CRIB
rdf2root
CRABAT
introductory manual

Yuji Sakaguchi, March, 2016

Introduction

I assume that you want to know how to convert rdf files into root files with rdf2root, or you have no knowledge of CRABAT though you have knowledge of ROOT and C++.

I assume that the environment of PCs for the conversion is that of analys2, which is used for online analysis in CRIB.

analys2 is shared by the member of CRIB, so you should take backups many times if you convert files with analys2.

CRABAT is compressed as “analysis_code.tar”, which is located in the directory including this manual.

I assume you finish making the environment of CRABAT. In other words, I assume you finish “make”.

This manual is made for beginners. Therefore you should read README and files directory if you want to know details.

Contents

- Convert rdf file into root file
  - rdf2root.C…designate directory, adjusting as your setup
  - sql file…location, modifying links,
  - Perform rdf2root
- Analyze root file with CRABAT
  - Outline of files in CRABAT directory
  - Adjust files
  - Perform CRABAT
  - Avoid waste of time
  - Make Analyzer.h

Convert rdf file into root file

Convert rdf file into root file

In experiments with CRIB, data are taken as rdf files because of the environment of Anapaw.

You are required to convert rdf files into root files in order to analyze with ROOT.

There is a macro, “rdf2root.C” in analys2. The conversion is performed with this macro.

I explain how to convert rdf files into root files in the environment of analys2 below.

Convert rdf file into root file

designate the directory for rdf files and the directory for root files

In the directory of “~/physics/rdf2root”, directories including rdf2root.C were made for each experiment.

This picture shows the example.

In ~/physics/rdf2root

For your experiment, you should make your directory by copying another directory.

View ~/physics/rdf2root/rdf2root_be10alpha/rdf2root.C. I use this file as an example.

Convert rdf file into root file

designate the directory for rdf files and the directory for root files

Change the sentence of “inPath” into your directory which includes rdf files.

Change the sentence of “outPath” into your directory planed to contain converted root files.

directory including rdf files

```
110 // Path where the evt and root files are going to be located
111 // CHANGE ME FOR YOUR COMPUTER!
112   char inPath[100],outPath[100];
113   sprintf(inPath,
114         "/home/crib/exp/be10_a/users/crib/rdf"
115         ); // where the rdf files are at
116   sprintf(outPath,
117         "/home/crib/physics/rdf2root/rdf2root_be10alpha/root_be10"); // wher
    e to output the root data
```

directory planed to contain converted root files

A part of rdf2root.C

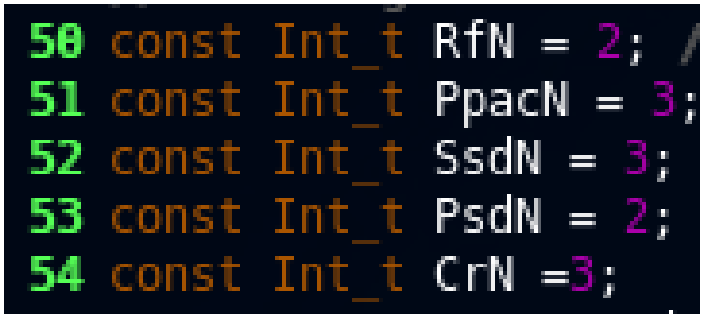
Convert rdf file into root file

Adjust for the setup of your experiment

The number of SSD, PSD or PPAC depends on setups.

There is a part you need to adjust for your experiment in rdf2root.C.

Adjust the part shown in the picture as your setup.



number of RF on TDC
number of PPAC
number of SSD
number of PSD
coin, single, pileup

A part of rdf2root.C

coin…event of coincidence between PPAC and PSD
single…event derived only from PPAC
pileup…event that over 2 particle hit a detector at the same time

As valuables in Analyzer.cxx
cr[0]…judge coin
cr[1]…judge single
cr[2]…judge pileup

Convert rdf file into root file

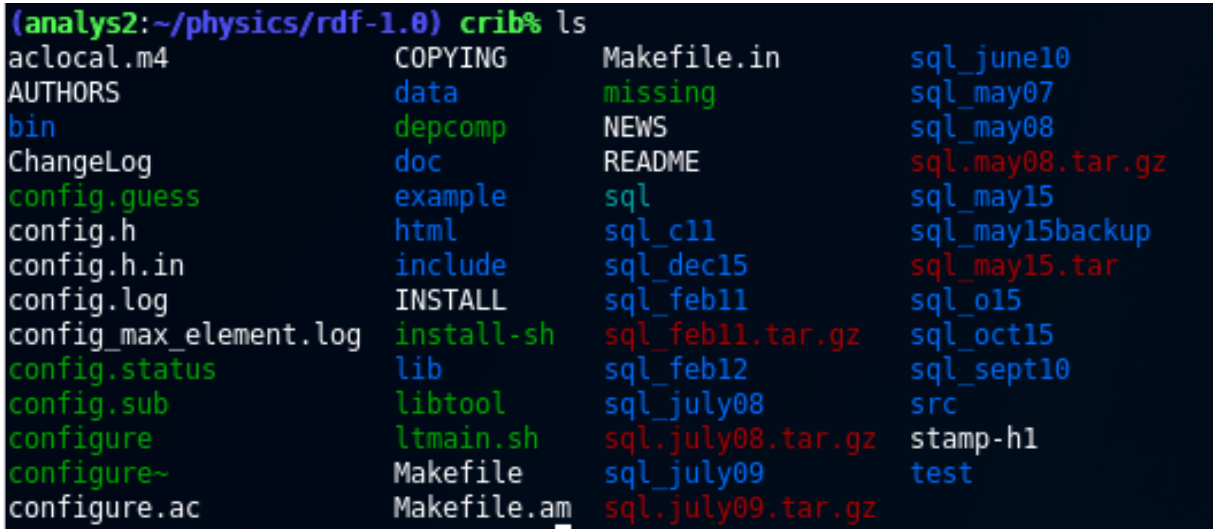
Location of sql file

You are required to write parameters such as those for calibration in sql files.

I explain how to write the parameters in sql files from here.

There are some directories for each experiment in ~/physics/rdf-1.0.

You should make a directory for your experiment by copying another directory.



In ~/physics/rdf-1.0

Convert rdf file into root file

Modify a link for sql file directory

In ~/physics/rdf-1.0, input “ls –l” into your terminal. After that certain information like this picture shows up.

```
lrwxrwxrwx 1 crib crib      9 2016-02-28 16:24 sql -> sql_may15
```

Part of terminal after “ls –l”

“sql” is a symbolic link to “sql_may15”. Link “sql” to your directory. You can do that by inputting “unlink sql” and “ln –s NAME OF DIRECTORY sql”.
After finishing linking, you are required to modify parameters in sql files. Enter your directory.

Convert rdf file into root file

Overview of sql files

The sql files you should focus on are as below.

- ppac.sql… sql file for PPAC
- ssd.sql… sql file for SSD
- psd.sql… sql file for PSD

I treat psd.sql as an example to explain how to modify parameters in sql files.

View psd.sql.

Convert rdf file into root file

Part of sql file you should change

Parts of psd.sql are written as these pictures. 9 parameters are arranged in the upper picture. The parameters you are required to change are located as the lower picture.

```
 9 CREATE TABLE `PSD0` (
10   `detector channel`       int NOT NULL,
11   `adc map`                integer,
12   `adc offset`             double,
13   `adc gain`               double,
14   `adc histogram channel`  integer,
15   `tdc map`                integer,
16   `tdc offset`             double,
17   `tdc gain`               double,
18   `tdc histogram channel`  integer,
19    PRIMARY KEY (`detector channel`)
20 );
```

You should understand what the numbers in the lower picture reveal. The order of the parameters in the upper picture corresponds to that in the lower picture. For example, “adc gain” shows up as the fourth parameter. Then, “adc gain” corresponds to “1.0”, the fourth parameter in the lower picture.

```
37 INSERT INTO PSD0 VALUES ( 0,  0, 0.0, 1.0,  0,  160,  0.0, 1.0,  0);
38 INSERT INTO PSD0 VALUES ( 1,  1, 0.0, 1.0,  1,  161,  0.0, 1.0,  1);
39 INSERT INTO PSD0 VALUES ( 2,  2, 0.0, 1.0,  2,  162,  0.0, 1.0,  2);
40 INSERT INTO PSD0 VALUES ( 3,  3, 0.0, 1.0,  3,  163,  0.0, 1.0,  3);
41 INSERT INTO PSD0 VALUES ( 4,  4, 0.0, 1.0,  4,  164,  0.0, 1.0,  4);
42 INSERT INTO PSD0 VALUES ( 5,  5, 0.0, 1.0,  5,  165,  0.0, 1.0,  5);
43 INSERT INTO PSD0 VALUES ( 6,  6, 0.0, 1.0,  6,  166,  0.0, 1.0,  6);
```

Same order

Part of psd.sql

Convert rdf file into root file

gain and offset in sql file

Energy is conventionally calculated with the equation of
Energy=gain × (channel-offset) in CRIB group.

Change these gain and offset into the values meeting your purpose in psd.sql, ssd.sql and ppac.sql.

Making the environment for rdf2root is very laborious. You need to install MySQL and ROOT adjusted to MySQL, and debag those. Most of you possibly use rdf2root in analys2 because it works well.

It is possible to write your calibration parameters in the sql files and make values in root files calibrated. However if the calibration parameters are wrong, you need to go to the CRIB monitor room and convert again. It's a waste of time.

Therefore if you decide to use rdf2root in analys2, fixing gain=1 and offset=0, converting rdf files of raw data into root files of raw data, and calibrating on CRABAT are wise way.

Convert rdf file into root file

Update MySQL after changing paramers

Be careful to the correspondence between values and valuables such as detector channel, adc map, adc histogram channel, tdc map and tdc histogram channel.

Guess the rule of the order by reading the sql files, looking up the VME rack in CRIB monitor room and looking up Anapaw mapping.

After changing the values in the sql files, you need to update MySQL.

If you change the values in ssd.sql, input “mysql –u crib –p < ssd.sql” in your terminal. If the sql file you change is another, change “ssd.sql” in the command above into the name of the sql file you change.

You are required to input the CRIB password after the command above.

Please ask the password of CRIB members.

Updating is completed after inputting the password.

Convert rdf file into root file

Perform rdf2root

After adsuting rdf2root.C and the sql files, you can convert.

Move to your directory, which contains rdf2root.C, input “~/physics/root/bin/root –l”.

If a ROOT is not optimized to the MySQL environment, you can not convert.

The command above starts the ROOT, which is optimized to the MySQL environment.

After starting the ROOT, input “.x rdf2root.C+(THE NUMBER OF RDF FILE YOU WANT TO CONVERT)”

The picture explaining the behavior of converting shows up in the next page.

Convert rdf file into root file

Perform rdf2root

```
Channel:   Map:     Gain:   Offset:  Histogram Map:
       0    116         1         0              0
Bit-mask = 0x000000000000100000
         = 00000000000000000000000000000000 00000000000000000000000000000000 000
00000000000000000000000000000000 00000000000000000001000000000000
TDC calibration:                                  0
Channel:   Map:     Gain:   Offset:
       0      0     0.122         0)0000000
Bit-mask = 0x0001
[illegible]
OpenFileUncompressed: Opening file /home/crib/exp/be10_a/users/crib/rdf/0100.rdf
File info:
Number   : RUN-0100
Start    : START => 17:50:31
Stop     :  STOP => XX:XX:XX
Time     : Print -> XX:XX:XX
Date     :  28-May-15
RemarkA  : continue
RemarkB  :

DecodeFera: ERROR - subaddress and pattern-word mismatch
DecodeFera: ERROR - subaddress and pattern-word mismatch
DecodeFera: ERROR - subaddress and pattern-word mismatch
DecodeFera: ERROR - subaddress and pattern-word mismatch
DecodeFera: ERROR - subaddress and pattern-word mismatch
DecodeFera: ERROR - subaddress and pattern-word mismatch
DecodeFera: ERROR - subaddress and pattern-word mismatch
DecodeFera: ERROR - subaddress and pattern-word mismatch
DecodeFera: ERROR - subaddress and pattern-word mismatch
DecodeFera: ERROR - subaddress and pattern-word mismatch
DecodeFera: ERROR - subaddress and pattern-word mismatch
DecodeFera: ERROR - subaddress and pattern-word mismatch
```

Terminal in which the conversion is being processed

Many ERROR show up, but you don’t have to solve those.

Please wait until the conversion is finished.

If the conversion is finished, the directory which was designated in outPath in rdf2root.C contains converted root file.

Convert rdf file into root file

Perform rdf2root automatically

```
5  #include "rdf2root.C"
6
7  void run_rdf2root()
8  {
9  //  TTree *t=new TTree("ri
10 int run_no;
11   bool status;
12
13  for (int i=11 i<14;i++)
14    {
15
16      run_no=i;
17      status=true;
18      //a loop to skip junk
19 //      if(run_no==5)
20   //        status=false;
21 //      TTree *t=new TTree(
22      if(status)
23        rdf2root(run_no);
24
25
26    }
```

The first number of the rdf file

The last number of the rdf file

Part of run_rdf2root.C

Basically there are tens of or hundreds of rdf files in one experiment.

Inputting commands for each rdf file takes you long time. It's a waste of time.

run_rdf2root.C enables you to convert automatically.

The run_rdf2root is located in the directory containing the rdf2root.C

You can designate the region of rdf files you want to convert in the run_rdf2root.C like the picture.

Start the ROOT, and input ".x run_rdf2root.C+". Then the conversion is performed automatically.

Convert rdf file into root file

Reference

The Web page made by Daid Kahl, who belonged to CRIB group is very helpful to construct the environment of the rdf2root in your PC.

http://www.cns.s.u-tokyo.ac.jp/~daid/physix/rdf2root-in-Linux.html

Analyze with CRABAT

Analyze with CRABAT

CRABAT is a system for analysis in which ROOT is assumed to used.

If you use CRABAT, many wastes of time are saved and you can spare more time for your research.

From here I explain how to use CRABAT for beginners.

Analyze with CRABAT

In nuclear experiments some physical quantities are recorded to some files. For example, calibration is measured five times, so there are five calibration files, and scattering is recorded to twenty files.

If you don’t devise, you write a code for analysis of scattering, compile, perform the executable file twenty times from your terminal. This is a waste of time. The time for your research is limited, so you should avoid it.

In addition, there is a case where you have a mistake in your code, and perform the executable file twenty times after modifying those. This is also a waste of time.

Analyze with CRABAT

With a function of CRABAT, analysis for data files whose purpose of measurements is same

is performed automatically. CRABAT brings out the best ability of your PC during analysis.

I explain how to use this function from here.

Analyze with CRABAT

Outline of the roles of CRABAT files

I explain the outline of the roles of CRABAT files.

- run.h definition of histograms
- run.conf designation of directory containing root files, designation of groups composed of raw data files which have the same purpose for measurements
- run.cxx definition of groups composed of raw data files which have the same purpose for measurements
- Analyzer.h definition of valuables, etc
- Analyzer.cxx calcuration of physical quantities, filling valuables to histograms

Please read README and the files if you want to know details.

Analyze with CRABAT

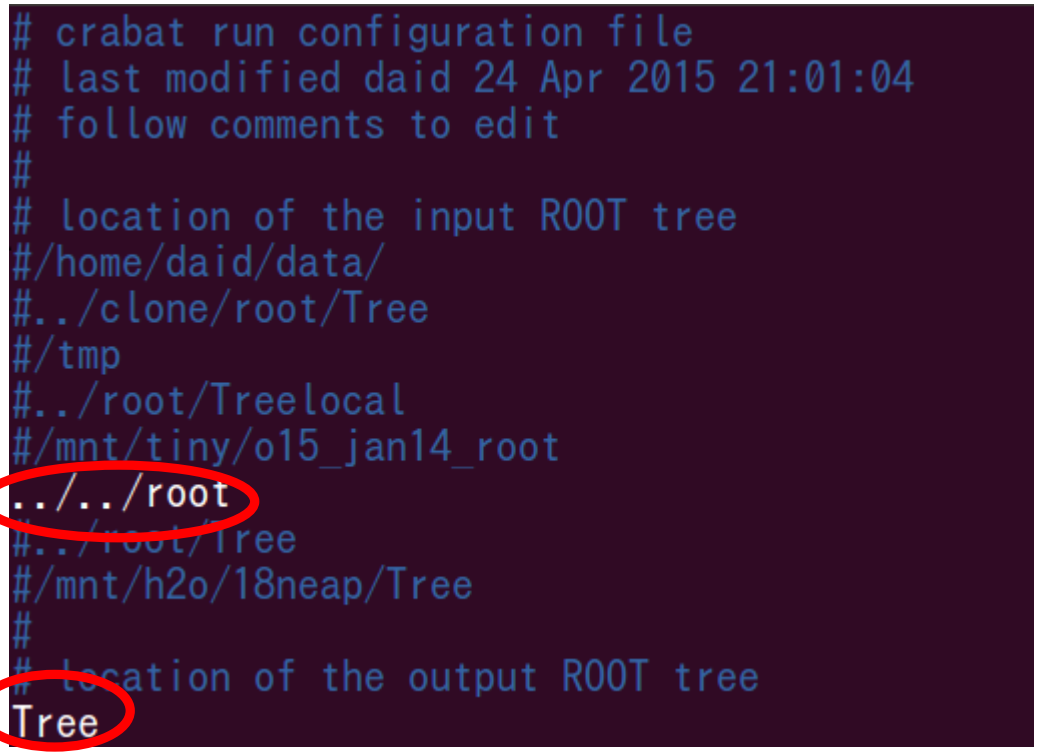
Designate data file directory

You are required to designate your directory containing root files planed to be analyzed and a directory you plan to send analyzed root files to. (With CRABAT, the result of analysis is recorded as root files.)
Edit run.conf. The sentenced are like this picture.
Change the parts like this picture as your environment.

Part of run.conf

Remove the comment out, and designate your directory containing raw root files.

Designate a directory you plan to send analyzed root files to.



Analyze with CRABAT

Data files which have the same purpose for measurements

In the lower side, the sentences are like this picture.

```
Hflag=1
#       0: alpha calib (PPAC or SSD)
#       1: physics (alpha scattering)
#       2  F3 beam measurement
#       3: background
#       4: F2 beam measurement
# set run header value here
```

Part of run.conf一部抜粋

This is my environment.
“alpha calib (PPAC or SSD)” means “calibration of PPAC and SSD with alpha sources”, and the ID of data group whose purpose is “alpha calib (PPAC or SSD)” is set as 0. “physics (alpha scattering)” measns “measurement of alpha scattering” and, the ID of data group whose purpose is “physics (alpha scattering)” is 1.

I explain how to designate these data groups and IDs in the next page.
I explained above that analysis for data files whose purpose of measurements is same is performed automatically with CRABAT. You can choose which data group you analyze with this “Hflag”. In this picture, “Hflag = 1” is written. Therefore the data group of “physics (alpha scattering)” is analyzed. You can choose by setting the number of “Hflag”.
Please determine which number corresponds to which data group on this step. This relation is required when you designate data groups and IDs.

Analyze with CRABAT

Data files whose purpose is same

You should write grouping data files and IDs in run.cxx.
In an experiment data of measurement are basically recorded about every fixed some hours. (For example two hours.) Every datum is recorded as a rdf file in the CRIB group. The name of the rdf file is determined according to the order. If the file is 100^{th}, the name is “0100.rdf”. The example is as follows.

0001.rdf～0015.rdf・・・calibration

…

0096.rdf～0122.rdf・・・elastic resonant scattering with alpha particles (capital measurement)

…

0130.rdf～0132.rdf・・・calibration after this experiment

example of relation between order of rdf file and measurement

For example , the capital measurement corresponds to data of 0096.rdf~0122.rdf.
Look up which rdf file corresponds to which purpose by reading the log file of your experiment. The relation of **rdf file ↔purpose of measurement ↔ID you determine** should be clarified. Then you are required to write this information in run.cxx View run.cxx

Analyze with CRABAT

There is a part in run.cxx like this picture. Write the relation of **rdf file ↔purpose of measurement ↔ID you determine** here. In CRABAT rdf files which do not meet the purpose of the measurement are avoided analyzing automatically. Hence designate rdf files which do NOT meet the purpose of the measurement. I use the capital measurement as the example and explain how to write.

Write the ID

The files of the capital measurement are 0096.rdf～122.rdf. Hence the other files are written.

Write “physics” which reveals the capital measurement.

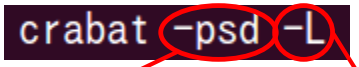
```
char file_run[150];
// to exclude a run, you can do it like this
if (flag_Header) {
  if (header_no == 0){
    if (!(run_no<=16 || run_no>=129)){
    cout << "Run number " << run_no << " is not alpha calibration...skipping" << endl;
    return 0;}
  }
  if (header_no == 1){
    if (run_no>=123 || run_no<=95){
    cout << "Run number " << run_no << " is not physics...skipping" << endl;
    return 0;}
  }
  if (header_no == 2){
    if (!(run_no==18 || run_no==20 || run_no==22 || run_no==23 || run_no==24 || run_no==25 || ru
n_no==26 || run_no==27 || run_no==28 || run_no==29 ||
      run_no==30 || run_no==31 || run_no==32 || run_no==33 || (run_no>=78 && run_no<=84 ))){
    cout << "Run number " << run_no << " is not F3 beam measurement...skipping" << endl;
    return 0;}
  }
  if (header_no == 3){
    if (run_no>=129 || (run_no>=95&&run_no<=122) || run_no<=88){
    cout << "Run number " << run_no << " is not background run...skipping" << endl;
    return 0;}
  }
  if (header_no == 4){
    if (!(run_no==17 || run_no==19 || run_no==21)){
    cout << "Run number " << run_no << " is not F2 run...skipping" << endl;
    return 0;}
  }
}
```

Part of run.cxx

Analyze with CRABAT

Perform CRABAT

After you finish designating the directories containing raw root files, the directory the analyzed root files are sent to, and **rdf file ↔purpose of measurement ↔ID you determine**, you can use the function of CRABAT. CRABAT brings out the best ability of your PC, and analyzes the data automatically .
Input like this picture in the CRABAT directory. The option is also explained.

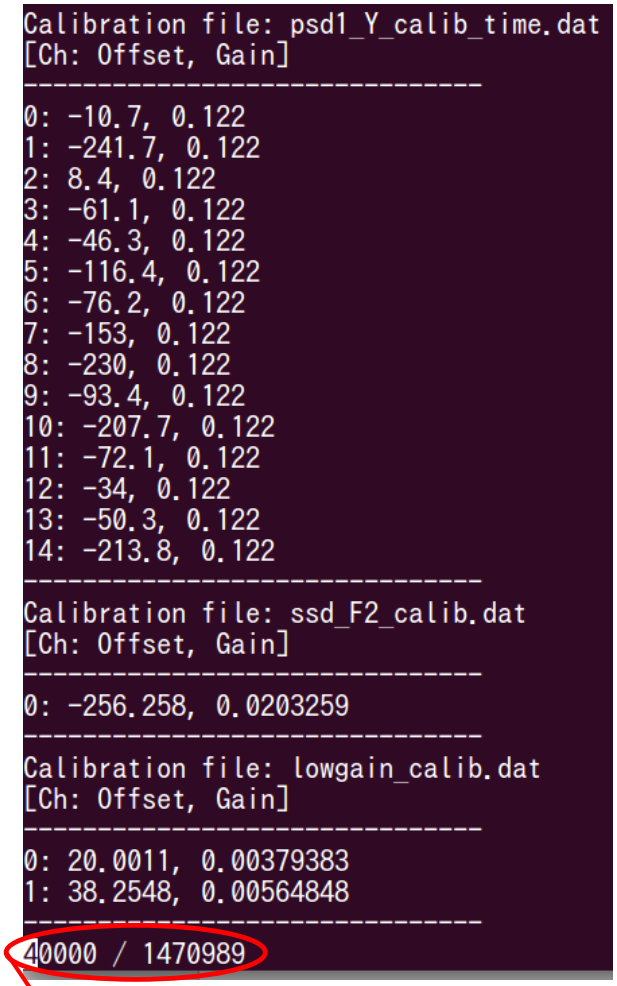


You can choose parts written in Analyzer.cxx by these options.
p…PPAC
s…SSD
d…detailed data
The details are written in README and Analyzer.cxx.

By inputting “-L”, CRABAT automatically analyzes rdf files whose purpose is same and whose ID corresponds to ID designated in “Hflag”.
PCs have cores, and the number of the cores is fixed according to PCs. By inputting “-L”, CRABAT automatically analyzes with all the cores used at the same time.

Analyze with CRABAT

Perform CRABAT



After inputting "crabat –psd –L" in your terminal, the terminal becomes like the picture.

All the core process Looping simultaneously.

CRABAT automatically selects the rdf files which meet the purpose of the measurement.

As Loop is processed, the number is increased.

Analyze with CRABAT

To avoid a waste of time

After adjusting the files for CRABAT, the process of analysis is defining histograms in run.h, calculation in Analyzer.cxx and filling valuables in histograms.

During analysis, you should keep in mind that most of the events are derived from noises, pedestals and something like those. Those are meaningless. Therefore it is a terrible waste of time for you to conduct functions for these meaningless events. Loop should be quit at the time when an event is judged as meaningless, and the next event should be Looped. In order to do that, you should use "continue" in Analyzer.cxx. The picture is an example explaining how to use "continue".

```
// Check if each PPAC is hit with valid event
PpacIsHit0 = false;
if (PPAC0_time[0]>0. && PPAC0_time[1]>0. && PPAC0_time[2]>0. && PPAC0_time[3]>0.) PpacIsHit0=true;
PpacIsHit1 = false;
if (PPAC1_time[0]>0. && PPAC1_time[1]>0. && PPAC1_time[2]>0. && PPAC1_time[3]>0.) PpacIsHit1=true;

    if(PpacIsHit0!=true || PpacIsHit1!=true)
    {
            continue;
    }
```

Part of Analyzer.cxx

If a value of PPAC0 is strange or a value of PPAC1 is strange, continue is performed and Loop is quitted. The next event is Looped.

Analyze with CRABAT

How to create Analyzer.h

The kind of Physical quantities depends on a setup. Hence definition of valuables in Analyzer.h depends on the setup. Looking up which physical quantity corresponds to which valuable in Analyzer.h and writing the relation in Analyzer.h are wastes of time and your vitality.

You can automatically create an optimized Analyzer.h for your experiment.
Input “root NAME OF ROOT FILE”, and enter ROOT system.
If a Tree name is riken_exp, input “riken_exp->MakeSelector (“Analyzer.h”)”.
Then Analyzer.h and Analyzer.C are created in the directory where you entered ROOT system.
This Analyzer.h is optimized to the root file. Please use this Analyzer.h

（Refer to　ftp://root.cern.ch/root/doc/21ExampleAnalysis.pdf)

Analyze with CRABAT

Then change and write Analyzer.h and Analyzer.cxx, etc as you like.
CRABAT contains other functions. Try to use the functions by reading README and the files.

I hope this manual is helpful for your research.