IPHC - Université de Strasbourg

DIGMAPS Documentation
A standalone software tool to study digitization with MAPS sensors
Author : Auguste BESSON (abessonin2p3.fr)
Date : October 24th 2011

This documentation is a short notice to explain how to run DIGMAPS.
First section shows how to run it in 2 minutes.
Section 2 reproduces the data card.
Section 3 describes the classes.
Section 4 is a short How to for developpers.

1 Quick start guide

1. Get the tar file *DIGMAPS.tar.gz*
2. Untar it :

   ```
   tar -xvzf DIGMAPS.tar.gz
   ```

   It creates a directory *DIGMAPS* which contains everything you need to start.
3. Compile (Root version $\geq$ 5.28)

   ```
   cd DIGMAPS/code
   Root
   root [0].x Compile.C
   Info in <TUnixSystem::ACLiC>: creating shared library ~/DIGMAPS/code/./digaction_cxx.so
   Info in <TUnixSystem::ACLiC>: creating shared library ~/DIGMAPS/code/./digadc_cxx.so
   Info in <TUnixSystem::ACLiC>: creating shared library ~/DIGMAPS/code/./digbeam_cxx.so
   Info in <TUnixSystem::ACLiC>: creating shared library ~/DIGMAPS/code/./digplane_cxx.so
   Info in <TUnixSystem::ACLiC>: creating shared library ~/DIGMAPS/code/./digtransport_cxx.so
   Info in <TUnixSystem::ACLiC>: creating shared library ~/DIGMAPS/code/./digparticle_cxx.so
   Info in <TUnixSystem::ACLiC>: creating shared library ~/DIGMAPS/code/./digreadoutmap_cxx.so
   Info in <TUnixSystem::ACLiC>: creating shared library ~/DIGMAPS/code/./digcluster_cxx.so
   Info in <TUnixSystem::ACLiC>: creating shared library ~/DIGMAPS/code/./digevent_cxx.so
   Info in <TUnixSystem::ACLiC>: creating shared library ~/DIGMAPS/code/./diginitialize_cxx.so
   Info in <TUnixSystem::ACLiC>: creating shared library ~/DIGMAPS/code/./dighistograms_cxx.so
   Info in <TUnixSystem::ACLiC>: creating shared library ~/DIGMAPS/code/./digmaps_cxx.so
   ```
4. Run :

   ```
   root [1] .x Run.C
   ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   ++                                                           ++
   ++                                                           ++
   ++                   WELCOME to DIGMAPS                      ++
   ++                                                           ++
   ++                                                           ++
   ++         Version: 1.01    Date: March 29th 2011            ++
   ++         Author: Auguste Besson abesson@in2p3.fr           ++
   ++                   (WITH OUTPUT FILE)                      ++
   ++                                                           ++
   +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
    Reading Setup from /var/autofs/nfs/rawcmos7/abesson/ILCSOFT/DIGITISEUR/code/input.txt
   [...]
   ... END
    REAL TIME = 10.9246 ; CPU TIME = 6.8
   root [1] .q
   ```

The output files are :
– output.txt which is an ASCII file for debugging. (all the print out will be put in this file).
– tree.root which stores everything in root class format.

5. Read the Root file you made and fill histograms :

```
Root
root [0] .x Read.C
[...]
... END
 REAL TIME = 4.18666 ; CPU TIME = 1.84
```

   It also creates a debugging output file *outputtree.txt*.

6. Print all the configurations you made :

```
root [1] myDIGMAPS2.PrintConfigurations()
**************************************************************
           List of all configurations
num |Thet|Phi |Pit x X y|epi |Nois|Temp|Nbits|lin|LSB  |gain   |Model|
   0| 0.0| 0.0|10.0X10.0|12.6|10.9|10.0| 1   |0  | 5.00|-999.00|1    |
   1| 0.0| 0.0|10.0X10.0|12.6|10.9|10.0| 1   |0  | 5.00|-999.00|2    |
   2| 0.0| 0.0|10.0X10.0|12.6|10.9|10.0| 1   |0  | 5.00|-999.00|3    |
   3| 0.0| 0.0|10.0X10.0|12.6|10.9|10.0|12   |1  | 0.64|   0.64|1    |
   4| 0.0| 0.0|10.0X10.0|12.6|10.9|10.0|12   |1  | 0.64|   0.64|2    |
   5| 0.0| 0.0|10.0X10.0|12.6|10.9|10.0|12   |1  | 0.64|   0.64|3    |
[...]
```

7. Plot the configuration number 5 :

```
root [3] myDIGMAPS2.PlotAConfiguration(5,1)
```

8. Superimpose configuration number 6 :

```
root [4] myDIGMAPS2.PlotAConfiguration(6,0)
```

2 Input card

All the parameters you could play with are set up in the input file (input.txt). The general idea is that you can run the program one time with different configurations to be able to compare it. You can chose :
- Beam (incident angle),
- Geometry (pitch, epitaxial layer thickness, Noise,
- Charge transport model,
- ADC/discris (N bits, thresholds).

Basically, the number of configurations will be :

$$configurations = NAngles \times NGeom \times NTransport \times NADC \qquad (1)$$

There will be *NumberOfEvents* events generated per configuration.
remark 1 : DO NOT USE Colons in Comments, the " :" is used as a marker to read the data card. So using it in comments would make the program unable to run.
remark 2 : units are all in micrometer.

```
// -+-+-+-+-+--+-+-+-+-+--+-+-+-+-+--+-+-+-+-+--+-+-+-+-+--+-+-+-+-+--+-+-+-+-+-
// -+-+-+-+-+--+-+-+-+-+--+-+-+-+-+--+-+-+-+-+--+-+-+-+-+--+-+-+-+-+--+-+-+-+-+-
//
// This is a Configuration File for Silicon Tracking Analysis DIGMAPS Package
//
// created   -> 18/03/2011
// Author = Auguste Besson abesson@in2p3.fr
// -+-+-+-+-+--+-+-+-+-+--+-+-+-+-+--+-+-+-+-+--+-+-+-+-+--+-+-+-+-+--+-+-+-+-+-
// -------------------- !!!! DO NOT USE Colons in Comments !!! -----------------
//
// -+-+-+-+-+--+-+-+-+-+--+-+-+-+-+--+-+-+-+-+--+-+-+-+-+--+-+-+-+-+--+-+-+-+-+-
// Action Parameter
// -+-+-+-+-+--+-+-+-+-+--+-+-+-+-+--+-+-+-+-+--+-+-+-+-+--+-+-+-+-+--+-+-+-+-+-
Model: "basic"

// -+-+-+-+-+--+-+-+-+-+--+-+-+-+-+--+-+-+-+-+--+-+-+-+-+--+-+-+-+-+--+-+-+-+-+-
// BEAM Parameter
// -+-+-+-+-+--+-+-+-+-+--+-+-+-+-+--+-+-+-+-+--+-+-+-+-+--+-+-+-+-+--+-+-+-+-+-
RunNumber: 1000
NumberOfEvents: 100
// Beam generation option.
//1=realistic beam with random number of particle per event
//2=1 particle per event with a hit in a central pixel
BeamOption: 2
//number of particles per mm^2 on the Plane per event (Lambda factor of a Poisson Law)
ParticleDensity: 5.0
//ParticleDensityWidth 0.5
// incident angle in degrees in cylindrical coordinates (theta and phi, in Degrees)
NAngles: 1
//ThetaIncidentDeg 0.0 10.0 20.0 30.0 40.0 50.0 60.0 70.0  0.0 10.0 20.0 30.0 40.0 50.0 60.0 70.0
//PhiIncidentDeg   0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0 30.0 30.0 30.0 30.0 30.0 30.0 30.0 30.0
ThetaIncidentDeg: 0.0
PhiIncidentDeg:   0.0

// -+-+-+-+-+--+-+-+-+-+--+-+-+-+-+--+-+-+-+-+--+-+-+-+-+--+-+-+-+-+--+-+-+-+-+-
// PLANE Parameters
// -+-+-+-+-+--+-+-+-+-+--+-+-+-+-+--+-+-+-+-+--+-+-+-+-+--+-+-+-+-+--+-+-+-+-+-

//-----------GENERAL GEOMETRY
//---pixel pitch in X and Y in microns
NGeom: 4
PitchX:             10.00 20.00 30.00 40.00
PitchY:             10.00 20.00 30.00 40.00
//---Noise in electrons
NoiseElectrons:     10.85  9.20  9.40  9.80
//---epitaxial thickness in microns
EpitaxialThickness: 12.65 10.80  8.69  7.78
//-----------OTHER PARAMETERS
//---number of pixels
NpixelsX: 21
NpixelsY: 21
//---Chip temperature
NTemperature: 1
Temperature: 10
//------------CHARGE TRANSPORT
//---ionization energy (eV)
IonizationEnergy: 3.6
//---Starting Segment size (in microns)
SegmentSize: 0.1
//---Maximum Segment size (in microns)
MaximumSegmentSize: 1.0
//---Maximum Charge Per Segment (in electrons)
MaximumChargePerSegment: 1.0
//---Diffusion Maximum Range in X and Y (in pitch units)
DiffusionMaximumRangeInX: 2.5
DiffusionMaximumRangeInY: 2.5
//---Reflexion Coefficient on the subtrat-epi border (1.0 means 100%) NOT USED
ReflexionCoefficient: 1.0
//---sigma of the gaussian width dispersion of charge at 10 microns depth
```

```
BasicModel_SigmaTenMicrons: 10.0
//------------CHARGE TRANSPORT MODEL
NTransport: 3
//-------------Transport 1
//---Chose Model (1=Lorentz2D , 2=2xGauss2D, 3=Lor+gaus)
ChargeModel: 1
RangeLimit_InPitchUnit: 2.5
//---Lorentz2D model
//---C term of the Lorentz width dispersion
Lorentz2DModel_Cp0: -0.340  //-0.340 //0.6607
Lorentz2DModel_Cp1: 0.36 // 0.360 //0.40  //0.400664
//---2xGauss2D Model
//sum of 2d gaussian The sigma are lineary dependant to the pitch
Gauss2DModel_sigma1_Cp0: 0.0
Gauss2DModel_sigma1_Cp1: 0.0
Gauss2DModel_sigma2_Cp0: 0.0
Gauss2DModel_sigma2_Cp1: 0.0
Gauss2DModel_weight: 0.0
//---Lor+gaus Model
LorGaussModel_Norm1_Cp0: 0.0
LorGaussModel_Norm1_Cp1: 0.0
LorGaussModel_Norm1_Cp2: 0.0
LorGaussModel_sigma_Cp0: 0.0
LorGaussModel_sigma_Cp1: 0.0
LorGaussModel_C_Cp0: 0.0
LorGaussModel_C_Cp1: 0.0
LorGaussModel_Norm_Cp0: 0.0
LorGaussModel_Norm_Cp1: 0.0
//-------------Transport 2
//---Chose Model (1=Lorentz2D , 2=2xGauss2D, 3=Lor+gaus)
ChargeModel: 2
RangeLimit_InPitchUnit: 2.5
//---Lorentz2D model
//---C term of the Lorentz width dispersion
Lorentz2DModel_Cp0: 0.0
Lorentz2DModel_Cp1: 0.0
//---2xGauss2D Model
//sum of 2d gaussian The sigma are lineary dependant to the pitch
Gauss2DModel_sigma1_Cp0: 1.12
Gauss2DModel_sigma1_Cp1: 0.35
Gauss2DModel_sigma2_Cp0: 1.16
Gauss2DModel_sigma2_Cp1: 0.83
Gauss2DModel_weight: 0.34
//---Lor+gaus Model
LorGaussModel_Norm1_Cp0: 0.0
LorGaussModel_Norm1_Cp1: 0.0
LorGaussModel_Norm1_Cp2: 0.0
LorGaussModel_sigma_Cp0: 0.0
LorGaussModel_sigma_Cp1: 0.0
LorGaussModel_C_Cp0: 0.0
LorGaussModel_C_Cp1: 0.0
LorGaussModel_Norm_Cp0: 0.0
LorGaussModel_Norm_Cp1: 0.0
//-------------Transport 3
//---Chose Model (1=Lorentz2D , 2=2xGauss2D, 3=Lor+gaus)
ChargeModel: 3
RangeLimit_InPitchUnit: 2.5
//---Lorentz2D model
```

```
//---C term of the Lorentz width dispersion
Lorentz2DModel_Cp0: 0.0
Lorentz2DModel_Cp1: 0.0
//---2xGauss2D Model
//sum of 2d gaussian The sigma are lineary dependant to the pitch
Gauss2DModel_sigma1_Cp0: 0.0
Gauss2DModel_sigma1_Cp1: 0.0
Gauss2DModel_sigma2_Cp0: 0.0
Gauss2DModel_sigma2_Cp1: 0.0
Gauss2DModel_weight: 0.0
//---Lor+gaus Model
LorGaussModel_Norm1_Cp0: 0.160573
LorGaussModel_Norm1_Cp1: -0.00184473
LorGaussModel_Norm1_Cp2: 5.07964e-05
LorGaussModel_sigma_Cp0: 0.95
LorGaussModel_sigma_Cp1: 0.60
LorGaussModel_C_Cp0: 0.171697
LorGaussModel_C_Cp1: 0.316165
LorGaussModel_Norm_Cp0: 5.69809
LorGaussModel_Norm_Cp1: 2.55373
// -+-+-+-+--+-+-+-+-+-+--+-+-+-+-+-+--+-+-+-+-+-+-+--+-+-+-+-+-+--+-+-+-+-+-+-+-
// ADC Parameters
// -+-+-+-+--+-+-+-+-+-+--+-+-+-+-+-+--+-+-+-+-+-+-+--+-+-+-+-+-+--+-+-+-+-+-+-+-
// number of different ADC to test
NADC: 3
// ADC parameters
// Nbits = number of bits of the ADC. There will be (2^Nbits - 1) thresholds.
// LSB, Electron_Conversion and ADC_thresholds are in Noise multiple units
// (e.g. 2.0 = 2.0 times the noise)
// There are 2 different ways to set the ADC
// ---------- EITHER --------------
// 1/ ADC_linear = 1 (the response is linear, so setting the LSB and the Electron Conversion
// factor allow to compute all thresholds.
// The thresholds will be = LSB, LSB+1xElectron_conversion, LSB+2xElectron_conversion etc.
// LSB= 1.5 = threshold of the first significant bit
// Electron_Conversion= 1.5
// ADC_thresholds= -
// ----------- OR --------------
// 2/ ADC_linear = 0 (the response is not linear, so enter all the different threshold values
// LSB= -
// Electron_Conversion= -
// ADC_thresholds= 2.0 4.0 5.0 etc.
// ---ADC 1
Nbits: 1
ADC_linear: 0
LSB: -
Electron_Conversion: -
ADC_thresholds: 5.0
// ---ADC 2
Nbits: 12
ADC_linear: 1
LSB: 0.64
Electron_Conversion: 0.64
ADC_thresholds: -
// ---ADC 3
Nbits: 12
ADC_linear: 1
LSB: 0.60
```

```
Electron_Conversion:  0.60
ADC_thresholds: -
// ---ADC 4
Nbits: 3
ADC_linear: 1
LSB: 1.0
Electron_Conversion: 1.0
ADC_thresholds: -
// ---ADC 5
Nbits: 4
ADC_linear: 1
LSB: 0.5
Electron_Conversion: 0.5
ADC_thresholds: -
// ---ADC 6
Nbits: 2
ADC_linear: 0
LSB: -
Electron_Conversion: -
ADC_thresholds: 2.0 4.0 5.0
```

3 Html documentation and class list

You can access to the Roothtml documentation in the directory *html*. It shows the different classes, data members and functions in a Root documentation way.

```
cd html
firefox ClassIndex.html
```

3.1 digmaps.h (DIGMAPS)

Main Class of DIGMAPS which contains pointers to all other classes and to the root tree. It contains :
- Run() function (loop on all configurations)
- ActionPlot() function (plot a configuration)
- RunConfiguration() loop on all events for a given configuration

3.2 diginitialize.h (DIGInitialize)

Class performing the initialization (reads the input data card and store it).

3.3 dighistograms.h (DIGHistograms)

Class containing histogram list stored in TObjArray.

3.4 digaction.h (DIGAction)

Class containing the main action foreseen by the program (make the tree, plot, etc.).

3.5 digbeam.h (DIGBeam)

Class containing incident particles / beam informations

3.6 digplane.h (DIGPlane)

Plane/Chip class which contains geometrical information on the chip (number of pixels, pitch, epitaxial layer thickness etc.).

3.7 digevent.h (DIGEvent)

Event class which contains particle list, cluster list and digital output of the plane (see DIGReadoutmap).

3.8 digparticle.h (DIGParticle)

particle class which contains :
- entry and exit point of the particle into the plane,
- segment list (Charge, position) of the track,
- pixel list (number, charge) where charge has been collected.

3.9 digtransport.h (DIGTransport)

Contains charge transport models parameters.

3.10 digadc.h (DIGADC)

Class containing the ADC/discri features (Nbits, thresholds, etc.).

3.11 digcluster.h (DIGCluster)

Class containing cluster information (pixel list, digital charge).

3.12 digreadoutmap.h (DIGReadoutmap)

Class containing the final output of the chip
- list of pixels with a collected charge ¿0,
- Analog charge list,
- Digital charge list.

3.13 digproto.h (DIGProto)

Dummy class to be used as a model.

4 Development and how to

4.1 Add a new parameter in the data card

Assume we want to add a parameter *A* to the DIGPlane class.

- Modify this function :

```
DIGInitialize::DIGInitialize(char *name, char *title, TString aCP, TString aCFN, TString action),
[...]
  read_item(PlaneParameter.A);
```

- Modify DIGInitialize.h :

```
 struct PlaneParameter_t {
 [...]
Float_t A;
 }PlaneParameter;
```

- Modify the data card accordingly (at the right place) : A : 0.5
- transfert the parameter into some class (DIGADC,DIGBeam, DIGPlane, etc.), for instance :

```
DIGMAPS::Run()
[...]
aDIGPlane->SetParameterA((fDIGInitialize->GetPlanePar().A[igeom]));
```

- add this parameter to the considered class, and create the corresponding Get/Set functions. For instance in the class DIGPlane :

```
 public:
  void SetParameterA(Float_t A){fA = A;}
  Float_t GetParameterA(){return fA;}
 protected:
    Float_t fA;
```

4.2 Add a new class

You can start from the digproto.h/digproto.cxx files which contain a dummy class (DIGProto). Then you need to compile it like the other class.

4.3 Add a new histogram

A given histogram is actually filled for a given configuration. So we need to create an TObjArray of histograms with one histogram for each configuration.

- Declare TObjArray in *dighistograms.h* :
  ```
  class DIGHistograms
  [...]
  TObjArray *Ar_h1_test;
  ```
- Book histogram:
  ```
  void DIGHistograms::BookHistograms(Int_t myNumberOfConfigurations){
  [...]
    Ar_h1_test = new TObjArray(fNumberOfConfigs);
  [...]
      sprintf(titre,"Ar_h1_test%d",i);
      h1temp= new TH1F(titre,titre,1000,0,2000);
      Ar_h1_test->Add(h1temp);
  ```
- Fill histogram :
  ```
  void  DIGMAPS::ActionPlot()
  [...]
  ((TH1F*)Ar_h1_test->At(Current_configuration))->Fill( XXX );
  ```
- Plot it on a canvas : Edit this method :
  ```
  void  DIGMAPS::PlotAConfiguration(Int_t confignumber, Bool_t newcanvas)
  ```

4.4 Add a new energy deposition model

It should be done at this place :

```
void  DIGMAPS::RunConfiguration(Int_t configcounter, Int_t BeamNumber, Int_t PlaneNumber,[...])
[...]
//---------Energy deposition generation
```

4.5 Add a new charge transport model

It should be done at this place :

```
void DIGParticle::ComputeChargeTransport(DIGPlane *aDIGPlane,DIGTransport *aDIGTransport)
```

Then add a new case for your new charge transport model.

4.6 Add a new clustering model

Right now, there is only a perfect clustering algorithm based on Monte-carlo information. It is done here :

```
void  DIGMAPS::RunConfiguration(Int_t configcounter, Int_t BeamNumber, Int_t PlaneNumber,[...])
[...]
fdigevent->BuildTrueClusters(GetPlane(PlaneNumber));
```

which calls this function :

```
void DIGEvent::BuildTrueClusters(DIGPlane *myDIGPlane)
```

So create a new function in the DIGEvent class with your new clustering algorithm.

4.7 Add a new beam model

The beam options are used here :

```
void  DIGMAPS::RunConfiguration(Int_t configcounter, Int_t BeamNumber, Int_t PlaneNumber,[...])
```

4.8 Modify the Analog To Digital Conversion model

Now, there is no fixed patern noise. The conversion is made here :

```
void DIGReadoutmap::AnalogToDigitalconversion(DIGADC *myDIGADC,  DIGPlane *myDIGPlane )
```

The random noise is computed for each particle, here :

```
void  DIGMAPS::RunConfiguration(Int_t configcounter, Int_t BeamNumber, Int_t PlaneNumber,[...])
      //---------random noise (should be removed if one wants to avoid double noise on double hit pixels)
      fdigparticle->AddRandomNoise(GetPlane(PlaneNumber));
```

So, to avoid adding double noise in case of double hits, this line should be removed, and the noise should be put at the ADC level.