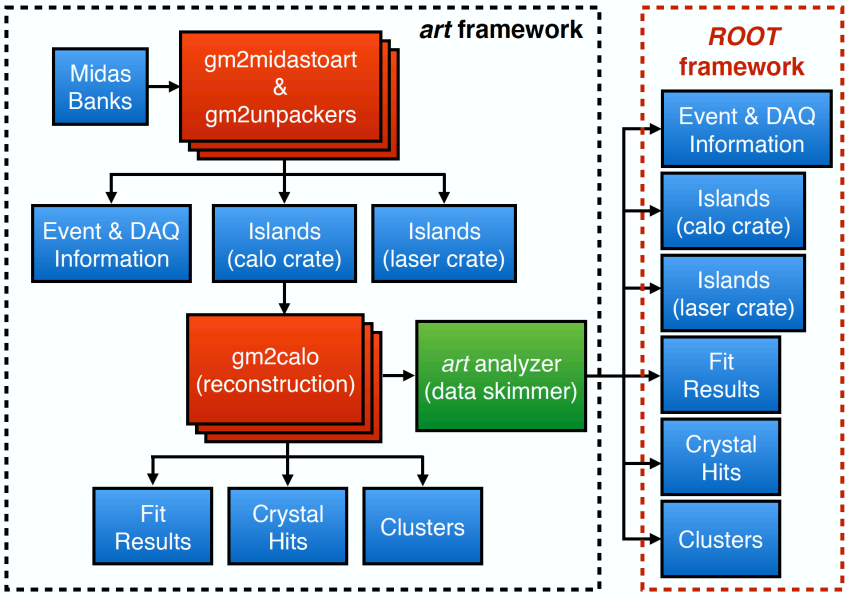

Figure 6.1: *Offline art and ROOT framework for the SLAC experimental data.*

The package `SLAC2016Ana` contains an example C++ framework to help you getting started. You can get it from the github.com by using the following command:

```
git clone https://github.com/kimsiang/SLAC2016+.
```

You can build your additional analysis code on top of this example or write a new one based on it. The example is already running but it's not doing much yet. You can compile the analysis code by first sourcing your ROOT environment

```
source thisroot.sh
```

and then followed by executing the command

```
make
```

This will read the necessary ingredients for compilation from the `Makefile` in the same directory. Don't have to worry much about this file at the moment unless you want to add in more classes to the analysis code. The point is that it creates an executable named `ana`. You can then execute the program by the command

```
./ana input.script
```

where `input.script` includes a path to the root file that you want to analyze (e.g. `./test.root`).

A description of the individual components of the example are given in the following list. Indicated are also the places where you should start adding your own code:

- `main.cxx`: This is the first starting point. It contains the `main()` function which is necessary for any `C++` program. The first step is to create instances of `MyAna()` class which is implemented in the files `MyAna.h` and `MyAna.C` (explained in the next items). The `TChain` represents the ROOT tree discussed in section 3. The files which should be read from disk are specified in the function `Add(filename)`. The tree is then read and processed by the `MyAna()` class which takes the `TChain` as argument. The real work is then done in the `Loop()` function of the `MyAna()` class which is discussed in the next two items.

- `MyAna.h`: Definition of the class `MyAna`, which inherits from the `TTree::MakeClass`. It declares variables and ROOT objects that will be used or stored in your analysis. Several basic functions that are common among event-based particle physics analysis like `initialize()`, `clear()`, `execute()` and`finalize()` are declared here.

- `MyAna.C`: The main function which is called automatically which processing the ROOT trees are `Loop()`. The `Loop()` function is called only once per run. In the `Loop()` function, `initialize()` is called at the beginning of the analysis run, `clear()` and `execute()` is called every event, and `finalize()` at the end of the analysis run.

- `t1.h`: Header file for the class `t1` created using `TTree::MakeClass`.

- `t1.C`: Source file for the class `t1` created using `TTree::MakeClass`. The class `Loop()` is used by `MyAna` to loop through each `TBranch`.

- `PlotAll.C`: A ROOT macro which can be used for automatic plotting of a set of histograms which are stored in a ROOT file. Please read the header of the file on how to use it.

6.1 Data format and structure for the ROOT tree

The data samples for this tutorial are stored in ROOT trees. The tree contains a collection of variables (called branches) which are filled once per event (could be 1 or more electrons). The list of variables along with their data type and further explanations are given in the following.

For studies using higher level objects

- `FitResult_EventNum` (`vector<int>`): event number of this fit result
- `FitResult_CaloNum` (`vector<int>`): calorimeter number of this fit result
- `FitResult_XtalNum` (`vector<int>`): crystal number of this fit result
- `FitResult_IslandNum` (`vector<int>`): island number of this fit result
- `FitResult_UtcaSlotNum` (`vector<int>`): utca slot number of this fit result
- `FitResult_ChannelNum` (`vector<int>`): rider channel number of this fit result
- `FitResult_Energy` (`vector<double>`): energy (number of photons) of this fit result
- `FitResult_Time` (`vector<double>`): time (clock tick of 800 MHz) of this fit result within the fill/event
- `FitResult_Pedestal` (`vector<double>`): pedestal of this fit result
- `FitResult_Chi2` (`vector<double>`): Chi squared of this fit result
- `FitResult_ClockCounter` (`vector<long long>`): time stamp (clock tick of 40 MHz) of this fit result from Rider header information
- `XtalHit_EventNum` (`vector<int>`): event number of this crystal hit
- `XtalHit_CaloNum` (`vector<int>`): calorimeter number of this crystal hit
- `XtalHit_XtalNum` (`vector<int>`): crystal number of this crystal hit
- `XtalHit_IslandNum` (`vector<int>`): island number of this crystal hit
- `XtalHit_UtcaSlotNum` (`vector<int>`): utca slot number of this crystal hit
- `XtalHit_ChannelNum` (`vector<int>`): rider channel number of this crystal hit
- `XtalHit_Energy` (`vector<double>`): energy (number of photons) of this crystal hit

- `XtalHit_Time` (`vector<double>`): time (clock tick of 800 MHz) of this crystal hit within the fill/event
- `XtalHit_ClockCounter` (`vector<long long>`): time stamp (clock tick of 40 MHz) of this crystal hit from Rider header information
- `Cluster_EventNum` (`vector<int>`): event number of this cluster
- `Cluster_CaloNum` (`vector<int>`): calorimeter number of this cluster
- `Cluster_IslandNum` (`vector<int>`): island number of this cluster
- `Cluster_Energy` (`vector<double>`): energy (number of photons) of this cluster
- `Cluster_Time` (`vector<double>`): time (clock tick) of this cluster
- `Cluster_X` (`vector<double>`): local x-position of this cluster (logarithmic-weighted)
- `Cluster_Y` (`vector<double>`): local x-position of this cluster (logarithmic-weighted)
- `Italiano_EventNum` (`vector<int>`): event number of this analyzed laser crate waveform
- `Italiano_CaloNum` (`vector<int>`): calorimeter number of this analyzed laser crate waveform
- `Italiano_XtalNum` (`vector<int>`): crystal number of this analyzed laser crate waveform
- `Italiano_IslandNum` (`vector<int>`): island number of this analyzed laser crate waveform
- `Italiano_UtcaSlotNum` (`vector<int>`): utca slot number of this analyzed laser crate waveform
- `Italiano_ChannelNum` (`vector<int>`): rider channel number of this analyzed laser crate waveform
- `Italiano_Amplitude` (`vector<double>`): amplitude (ADC samples) of this analyzed laser crate waveform
- `Italiano_Time` (`vector<double>`): time (clock tick of 800 MHz) of this analyzed laser crate waveform within the fill/event
- `Italiano_Pedestal` (`vector<double>`): pedestal of this analyzed laser crate waveform

- `Italiano_Area` (`vector<double>`): Area of this analyzed laser crate waveform
- `Italiano_ClockCounter` (`vector<long long>`): time stamp (clock tick of 40 MHz) of this analyzed laser crate waveform from Rider header information

For studies using lower level objects

Blow are the Tleaves of the `islandTree`.

- `RunNum` (`int`): run number of this island
- `EventNum` (`int`): event number of this island
- `FillType` (`int`): fill type of this island (1 is muon fill, 2 is laser fill)
- `TriggerNum` (`int`): trigger number of this island
- `CaloNum` (`int`): calorimeter number of this island
- `XtalNum` (`int`): crystal number of this island
- `IslandNum` (`int`): island number of this island
- `UtcaSlotNum` (`int`): utca slot number of this island
- `ChannelNum` (`int`): rider channel number of this island
- `Length` (`int`): number of sample of this island
- `Time` (`int`): time (clock tick of 800 MHz) of the first sample of this island within the fill/event
- `ClockCounter` (`long`): time stamp (clock tick of 40 MHz) of this island from Rider header information
- `Trace` (`vector<short>`): ADC samples of this island

Chapter 7

ROOT-based offline event display

A ROOT-based offline event display is also developed to increase the user friendliness of the data analysis. As shown in Figure 7.1, this GUI interface allows the user to inspect the fit results of the template fit algorithm by overlaying it with the island samples. More functionality will be added in the coming days.

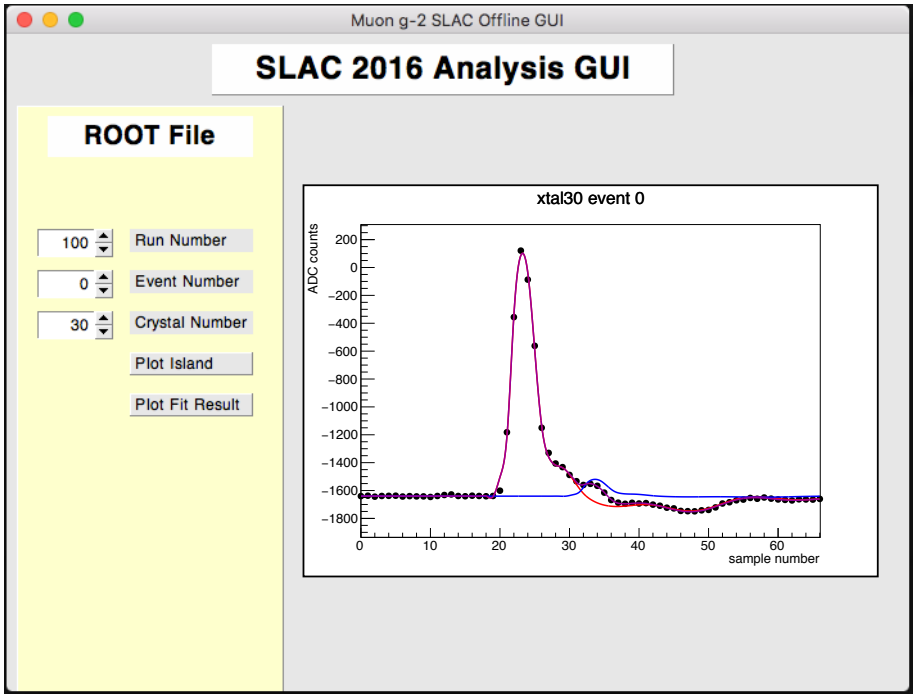

Figure 7.1: *SLAC Offline analysis GUI. This is a very preminary prototype.*

Chapter 8

Napoli DAQ

In this chapter, the Napoli DAQ will be explained. The codes for the plots are in https://github.com/gm2-it/test-beam-slac-jun16/tree/master/NapoliDaq/analyzer.

8.1 Overview of the electronics

A single Monitoring Board (MB) manages the entire signal processing for one of six Source Monitor (SM) elements. Light signals from the SM are opportunely divided among three channels: two PIN photodiodes and one PMT.

The signal at the output of each photodetector is processed through a multistage electronics chain based on the following: (1) preamplifier; (2) pulse shaper; (3) baseline restorer; and (4) 14 bit analog to digital converter. The preamplifier circuit has been designed on a separate daughter board to allow for a custom, optimized interface for the specified sensor (PIN or PMT).

For each channel the complete ADC readout and the control logic block are performed on a frontend FPGA where various slow control measurements (e.g. temperatures, bias voltage) are monitored. A 14 bit DAC on the same FPGA performs a self-calibration of the channel while a fourth, separate FPGA collects the signals from the three aforementioned FPGAs and communicates with the external interface, that is, provides the MB signals to the riders in order to be digitized.

8.2 Data format

Data is acquired in ASCII format, an example is shown in figure 8.1. Each line corresponds to an event and contains all the information readout by one of the three MB channels described previously. The data is recorded every time that one of the devices measures a

```
2016 6 2 56131 0 28 00000000 00000000 0004 02C0 0000 0000 0000 0000 02B8
2016 6 2 56131 0 28 00000000 00000000 0005 02C0 0000 0000 0000 0000 02B5
2016 6 2 56131 0 28 00000000 00000000 0006 02C0 0000 0000 0000 0000 02B8
2016 6 2 56131 0 28 00000000 00000000 0007 02C0 0000 0000 0000 0000 02B6
2016 6 2 56131 0 28 00000000 00000000 0008 02C0 0000 0000 0000 0000 02BA
2016 6 2 56131 0 28 00000000 00000000 0009 02C0 0000 0000 0000 0000 02B4
2016 6 2 56131 0 28 00000000 00000000 000A 02C0 0000 0000 0000 0000 02B3
2016 6 2 56131 0 28 00000000 00000000 000B 02C0 0000 0000 0000 0000 02B6
2016 6 2 56131 0 28 00000000 00000000 000C 02C0 0000 0000 0000 0000 02B8
2016 6 2 56131 0 28 00000000 00000000 000D 02C0 0000 0000 0000 0000 02B8
2016 6 2 56131 0 28 00000000 00000000 000E 02C0 0000 0000 0000 0000 02B4
```

Figure 8.1: *Example of the ASCII files written by the Naples DAQ.*

signal that exceeds a fixed threshold.

For each event the following information (columns of figure 8.1) is collected:

1^{st}, 2^{nd} and 3^{rd} columns: date (year, month and day) when the data are recorded (UTC time);

4^{th} column: time of the event (*i.e.*, the second of the day when the event happened);

5^{th} column: error code;

6^{th} column: number of bytes written;

7^{th} column: number of begin-of-fills;

8^{th} column: time in which the trigger occurred;

9^{th} column: number of triggers inside the fill;

10^{th} column: information about the MB and the signal (board address, channel, type of signal);

11^{th}, 12^{th}, 13^{th} columns: temperatures;

14^{th} column: information about the bias voltage of the device;

15^{th} column: charge of the signal.

8.2.1 From Raw Data to the ROOT tree

Data, with the exception of the time, the error code and the space information (first six columns of fig. 8.1) are recorded in hexadecimal. In order to make this data easy to analyze the raw data is converted into decimal numbers and stored in root files. These root files are generated using the program that can be found here: https://github.com/agioiosa/test-beam-slac-2016/tree/master/DAQ_Naples/SLAC2016Cruncher. The output of the program is a root file that contains a `TTree`, named `ntMonFrame` with three branches: `pmt`, `pin1` and `pin2`. Each branch includes the following information:

`ErrorCode`: is an error flag, if there are no errors it is set to be 0;

`frameloc`: is the numbers of bytes written. The expected value is 28, others numbers are an indication that an error is present;

`NBOF`: is the number of begin-of-fills occurred during the run (it is expected to be progressive);

`NtrgBOF`: numbers of triggers during the fill;

`NTimeTrgBOF`: time of the trigger with respect to the time of the BOF;

`PulseType`: set to be 1 in case of laser pulses and 2 in case of americium pulses.

`BoardAdr`: is the board address number (ranges from 0 to 11);

`ChBoard`: mother board channel that acquired the data. There are 3 channels A, B, C that correspond to 10, 11, 12, in decimal value. During the whole test beam, channel 10 was reading out the PMT and channels 11 and 12 the PIN1 and PIN2 outputs respectively;

`boardTemp`: MB temperature readout by a probe that is close to the channel that is acquiring the event (each channel has its own temperature sensor);

`cspTemp`: temperature of the preamplifier board related to the channel that is acquiring (*e.g.*, if ChBoard = 10, the cspTemp refers to the temperature of the PMT preamplifier board);

`extTemp`: temperature of the probe connected to the channel. During the test beam at SLAC we used two of the three probes. For the position of the probes please refer to UW elog post 78 [https://muon.npl.washington.edu/elog/g2/SLAC+Test+Beam+2016/78];

`Vbias`: bias voltage of the device related to the acquiring channel;

`ADCVal`: charge of the signal;

`t_year`: temporal informations from rs232 board: year;

`t_mon`: temporal informations from rs232 board: month;

`t_day`: temporal informations from rs232 board: day;

`t_secday`: temporal informations from rs232 board: seconds of day.

During this first stage, events with error code that differs from 0 are discarded. Events with other types of errors (*e.g.*, negative NBOF) might be still present and need to be removed during the further steps of the analysis. The decision of keeping also the "bad" events is supported by the fact that it is interesting to analyze the reason why the errors occurred and how they affect the data.

For the 2016 test beam at SLAC PulseType, cspTemp and one of the extTemp were not available.

8.2.2 Example of an Analysis

A preliminary off-line analysis of the data can be performed using the code posted in `https://github.com/agioiosa/test-beam-slac-2016/tree/master/DAQ_Naples/SLAC2016Ana`. The output of the code is a root file that contains the following plots:

prof_pmt_ls_adcval: charge of the laser pulse recorded by the PMT as a function of the time measured in days;

prof_pmt_am_adcval: charge of the Am-pulse recorded by the PMT as a function of the time measured in days;

h1_pmt_adcval: histogram of the charge of the signals recorded by the PMT;

h2_pmt_nbof NBOF (recorded in the PMT channel) as a function of the time measured in days;

h1_pmt_ntimetrgbof histogram of NTimeTrgBOF recorded in the PMT channel;

h2_pmt_ntimetrgbof NTimeTrgBOF recorded in the PMT channel as a function of the time measured in days;

prof_pmt_vbias PMT bias voltage as a function of the time measured in days;

prof_pmt_extemp temperature measured by one of the external probes during the data acquisition;

prof_pmt_boardtemp: temperature of the MB recorded in the PMT channel as a function of the time measured in days;

prof_pin1_adcval: charge of the laser pulse recorded by the PIN1 as a function of the time measured in days;

h1_pin1_adcval: histogram of the charge of the signals recorded by the PIN1;

h2_pin1_nbof NBOF (recorded in the PIN1 channel) as a function of the time measured in days;

h1_pin1_ntimetrgbof histogram of NTimeTrgBOF recorded in the PIN1 channel;

h2_pin1_ntimetrgbof NTimeTrgBOF recorded in the PIN1 channel as a function of the time measured in days;

prof_pin1_vbias: PIN1 bias voltage as a function of the time measured in days;

prof_pin1_extemp: temperature measured by one of the external probes during the data acquisition;

prof_pin1_boardtemp: temperature of the MB recorded in the PIN1 channel as a function of the time measured in days;

prof_pin2_adcval: charge of the laser pulse recorded by the PIN2 as a function of the time measured in days;

h1_pin2_adcval: histogram of the charge of the signals recorded by the PIN2;

h2_pin2_nbof: NBOF (recorded in the PIN2 channel) as a function of the time measured in days;

h1_pin2_ntimetrgbof: histogram of NTimeTrgBOF recorded in the PIN2 channel;

h2_pin2_ntimetrgbof: NTimeTrgBOF recorded in the PIN2 channel as a function of the time measured in days;

prof_pin2_extemp: PIN2 bias voltage as a function of the time measured in days;

prof_pin2_boardtemp: temperature of the MB recorded in the PIN2 channel as a function of the time measured in days;

prof_pin1Dpin2_adcval: ratio between the signal recorded by the PINs as a function of the time measured in days;

h1_pin1Dpin2_adcval: ratio between the signal recorded by the PINs;

h1_ratio_pin1_pin2_adcval: histogram of the ratio between the signal recorded by the PINs;

h2_pin1Vpin2_adcval: correlation between PIN1 and PIN2 signals;

h1_lsDam_adcval: histogram of the ratio between laser and americium pulses recorded by the PMT.

Temperatures are recorded in arbitrary units (*temp*) and then transformed into degrees Celsius by using the following relation: $temp \cdot 200/2047 - 50$.

Chapter 9

Fiber harp

This chapter covers the fiber harp unpacking and analysis software used during the 2016 SLAC test beam.

9.1 Overview

In addition to the calorimeter, the scintillating fiber harp detector was also tested during the SLAC 2016 test beam. A single fiber harp with 7 fibers was used, and data runs were taken with the harp orientated in both the in standard (transverse to beam) and calibrate (longitudinal in beam) positions.

As well as testing the fiber harp, a t0 counter scintllator detector was also installed for later runs and tested simultaneously with the fiber harp.

A laser sync pulse from the calorimeter laser calibration system was used in one rider channel for synchronisation purposes.

Two support detectors from SLAC were also used simultaneously; a downstream SciFi detector and an upstream ePix silicon detector.

The offline code described below includes data from the fiber harp, t0 counter, sync pulse and SciFi. The ePix detector was stored separately in a different data stream, and is not considered here.

The fiber harp readout was performed using a modified version of the calorimeter DAQ. This used a single AMC13 with 3 Riders. GPU island chopping was used, but with the GPUs only triggering from the sync pulse and SciFi waveforms. A trigger in either of these channels results in a waveform being recorded in all channels. Since the SciFi detects all primary beam particles, waveforms triggered by the SciFi are beam events, meaning that signals above pedestal may be expected in the fiber harp and t0 counter. Waveforms triggered by the sync pulse are outside of the beam window, and hence the waveforms in the fiber harp and t0 counter for these islands should be consistent with pedestal.

9.2 Installation

The fiber harp unpacking and reconstruction software overlaps heavily with the calorimeter, but with one additional package `gm2aux`.

Follow the guide in section 5.1.1 to set up the environment.

Follow the guide in section 5.1.2 to check out all required unpacking and reconstruction packages, but in addition checkout the package `gm2aux`.

```
cd $MRB_SOURCE
mrb g -b feature/SLAC2016 gm2aux
```

After check out this additional package, re-build the code using `mrb b` (not `ninja`):

```
mrb b --generator ninja
```

9.3 Unpacking the data

The fiber harp data is unpacked directly from MIDAS files. Run the unpacker as follows:

```
gm2 -c $MRB_SOURCE/gm2aux/fcl/unpackFiberHarp.fcl -s <.mid file(s)>
```

The output will be the file `gm2slac2016_unpackFiberHarp.art`, which contains the data product

```
gm2aux::FredinoArtRecordCollection
```

with module label `fiberHarpProcessor` and instance name `harp` which contains the unpacked data. This data product is filled once per island per fill. There are typically two islands per fill, one triggered by the sync pulse and the other by the beam. The fill and island numbers in each entry are given by the `fillNum` and `islandNum` variables in the data product. Also within the data product are containers of "hits" in the island for each detector type in the system, where a "hit" is the waveform in a given channel in the island, including some processed information usch as time of peak, pedestal, waveform area, etc.

- `harpHits`: Fiber harp island waveforms
- `syncLineHits`: Sync pulse island waveforms
- `sciFiHits`: SciFi island waveforms

- `t0CounterHits`: t0 counter island waveforms

Each "hit" in the container contains member date. The most commonly used members are listed below:

- `fiberNum`: Fiber harp only. Fiber number in range 1-7. Sync line, t0 counter and SciFi each have one channel only.
- `time`: Time of waveform peak from start of fill [*samples*]
- `pedestal`: Pedestal ADC value [*ADC*]
- `amplitude`: Height of peak above pedestal in ADC samples [*ADC*]
- `area`: Total size area under waveform above pedestal [$ADC \times samples$]

An example `art analyzer` for plotting fiber harp data can be found in :

```
gm2aux/analyses/FiberHarpSanityPlots_module.cc
```

Run it using:

```
gm2 -c $MRB_SOURCE/gm2aux/fcl/fiberHarpSanityPlots.fcl
-s <unpacked art file>
```

e.g.

```
gm2 -c $MRB_SOURCE/gm2aux/fcl/fiberHarpSanityPlots.fcl
-s gm2slac2016_unpackFiberHarp.art
```

Appendix A

T-536 program

This chapter contains the full program we have planned and executed during the SLAC run. A better organized runlog can be found under http://gm2-docdb.fnal.gov:8080/cgi-bin/RetrieveFile?docid=3964&filename=CondensedRunLog.xlsx&version=5.

Study	Comment / Instruction	Run Numbers
Vertical Sweep		1649 - 1653
Calibration Sequnce		1665 - 1672
Template xtal 24	One hour	1673 - 1680 (excluding 1677, 1679)
crystal centers in the cluster of 4x4 crystals:	2000 events per run	1759-1765, 1768-1775, 1872-1873, 1875-1878
34, 33, 32, 31 25, 24, 23, 22 16,15,14,13	use relPos buttons to move red means done	
high statistics seg 24	1 hour	1879-1886, 1888-1893
Fine position scan of segment 24	for y in 0, and 10 for x from -32 to 32 in steps of 4 (64 mm scanned out of 64 for y=0) Current position x=277, y=95, Xtal 23 3000 events per run total run time: 6 hours	1923-1943, 2092-2095, 2098-2099, 2122-2125, 2141-2146, 2149-2151
Cross of 4 segments	Intersection of segment 24, 25, 33, 34 Intersection of segment 25, 26, 34, 35	2152-2154
Edge scan outward of segment 26	For y=0, 6 points from center to calo edge For y=10, 6 points from center to calo edge Remain to be done: x={332.6,330.1,327.6}, y=95	2155-2161, 2164-2170
Crack scan	Crack 24-25 ; 25-26 ; fine for 24-25-33-34	2401-2404, 2407-2412; 2413, 2416-2417; 2418-2425

Energy scan at 3 points for each 3.5 GeV 4.0 GeV 4.5 GeV 5.0 GeV 2.5 GeV	4 X,Y Locations at each energy (277,105) (291.8,105) (304.4,105) (291.8,92.4) (279.2,105) (291.8,105) (304.4,105) (291.8,92.4) (279.2105) (291.8,105) (304.4,105) (291.8,92.4) (279.2,105) (291.8,105) (304.4,105) (291.8,92.4) (279.2,105) (291.8,105) (304.4,105) (291.8,92.4) (279.2,105)	3: 2269-2274; 3.5: 2283-2286, 2288-2290; 4: 2302-2308; 4.5: 2317-2326; 5: 2344-2348
Edge scan outward of segment 26	For y=95, x=347.45 (outside); 344.95 (outside); 342.45 ; 339.95 ; 337.45; 334.95; 332.45; 329.95 Repeat for y=105, x=347.45 (outside); 344.95 (outside); 342.45 ; 339.95 ; 337.45; 334.95; 332.45; 329.95	2428-2438
Angle Scan at 5 degrees	3000 events per step Y = 105; X = [284, 288, 292, 296, 300, 304, 308, 312, 316, 320, 324, 328, 332, 336, 340, 344, 348, 352, 356, 360] 3000 events per step Y = 95; X = [284, 288, 292, 296, 300, 304, 308, 312, 316, 320, 324, 328, 332, 336, 340, 344, 348, 352, 356, 360]	2839-2864, 2902-2923
Angle Scan at 10 degrees	3000 events per step Y = 105; X = [286, 290, 294, 298, 302, 306, 310, 314, 318, 322, 326, 330, 334, 338, 342, 346, 350, 354, 358, 362] 3000 events per step Y = 95; X = [286, 290, 294, 298, 302, 306, 310, 314, 318, 322, 326, 330, 334, 338, 342, 346, 350, 354, 358, 362] then a long run at Y = 95, X = 306 till 9AM	2948-2951, 2960-2982, 2984, 2989-2992, 2994-3002

Angle Scan at 15 degrees	3000 events per step Y = 105; X = [294, 298, 302, 306, 310, 314, 318, 322, 326, 330, 334, 338, 342, 346, 350, 354, 358, 362, 366, 370] 3000 events per step Y = 95; X = [294, 298, 302, 306, 310, 314, 318, 322, 326, 330, 334, 338, 342, 346, 350, 354, 358, 362, 366, 370] then a long run at Y = 95, X = 310 till 9PM with filter wheel calibration in parallel	3034, 3036-3038, 3040-3043, 3045, 3047-3048, 3057-3067, 3080-3097, 3099-3100
Moving Flight Simulator Tests (1)	FSrate=32; FWstate=6; 1/2 mode; T0 = -110950 (start of exp sequence) scan the first 20 usec, 3000 events per run trigger delay -120950, -118950, -116950, -114950, -112950, -110950, -120950, -122950, -124950, -126950, -128950, -130950, scan the first 100 usec, 3000 events per run trigger delay -130950, -140950, -150950, -160950, -170950, -190950, -210950 60 min runs Delay = 15 us, 30 us, 60 us, 120 us, 240 us, 480 us; Check statistics along way for run length precision in ratio of on/off	3153-3203

Moving Flight Simulator Tests (2)	FSrate=32; FWstate=5; 1/2 mode; T0 = -110950 (start of exp sequence) scan the first 20 usec, 3000 events per run trigger delay -120950, -118950, -116950, -114950, -112950, -110950, -120950, -122950, -124950, -126950, -128950, -130950, scan the first 100 usec, 3000 events per run trigger delay -130950, -140950, -150950, -160950, -170950, -190950, -210950	
Moving Flight Simulator Tests (3)	FSrate=32; FWstate=1; 1/2 mode; T0 = -110950 (start of exp sequence) scan the first 20 usec, 6000 events per run trigger delay -120950, -116950, -112950 (stopped), -124950, -130950, -140950, -160950	
Fine vertical scan seg 24	3000 events per run at X = 284, and Y = [85, 89, 93, 97, 101, 105, 109, 113, 117, 121, 125]	3220-3231
Horizontal scan seg 24	3000 events per run at Y = 105 and X = 280, 284, 288	
Front face scan	widen island for template gen (100 pre, 200 post) 4500 events per segment 26, 25, 24, 23, 22, 21, 20, 19, 18, 9, 10, 11, 12, 13, 14, 15, 16, 17, 35, 34, 33, 32, 31, 30, 29, 28, 27, 36, 37, 38, 39, 40, 41, 42, 43, 44, 53, 52, 51, 50, 49, 48, 47, 46, 45, 0, 1, 2, 3, 4, 5, 6, 7, 8	3232-3239, 3245-3254, 3256-3258, 3263, 3265-3269, 3271-3278, 3281-3289, 3291-3298, 3300-3304
undo wide islands for templates back to 8 pre-trigger, and 24 post-trigger		3305-3307

Flight sim test, FW#1, 96 shots per fill	FSrate=96; FWstate=1; 1/2 mode; 9000 events per run trigger delay -120950, -118950, -116950, -114950, -112950, -122950, -124950, -126950, -128950, -130950	3320-3322, 3358-3367
redo ping seg 8, 3	switch laser to 100kHz fixed widen islands for template generation collect 4500 events undo wide islands	3368-3369
low QE run	be creative	3371, 3373, 3375, 3377, 3379-3388, 3396-3398
long run	25 deg, two diff positions, same beam, 6 hours each, filter wheel calibration before, after and during	3413-3415, 3424-3429, 3431-3513, 3522-3556
position scan for 25 deg	9000 events per point Y = 105, X = 284.0, 288.0, 292.0, 296.0, 300.0, 304.0, 308.0, 312.0, 316.0, 320.0, 324.0, 328.0, 332.0, 336.0, 340.0, 344.0, 348.0, 352.0, 356.0, 360.0, 364.0, 368.0, 372.0, 376.0, 380.0 9000 events per point Y = 95, X = 284.0, 288.0, 292.0, 296.0, 300.0, 304.0, 308.0, 312.0, 316.0, 320.0, 324.0, 328.0, 332.0, 336.0, 340.0, 344.0, 348.0, 352.0, 356.0, 360.0, 364.0, 368.0, 372.0, 376.0, 380.0	3580-3619

horizontal scan of fiber harp	3000 events per run Y = 114, 110, 106, 102, 98 X = 240, 244, 248, 252, 256, 260, 264, 268, 272, 276, 280, 284, 288, 292, 296 300, 304, 308, 312, 316, 320, 324, 328, 332, 336, 340, 344, 348, 352, 356, 360	3961-3973, 3986-3990, 3999-4084, 4091-4129
horizontal scan of fiber harp in calibration position, vertically centered in beam	3000 events per run Y = 105 X = 240, 244, 248, 252, 256, 260, 264, 268, 272, 276, 280, 284, 288, 292, 296, 300, 304, 308, 312, 316, 320, 324, 328, 332, 336, 340, 344, 348, 352, 356, 360	4148-4178
scan of intensity and bias voltage	at Y = 105, X = 260 mm bias voltages = 65.5, 66.0, 66.5, 67.0, 67.5, 68.0, 68.5 VSL10 = 20, 15, 10, 5 mm C24_H = 1, 2, 4 mm 6000 events if SL10*C24_H <= 10 mmˆ2; otherwise 3000 events	4181-4195, 4198-4253, 4256-4269

horizontal scan of fiber harp in calibration position with narrow beam	3000 events per run Y = 105, X = 240, 244, 248, 252, 256, 260, 264, 268, 272, 276, 280, 284, 288, 292, 296, 300, 304, 308, 312, 316, 320, 324, 328, 332, 336, 340, 344, 348, 352, 356, 360, X = 242, 246, 250, 254, 258, 262, 266, 270, 274, 278, 282, 286, 290, 294, 298, 302, 306, 310, 314, 318, 322, 326, 330, 334, 338, 342, 346, 350, 354, 358	4270-4286, 4289-4291, 4294-4334
long run	1 hour = 36,000 events total 6 runs of 6000 events each (10 min) Y = 105, X = 260 SL10 = 20, C24 = 1.5	4337 - 4343
bias voltage scan	3000 events, Y = 105, X = 260, SL10 = 20, C24 = 1.5, bias voltages = 65.5, 66.0, 66.5, 67.0, 67.5, 68.0, 68.5 V	4344 - 4350

horizontal scan of fiber harp in calibration position, new beam width	3000 events per run SL10 = 20 mm, C24 = 1.5 mm, Y = 105, X = 240, 244, 248, 252, 256, 260, 264, 268, 272, 276, 280, 284, 288, 292, 296, 300, 304, 308, 312, 316, 320, 324, 328, 332, 336, 340, 344, 348, 352, 356, 360	4351 - 4357
scan of intensity and bias voltage at ideal position along fibers	at Y = 105, X = 276 mm bias voltages = 65.5, 66.5, 67.5, 68.5 V, C24_H = 1.5, 2, 4 mm SL10 = 20, 10, 5 mm, 3000 events	4358 - 4397
scan of intensity and bias voltage at ideal position along fibers	at Y = 105, X = 276 mm bias voltages = 66, 67, 68 V (at C24 = 2, SL10 = 20, do more points: 65.5, 66, 66.5, 67, 67.5, 68, 68.5 V) C24_H = 1.5, 2, 4 mm SL10 = 20, 10, 5 mm 3000 events	4458 - 4492
long run at ideal position along fibers	1 hour = 36,000 events total 6 runs of 6000 events each (10 min) Y = 105, X = 276 SL10 = 20, C24 = 2	4493-4495, 4503-4505

long run with white paint on fiber ends	1 hour = 36,000 events total 6 runs of 6000 events each (10 min) Y = 105, X = 276 SL10 = 20, C24 = 2	4512, 4514-4516
horizontal scan of fiber harp in calibration position with white paint	3000 events per run SL10 = 20 mm, C24 = 2 mm, Y = 105, X = 264, 268, 272, 276, 280, 284, 288, 292, 296, 300, 304, 308, 312, 316, 320, 324, 328, 332, 336, 340, 344, 348, 352, 356	4517-4518, 4520-4541
vertical scan of T0 counter	3000 events per run SL10 = 20 mm, C24 = 2 mm X = 288 Y = 65, 75, 85, 95, (105), 115, 125, 135, 145	4542-4545, 4547-4551
laser before beam at different time separations	3000 events, Y = 105, X = 276, SL10 = 20, C24 = 2, deltaT = 10, 20, 30, 40, 50, 75, 100, 150, 300 ns	4585, 4602-4603, 4605-4608, 4610-4614

Appendix B

Data location

This chapter summarizes the location of the files.

B.1 Data samples

This section explains the naming and the locations of the data files (Local location will be obsolete once we move everything to the PNFS disk at FNAL).

B.1.1 File naming

All the created art/ROOT and ROOT files will have the same run number as the MIDAS they are extracted from.

Table B.1: *Data samples and descriptions. The ROOT full data includes raw waveforms, chopped islands and processed data like fit results and crystal hits. The ROOT analysis data has only processed data.*

filename	type
gm2slac_run0xxxx.art	data ready for art-based analysis
gm2slac_run0xxxx_raw.root	full data for standalone C++ analysis
gm2slac_run0xxxx.root	analysis data for standalone C++ analysis

B.1.2 Location of the files (During SLAC test runs)

For the following steps, you have to be connected to the local Wifi network `T536-local`. All the art/ROOT and ROOT files are stored in the 2×3TB HDD in the `g2analysis` machine. Use the following command to copy over the data files that you want to analyze:

```
scp -r g2muon@g2analysis:/data1/slac2016/root/gm2slac_run0xxxx.root .
scp -r g2muon@g2analysis:/data2/slac2016/art/gm2slac_run0xxxx.art .
```

Or you can use `rsync` if you have enough space in your hard drive:

```
rsync -avurt g2muon@g2analysis:/data1/slac2016/root/ YOUR_LOCAL_DISK/
rsync -avurt g2muon@g2analysis:/data2/slac2016/art/ YOUR_LOCAL_DISK/
```

Or a better way is to use `sshfs` to mount the disk onto your laptop:

```
sshfs g2muon@g2analysis:/data1/slac2016/analysis YOUR_LOCAL_DISK
sshfs g2muon@g2analysis:/data2/slac2016/art YOUR_LOCAL_DISK
```

B.1.3 Location of the files (At Fermilab storage)

The final locations of the files are as the following:

```
RAW = /pnfs/GM2/slac2016/midas
ART = /pnfs/GM2/persistent/slac2016/art
ROOT = /pnfs/GM2/persistent/slac2016/root
```

Appendix C

Daq information

This chapter summarizes the minimum information from the DAQ needed for the analysis.

C.1 DAQ

C.1.1 Header information

This section provides the information regarding user interface to the event, AMC13 and rider header information. All the information are stored in the art/ROOT files and standalone ROOT files with the TBranch structures in the following sections.

C.1.2 Header and trailer formats

This section compiles all the available header data formats. Please refer to http://gm2-docdb.fnal.gov:8080/cgi-bin/ShowDocument?docid=3409 for more details.

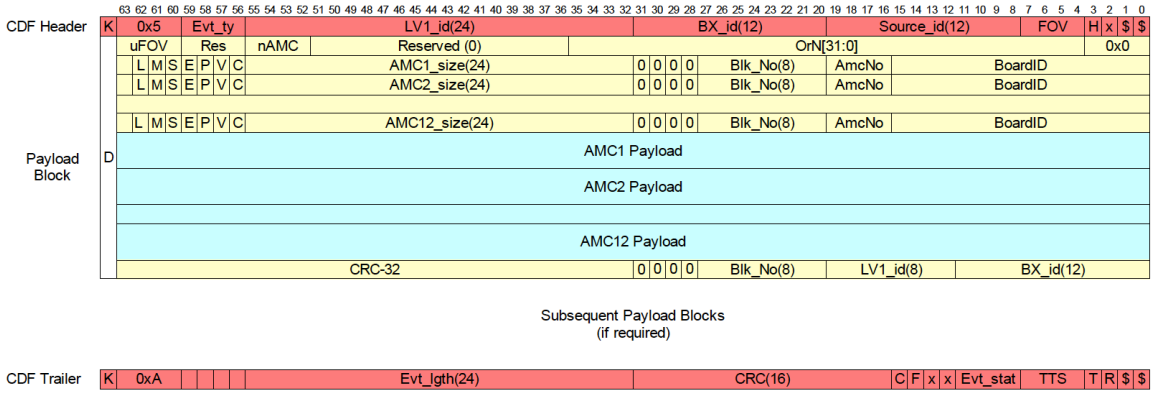

Figure C.1: *AMC13 to DAQ data format.*

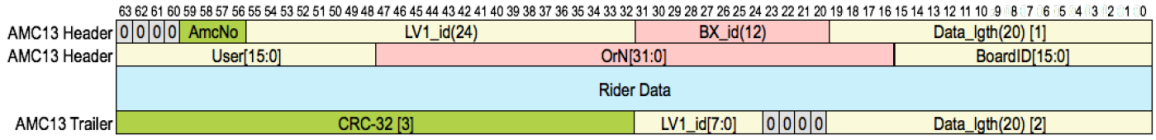

Figure C.2: *Rider to AMC13 data format.*

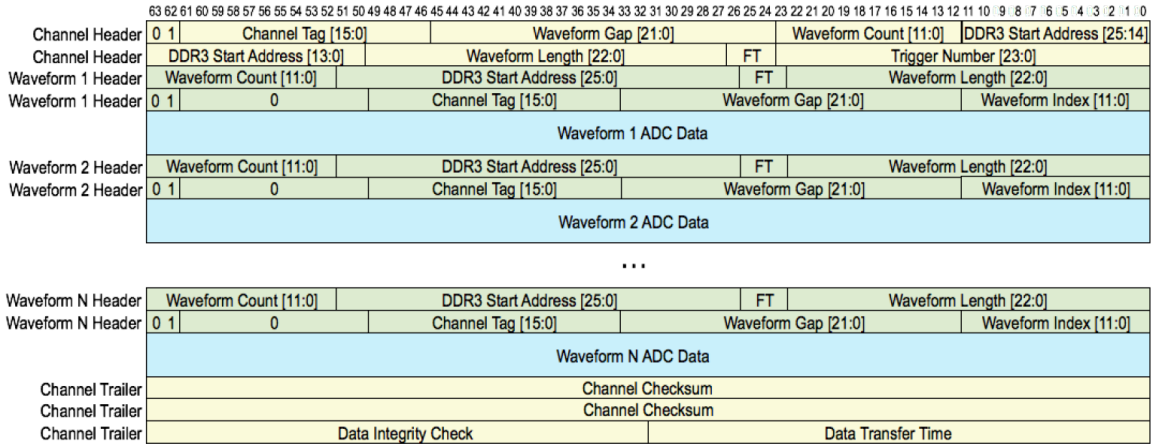

Figure C.3: *Rider Channel data format.*

C.2 Slow control data

The only slow control data recorded by the MIDAS DAQ is the temperature sensor SCS-3000. It was initially installed in the control room and was moved into the experimental hall on Jun 8 2016 in the night. It has 6 channels and the corresponding locations are summarized in Table C.1.

Table C.1: *Locations of the temperature sensors of SCS-3000.*

ID	Location
1	Air in
2	Loose on table in front of calorimeter
3	Loose on table in front of calorimeter
4	Loose on table in front of calorimeter
5	Air out
6	Tunnel entrance

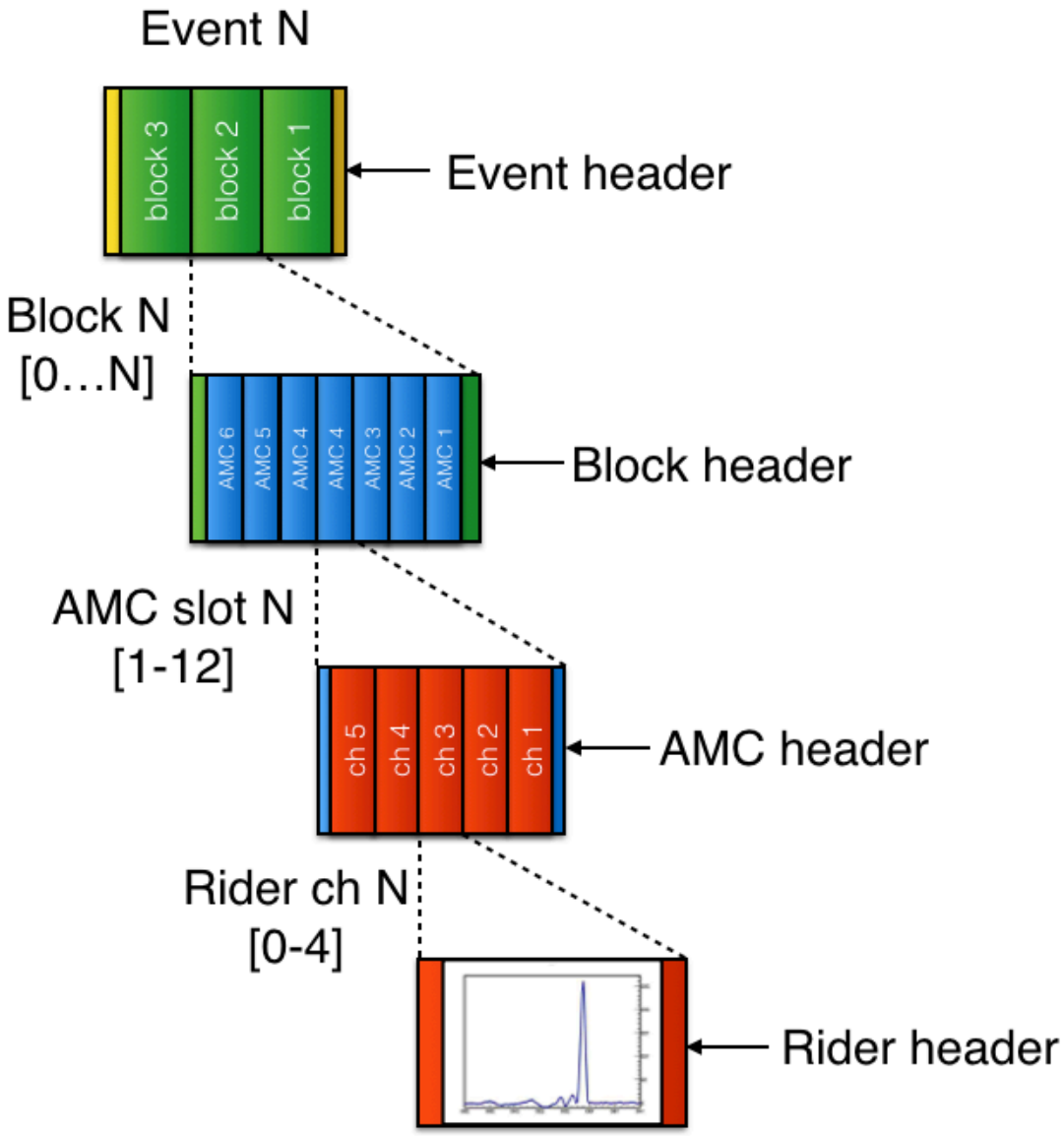

Figure C.4: *Per event data format*

Appendix D

Job submission in Fermigrid

To submit a job to the grid, the following two files are needed.

```
#!/bin/sh
# Usage: gridSetupAndSubmit-local.sh [FCL filename] [RUNNUM]

#[ ! -r $PWD/submit-localrelease.sh ] \
        && echo "Unable to access submit-localrelease.sh at $PWD !" \
        && exit

setup jobsub_client

NOW=$(date +"%F-%H-%M-%S")
SCRATCH_DIR=/pnfs/GM2/scratch/users/${USER}/slac2016_grid

echo "Scratch directory: ${SCRATCH_DIR}"

# only needed for the first time
mkdir ${SCRATCH_DIR}
mkdir ${SCRATCH_DIR}/logs
mkdir ${SCRATCH_DIR}/art
mkdir ${SCRATCH_DIR}/root
chmod -R g+w ${SCRATCH_DIR}

#enter the FHiCL file you want to use for the submission
MAINFCLNAME=${1:-ProductionMuPlusMuonGasGun}
echo "Main FCL: ${MAINFCLNAME}.fcl"
```

```
RUN=$2

#This submits the job to the grid using local release:
jobsub_submit -G gm2 -M --OS=SL6  \
        --resource-provides=usage_model=DEDICATED,OPPORTUNISTIC \
        --role=Analysis file://$PWD/submit-localrelease.sh \
        ${MAINFCLNAME} ${RUN} ${SCRATCH_DIR}

#!/bin/sh
# Usage: submit-localrelease.sh [FCL filename] [RUNNUM] [OUTPUTDIR]

MYDIR=/gm2/app/users/kkhaw/Work/slac2016/dev
localsetup=${MYDIR}/localProducts_gm2_v6_04_00_prof/setup
[ ! -f $localsetup -o ! -r $localsetup ] \
    && echo -e "\nUnable to access local setup file $localsetup\n" \
    && exit

source /cvmfs/fermilab.opensciencegrid.org/products/common/etc/setups
source /cvmfs/fermilab.opensciencegrid.org/products/larsoft/setup
setup ifdhc v1_6_2 -z /cvmfs/fermilab.opensciencegrid.org/products/common/db

#Set MIDAS and ROME environments
export MIDASSYS="${MYDIR}/srcs/gm2midas/midas"
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:${MYDIR}/srcs/gm2midas/midas/linux/lib
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:${MYDIR}/srcs/gm2midas/rome/
export ROMESYS="/${MYDIR}/srcs/gm2midas/rome"
#Set MIDAS event size (must match size the MIDAS file was created with)
export MIDAS_MAX_EVENT_SIZE=0x10000000

echo "Your environment in this job: " > job_output_${CLUSTER}.${PROCESS}.log
env >> job_output_${CLUSTER}.${PROCESS}.log

if [ "${PROCESS}" == "0" ]; then
    echo ${JOBSUBJOBID} > ${JOBSUBJOBID}
    ifdh cp -D ${JOBSUBJOBID} $3
fi

source /cvmfs/gm2.opensciencegrid.org/prod/g-2/setup
source $localsetup
```

```
#need to setup local products before running
. mrb slp

printf -v RUNNUM "%05g" $2

MIDASFILE=run${RUNNUM}.mid
ifdh cp /pnfs/GM2/slac2016/midas/${MIDASFILE} .

gm2 -c $1.fcl -s ${MIDASFILE} \
        -o gm2slac_run${RUNNUM}.art -T gm2slac_run${RUNNUM}.root

ifdh cp -D job_output_${CLUSTER}.${PROCESS}.log $3/logs
ifdh cp -D gm2slac_run${RUNNUM}.art $3/art
ifdh cp -D gm2slac_run${RUNNUM}.root $3/root
rm ${MIDASFILE}
```

Bibliography

[1] A. Fienberg et. al., *Studies of an array of PbF_2 Cherenkov crystals with large-area SiPM readout*, Nuclear Instruments and Methods in Physics Research Section A, Vol. 783, 12-21 (2015)

[2] J. Kaspar et. al., *Design and performance of SiPM-based readout of PbF_2 crystals for high-rate, precision timing applications*, JINST 12 P01009 (2017)

[3] Rene Brun and Fons Rademakers, *ROOT - An Object Oriented Data Analysis Framework*, Proceedings ALHENP'96 Workshop, Lausanne, Sep. 1996, Nucl. Inst. & Meth. in Phys. Res. A 389 (1997) 81-86. See also http://root.cern.ch.

[4] K. S. Khaw, *Offline structure group report*, FNAL E989 g-2 Experiment Document, FM2-doc-3781-v3 (2016). http://gm2-docdb-fnal.gov:8080/cgi-bin/ShowDocument?docid=3781

[5] TRIUMF MIDAS homepage, accessed May 13, 2016. https://midas.triumf.ca/MidasWiki/index.php/Main_Page