HVCheck

October 10, 2016

1 Program do analizy kalibracji kanałów wysokiego napięcia detektora promieniowania przejścia eksperymentu ATLAS

1.0.1 Pobieranie danych

Pierwszym krokiem przed rozpoczęciem analizy jest pobranie danych o napięciu zadanycm i zmierzonym, dla każdej komórki detektora. Odbywa się to za pośrednictwem poniższego skryptu. Dane są zapisywane w osobnych plikach i segregowane do opowiednich katalogów.

In [ ]:
```
import os
from datetime import datetime


#colors
YELLOW = '\033[93m'
END = '\033[0m'
#schemata


CellNr = 0

#WCZYTYWANIE PLIKU Z DANYMI:

with open("cell_list.txt","r") as file:
    target_cell_list = file.readlines()

file.close()

print YELLOW+'Wczytano plik z danymi...'
#=============================================================================
#USTALANIE DATY:

start_date = '12-03-2011 11:22'
end_date = '19-03-2011 12:22'

cmd_date = ' ,'+ start_date + ' ,' + end_date +'!'
#=============================================================================
#ZPYTANIA:
```

```
t1 = datetime.now()

while(CellNr< len(target_cell_list)):
        url = '" http://atlas-ddv.cern.ch:8089/multidata/downloadTxtData'
        cmd_start = 'wget --post-data '
        cmd_query_start ='"queryInfo='
        query1 = 'atlas_pvsstrt, element_name, ' + target_cell_list[CellNr]
        FullCmd = cmd_start + cmd_query_start + query1 + cmd_date + url
        print FullCmd
        os.system(FullCmd)
        with open('downloadTxtData',"r") as file:
                    Data = file.readlines()
        file.close()
        name = target_cell_list[CellNr][16:38]
        n = name.split("/")
        fname ='Output_Voltage/'+ n[0] + '.' + n[1]+'.'+n[2] +'.txt'
        output = open(fname,"w")
        for val in Data:
                output.write(val)
        output.close()
        os.remove('downloadTxtData')
        CellNr+=1

CellNr=0

while(CellNr< len(target_cell_list)):
        url = '" http://atlas-ddv.cern.ch:8089/multidata/downloadTxtData'
        cmd_start = 'wget --post-data '
        cmd_query_start ='"queryInfo='
        query1 = 'atlas_pvsstrt, element_name, ' + target_cell_list[CellNr]
        FullCmd = cmd_start + cmd_query_start + query1 + cmd_date + url
        print FullCmd
        os.system(FullCmd)
        with open('downloadTxtData',"r") as file:
                Data = file.readlines()
        file.close()
        name = target_cell_list[CellNr][16:38]
        n = name.split("/")
        fname ='Output_HVSetpoint/'+ n[0] + '.' + n[1]+'.'+n[2] +'.txt'
        output = open(fname,"w")
        for val in Data:
                output.write(val)
        output.close()
        os.remove('downloadTxtData')
        CellNr+=1

t2 = datetime.now()
```

```
#=========================================================================
dt = t2-t1
SEC =  dt.seconds
MIL = dt.microseconds/1000
print MIL


#colors
YELLOW = '\033[93m'
END = '\033[0m'

print YELLOW+"Query circle is "+str(SEC)+"."+str(MIL)+" seconds"+END
```

1.0.2 Konwertowanie danych z plików txt do ROOT

Nastepnie dane są przepisywane z wielu plików tekstowych do dwóch plików o rozszerzeniu ROOT (jedne dla napięcia zmierzonego, drugi dla zadanego). Po przepisaniu dane przybierają formę TNtupla - typu który może przechowywać duże ilości informacji i jest zoptymalizowany do obliczeń w środowisku ROOT.

```
In [ ]: import os
        import ROOT
        import time
        from datetime import datetime

        def repochtuple(str):
                d=int(str[0:2])
                m=int(str[3:5])
                y=int(str[6:10])
                h=int(str[10:13])
                mi=int(str[14:16])
                s=int(str[17:19])
                ms=int(str[20:23])
                t=(y,m,d,h,mi,s,ms,0,0)
                return t

        def beamcheck(time,sbf):
                i=0
                ret=0
                while i<len(sbf):
                        if i==0:
                                if time>=sbf[0][0] and time<sbf[1][0] and sbf[0][1]
                                        ret=1
                        elif i+1==len(sbf):
                                if time>=sbf[len(sbf)-1][0] and sbf[len(sbf)-1][1]=
                                        ret= 1
                        elif i>0 and i<len(sbf)-1:
                                if time>=sbf[i][0] and time<sbf[i+1][0] and sbf[i]|
```

```
                                ret= 1
                else:
                        ret= 0
                i+=1
        return ret


with open("ATLGCSLHC.lhcStatus.StableBeamsFlag.txt","r") as file:
        SBFlags = file.readlines()

sbflag_data=[]
vec=[]
for data in SBFlags:
        vec=[]
        if len(data)>2:
                t=time.mktime(repochtuple(data[38:61]))
                vec.append(t)
                vec.append(float(data[64:]))
                sbflag_data.append(vec)

print sbflag_data

with open("cell_list.txt","r") as file:
        target_cell_list = file.readlines()

file.close()

CellNr=0

ATLTRTHVA_HVSYS_OutputVoltage = []
ATLTRTHVA_HVSYS_HVSetpoint = []
CRATE=1
BRANCH=2
CELL=3
VALUE=6
DATE=4
TIME=5



Voltage_Value = ROOT.TNtuple("Voltage_Val","ATLTRTHVA_HVSYS_OutputVoltage",
HVSetpoint_Value = ROOT.TNtuple("HVSetpoint_Val","ATLTRTHVA_HVSYS_HVSetpoir

root_output_f = ROOT.TFile("root_out.root","RECREATE")

#VOLTAGE GET DATA============================================================

print "Processing VoltageOutput data...."
```

```
while(CellNr<len(target_cell_list)):
        name = target_cell_list[CellNr][16:38]
        n = name.split("/")
        fname ='Output_Voltage/'+ n[0] + '.' + n[1]+'.'+n[2] +'.txt'
        fname_out ='Output_Voltage_'+ n[0] + '_' + n[1]+'_'+n[2]
        output = open(fname,"r")
        Voltage_Value = ROOT.TNtuple(fname_out,"ATLTRTHVA_HVSYS_OutputVolta
        with open(fname,"r") as input:
                input_data = input.readlines()
        for data in input_data:
                if len(data)>6:
                        t = time.mktime(repochtuple(data[59:82]))
                        sbf_check = beamcheck(t,sbflag_data)
                        if sbf_check == 1:
                                ATLTRTHVA_HVSYS_OutputVoltage.append([int(n
                                Voltage_Value.Fill(t,float(data[85:]))
        CellNr+=1
        Voltage_Value.Write()

root_output_f.Close()
print "Done"

#HVS DATA GET=================================================================

CellNr=0
root_output_f = ROOT.TFile("root_hvs.root","RECREATE")

print "Processing HVS data...."
while(CellNr<len(target_cell_list)):
        name = target_cell_list[CellNr][16:38]
        n = name.split("/")
        fname ='Output_HVSetpoint/'+ n[0] + '.' + n[1]+'.'+n[2] +'.txt'
        output = open(fname,"r")
        fname_out ='Output_HVSetpoint_'+ n[0] + '_' + n[1]+'_'+n[2]
        HVSetpoint_Value = ROOT.TNtuple(fname_out,"ATLTRTHVA_HVSYS_OutputVo
        with open(fname,"r") as input:
                input_data = input.readlines()
        for data in input_data:
                if len(data)>6:
                        #print data[57:80]
                        t = time.mktime(repochtuple(data[56:80]))
                        sbf_check = beamcheck(t,sbflag_data)
                        if sbf_check ==         1:
                                ATLTRTHVA_HVSYS_HVSetpoint.append([int(n[0]
                                HVSetpoint_Value.Fill(t,float(data[82:]))
        CellNr+=1
        HVSetpoint_Value.Write()
```

```
root_output_f.Close()
print "Done"
```

Dodatkowo rysowane są wykresy dla poszczególnych komórek, przedstawiających zmianę napięć w czasie.

```
In [ ]: import os
        import ROOT
        import time
        from ROOT import gROOT
        from array import array


        #colors
        YELLOW = '\033[93m'
        END = '\033[0m'
        WARNING = '\033[91m'
        #schemata


        from datetime import datetime

        def repochtuple(str):
                d=int(str[:2])
                m=int(str[3:5])
                y=int(str[6:10])
                h=int(str[10:13])
                mi=int(str[14:16])
                s=int(str[17:19])
                ms=int(str[20:23])
                t=(y,m,d,h,mi,s,ms,0,0)
                return t

        def beamcheck(time,sbf):
                i=0
                ret=0
                while i<len(sbf):
                        if i==0:
                                if time>=sbf[0][0] and time<sbf[1][0] and sbf[0][1]
                                        ret=1
                        elif i+1==len(sbf):
                                if time>=sbf[len(sbf)-1][0] and sbf[len(sbf)-1][1]=
                                        ret= 1
                        elif i>0 and i<len(sbf)-1:
                                if time>=sbf[i][0] and time<sbf[i+1][0] and sbf[i]|
                                        ret= 1
                        else:
                                ret= 0
```

```
                i+=1
        return ret

with open("ATLGCSLHC.lhcStatus.StableBeamsFlag.txt","r") as file:
        SBFlags = file.readlines()

sbflag_data=[]
vec=[]
for data in SBFlags:
        vec=[]
        if len(data)>2:
                t=time.mktime(repochtuple(data[38:61]))
                vec.append(t)
                vec.append(float(data[64:77]))
                sbflag_data.append(vec)

print sbflag_data


with open("cell_list.txt","r") as file:
    target_cell_list = file.readlines()

file.close()

CellNr=0


gROOT.Reset()


root_output_f = ROOT.TFile("root_graph_out.root","RECREATE")

#VOLTAGE GET DATA=============================================================
print YELLOW+'======================DRAWING=============================
print "Processing VoltageOutput data...."
print "ilosc wszystkich cell ",len(target_cell_list),END
while(CellNr<len(target_cell_list)):
        name = target_cell_list[CellNr][16:38]
        n = name.split("/")
        fname_vv ='Output_Voltage/'+ n[0] + '.' + n[1]+'.'+n[2]
        gname_vv = 'Output_Voltage_'+n[0] + '_' + n[1]+'_'+n[2]
        with open(fname_vv+".txt","r") as input:
                input_data_vv = input.readlines()
        nb_vv=len(input_data_vv)
        input.close()
        #print name,' ',fname,' ',nb
        x_vv=0
        y_vv=0
```

```
        x_vv, y_vv = array('f'), array('f')
        if nb_vv>1:
                for data in input_data_vv:
                        if len(data)>6:
                                t = time.mktime(repochtuple(data[59:82]))
                                sbf_check = beamcheck(t,sbflag_data)
                                if sbf_check == 1:
                                      x_vv.append(float(t))
                                      y_vv.append(float(data[85:]))
        else :
                print 'nr petli ',CellNr,' ',fname_vv,WARNING+' --->File co
        mb_vv = len(x_vv)
        #print mb-vv, ' ' , x_vv[0],' ',y_vv[0],' ',CellNr
        if mb_vv>1:
                Voltage_Value = ROOT.TGraph(mb_vv,x_vv,y_vv)
                Voltage_Value.SetTitle(gname_vv)
                Voltage_Value.SetName(gname_vv)
                Voltage_Value.Write()
        CellNr+=1

root_output_f.Close()
print YELLOW+"Done"+END

#HVS DATA GET=================================================================

CellNr=0
root_output_f = ROOT.TFile("root_graph_hvs.root","RECREATE")

print YELLOW+ "Processing HVS data...."+END

while(CellNr<len(target_cell_list)):
        name = target_cell_list[CellNr][16:38]
        n = name.split("/")
        fname_hvs ='Output_HVSetpoint/'+n[0] + '.' + n[1]+'.'+n[2]
        gname_hvs ='Output_HVSetpoint_' + n[0] + '_' + n[1]+'_'+n[2]
        with open(fname_hvs+".txt","r") as input:
                input_data_hvs = input.readlines()
        nb_hvs=len(input_data_hvs)
        input.close()
        #print name,' ',fname,' ',nb
        x_hvs=0
        y_hvs=0
        x_hvs, y_hvs = array('f'), array('f')
        if nb_hvs>1:
                for data in input_data_hvs:
                        if len(data)>6:
                                t = time.mktime(repochtuple(data[56:80]))
                                sbf_check = beamcheck(t,sbflag_data)
```

```
                                    if sbf_check == 1:
                                        x_hvs.append(float(t))
                                        y_hvs.append(float(data[82:]))
                else :
                            print 'nr petli ',CellNr,' ', fname_hvs,WARNING+' ---
                mb_hvs = len(x_hvs)
                if mb_hvs>1:
                        #print mb_hvs, ' ' , x_hvs[0],' ',y_hvs[0],' ',CellNr,'===
                        HVSetpoint_Value = ROOT.TGraph(mb_hvs,x_hvs,y_hvs)
                        HVSetpoint_Value.SetTitle(gname_hvs)
                        HVSetpoint_Value.SetName(gname_hvs)
                        HVSetpoint_Value.Write()
                CellNr+=1


        c1=ROOT.TCanvas('c','c',1024,768)
        c1.SetGridx()
        c1.SetGridy()

        print type(HVSetpoint_Value)
        print type(Voltage_Value)

        HVSetpoint_Value.Draw()
        Voltage_Value.Draw("Same")

        c1.Update()
        c1.Write()
        root_output_f.Close()
        print YELLOW+"Done"+END
```

1.0.3 Analiza stopnia skalibrowania

Ostatnim krokiem jest przeprowadzenie alnalizy i stworzenie histogramów, które będą obrazować dokładność kalibracji. Komórki które nie sełnią kryteriów analizy zostaną oznaczone jako rozkalibrowane i wypisane na końcu obliczeń.

In [ ]:
```
import os
import ROOT

def cls():
    os.system(['clear','cls'][os.name == 'nt'])

#colors
YELLOW = '\033[93m'
END = '\033[0m'
WARNING = '\033[91m'
#schemata
```

```
input_f_vv = ROOT.TFile("root_out.root","READ")
input_f_hvs = ROOT.TFile("root_hvs.root","READ")

CellNr = 0
error_list=[]
with open("cell_list.txt","r") as file:
        target_cell_list = file.readlines()
file.close()

ROOT.gROOT.Reset()
th1f_A_all = ROOT.TH1F("HVA_mid_diff_all_histo","HVA_mid_diff_all_histo",10
th1f_B_all = ROOT.TH1F("HVB_mid_diff_all_histo","HVB_mid_diff_all_histo",10
th1f_C_all = ROOT.TH1F("HVC_mid_diff_all_histo","HVC_mid_diff_all_histo",10


ntuple_srch_A = ROOT.TNtuple("HVA_mid_diff_all","HVA_mid_diff_all","value")
ntuple_srch_A_1 = ROOT.TNtuple("HVA_mid_diff_lastP","HVA_mid_diff_lastP","v

ntuple_srch_B = ROOT.TNtuple("HVB_mid_diff_all","HVB_mid_diff_all","value")
ntuple_srch_B_1 = ROOT.TNtuple("HVB_mid_diff_lastP","HVB_mid_diff_lastP","v

ntuple_srch_C = ROOT.TNtuple("HVC_mid_diff_all","HVC_mid_diff_all","value")
ntuple_srch_C_1 = ROOT.TNtuple("HVC_mid_diff_lastP","HVC_mid_diff_lastP","v

output_f_data = ROOT.TFile("root_minusData.root","RECREATE")
CrateNr=0


print YELLOW+'>PROCESS START>'+END
while CellNr<len(target_cell_list):
        #getting ntuple name=================================================
        name = target_cell_list[CellNr][16:38]
        n = name.split("/")
        CrateNr = n[0][-1]
        fname_out ='Output_Voltage_'+ n[0] + '_' + n[1]+'_'+n[2]
        name_vv ='Output_Voltage_'+ n[0] + '_' + n[1]+'_'+n[2]
        name_hvs ='Output_HVSetpoint_'+ n[0] + '_' + n[1]+'_'+n[2]
        #reading ntuples from files=========================================
        ntuple_vv = input_f_vv.Get(name_vv)
        ntuple_hvs = input_f_hvs.Get(name_hvs)
        ntuple_vv.SetName('Voltage/'+name)
        ntuple_hvs.SetName('HVSetpoint'+name)
        entries_vv = ntuple_vv.GetEntries()
        entries_hvs = ntuple_hvs.GetEntries()
#        print 'hvs entries: ',entries_hvs
#        print 'vv entries: ',entries_vv
#seting up for cuting hvs============================================
# time_table :=[
```

```
#         [cut start]
#         [hvs]
#         [cut end/new cut start]]
        ientry=0
        val_stamp=0
        val_stamp_current=0
        time_stamp=0
        time_stamp_prev=0
        time_table=[]
        vec=[]
        for entry in ntuple_hvs:
                val_stamp_current = entry.value
                if val_stamp != entry.value:
                        vec.append(time_stamp)
                        time_table.append(vec)
                        vec=[]
                        vec.append(time_stamp)
                        vec.append(val_stamp_current)
                        val_stamp = entry.value
                        time_stamp = entry.time
#                        print '-=-'
                time_stamp_prev = entry.time
                ientry+=1
#                print entry.time , ' ' , ientry
        vec.append(time_stamp_prev)
        time_table.append(vec)
        val_stamp = entry.value
        time_stamp = entry.time
#         print ientry
        ientry=0
        #for data in time_table:
        #        print data
#=============================================================================
# diff_table:=[
#         timeperoid:=[
#                 [diff value]
#                 [time]
#                 ]
#         ]
        itabl=1
        x=0
        y=0
        mid_val_lp=0
        mid_val_all=0
        diff_table=[]
        combined_val=[]
        just_val=[]
#         SWITCH CONTROLS  <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
```

```
        switch = 0
#        print time_table
        if len(time_table)<=1:
                print '-----------------------------------------------'
                print 'error == 1: ',name, ' ' ,CellNr
                error_list.append('blad == 1: '+name)
                for l in time_table:
                        print l
        else:
#                print name,'================================',CellNr
#                print time_table
                for entry in ntuple_vv:
                        #trip prevent system
                        if not(entry.value<=800):
                                val=[]
                                if itabl>=len(time_table):
                                        itabl-=1
                                if not(entry.time>=time_table[itabl][0] and
                                        diff_table.append(combined_val)
                                        combined_val=[]
                                        itabl+=1
                                        if itabl<len(time_table):
                                                diff_val = time_table[itabl
                                                val.append(entry.time)
                                                val.append(diff_val)
                                                combined_val.append(val)
                                                x+=1
                                                y+=1
                                                mid_val_all+=diff_val
                                                mid_val_lp +=diff_val
                                                just_val.append(val)
                                else:
                                        diff_val = time_table[itabl][1]-ent
                                               val.append(entry.time)
                                        val.append(diff_val)
                                        combined_val.append(val)
                                        y+=1
                                        mid_val_all+=diff_val
                                        just_val.append(val)
                        else:
                                print WARNING+"tripped on ["+END,entry.time
                                break
        ntuple_diff = ROOT.TNtuple(name,name,"time:value")
#        print len(just_val)
        if not(x==0):
                for data in just_val:
                        ntuple_diff.Fill(data[0],data[1])
#                ntuple_diff.Write()
```

```
                srch_all=mid_val_lp/x
                srch_lp=mid_val_all/y
                #-----------------------------------------------HVA
                if CrateNr=="3" or CrateNr=="4":
                        ntuple_srch_A.Fill(srch_all)
                        th1f_A_all.Fill(name,srch_all)
                        ntuple_srch_A_1.Fill(srch_lp)
                #-----------------------------------------------HVB
                elif CrateNr=="1" or CrateNr=="2":
                        ntuple_srch_B.Fill(srch_all)
                        th1f_B_all.Fill(name,srch_all)
                        ntuple_srch_B_1.Fill(srch_lp)
                #-----------------------------------------------HVC
                elif CrateNr=="5" or CrateNr=="6":
                        ntuple_srch_C.Fill(srch_all)
                        th1f_C_all.Fill(name,srch_all)
                        ntuple_srch_C_1.Fill(srch_lp)
                else:
                        print CrateNr," error nr 3"
                if srch_all>5 or srch_all <-5 or srch_lp>5 or srch_lp <-5:
                        print YELLOW+name," === suspicous data: "+END,srch_
#                print 'suma wejsc w diffT ',x,'srednia wartosc bledu: ',m
        else:
                print'error == 2: ',name
                print '-----------------------------------------------'
                error_list.append('blad ==2 '+name)
        x=0
        CellNr+=1
th1f_A_all.LabelsOption(">","X")
th1f_A_all.Write()
th1f_B_all.LabelsOption(">","X")
th1f_B_all.Write()
th1f_C_all.LabelsOption(">","X")
th1f_C_all.Write()
ntuple_srch_A.Write()
ntuple_srch_A_1.Write()
ntuple_srch_B.Write()
ntuple_srch_B_1.Write()
ntuple_srch_C.Write()
ntuple_srch_C_1.Write()
print YELLOW+ 'all jobs done'+END
#for data in diff_table:
#        for d in data:
#                x+=1
#                print d
#        print x,'-=-'
```