

ROOT - A brief introduction

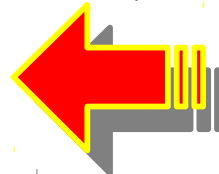
Survival *with* ROOT != Survival *at* ROOT != Survival *despite* ROOT



The goals we have...



- ◆ Introduction to ROOT
 - ◆ What is it ?
 - ◆ Why is it good ?
- ◆ Using ROOT
 - ◆ Command line
 - ◆ Scripting/Interpretation
 - ➔ Example script comparing two current-mode D/A converter architectures
 - ◆ Compilation
 - ➔ Compiling a script into a "*.so" shared object library
 - ➔ Compiling standalone
 - ➔ Application development
 - ➔ Example standalone application
- ◆ GUI of ROOT
 - ◆ Human interaction
 - ◆ Creating a GUI
- ◆ Survival with/at/despite ROOT
 - ◆ User's guide
 - ◆ Referring to:
 - ➔ \$ROOTSYS/tutorials
 - ➔ \$ROOTSYS/test
 - ◆ HTML source code documentation
- ◆ External library usage from within ROOT
 - ◆ DQM of ALICE experiment @ CERN
 - ➔ Simplified DAQ operation
 - ◆ Understanding the detector data
 - ➔ Accessing and decoding data



What is it ?

Well, it is:

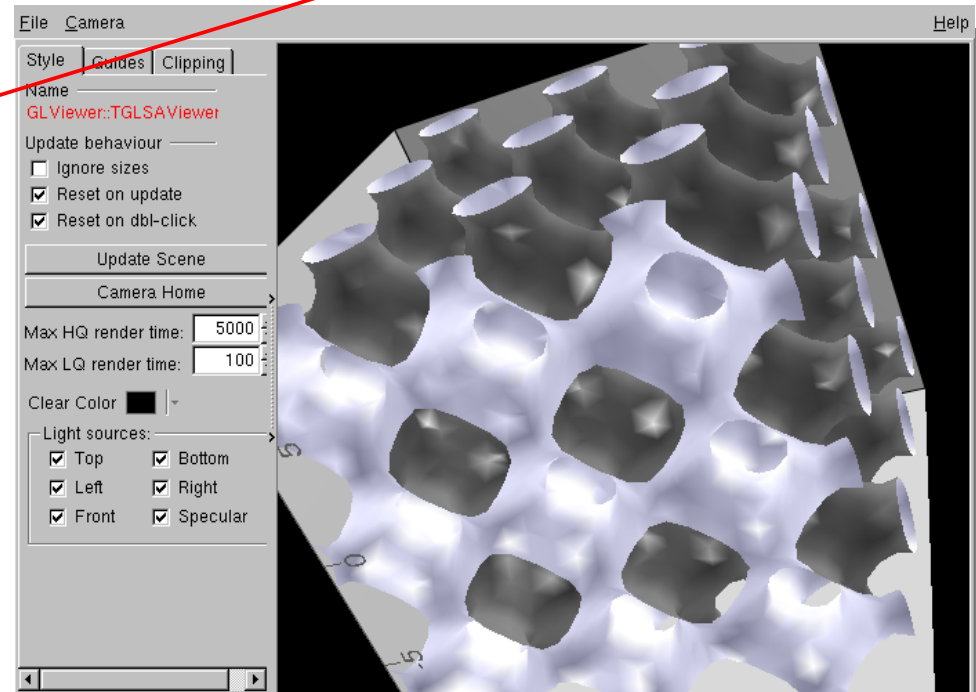
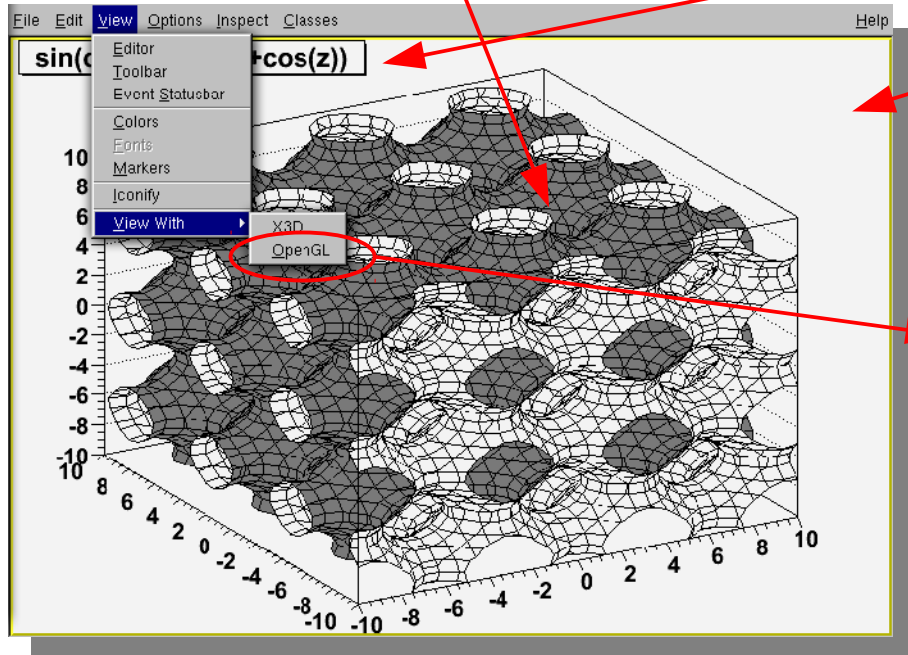
- ◆ **A library:** ROOT is a very **large class library** consisting of specialized smaller libraries. ROOT libraries can be included in **user programs** and be used as external libraries:
 - **GUI development library**
 - Library providing **signal-slot** mechanism
 - Libraries encapsulating functions for **data analysis**, etc.
- ◆ **A C/C++ interpreter:** ROOT has CINT -an embedded C **interpreter**- which allows easy algorithm development. You can execute C/C++ statements just like executing commands on a Linux shell or like writing shell scripts.
 - This way of working generates applications running slower, however:
 - Development is **faster** and in case of necessity the scripts can easily be compiled into *.so objects/libraries which allows faster execution
- ◆ **A framework:** ROOT provides the **infrastructure** needed by the physicists to work
 - Interpreter, histogramming and analysis functions, GUI development capability, I/O functions, class libraries, parallel processing and threads, functions specific to sockets and network communication, etc. These are all **ready to use**, minimizing the effort of development.

Why is it good ?

Because of the supposed fact that:

- ◆ There are advantages of working within a **framework**, such as the following:
 - No need to write **many commands** to achieve a **specific functionality**
 - High **reliability** of developed code due to extensive library usage
 - **Consistent** class hierarchy of the developed code
 - **Flexibility** of a modular architecture, thus “develop-and-reuse” is easier
 - Physicist can **focus** on his/her subject more

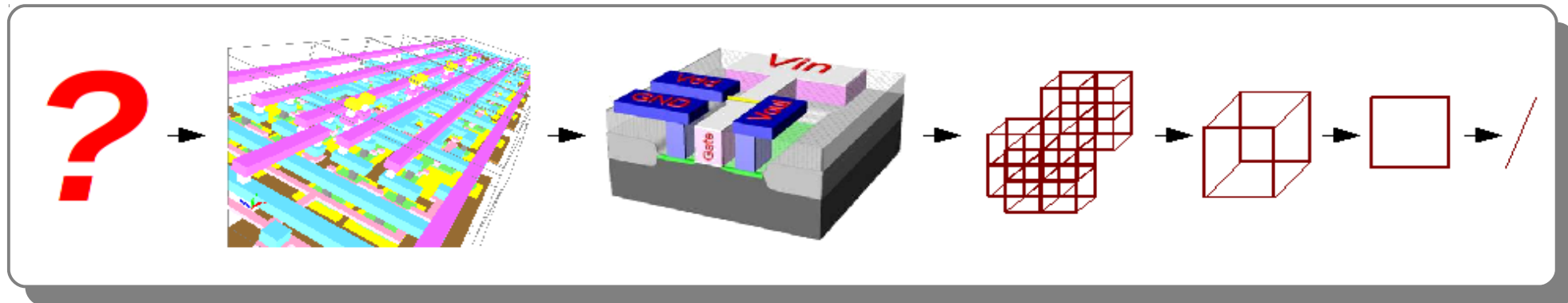
```
root [0] TF3 f1("HelloWorld", "sin(cos(x)+sin(y)+cos(z))", -10,10,-10,10,-10,10)
root [1] f1->Draw()
<TCanvas::MakeDefCanvas>: created default TCanvas with name c1
root [2] HelloWorld->SetTitle("Something else !..")
```



Why is it good ?

Because of the supposed fact that:

- ♦ There are advantages of working within an **object-oriented** framework:
 - Specific to object-oriented languages, the data are embedded within the class instances, allowing easier **abstraction**, and leading to more development and **reuse** of library components
 - Generation of classes out of other classes allows **further** development and/or modification of already existing classes
 - **Hierarchy of classes** resembles conceptual hierarchy of real-world objects allowing easy abstraction/development



- Developed code is far from being **complex**, the data are localized within class instances and are not distributed over many global variables as it is the case for procedural approach (e.g. C or FORTRAN)
- Removing and/or adding new classes into the developed code does not change the **architecture** therefore an architecture can be maintained during development

ROOT - A brief introduction

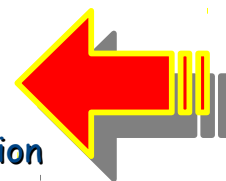
Survival with ROOT != Survival at ROOT != Survival despite ROOT



The goals we have...



- ◆ Introduction to ROOT
 - ◆ What is it ?
 - ◆ Why is it good ?
- ◆ Using ROOT
 - ◆ Command line
 - ◆ Scripting/Interpretation
 - ➔ Example script comparing two current-mode D/A converter architectures
 - ◆ Compilation
 - ➔ Compiling a script into a "*.so" shared object library
 - ➔ Compiling standalone
 - ➔ Application development
 - ➔ Example standalone application
- ◆ GUI of ROOT
 - ◆ Human interaction
 - ◆ Creating a GUI
- ◆ Survival with/at/despite ROOT
 - ◆ User's guide
 - ◆ Referring to:
 - ➔ \$ROOTSYS/tutorials
 - ➔ \$ROOTSYS/test
 - ◆ HTML source code documentation
- ◆ External library usage from within ROOT
 - ◆ DQM of ALICE experiment @ CERN
 - ➔ Simplified DAQ operation
 - ◆ Understanding the detector data
 - ➔ Accessing and decoding data



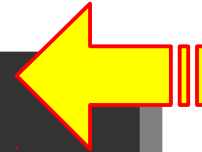
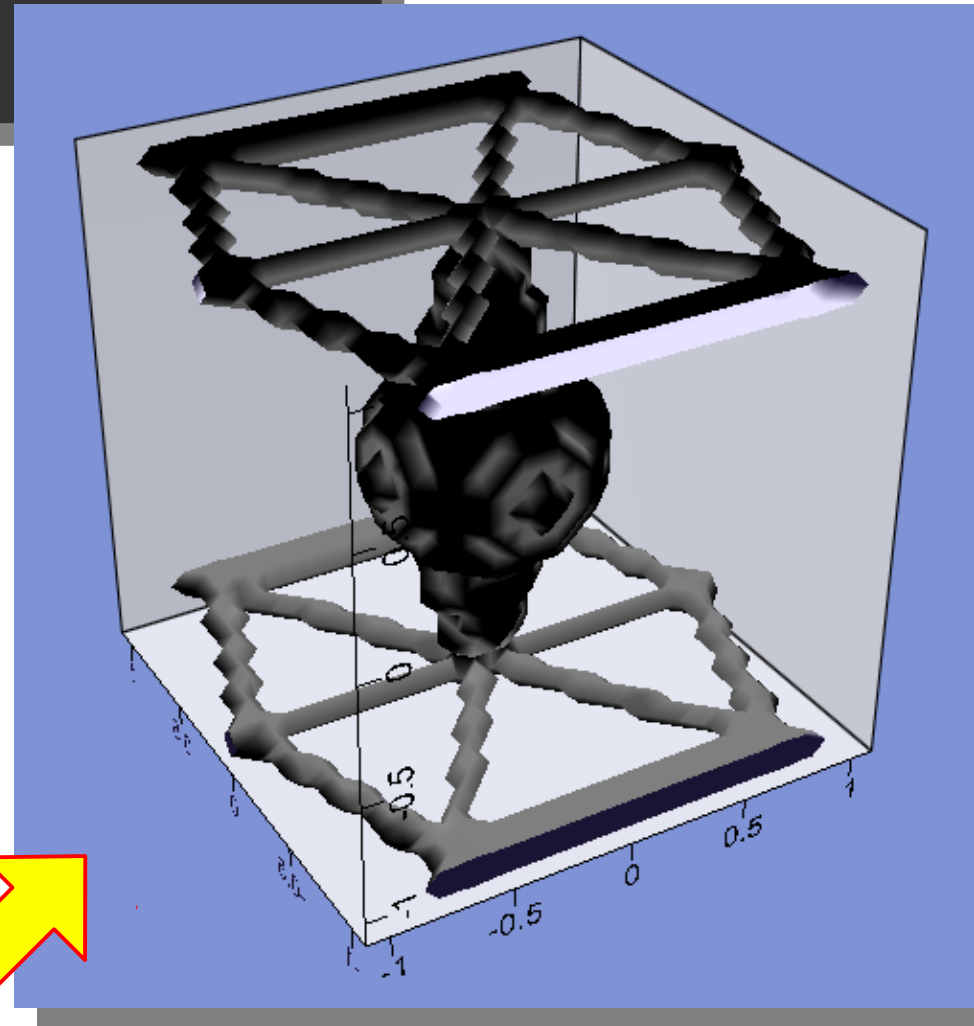
Installation & tie fighter math

Intended usage ?

```
> tar xvfz root_v5.22.00.source.tar.gz
> export ROOTSYS=$HOME/root
> export PATH=$PATH:$ROOTSYS/bin
> export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$ROOTSYS/lib
> cd root
> ./configure
> make
> sudo make install
> _
```

- ◆ Set environment, configure, compile and ROOT is ready to be used.

```
# root -l
> TF3 *tieFighter = new
TF3("tieFighter", "(x^2+y^2+z^2<0.2)+
((y^2+z^2<0.08)*(x<0.4)*(x>0))+
(x^2+4*y^2<(1-TMath::Abs(z))*0.12)+
((TMath::Abs(z)<0.95)*(TMath::Abs(z)
>0.9)*(TMath::Abs(x)
+TMath::Abs(y)*0.3<1))+
((TMath::Abs(z)<1)*(TMath::Abs(z)>0.
89))*(TMath::Abs(x)<0.7)*(TMath::Ab
s(y)>0.9)+(TMath::Abs(y)<0.035)+
(x>y*0.7-0.05)*(x<y*0.7+0.05)+(-
x>y*0.7-0.05)*(-x<y*0.7+0.05)+
((TMath::Abs(x)
+TMath::Abs(y)*0.3<1.05)*(TMath::Abs
(x)+TMath::Abs(y)*0.3>0.95))", -
1.1,1.1,-1.1,1.1,-1.1,1.1);
> tieFighter->Draw()
```



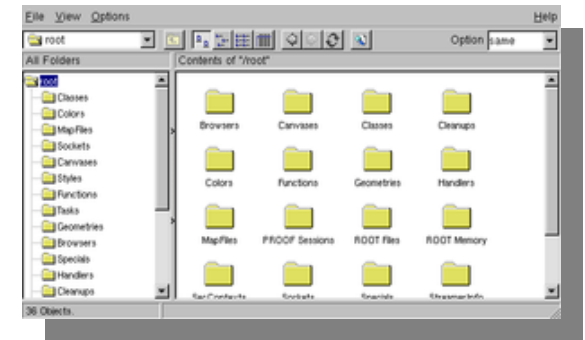
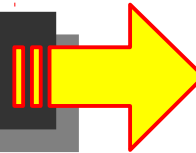
Usage - ROOT Command Line

Invoking commands

- ◆ Invoking C/C++ statements just like invoking shell commands

→ Almost all C/C++ phrases and all classes of ROOT can be used without the need for inclusion such as “`#include<stdio.h>`”. You even do not have to start a function properly like: “`int main()`”. Example: we create an instance of the ROOT class named TBrowser and call its instance as “`myBelovedBrowser`” below:

```
root [0] TBrowser myBelovedBrowser
root [1] _
```



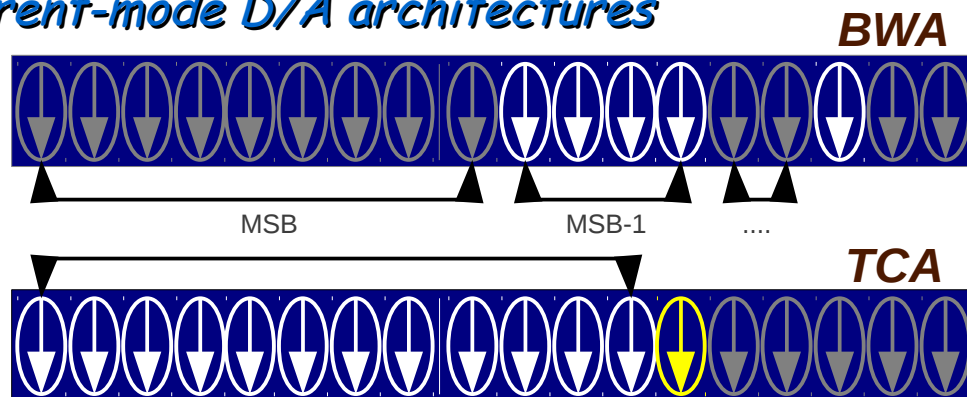
→ Or we write a loop on the fly:

```
root [0] for (int i=0 ; i<10 ; i++) {
end with '}', '@':abort > printf("Square root of %d is %d \n", i*i, i);
end with '}', '@':abort > }
Square root of 0 is 0
Square root of 1 is 1
Square root of 4 is 2
Square root of 9 is 3
Square root of 16 is 4
Square root of 25 is 5
Square root of 36 is 6
Square root of 49 is 7
Square root of 64 is 8
Square root of 81 is 9
root [1] _
```

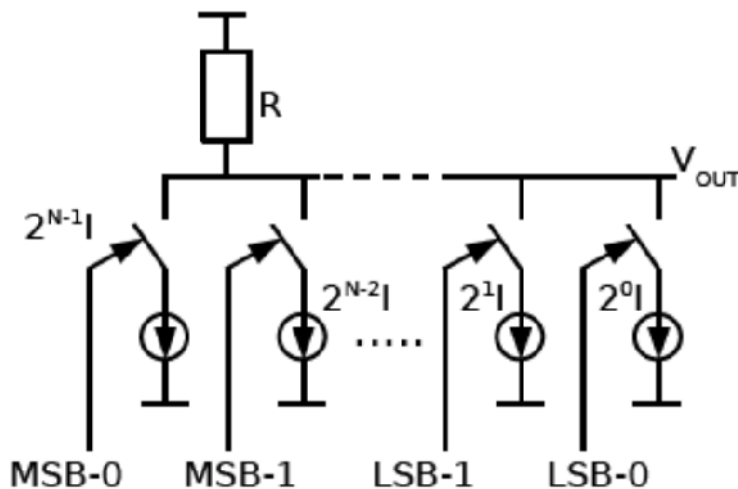
Usage - Scripting

An example script to compare two current-mode D/A architectures

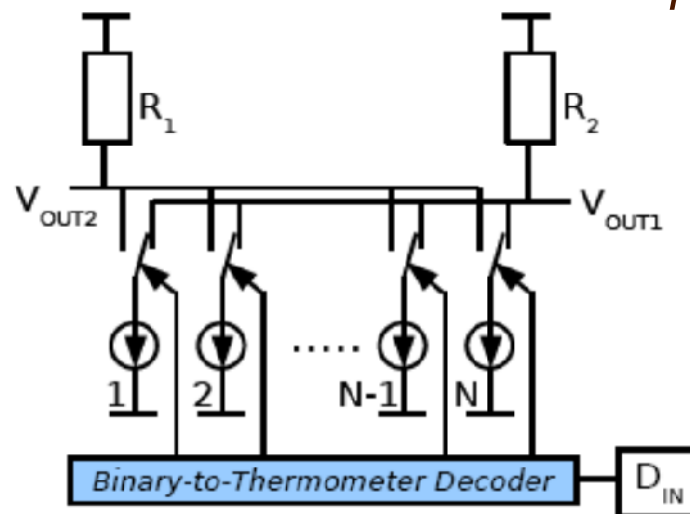
- ◆ A **10-Bit** current-mode D/A converter
- ◆ Two possible **architectures**
- ◆ Have to **choose one**
- ◆ Need for **qualitative comparison**
- ◆ Monte Carlo (MC) is a **must**



- ➔ Generate random numbers out of a Gaussian with a x_c of **1** and a σ of **0.02**
- ➔ Let these numbers to be the unit current sources forming the two D/As
 - ➔ **TCA Case:** each step is represented by an addition of a single current source defining the output voltage
 - ➔ **BWA Case:** let the sum of the first $2^{(N-1)}$ sources form MSB, sum of the next $2^{(N-2)}$ sources to form the next bit after MSB, and so on.
- ➔ Calculate INL and DNL for both of the architectures in RMS and compare.



Binary weighted (BWA)



Thermometer coded (TCA)

Usage - Scripting

An example script to compare two current-mode D/A architectures

```
File Edit View Search Tools Documents Help
New Open Save Print... Undo Redo Cut Copy Paste Find Replace
mimariKarsilastirci.C x mimariKarsilastirci2.C x
27 {
28   gROOT->Reset();
29
30   // edit these parameters for other simulations -----
31   int noofbits=10;
32   int noofiteration4rms=100, nob=100; // nob : no of bins in distro
33   double sigma=0.92, centroid=1.0;
34   const Int t KUPDATE = 50;
35   bool update=kTRUE;
36   //-----
37
38   int nop=pow(2.0, noofbits);
39   int boyut = pow(2, 1*noofbits);
40   double Iu[boyut]; // 2^noofbits current source
41   int lastincremented=0;
42   TCanvas *c1 = new TCanvas("c1", "Binary vs Thermometer", 1200, 850);
43   TPad *pad1 = new TPad("pad1", "Unit Current Source", 0, 0, 1, 1);
44   pad1->Draw();
45   pad1->Divide(5, 4, 0.001, 0.001);
46   pad1->cd(5)->SetFillColor(47); pad1->cd(5)->SetGrid();
47   pad1->cd(6)->SetFillColor(47); pad1->cd(6)->SetGrid();
48   pad1->cd(7)->SetFillColor(47); pad1->cd(7)->SetGrid();
49   pad1->cd(8)->SetFillColor(47); pad1->cd(8)->SetGrid();
50   pad1->cd(9)->SetFillColor(47); pad1->cd(9)->SetGrid();
51   pad1->cd(10)->SetFillColor(47); pad1->cd(10)->SetGrid();
52   pad1->cd(11)->SetFillColor(21); pad1->cd(11)->SetGrid();
53   pad1->cd(12)->SetFillColor(21); pad1->cd(12)->SetGrid();
54   pad1->cd(13)->SetFillColor(21); pad1->cd(13)->SetGrid();
55   pad1->cd(14)->SetFillColor(21); pad1->cd(14)->SetGrid();
56   pad1->cd(15)->SetFillColor(21); pad1->cd(15)->SetGrid();
57   pad1->cd(16)->SetFillColor(21); pad1->cd(16)->SetGrid();
58
59   pad1->cd(1);
60   TH1F *h1f = new TH1F("h1f", "Test random numbers", nob, 0.9, 1.1);
61   TH1F *h1f_ins = new TH1F("h1f_ins", "Current Iu[i] Set", nob, 0.9, 1.1);
62   TH1F *h1f2 = new TH1F("h1f2", "Test random numbers COPY", nob, 0.9, 1.1);
63   TGraph *graph = new TGraph(nop);
64   graph->SetMarkerSize(0);
65   TF1 *gaus = new TF1("gaus", "gaus(0)", 0.9, 1.1);
66   gaus->SetParameters(1, centroid, sigma);
67   gaus->Draw();
68   if (update) c1->Update();
69
70   TH1F *rms_dnl = new TH1F("rms_dnl", "DNL in RMS [LSB unit]", nop-2, 0, nop-2);
71   TH1F *rms_inl = new TH1F("rms_inl", "INL in RMS [LSB unit]", nop, 0, nop);
72   TH1F *rms_dac = new TH1F("rms_dac", "D/A Output in RMS [LSB unit]", nop, 0, nop);
73   TH1F *b_rms_dnl = new TH1F("b_rms_dnl", "b DNL in RMS [LSB unit]", nop-2, 0, nop-2);
```

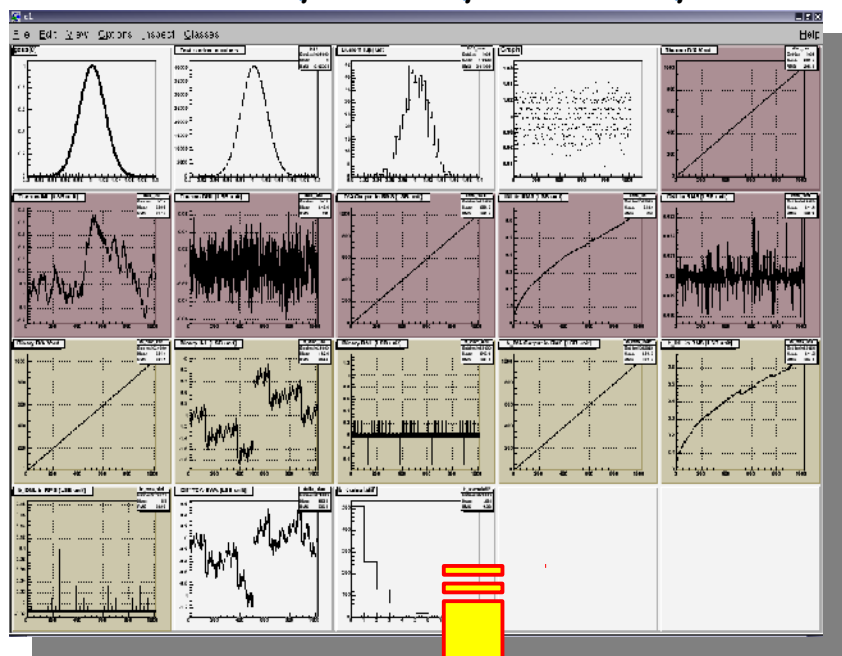
```
File Edit View Search Tools Documents Help
New Open Save Print... Undo Redo Cut Copy Paste Find Replace
mimariKarsilastirci.C x mimariKarsilastirci2.C x
182   b_inl[i]=(1+1.0*b_dac[i]);
183   b_rms_inl->SetBinContent(i, b_rms_inl->GetBinContent(i)+b_inl[i]*b_inl[i]);
184   if (i>0 && i<nop) {
185     b_dnl[i-1]=Iu[0]-b_dac[i]+b_dac[i-1];
186     b_rms_dnl->SetBinContent(i-1, b_rms_dnl->GetBinContent(i-1)+(b_dnl[i-1]-Iu[0]
187   }
188   b_dac_inl->SetBinContent(i, b_inl[i]-Iu[0]);
189   if (i>0) b_dac_dnl->SetBinContent(i-1, b_dnl[i-1]);
190 }
191 pad1->cd(12);
192 b_dac_inl->Draw();
193 pad1->cd(13);
194 b_dac_dnl->Draw();
195 if (update) c1->Update();
196 pad1->cd(17);
197 delta_dac->Draw();
198 pad1->cd(18);
199 b_cumulatif->Draw();
200 pad1->cd(11);
201 b_dac_ler->Draw();
202
203 } // end loop
204
205 for (int i=0 ; i<nop ; i++) { // calculate RMS
206   rms_dac->SetBinContent(i, sqrt(rms_dac->GetBinContent(i)/noofiteration4rms));
207   if (i<nop-1) rms_dnl->SetBinContent(i, sqrt(rms_dnl->GetBinContent(i)/noofiterat
208   rms_inl->SetBinContent(i, sqrt(rms_inl->GetBinContent(i)/noofiteration4rms));
209   b_rms_dac->SetBinContent(i, sqrt(b_rms_dac->GetBinContent(i)/noofiteration4rms));
210   if (i<nop-1) b_rms_dnl->SetBinContent(i, 1.55-3*(-Iu[0]/2+sqrt(b_rms_dnl->GetBin
211   b_rms_inl->SetBinContent(i, sqrt(b_rms_inl->GetBinContent(i)/noofiteration4rms));
212 }
213 pad1->cd(8);
214 rms_dac->Draw();
215 pad1->cd(9);
216 rms_inl->Draw();
217 pad1->cd(10);
218 rms_dnl->Draw();
219 pad1->cd(14);
220 b_rms_dac->Draw();
221 pad1->cd(15);
222 b_rms_inl->Draw();
223 pad1->cd(16);
224 b_rms_dnl->Draw();
225 if (update) c1->Update();
226
227 gBenchmark->Stop("binary_vs_thermometer");
228 }
```

ArchitectureComparer.C

- ➔ Beware that the script does **not** have a **name**
- ➔ Note how the unnamed script starts (**no** header inclusion) and ends (**no** return value).

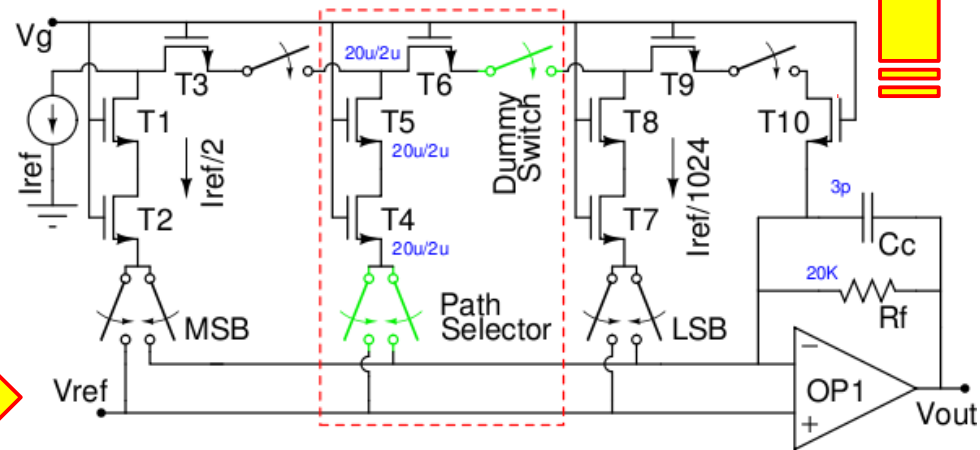
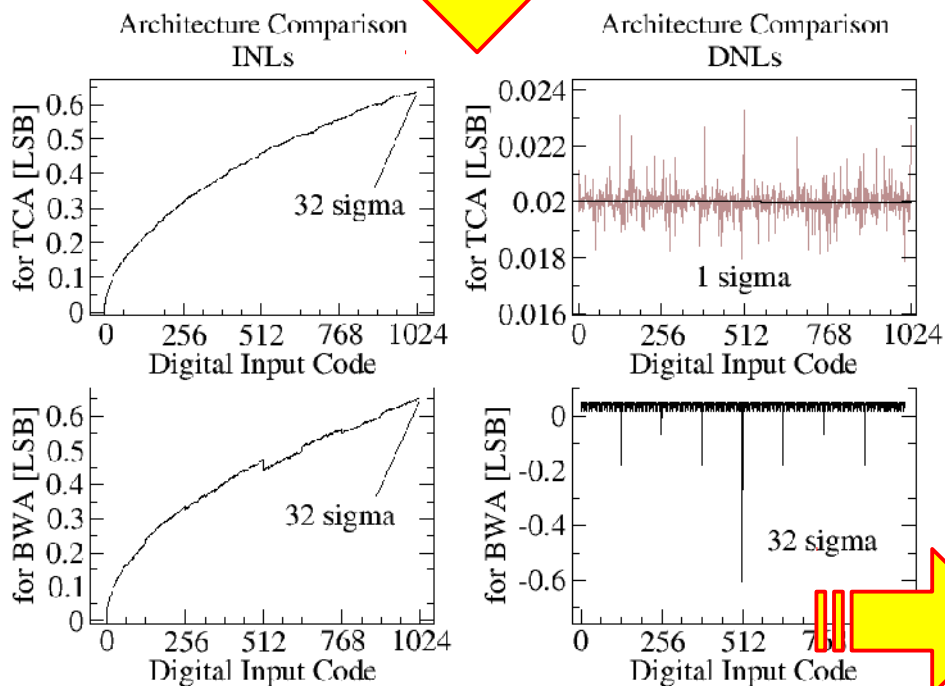
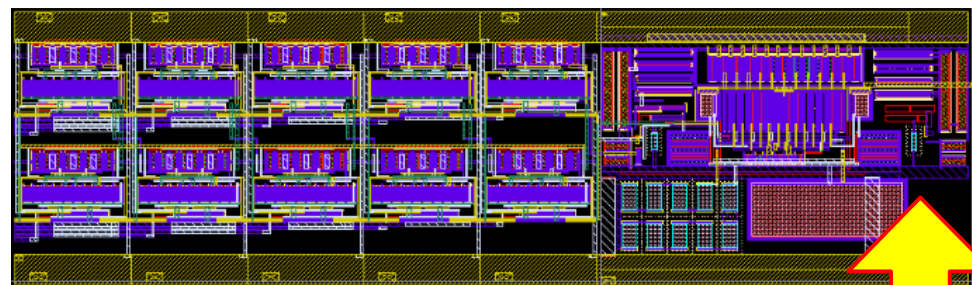
Usage - Scripting

An example script to compare two current-mode D/A architectures



- ◆ INLs are almost identical.
- ◆ DNLs are quite different, TCA behaves much better, however it will occupy a huge space on the chip layout
- ◆ Decided, designed the D/A accordingly, delivered it to the experiment people and published the paper
- ◆ Mission complete !!

Used within the CMAD front-end ASIC designed for RICH-I detector of the COMPASS experiment at CERN



Usage - Compilation

Compiling a script into a `*.so` library

- ◆ ArchitectureComparer.C is **interpreted** by CINT **“slowly”**
- ◆ When **compiled**, the script will be **executed**, instead of being interpreted, this is **“fast”**
- ◆ With **almost** no modification, one can compile the script into a `*.so` library
- ◆ **“..almost no modification..”** actually means:
 - ➔ Header files of the classes used must be included (e.g. if `TCanvas` is used then I need the following statement to be inside my code: `#include<TCanvas.h>`)
 - ➔ The function must be given a name, preferably same as the file name. (e.g. for `“name.C”` as the file name, I would write `int name() {}`)
- ◆ Now the script is ready for being compiled into an **“*.so”** library:

```
> root ArchitectureComparer.C++
root [0] Processing ArchitectureComparer.C++.
Info in <TUnixSystem::ACLiC>: creating shared library
/home/oc/Documents/HEP_0kulu/workDir/root/./ArchitectureComparer_C.so
```

- ◆ I can use my library later at any time I wish so:

```
oc@olmak2:~/Documents/HEP_0kulu/workDir/root$ root -l
root [0] .L ArchitectureComparer_C.so
root [1] ArchitectureComparer()
```

Usage - Compilation

Compiling a script into a *.so library

```
File Edit View Search Tools Documents Help
New Open Save Print... Undo Redo Cut Copy Paste Find Replace
mimariKarsilastirici.C mimariKarsilastirici2.C
27 #include <TFile.h>
28 #include <TNTuple.h>
29 #include <TH2.h>
30 #include <TProfile.h>
31 #include <TCanvas.h>
32 #include <TFrame.h>
33 #include <TROOT.h>
34 #include <TSystem.h>
35 #include <TRandom.h>
36 #include <TBenchmark.h>
37 #include <T Cint.h>
38 #include <TGraph.h>
39 #include <TF1.h>
40
41 int mimariKarsilastirici2() {
42     gROOT->Reset();
43
44     // edit these parameters for other simulations -----
45     int noofbits=10;
46     int noofiteration4rms=100, nob=100; // nob : no of bins in distro
47     double sigma=0.2, centroid=1.0;
48     const Int_t kUPDATE = 50;
49     bool update=kTRUE;
50     //-----
51
52     int nop=pow(2.0, noofbits);
53     int boyut = pow(2, *noofbits);
54     double Iu[boyut]; // 2^noofbits current source
55     int lastincremented=0;
56     TCanvas *c1 = new TCanvas("c1","Binary vs Thermometer",1200,850);
57     TPad *pad1 = new TPad("pad1","Unit Current Source",0.0,0.0,1.0,1.0);
58     pad1->Draw();
59     pad1->Divide(5, 4, 0.001, 0.001);
60     pad1->cd(5)->SetFillColor(47); pad1->cd(5)->SetGrid();
61     pad1->cd(6)->SetFillColor(47); pad1->cd(6)->SetGrid();
62     pad1->cd(7)->SetFillColor(47); pad1->cd(7)->SetGrid();
63     pad1->cd(8)->SetFillColor(47); pad1->cd(8)->SetGrid();
64     pad1->cd(9)->SetFillColor(47); pad1->cd(9)->SetGrid();
65     pad1->cd(10)->SetFillColor(47); pad1->cd(10)->SetGrid();
66     pad1->cd(11)->SetFillColor(21); pad1->cd(11)->SetGrid();
67     pad1->cd(12)->SetFillColor(21); pad1->cd(12)->SetGrid();
68     pad1->cd(13)->SetFillColor(21); pad1->cd(13)->SetGrid();
69     pad1->cd(14)->SetFillColor(21); pad1->cd(14)->SetGrid();
70     pad1->cd(15)->SetFillColor(21); pad1->cd(15)->SetGrid();
71     pad1->cd(16)->SetFillColor(21); pad1->cd(16)->SetGrid();
72
73     pad1->cd(1);
```

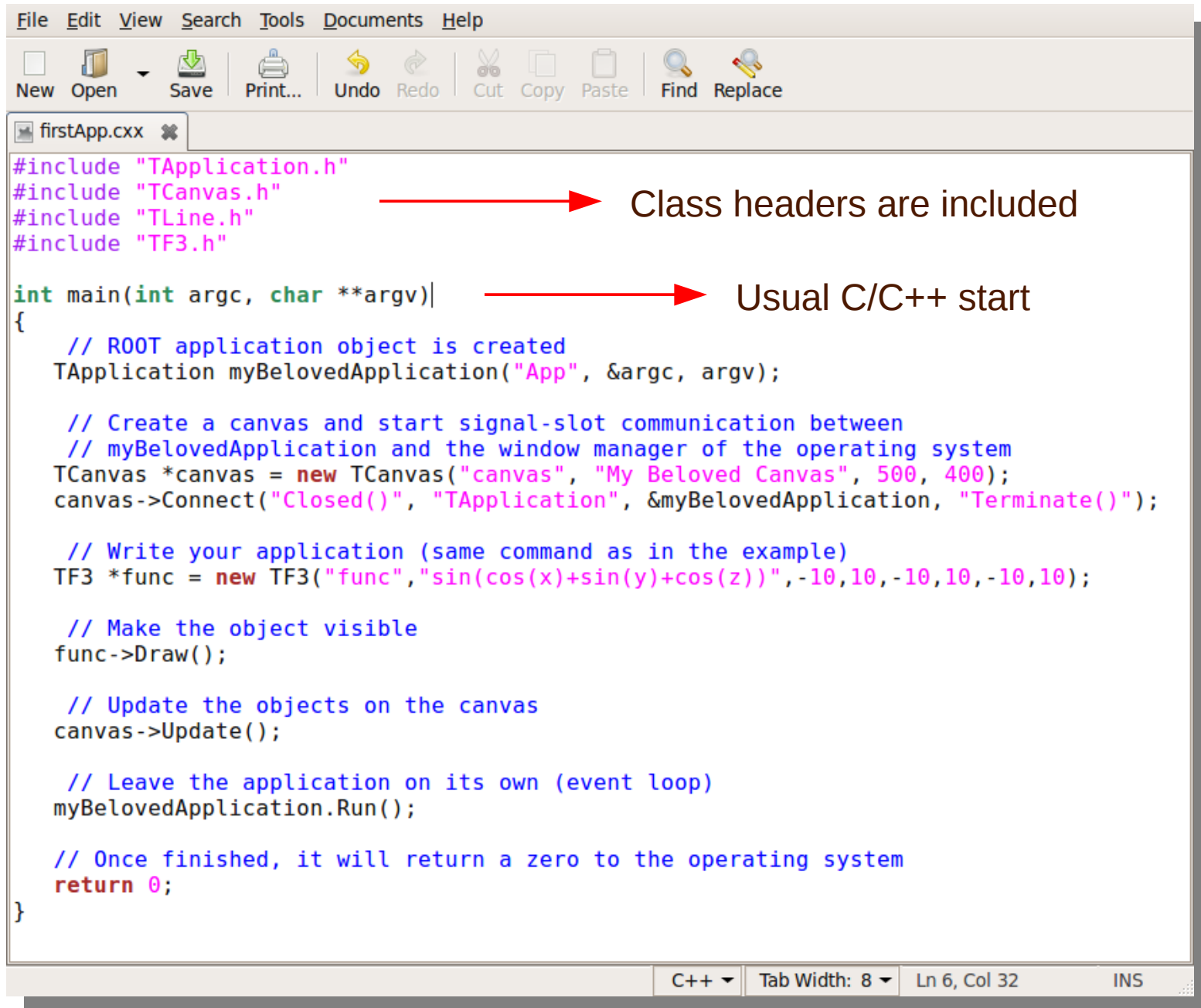
```
File Edit View Search Tools Documents Help
New Open Save Print... Undo Redo Cut Copy Paste Find Replace
mimariKarsilastirici.C mimariKarsilastirici2.C
198     if (i>0 && i<nop) {
199         b_dnl[i-1]=Iu[0]-b_dac[i]+b_dac[i-1];
200         b_rms_dnl->SetBinContent(i-1, b_rms_dnl->GetBinContent(i-1)+(b_dnl[i-1]-Iu[0])*(i-1));
201     }
202     b_dac_inl->SetBinContent(i, b_inl[i]-Iu[0]);
203     if (i>0) b_dac_dnl->SetBinContent(i-1, b_dnl[i-1]);
204 }
205 pad1->cd(12);
206 b_dac_inl->Draw();
207 pad1->cd(13);
208 b_dac_dnl->Draw();
209 if (update) c1->Update();
210 pad1->cd(17);
211 delta_dac->Draw();
212 pad1->cd(18);
213 b_cumulatif->Draw();
214 pad1->cd(11);
215 b_dac_ter->Draw();
216
217 } // end loop
218
219 for (int i=0 ; i<nop ; i++) { // calculate RMS
220     rms_dac->SetBinContent(i, sqrt(rms_dac->GetBinContent(i)/noofiteration4rms));
221     if (i<nop-1) rms_dnl->SetBinContent(i, sqrt(rms_dnl->GetBinContent(i)/noofiteration4rms));
222     rms_inl->SetBinContent(i, sqrt(rms_inl->GetBinContent(i)/noofiteration4rms));
223     b_rms_dac->SetBinContent(i, sqrt(b_rms_dac->GetBinContent(i)/noofiteration4rms));
224     if (i<nop-1) b_rms_dnl->SetBinContent(i, 1.55*3*(-Iu[0]/2+sqrt(b_rms_dnl->GetBinContent(i)/noofiteration4rms));
225     b_rms_inl->SetBinContent(i, sqrt(b_rms_inl->GetBinContent(i)/noofiteration4rms));
226 }
227 pad1->cd(8);
228 rms_dac->Draw();
229 pad1->cd(9);
230 rms_inl->Draw();
231 pad1->cd(10);
232 rms_dnl->Draw();
233 pad1->cd(14);
234 b_rms_dac->Draw();
235 pad1->cd(15);
236 b_rms_inl->Draw();
237 pad1->cd(16);
238 b_rms_dnl->Draw();
239 if (update) c1->Update();
240
241 gBenchmark->Stop("binary_vs_thermometer");
242 return 0;
243 }
244
```

ArchitectureComparer2.C

- ➔ Beware that the script **does** have a name
- ➔ Note how the **named** script starts (**header inclusion**) and ends (**with a return value**).

Usage - Compiling Standalone

Standalone application version of the example



```
File Edit View Search Tools Documents Help
New Open Save Print... Undo Redo Cut Copy Paste Find Replace
firstApp.cxx x
#include "TApplication.h"
#include "TCanvas.h"
#include "TLine.h"
#include "TF3.h"

int main(int argc, char **argv)
{
    // ROOT application object is created
    TApplication myBelovedApplication("App", &argc, argv);

    // Create a canvas and start signal-slot communication between
    // myBelovedApplication and the window manager of the operating system
    TCanvas *canvas = new TCanvas("canvas", "My Beloved Canvas", 500, 400);
    canvas->Connect("Closed()", "TApplication", &myBelovedApplication, "Terminate()");

    // Write your application (same command as in the example)
    TF3 *func = new TF3("func", "sin(cos(x)+sin(y)+cos(z))", -10, 10, -10, 10, -10, 10);

    // Make the object visible
    func->Draw();

    // Update the objects on the canvas
    canvas->Update();

    // Leave the application on its own (event loop)
    myBelovedApplication.Run();

    // Once finished, it will return a zero to the operating system
    return 0;
}
```

Class headers are included

Usual C/C++ start

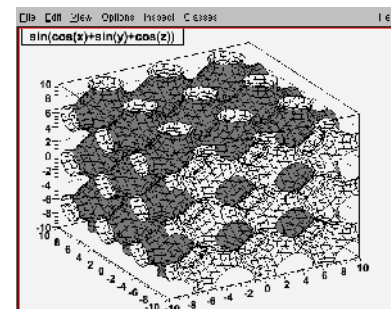
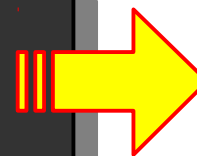
C++ Tab Width: 8 Ln 6, Col 32 INS

Usage - Compiling Standalone

Standalone application version of the example

- An elegant method to write applications is via **Makefile**, but we will not do it here
- To compile the code in the previous page:
 - ➔ **g++** -L/usr/lib/root -lCore -lCint -lRIO -lNet -lHist -lGraf -lGraf3d -lGpad -lTree -lRint -lPostscript -lMatrix -lPhysics -lz -pthread -lm -ldl -rdynamic -pthread -m64 -I/usr/include/root -L/usr/lib/root -lCore -lCint -lRIO -lNet -lHist -lGraf -lGraf3d -lGpad -lTree -lRint -lPostscript -lMatrix -lPhysics -lz -lGui -pthread -lm -ldl -rdynamic **firstApp.cxx** -o **firstApp**
- Remembering all the above things is hard, therefore we will use:
 - ➔ **root-config**: a command-line tool to make lives of ROOT users easy (libraries usually have tools like this one)
 - ➔ It returns appropriate lines needed for compilation
 - ➔ Usually used in-between “ ` ”, aka escape symbol
- ➔ **g++** `root-config --glibs --cflags` **firstApp.cxx** -o **firstApp**

```
oc@olmak2:~/ISOTDAQ/root$ ./firstApp &  
[1] 16811  
oc@olmak2:~/ISOTDAQ/root$ _
```



ROOT - A brief introduction

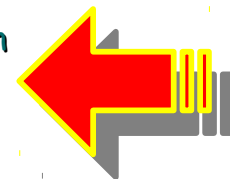
Survival with ROOT != Survival at ROOT != Survival despite ROOT



The goals we have...



- ◆ Introduction to ROOT
 - ◆ What is it ?
 - ◆ Why is it good ?
- ◆ Using ROOT
 - ◆ Command line
 - ◆ Scripting/Interpretation
 - ➔ Example script comparing two current-mode D/A converter architectures
 - ◆ Compilation
 - ➔ Compiling a script into a "*.so" shared object library
 - ➔ Compiling standalone
 - ➔ Application development
 - ➔ Example standalone application
- ◆ GUI of ROOT
 - ◆ Human interaction
 - ◆ Creating a GUI
- ◆ Survival with/at/despite ROOT
 - ◆ User's guide
 - ◆ Referring to:
 - ➔ \$ROOTSYS/tutorials
 - ➔ \$ROOTSYS/test
 - ◆ HTML source code documentation
- ◆ External library usage from within ROOT
 - ◆ DQM of ALICE experiment @ CERN
 - ➔ Simplified DAQ operation
 - ◆ Understanding the detector data
 - ➔ Accessing and decoding data



*

Usage

GUI - Human interaction



- ◆ All created windows and everything but everything on these windows (canvases, pads, histograms, titles, pave texts, sub-windows, axes, etc.) are either ROOT class instances or are instances of derived classes.
- ◆ Right click would bring a menu of some of the member functions of the instances created on the heap

The screenshot shows the ROOT GUI interface. At the top, there are several plots. The central plot is a histogram with a fit curve. A yellow box highlights the histogram and its fit panel. A context menu is open over the histogram, listing various member functions. A red arrow points from the 'FitPanel' option in the menu to the 'Set Parameters...' button in the fit settings dialog. Another red arrow points from the 'Pre-defined fit function' text to the 'gaus' dropdown in the dialog. A blue arrow points from the 'Many options available' text to the 'Advanced...' button in the dialog. At the bottom, there is a 'Set Range' dialog box with a table of parameters.

Name	Fix	Bound	Value	Min	Set Range	Max	Step	Errors
Constant	<input type="checkbox"/>	<input type="checkbox"/>	1	-3		3	0.3	-
Mean	<input type="checkbox"/>	<input type="checkbox"/>	1	-3		3	0.3	-
Sigma	<input type="checkbox"/>	<input type="checkbox"/>	0.02	-0.06		0.06	0.006	-

Buttons: Immediate preview, Reset, Apply, OK, Cancel

Fit Settings Dialog:

- Function: gaus
- Predefined: Operation (Nop, Add, Conv)
- Selected: gaus
- Method: Chi-square
- Robust: 1.00
- Fit Options: Integral, Best errors, All weights = 1, Empty bins, weights=1, Use range, Improve fit results, Add to list
- Draw Options: SAME, No drawing, Do not store/draw

Buttons: Set Parameters..., User-Defined..., Advanced..., Fit, Reset, Close

- Pre-defined fit function
- Many options available

Usage

GUI - Human interaction

The top two panels show the configuration for the histogram's statistics panel (stats::TPaveStats). The bottom two panels show the configuration for the fit function (fitFunc::TF1).

The screenshot shows a histogram with a Gaussian fit. A context menu is open over the fit curve, and a fit settings dialog is also visible. Green arrows point from the fit settings dialog to the histogram, and red arrows point from the histogram to the fit settings dialog.

Current lu[i] Set

h1f_ins

Entries	1024
Mean	0.9999
RMS	0.02048

TF1::fitFunc

- DrawDerivative
- DrawIntegral
- SetMaximum
- SetMinimum
- SetNpx
- SetRange
- SetParameters
- SetParNames
- SetName
- SetTitle
- Delete
- DrawClass
- DrawClone
- Dump
- Inspect
- SaveAs
- SetDrawOption
- SetLineAttributes**
- SetFillAttributes
- SetMarkerAttributes

Current selection: h1f_ins:TH1F

General | Minimization

Function

Predefined: **gaus** Operation: Nop Add Conv

Selected: gaus Set Parameters...

Fit Settings

Method: **Chi-square** User-Defined...

Linear fit Robust: 1.00 No Chi-square

Fit Options

- Integral
- Best errors
- All weights = 1
- Empty bins, weights=1
- Use range
- Improve fit results
- Add to list

Draw Options

- SAME
- No drawing
- Do not store/draw

Advanced...

X:

Fit **Reset** **Close**

LIB Minuit MIGRAD Itr: 5000 Prn: DEF

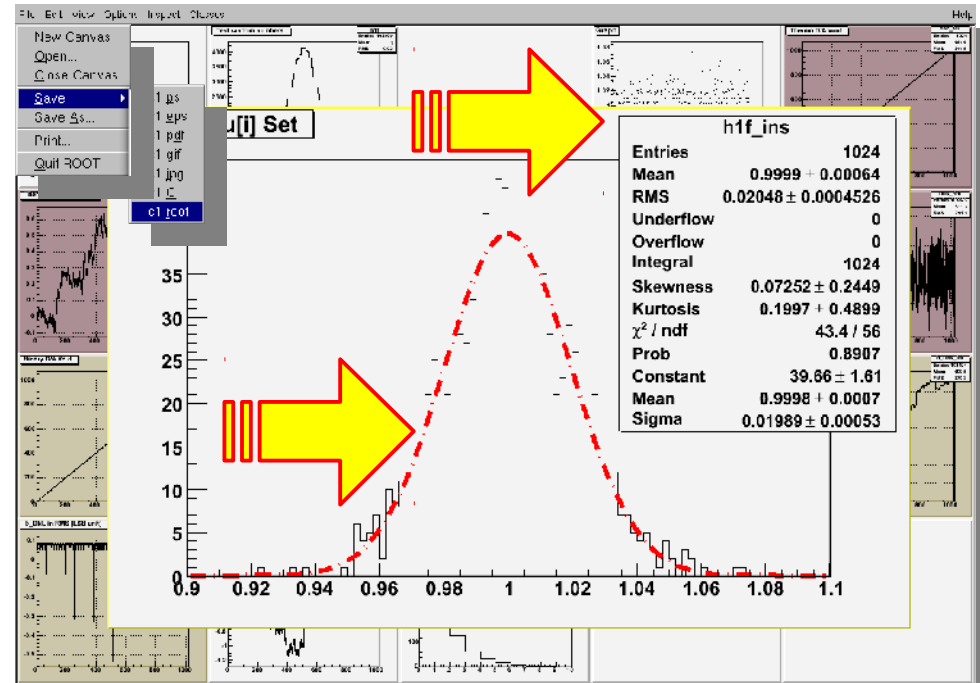
- After fitting ends, adding make-up is also possible
- I want my fit curve to be **thick, dashed and red**
- And I want more statistical information on the histogram

Usage

GUI - Human interaction

- ◆ We have developed the code needed, performed the analysis, made the results more visible by adding make-up, **BUT** the mission is not yet complete !
- ◆ The **work** should be **saved** !
- ◆ There are many available methods:
 - It can be saved as a *.C source code
 - ➔ To get the results again it must be reinterpreted by ROOT (e.g. **“root -l code.C”**)
 - It can be saved as a *.root file
 - ➔ Content can be directly browsed by a TBrowser object (e.g. **“TBrowser a”**)
 - It can be saved as a picture:
 - ➔ ps, eps, gif v.b.

- **Thicker, dashed and red**
- **More statistical information**

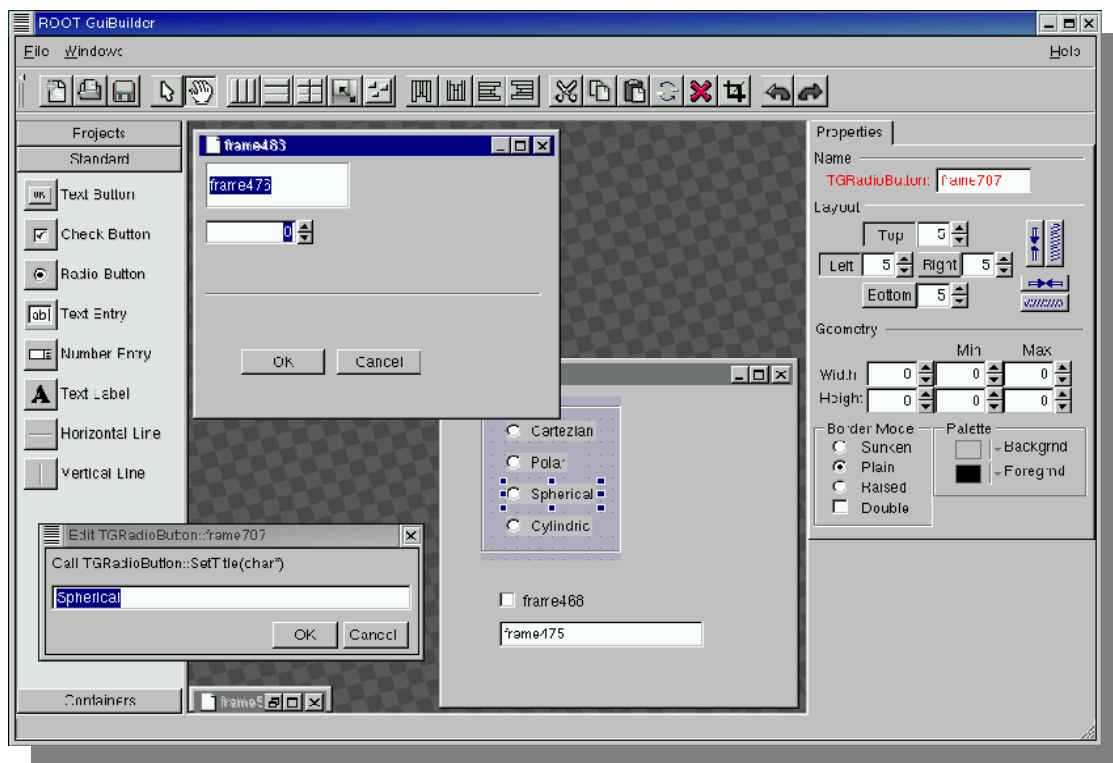


Example

Creating a GUI

- ◆ Users can design GUIs for their applications developed with ROOT libraries
- ◆ There are two main ways of doing so, usually iteratively used:
 - ◆ Writing source code
 - ◆ Using an instance of “**TRootGuiBuilder**” class

```
oc@olmak2:~/ISOTDAQ/workDir/root$ root -l
root [0] TRootGuiBuilder a
root [1] _
```



- Graphically prepared application is saved as *.C script and development of actual code is continued over this “template”.
- **Educative: design** the GUI, **save** it, **read** it !!
- **Saves time**: you do not have to memorize ROOT GUI classes
- **Save frequently** against frequent **crashes** not to lose work !!

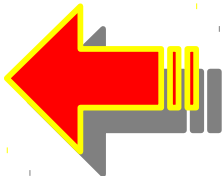
ROOT - A brief introduction

Survival *with* ROOT != Survival *at* ROOT != Survival *despite* ROOT



The goals we have...



- ◆ Introduction to ROOT
 - ◆ What is it ?
 - ◆ Why is it good ?
- ◆ Using ROOT
 - ◆ Command line
 - ◆ Scripting/Interpretation
 - ➔ Example script comparing two current-mode D/A converter architectures
 - ◆ Compilation
 - ➔ Compiling a script into a "*.so" shared object library
 - ➔ Compiling standalone
 - ➔ Application development
 - ➔ Example standalone application
- ◆ GUI of ROOT
 - ◆ Human interaction
 - ◆ Creating a GUI
- ◆ Survival with/at/despite ROOT 
 - ◆ User's guide
 - ◆ Referring to:
 - ➔ \$ROOTSYS/tutorials
 - ➔ \$ROOTSYS/test
 - ◆ HTML source code documentation
- ◆ External library usage from within ROOT
 - ◆ DQM of ALICE experiment @ CERN
 - ➔ Simplified DAQ operation
 - ◆ Understanding the detector data
 - ➔ Accessing and decoding data

*

Survival with/at/despite ROOT

User's Guide

- ◆ Meeting with a new library stands upon 4 legs: (*subjective*)
 - i. User's manual
 - ii. Source code documentation
 - iii. Set of examples showing library usage (tutorial, test, example, etc.)
 - iv. You, reading these resources and trying them out
- ◆ **We are lucky: ROOT has all these things !!** Meeting with ROOT **is trivial !!**
- ◆ **User's guide**, tells us on which ideals the library has been developed, its architecture and “**hello world**” examples with a lot of relevant explanations.
- ◆ It **does not change** fast, it is rather **static**.
- ◆ It must be studied/digested **once** at the beginning
- ◆ For many of us, only having a **skin-deep look** renders **enough** for every new **major release** (i.e. change of major version number)
- ◆ **All** the things mentioned within this lecture **and more**, written **fluently** with a lot of useful tricks can be found in ROOT User's Guide.

Survival with/at/despite ROOT

User's Guide

- ◆ Meeting with a new library stands upon 4 legs: (*subjective*)
 - i. User's manual
 - ii. Source code documentation
 - iii. Set of examples showing library usage (tutorial, test, example, etc.)
 - iv. You, reading these resources and trying them out
- ◆ **We are lucky: ROOT has all these things !!** Meeting with ROOT **is trivial !!**
- ◆ **User's guide**, tells us on which ideals the library has been developed, its architecture and “**hello world**” examples with a lot of relevant explanations.
- ◆ It **does not change** fast, it is rather **static**.
- ◆ It must be studied/digested **once** at the beginning
- ◆ For many of us, only having a **skin-deep look** renders **enough** for every new **major release** (i.e. change of major version number)
- ◆ **All** the things mentioned within this lecture **and more**, written **fluently** with a lot of useful tricks can be found in ROOT User's Guide.



Why do we
lose time
here then ?

*

Survival with/at/despite ROOT

User's Guide

ROOT User's
Guide is more
than **~500 pages!**
Moreover ...



*

... it just teaches a
language but **not**
what to say in that
language !

Survival with/at/despite ROOT

Effective function of \$ROOTSYS/tutorials directory

- ◆ The line separating **death** and **life** while **scripting**
- ◆ Not to be memorized; to be **referred continuously** while ROOTing; the lowest level of scripting examples, a teacher
- ◆ Learning to use a new library, especially in languages where almost everything is hard without library support, is **equivalent** to learning a new programming language
- ◆ A language is best learned by **practicing**
- ◆ This directory is where you practice ROOT **scripting** language
- ◆ To get acquainted with ROOT, open (with your favorite text editor) **all but all** the scripts, have a look at each code, run with **root -l xxx.C** and understand what they do; not deeply, just to have an idea
- ◆ This is vital because:
 - Understanding **what percentage** of the tutorial codes can actually run without problems is important. Is the library you will use **perfect** ?
 - The scripts within this directory are the **starting points** for your future developments.
 - You do not have to remember which one does what.
 - However, you **must be able to say** “**there was something in the tutorials directory that does a similar thing...**” when you need assistance

Survival with/at/despite ROOT

Effective function of \$ROOTSYS/test directory

- ◆ The line separating **death** and **life** while **compiling standalone applications**
- ◆ Not to be memorized; to be **referred continuously** while ROOTing; the lowest level of application examples, a teacher
- ◆ Learning to use a new library, especially in languages where almost everything is hard without library support, is **equivalent** to learning a new programming language
- ◆ A language is best learned by **practicing**
- ◆ This directory is where you practice **application development** with ROOT
- ◆ To get acquainted with ROOT, open (with your favorite text editor) **all but all** the source codes, have a look at each code, compile with **make xxx** and understand what they do; not deeply, just to have an idea
- ◆ This is vital because:
 - ➔ Understanding **what percentage** of the test codes can actually run without problems is important. Is the library you will use **perfect** ?
 - ➔ The source codes within this directory are the **starting points** for your future developments.
 - ➔ You do not have to remember which one does what.
 - ➔ However, you **must be able to say** “**there was something in the test directory that does a similar thing...**” when you need assistance

Survival with/at/despite ROOT

Usage of HTML documentation

- ◆ **The line** in-between death and life
- ◆ It is not something to learn, it is to refer in a cyclic manner while ROOTing (documentation which is the closest to what you actually use)
- ◆ **On-line/live:** it is generated out of the version you use at the time, therefore:
 - ➔ Un-like user's guides, it is **not static**; it is not full of "old" knowledge; it is **valid and up-to-date**, truly useful
 - ➔ Equivalent to reading the source code of the ROOT library, **least error-prone** technique to learn what the command you use in your code actually does

Your favorite text editor

```
File Edit View Search Tools Documents Help
New Open Save Print... Undo Redo Cut Copy Paste Find Replace

mimariKarsilastirici3.C x
153 // for binary weighted architecture
154
155 int sayac=0;
156 double MSB[noofbits];
157 for (int i=0 ; i<noofbits ; i++) MSB[i]=0.0;
158
159 for (int i=0 ; i<noofbits ; i++) { // form the binary weighted
160     for (int j=0 ; j<pow(2.0, noofbits-i-1) ; j++) {
161         MSB[i]+=1u[sayac++];
162         b_cumulatif->SetBinContent(i+1, MSB[i]);
163     }
164 }
165
166 for (int i=0 ; i<noofbits ; i++) { // calculate full scale D/A output
```

Search for `b_cumulatif` via **Ctrl-F** to see what objects instance it is: an instance of **TH1F** class

Your favorite html browser

```
File Edit View History Bookmarks Tools Help
file:///home/sagit
TH2F x Lolcats '... Lolcats '... CDR Cern Tur... bir >
void TArrayF::Set (Int_t n, const Float_t* array)
virtual void TArrayF::SetAt (Double_t v, Int_t i)
virtual void TH1::SetAxisColor (Color_t color = 1, Option_t* axis = "X")
virtual void TH1::SetAxisRange (Double_t xmin, Double_t xmax, Option_t* axis = "X")
virtual void TH1::SetBarOffset (Float_t offset = 0.25)
virtual void TH1::SetBarWidth (Float_t width = 0.5)
virtual void TH1::SetBinContent (Int_t bin, Double_t content)
virtual void TH1::SetBinContent (Int_t binx, Int_t biny, Double_t content)
virtual void TH1::SetBinContent (Int_t binx, Int_t biny, Int_t, Double_t content)
virtual void TH1::SetBinError (Int_t bin, Double_t error)
virtual void TH1::SetBinError (Int_t binx, Int_t biny, Double_t error)
virtual void TH1::SetBinError (Int_t binx, Int_t biny, Int_t binz, Double_t error)
virtual void TH1::SetBins (Int_t nx, const Double_t* xBins)
virtual void TH1::SetBins (Int_t nx, Double_t xmin, Double_t xmax)
```

```
File Edit View Search Tools Documents Help
New Open Save Print... Undo Redo Cut Copy Paste Find Replace

mimariKarsilastirici3.C x
68 rms_dnl = new TH1F("rms_dnl", "DNL in RMS [LSB unit]", nop-2, 0,
69 rms_inl = new TH1F("rms_inl", "INL in RMS [LSB unit]", nop, 0, no
70 rms_dac = new TH1F("rms_dac", "D/A Output in RMS [LSB unit]", nop
71 b_rms_dnl = new TH1F("b_rms_dnl", "b_DNL in RMS [LSB unit]", nop-
72 b_rms_inl = new TH1F("b_rms_inl", "b_INL in RMS [LSB unit]", nop,
73 b_rms_dac = new TH1F("b_rms_dac", "b_D/A Output in RMS [LSB unit]
74 b_cumulatif = new TH1F("b_cumulatif", "b_cumulatif", noofbits, 0
75 delta_dac = new TH1F("delta_dac", "Diff TCA-BWA [LSB unit]", nop
76
77 rms_dnl->Reset();
78 rms_inl->Reset();
79 rms_dac->Reset();
80 b_rms_dnl->Reset();
81
```

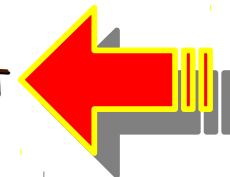
ROOT - A brief introduction

Survival with ROOT != Survival at ROOT != Survival despite ROOT



The goals we have...

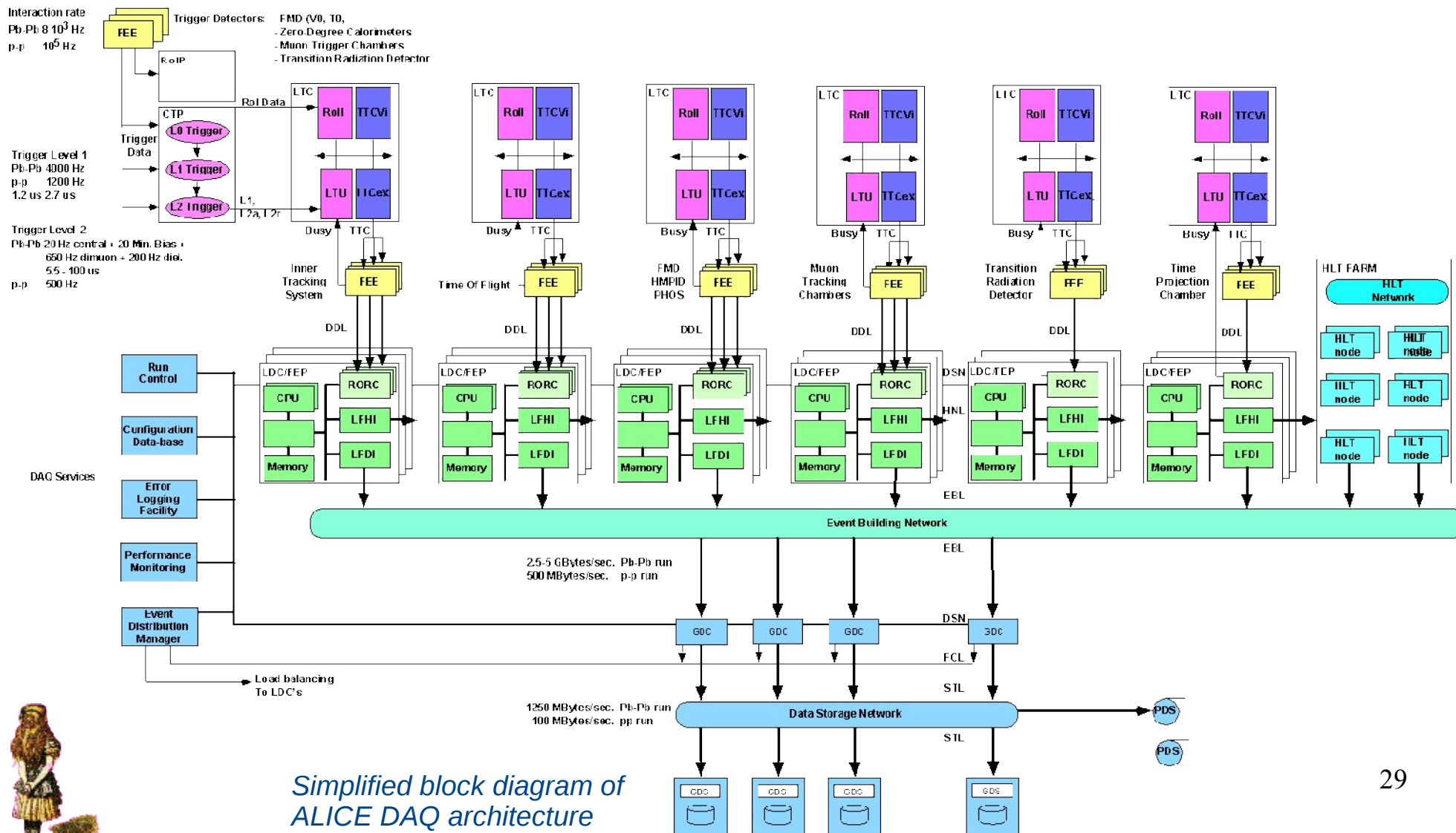
- ◆ Introduction to ROOT
 - ◆ What is it ?
 - ◆ Why is it good ?
- ◆ Using ROOT
 - ◆ Command line
 - ◆ Scripting/Interpretation
 - ➔ Example script comparing two current-mode D/A converter architectures
 - ◆ Compilation
 - ➔ Compiling a script into a "*.so" shared object library
 - ➔ Compiling standalone
 - ➔ Application development
 - ➔ Example standalone application
- ◆ GUI of ROOT
 - ◆ Human interaction
 - ◆ Creating a GUI
- ◆ Survival with/at/despite ROOT
 - ◆ User's guide
 - ◆ Referring to:
 - ➔ \$ROOTSYS/tutorials
 - ➔ \$ROOTSYS/test
 - ◆ HTML source code documentation
- ◆ External library usage from within ROOT
 - ◆ DQM of ALICE experiment @ CERN
 - ➔ Simplified DAQ operation
 - ◆ Understanding the detector data
 - ➔ Accessing and decoding data



External library usage from within ROOT

Example: Data quality monitoring application for ALICE experiment at CERN

- Large-scale experiments having a deep hierarchy of sub-systems
- These systems need to be **monitored at different levels** (e.g. LDC, GDC, etc). This is because the quality of the data depends on many parameters (human, accelerator, detector slow control, etc.)
- Access to data is established by a **C/C++ library** provided by the **DAQ team**

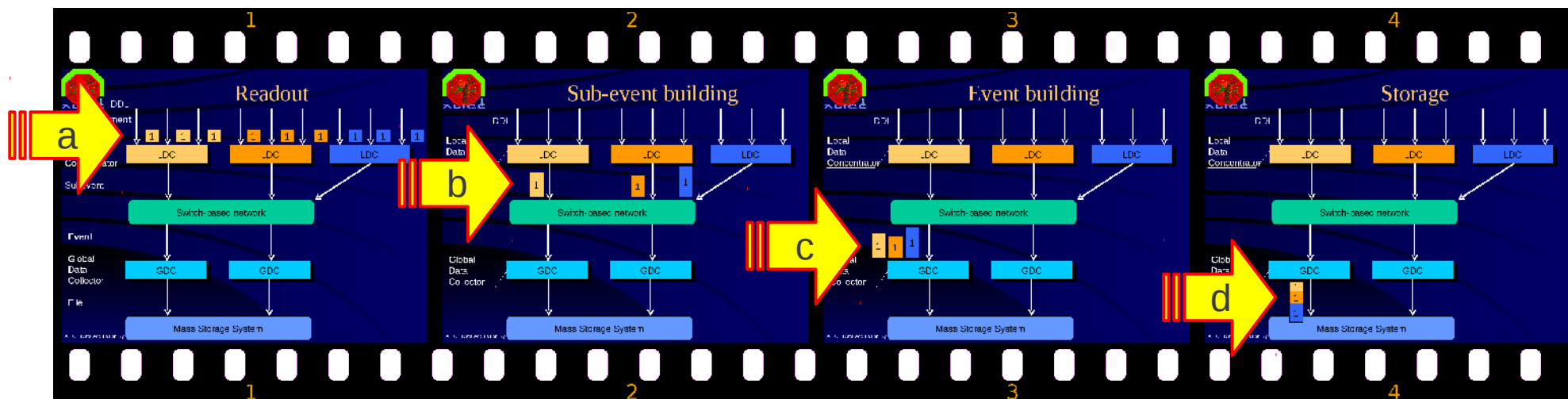


Simplified block diagram of ALICE DAQ architecture



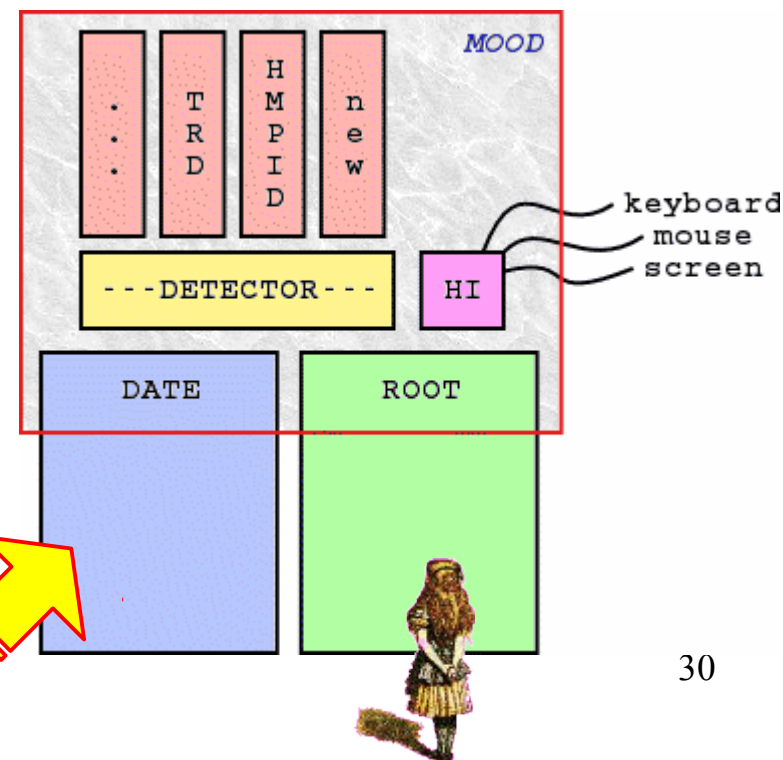
External library usage from within ROOT

Simplified DAQ operation



* Further simplified architecture (from Pierre Vande Vyvre's presentation)

- FE ASICs reading out the detectors generate **payloads (a)**
- Payloads travel further down within the DAQ network into LDCs (Local Data Collector) to get together, forming the **event fragments (b)**
- Event fragments move further down to GDCs (Global Data Concentrator) and are put together, forming **sub-events (c)**
- Sub-events get together forming the **super-event** and are sent to **permanent data storage** for **off-line analysis (d)**
- **Our task** is to develop a **monitoring application** using **DATE** library for data access and **ROOT** class library for GUI, event display, and analysis.



External library usage from within ROOT

Accessing data via monitoring functions provided by an external library

```
oc@o1mak-x200:~$ root -l
root [0] gSystem->Load("${DATE_MONITOR_DIR}/{DATE_SYS}/libmonitor.so");
root [1] monitorSetDataSource(":"); /* Enable local on-line monitoring*/
root [2] int *event;
root [3] monitorGetEventDynamic(&event);
root [4] printf("%08x %08x \n", event[0], event[1]);
0000878c da1e5afe
Root [5] .q
oc@o1mak-x200:~$ _
```

DATE Event Format

- 0 - Total size of the event (0000878c)
 - 1 - Unique DATE event signature (da1e5afe)
 - 2 - Size of the header (base & extension)
 - 3 - Base event header structure version
 - 4 - Type of event
 - 5 - Number of the run associated to the event
 - 6 - Unique event identification
 - 7 - Level 2 trigger associated to the event
 - 8 - Detector pattern associated to the event
 - 9 - Attributes associated to the event
 - 10 - ID of the LDC
 - 11 - ID of the GDC
 - 12 - Time stamp at the creation of the event
- <Equipment Header "n">
<Data associated to the equipment "n">
<Equipment Header "n-1">
<Data associated to the equipment "n-1">

Summary

- Root [0] → Load the monitoring library
- Root [1] → Set data source
- Root [2] → Variable to cast data onto
- Root [3] → Get the data into the variable
- Root [4] → Show the first two "words"
- Root [4] → End the session

Super Event header

Data generated by a specific part of a detector (payload)

Payload

This is ~literally more information than your ID card has about you :)

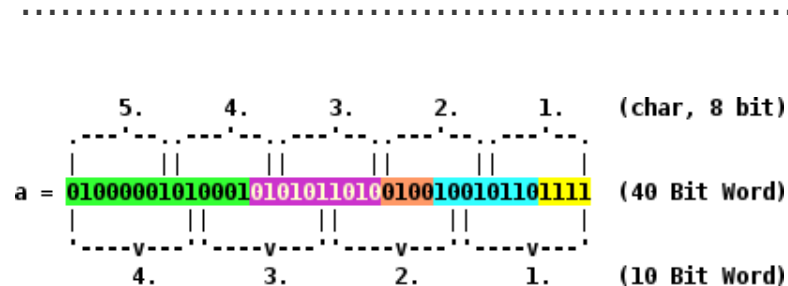
...

External library usage from within ROOT

Parsing and/or casting the detector data - Decoding payload

- Beginning of an ALICE **TPC** (Time Projection Chamber) event (total size is roughly 60 Mbyte)
- Yellow**, **green** and **magenta** highlighted parts represent data headers generated by different levels of hardware within the DAQ chain (**event** / **equipment** / **hardware** headers)
- The rest is **raw data** generated by the TPC
- A TPC "**word**" is also analyzed below (a decoding example):
 - A phrase composed of 4 words of **10 bits**
 - A phrase composed of 5 words of **8 bits**
 - A single word of **40 bits**
 - A phrase composed of 5 words of, from left to right **4, 8, again 4, 10, and 14 bits** (we are interested in this level)

0000:0000	8c870000	fe5a1eda	44000000	04000300	07000000	78130000	00000000	f842bf06
0000:0020	01000000	00000000	00000000	f0000000	00000000	20000000	03000000	ffffff
0000:0040	d30e0b43	48870000	11000000	69000000	00000000	00000000	00000000	04000000
0000:0060	01000000	02000000	03000000	04000000	05000000	06000000	07000000	08000000
0000:0080	3ef4d043	0f3df4e0	430f3df8	d0430f3d	f8e0430f	3df4d043	0f3ef4d0	430f3ef4
0000:00a0	d0430f3d	f4d0830f	3ef4d043	0f3ef8d0	430f3df4	d0430f3d	f8d0830f	3df8e003
0000:00c0	0f3ef8d0	430f3df4	d0430f3d	f4e0830f	3ef4d083	0f3df0d0	830f3df4	d0830f3d
0000:00e0	f4e0430f	3ef4d043	0f3ef8d0	430f3df4	e0830f3d	f0d0430f	3df4e043	0f7298a1
0000:0100	aaaa00a8	66a8aa3d	f8d0830f	3ef4d043	0f3ef4d0	430f3df4	e0830f3e	f4d0430f
0000:0120	3df0d043	0f3ef8d0	430f3df8	e0430f3d	f4e0430f	3df4e043	0f3df8e0	430f3df4
0000:0140	d0830f3e	f4d0830f	3ef4e083	0f3ef8d0	430f3ef4	e0430f3d	f4d0430f	3df4d043
0000:0160	0f3df4e0	430f3df4	d0830f3d	f4d0430f	3df4e043	0f3ef8c0	430f3df4	e0430f3c
0000:0180	f8e0830f	7298a1aa	aa01a866	a8aa2bac	b0c20a2a	b0c0c20a	2bacb0c2	0a2bb0b0
0000:01a0	c20a2aac	b0c20a2b	acb0c20a	2ab0b082	0a2aacb0	c20a2bb0	b0820a2b	acb0c20a
0000:01c0	2bacb0c2	0a2bb0c0	c20a2bac	b0820a2b	b0b0c20a	2bb0b0c2	0a2bacc0	020b2bac
0000:01e0	b0c20a2b	acb0c20a	2aacb0c2	0a2bacb0	c20a2bac	b0020b2b	acb0c20a	2bacb0c2
0000:0200	0a2bacc0	c20a2cac	b0c20a72	98a1aaaa	02a866a8	aa2bacb0	020b2cac	c0c20a2c
0000:0220	b0c0020b	2cb0c0c2	0a2cb0c0	c20a2cb0	c0c20a2b	acc0020b	2cb0c002	0b2bb0c0
0000:0240	020b2cac	c0c20a2c	b0c0020b	2cb0c002	0b2cb0b0	c20a2bb0	c0020b2c	acc0c20a
0000:0260	2bacc002	0b2cb0c0	c20a2cb0	c0c20a2c	acb0c20a	2bb0c002	0b2bacc0	c20a2cb0
0000:0280	c0020b2c	b0c0020b	2bacc0c2	0a2bb0c0	c20a7298	a1aaaa03	a866a8aa	28a070c2
0000:02a0	09279c60	c209279c	80020a27	9c70c209	289c70c2	09279c70	c20928a0	80c20928
0000:02c0	a070c200	72a070c2	0072a000	c200720c	70070c27	0c80c200	720c70c2	00709c70

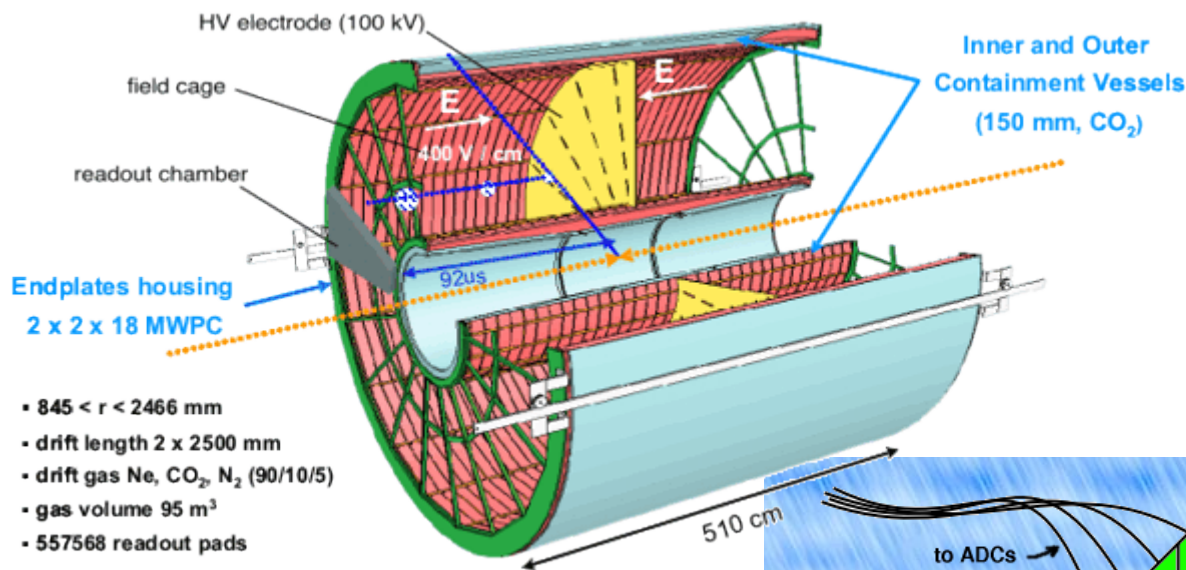


Seq.	bin	dec	(10 Bit Words, trailer)
Channel.	1111	15	(yellow)
Hardware.	10010110	150	(light blue)
Pattern A	0100	4	(orange)
10BitWord	0101011010	346	(magenta)
2AAA Pat.	01000001010001	4177	(green)

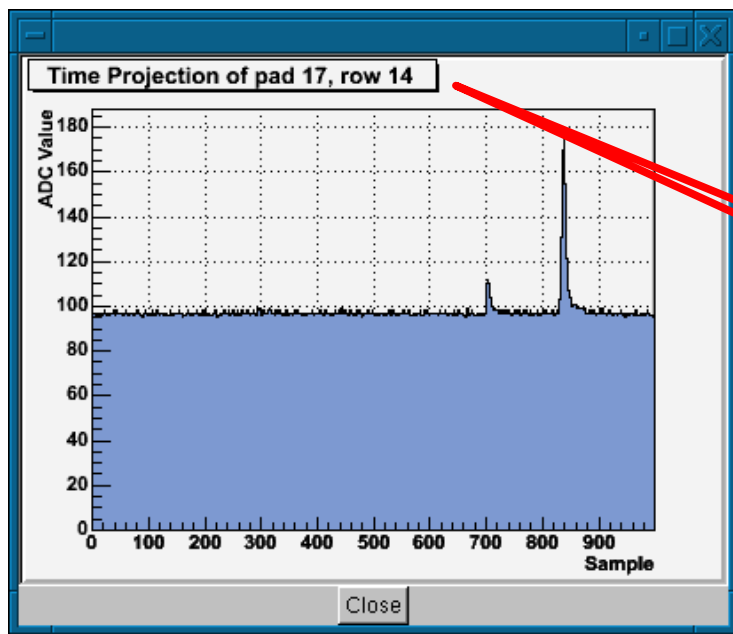
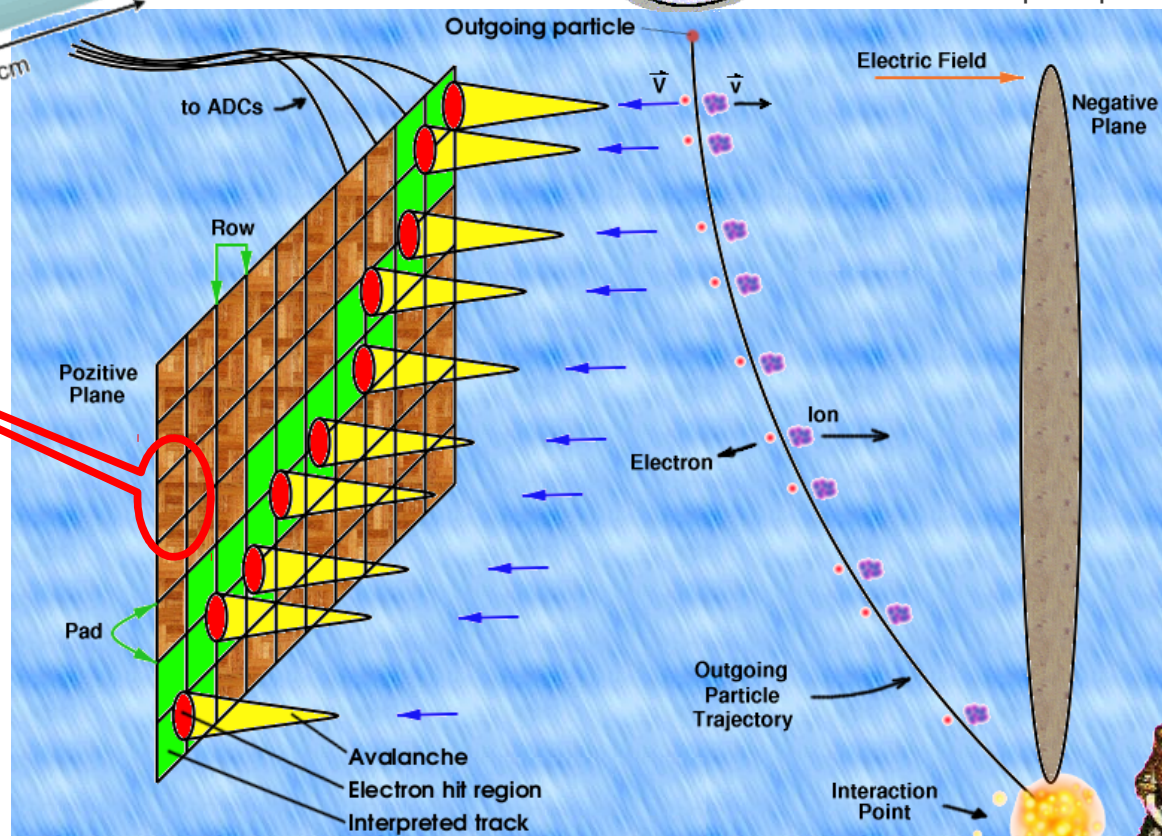
External library usage from within ROOT

Understanding the detector data

- For our example, let us choose the **Time Projection Chamber** detector of **ALICE** experiment.



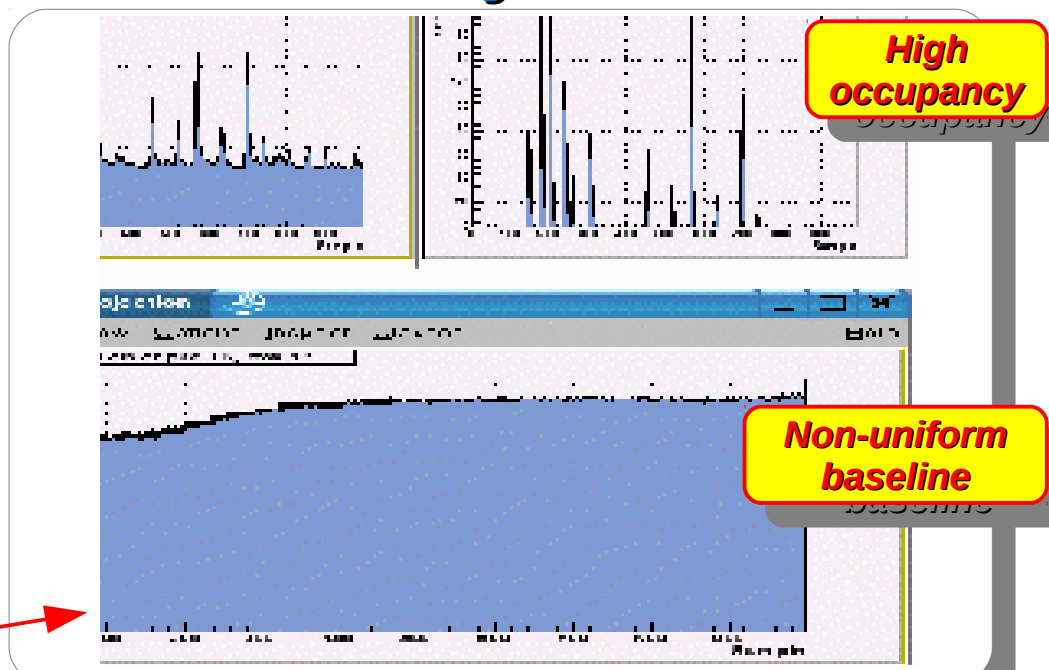
TPC detection principle



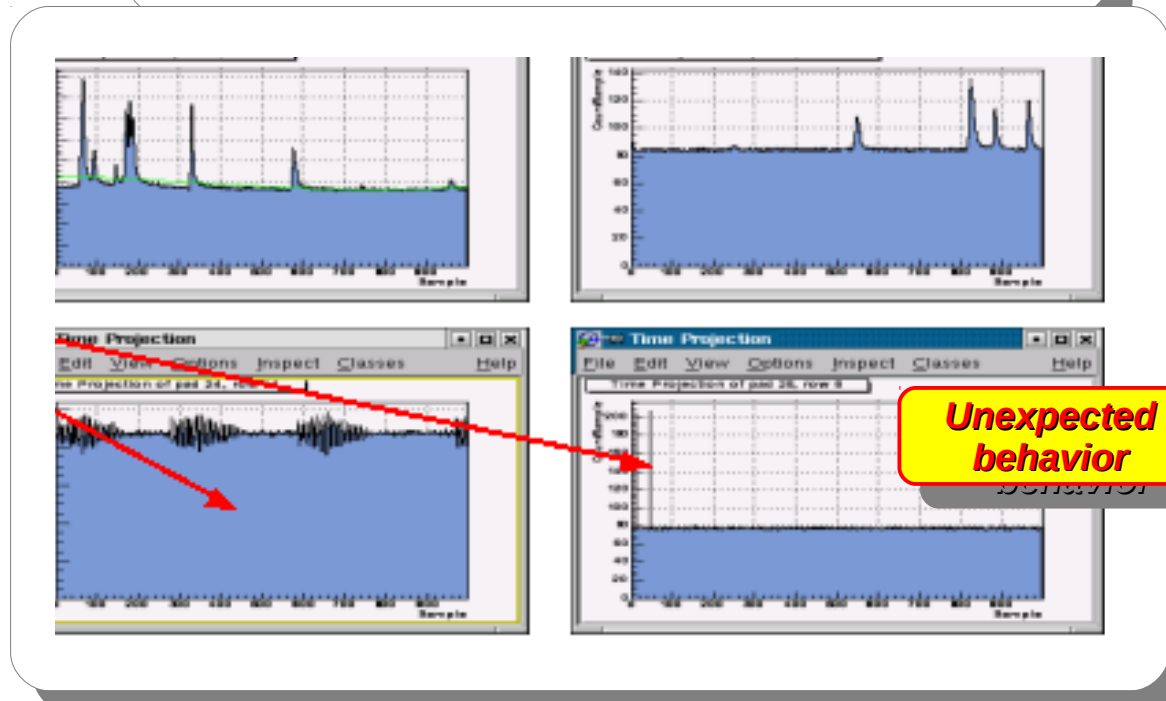
What do we expect to see ?

What errors do we expect to catch via the monitoring tool ?

- **Electronic issues**
 - ➔ Stack-at errors
 - ➔ Non-configurable FE, etc.
 - ➔ Unexpected behavior
- **Detector issues**
 - ➔ Gas mixture and dynamics
 - ➔ Occupancy, etc.
 - ➔ Unexpected behavior
- **General**
 - ➔ Temperature & supply variations
 - ➔ Peak formation
 - ➔ Baseline fluctuations, etc.
 - ➔ Unexpected behavior



- ➔ A channel generating the same value all the time
- ➔ Un-familiar condition within the detector

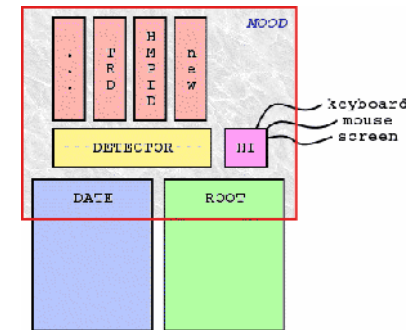


Compiling Standalone

Standalone application

- An elegant method to write applications is via **Makefile**, but we will not do it here
- To compile a code similar to the one on the previous page:
 - ➔ **g++** -L/date_home -lbase -laz -lgui -thread -lm -lmst -ldynamic -lnonStand -lshift -lcustomDet -lmntrno {...} **secondApp.cxx** -o **secondApp**
- Remembering all the above things is hard, therefore we will use:
 - ➔ **date-config**: a command-line tool to make lives of DATE users easy (libraries usually have tools like this one, remember ?)
 - ➔ It returns appropriate lines needed for compilation
 - ➔ Usually used in-between “ ` ”, aka escape symbol
- ➔ **g++** `date-config --glibs --cflags` **secondApp.cxx** -o **secondApp**

```
g++  
`root-config --glibs --cflags`  
`date-config --glibs --cflags`  
thirdApp.cxx -o thirdApp
```



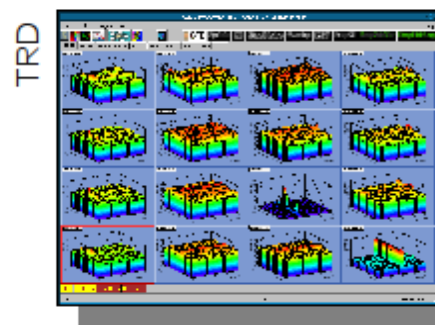
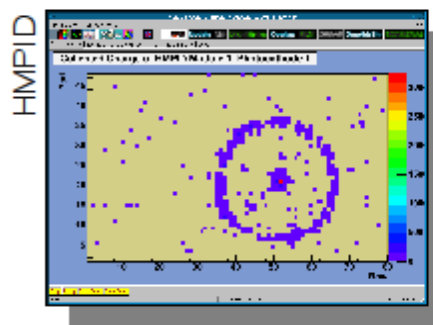
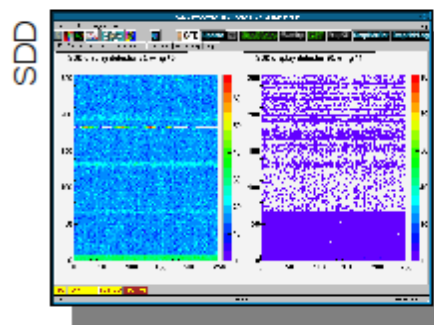
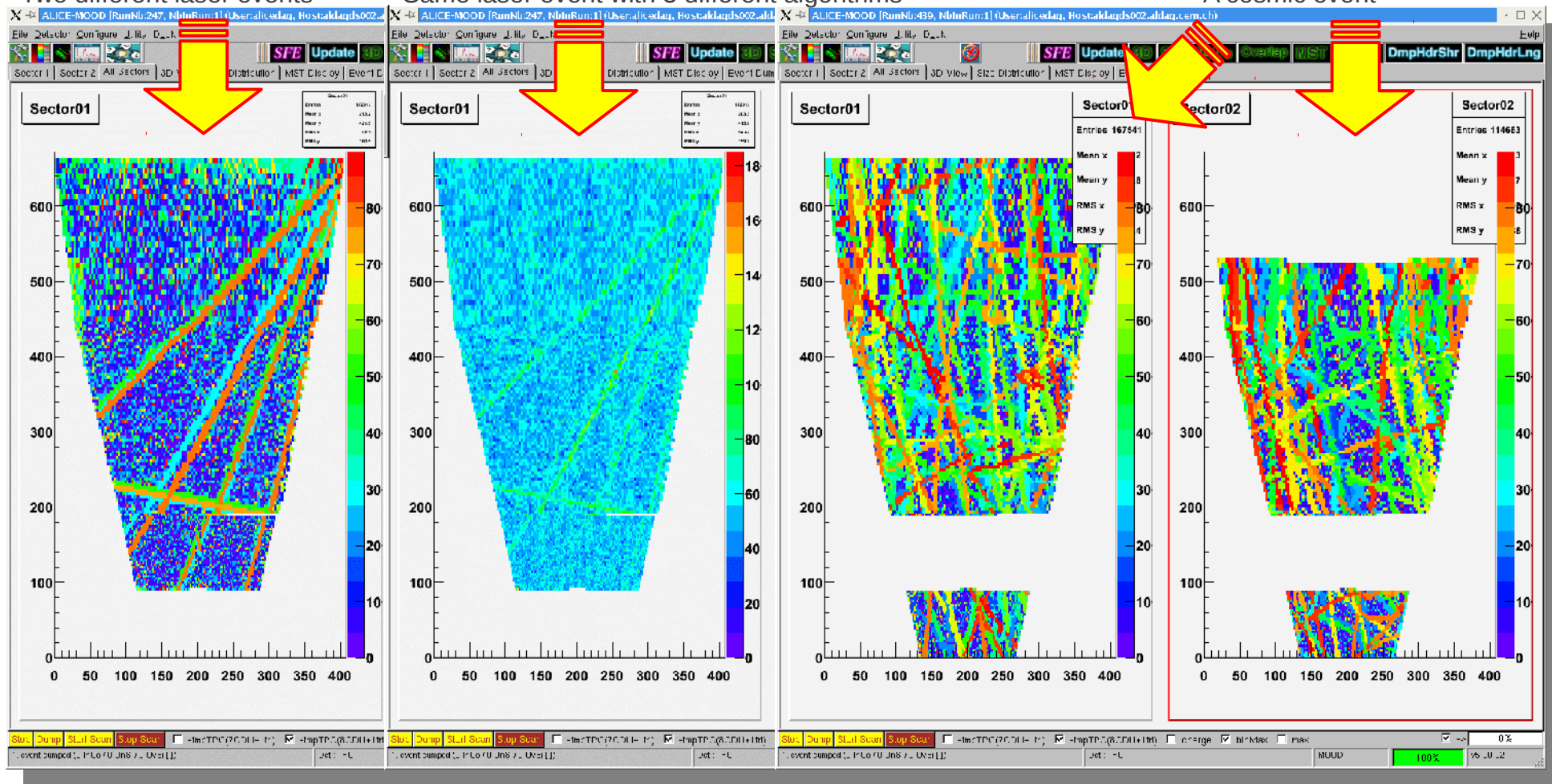
External library usage from within ROOT

Full featured application being delivered to detector people

Two different laser events

Same laser event with 3 different algorithms

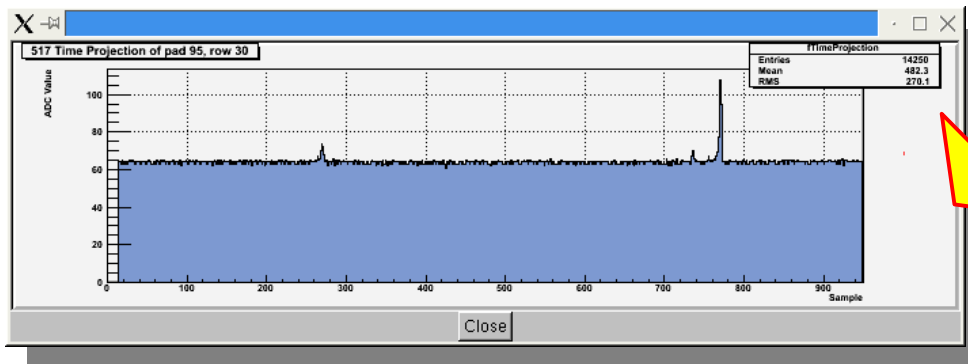
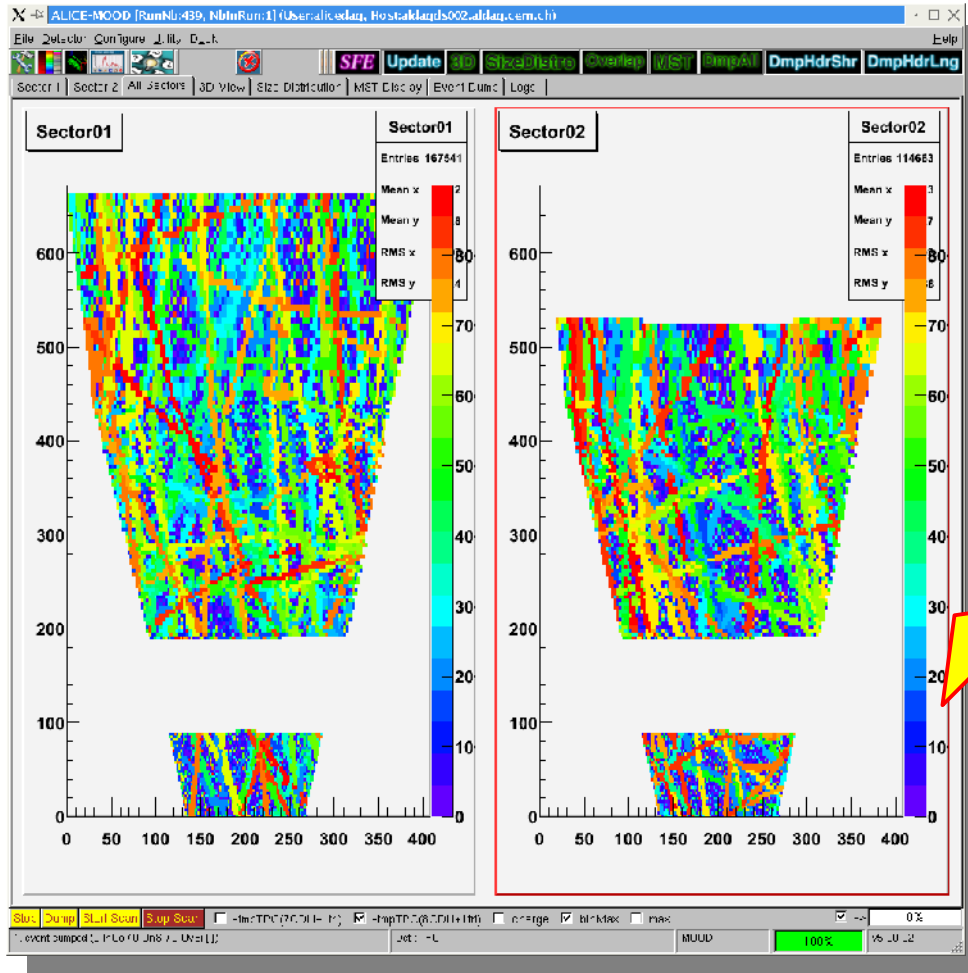
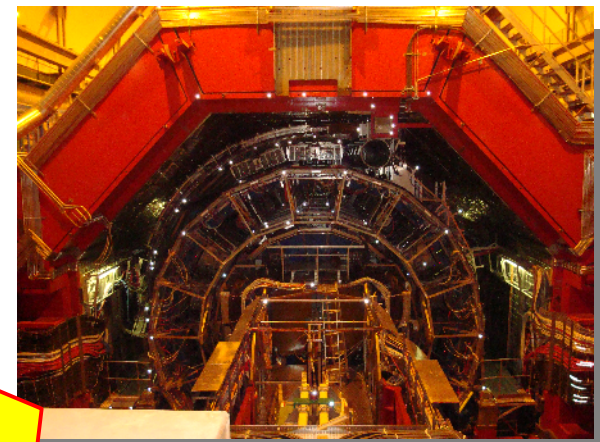
A cosmic event

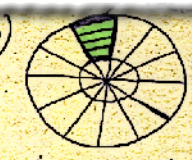


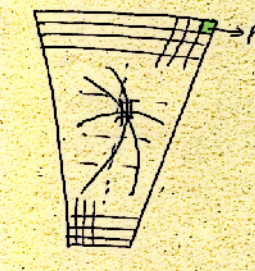
Coding or design ?

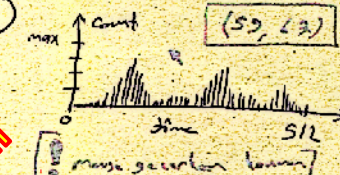
Think, design, write, implement and only then start coding


From
2002
To
today



1)  inner side of TPC

2)  Pad
Ditim Selected

3)  count
time
S12
(59, 23)

4)  DATE

Ditim klbkendiginde 2. nolu pencere olusabilir. Save (PS), Exit, Help, Print butonlari oluslur. iz gostergeci yaktur. Cluster gostergeci yaktur mouse surene geldiginde parlatimim ve kisa birer veriyi 15Ayarim.

(mouse) Pad klbkendiginde 3. nolu pencere oluslur. Save (PS), Exit, Print, Help butonlari burada da oluslur. Fined padlar burada gosterilmis ve charge collection gune renk bilgisi ile (blurred olamak tabiri) iz gosterim hale getirilmislerdir. Update butonu ile gosterim guncellenmislerdir. mouse surene gelen pa parlatimlari (mouse-3) sas klbkendiginde fine kamm bilgisi gosterilmislerdir.

Charge collection bilgisi zamanin fonksiyonu olarak gosterilmislerdir. Update butonu basildiginde aynı pad için gazlari duyer tabiri olusur ve spectrum guncellenmislerdir. Save, Print, Exit, Help butonlari zaten oluslur. Hangi fiziksel pad oldugu ve nun nuni bilgisi de basilmislerdir.

DATE 'den gelen cluster bilgisi pad'a onta klbkelyerek alınabilir. Data isinlari cluster algoritması ile analiz edilerek kucuklari ve daha net fikirlerin

ROOT - A brief introduction

Survival with ROOT != Survival at ROOT != Survival despite ROOT



- ◆ Introduction to ROOT
 - ◆ What is it ?
 - ◆ Why is it good ?
- ◆ Using ROOT
 - ◆ Command line
 - ◆ Scripting/Interpretation
 - ➔ Example script comparing two current-mode D/A converter architectures
 - ◆ Compilation
 - ➔ Compiling a script into a "*.so" shared object library
 - ➔ Compiling standalone
 - ➔ Application development
 - ➔ Example standalone application
- ◆ GUI of ROOT
 - ◆ Human interaction
 - ◆ Creating a GUI
- ◆ Survival with/at/despite ROOT
 - ◆ User's guide
 - ◆ Referring to:
 - ➔ \$ROOTSYS/tutorials
 - ➔ \$ROOTSYS/test
 - ◆ HTML source code documentation
- ◆ External library usage from within ROOT
 - ◆ DQM of ALICE experiment @ CERN
 - ➔ Simplified DAQ operation
 - ◆ Understanding the detector data
 - ➔ Accessing and decoding data



*