My ROOT version: 6.22/06

ROOT6 is recommended. Some syntax is only supported by C++11 and after.

FilePath:

Assumed that there are two directories **LST** and **rootfiles** under same directory.

Like: Run01 under which there are LST and rootfiles

**LST**: store .lst files of tof and beta

**rootfiles**: FP\_092Rb@700\_191.root // raw tof

FP\_092Rb@700\_191\_dcorrect.root // tof after drift correction

tbeta\_beta\_tree.root // beta-beta coincidence

tbeta\_tof\_tree.root // beta-tof coincidence

TOFMarker\_FP\_092Rb@700\_191.txt // TOF marker file

beta-tof.log // beta-tof coincident event log

cutfile.root // cut for beta-beta coincident event

Important containers:

Ref:

Time of flight: double tof\_ref\_cento; double tof\_ref\_cento\_err;

int q\_ref = 1; // charge

double bref; // b value

int laps\_ref; // laps of ref

double t0; // t0, default = 130 ns

Ion X:

double tof\_x\_cento[10]; double tof\_x\_cento\_err[10]; // support 10 peaks

int q\_x; // charge of ion X

int laps\_x; // laps of ion X

int NumOfPeaks = 1; // number of peaks for multipeak fitting

funcS\* fs; // sample function

funcN\* fext; // sample function

TCutG \*mycut[5]; // cut

RejectionList rejectlist; // rejection list

TF1\* tem\_func; // current fitting function **global handler**

**// reject multi-hit**

double **Tof\_veto**=0; //[ns] veto reject hits of the same sweeep in the window

int **Tof\_Channel2Read**=1; // 🡨 load data from which channel

Useful methods：

Double\_t\* SearchAME(int Anum = 0, const char\* element = "", bool verbose = false, const char\* filename = "AME2020.txt")；

Double\_t\* SearchMolMass(string formula = "12C;1 16Cl;4", bool verbose = false)；

double Lifetime2Halflife(double Inlivetime\_ms = 150)；

double Halflife2Lifetime(double Inhalflife\_ms = 150)

**Convert micro-amu to keV**

double\* MassExcess(double mass\_value, double mass\_value\_err, bool verbal = true);

Decode and drift correction:

**Refer to Drift correction panel.docx for detail.**

Mass analyze (use of preview window):

1. .L preview5.C+
2. Set **Tof\_veto and Tof\_Channel2Read to define a time window to reject multi-hit and the channel for loading data**

Tof\_veto = 0; 🡪 no reject

Tof\_Channel2Read=1; 🡪 read data from channel 1

3.

void autopreview(string PATH = "../rootfiles/", char origin\_correct = 'o', double \_intime\_ref = 0)

**run**: autopreview(“filepath into rootiles/”, ”filename of tof.lst without .lst”，‘o FOR RAW TOF SPECTRUM OR c FOR DRIFT CORRECTED SPECTRUM’, reference peak center in ns)

It will load the .root file (before drift correction) or \_dcorrect.root file(after dirft correction) basing on the list of file get in LST folder.

If there is only one file of tof.lst in “LST” 🡪 open corresponding file in “rootfiles” folder.

If have degrader scan files in “LST”, 🡪 open the one with shorter name or “smaller” name.

For example:

If [J\_097Ag@690\_78.lst](mailto:J_097Ag@690_78.lst) 🡨 open [J\_097Ag@690\_78.root](mailto:J_097Ag@690_78.root) or J\_097Ag@690\_78\_dcorrect.root

[J\_097Ag@690\_78-S01.lst](mailto:J_097Ag@690_78-S01.lst)

[J\_097Ag@690\_78-S02.lst](mailto:J_097Ag@690_78-S02.lst)

If [J\_097Ag@690\_78-S01.lst](mailto:J_097Ag@690_78-S01.lst) 🡨 open [J\_097Ag@690\_78-S01.root](mailto:J_097Ag@690_78-S01.root) or

J\_097Ag@690\_78-S01\_dcorrect.root

[J\_097Ag@690\_78-S02.lst](mailto:J_097Ag@690_78-S02.lst)

Temporarily relocate the tof.lst in “LST” can help to auto open the correct file. **OR** using the preview() function to specify the name of file to open. By default, peak center is not needed to input manually, unless it drifted too much away from the value given in .lst

1. **Option:** void preview(string PATH = "../rootfiles/", string filename = "", char origin\_correct = 'o', double eje0 = 0, double eje1 = 0, double tof\_ref = 0)

**run:**

preview(“filepath into rootiles/”, ”filename of tof.lst without .lst”，‘o FOR RAW TOF SPECTRUM OR c FOR DRIFT CORRECTED SPECTRUM’, ejection tag0 if known, ejection tag1 if known, reference peak center if known)

just using default values for following arguments if tof.lst file in “LST”. Corresponding arguments are got for the tof.lst automatically.

**If everything goes well:**

A window pops out => pad1: full spectrum of tag0; pad2: peak of reference ion of tag1; pad3: full spectrum of tag1; pad4: tof VS sweeps(2D) of reference peak

If the file header in tof.lst is not complete or have special character occasionally in some case, spectrum can not be loaded correctly.

**Solution1:** specify peak center and keep eje0=0, eje1=0; if only the peak center is not off.

**Solution2:** if eje0, eje1 and peak center information is not complete. Input eje0, eje1 and

peak center in ns

**Solution3:** Copy the information from another .lst file

**Solution4:** if eje0, eje1 and tof\_ref are input correctly, but the display of spectrum is not

correct. The most possible reason is the eje0 and eje1 are input reversely. In this

case just use **execute ReverseEjeTime()** and then run preview() again.

1. **Option:** PrintInfo(); // check initial setting of Ref and ion X
2. **Option:**

If setting is not ok for **reference ion**, use:

void ResetRef(double mass\_ref = -1, double mass\_ref\_err = -1, int \_qref = -1, int \_laps\_ref = -1, double b\_ref = -1, double tofref = -1, double tofref\_err = -1)

void ResetRefMassTime(int Anum = 0, const char\* elename = " ", int \_qref = -1, int \_laps\_ref = -1, double b\_ref = -1, double tofref = -1, double tofref\_err = -1)

// feed with -1 for terms you don’t want to change

**Set atomic mass as reference**

// set mass: ResetRefMassTime(85,”Rb”); ResetRefMassTime(133,”I ”); // single letter element

**Set molecular mass as reference**

void ResetRefMassTime(string formula = "12C;1 16O;2", int \_qref = -1, int \_laps\_ref = -1, double b\_ref = -1, double tofref = -1, double tofref\_err = -1)

**Or access to containers directly:**

q\_ref; laps\_ref; bref;

If setting is not ok for **X ion**, access to containers directly:

q\_x; laps\_x;

**Set t0 value by t0= xxx; and the uncertainty of it err\_t0 = xxx; directly.**

**Set uncertainty of b value by bref\_err = xxx;**

**Change the default setting of display when execute preview(……..):**

void para\_modifier(int FullWidth\_ns = 36000, int binsFW = 10000, int refHalfWidth\_ns = 150, int bins\_refSingle = 200, int bins\_event = 100, int whichtag2D = 1)

( the width of histogram at pad1 and pad3, bin number used for histogram of pad1 and pad3,

Half width of histogram at pad2 [total width = 2X] , number of bins of histogram at pad2, number of event or sweeps per bin of X axis of 2D histogram, which tag to be shown in 2D histogram🡺 [1: tag1, 2: tag2 else: both] )

Note: refHalfWidth and refSingle determine the setting of Y axis of 2D histogram

void DisplaySetting() 🡪 show the current display setting

**After changing the setting of display, Run preview(…..) again to refresh;**

1. StartInteractionMode() // under interaction mode pad1 and pad2 will enable cursors

‘z’ => zoom in ( mouse left click to give a range)

‘x’ => zoom out

‘space’ => open a new histogram of tag0 (by giving a range on pad1); using function displayed on terminal to set number of bins to change bin width if necessary (just change the 2nd argument).

histo\_zoom\_in\_x(int tag = 0, int **bins** = 0, double histoL = 0, double histoR = 0)

‘l’ => switch between linear and log scale

‘double click + r’ => set ROI on pad1 to show counts // more descriptions later

‘double click + R’ => renew current ROI on pad1 // more descriptions later

‘s’ => sample a peak at pad4 (tag1); “NumOfPeaks” global variable to set number of peaks to fit after sample // more descriptions later

‘S’ => sample a peak at pad2 (tag0 usually) // more descriptions later

‘m’ => calculate mass of Xion // more descriptions later

‘c’ => compare tof and beta time stamp for beta-tof analysis // more descriptions later

Double click on pad1 => show list of possible mass candidates with up to triple charge at different laps

‘,’ => release fitting range for sampling fit //// more descriptions later

‘-‘ => minus sign shows the time difference between two continuous clicks of cursor.

‘e’ fit reference peak in h\_ref with “fext” curve by draw arrow to define the fitting range.

“fext->Norder2Set” default 0 order (left tail + gaussian center + right tail”; order =1 🡺 +1 right tail; order = 2🡺 + 1 left tail …. Using “NumOfPeaks” global variable to set the number of peaks to fit

‘E’ fit reference peak in h\_zoom\_x with “fext” curve by set a window by cursor ahead.

‘u’ after ‘f’ -> unbinned fitting by current fitting function

1. histo\_zoom\_in\_ref(int bins = 0, double tof\_ref = 0, double halfwidth = 0, double TAG = 1);

// create spectrum of reference peak on pad4

// usually just need to change ‘bins’ to get finer spectrum

1. **Extract tof of reference**

For convenience, only sampling fit is introduced. (Maybe giving more details on fitting in another introduction); **“fext” curve follows similar steps.**

🡺Click pad1 or pad2 to activate.

🡺For sampling at pad4: Press ‘s’ => drag to draw an arrow to give a sampling range

For sampling at pad2: first click to give a range, then press ‘S’.

// in case arrow is not activated automatically, you can get an arrow in toolbar

// change smooth level for sampling: fs->smoothlevel =4 (set to 4, default = 2)

After sampling, tof of reference is extracted

**Sampling at tag0 is also available:**

🡺using cursor clicking to define sampling range

🡺press capital ‘S’ to sample

1. Extract tof of Xion

🡺Mouse click to include a peak of interest.

🡺press ‘space’ to make a new histogram on pad2

🡺 histo\_zoom\_in\_x(int tag = 0, int **bins** = 0, double histoL = 0, double histoR = 0) // change bins number to adjust

🡺mouse click to give fitting range

🡺press ‘f’ to fit with sampling function

🡺repeat these two step to get stable

🡺option: if want to change fitting range, press ‘,’ to free fitting range(width for fitting will be fixed after 1st first fit after sampling)

After fitting, tof of ionX is stored automatically

For multipeak fitting:

🡺set NumOfPeaks = 2 ( if there is two peaks to fit)

🡺mouse click to give a range covering these peaks

🡺press ‘f’ to fit

🡺draw arrow from the submit of the 1st peak to its FWHM (both left and right are ok)

Repeat drawing arrows for all peaks to give initial conditions

🡺finishing fitting after giving all conditions

1. Mass calculate

if laps\_x and q\_x are what we want and it is single peak fitting:

🡺Press ‘m’ ==> done

Otherwise: make sure laps\_x, q\_x, and NumOfPeaks are current conditions before

🡺press ‘m’

Mark TOF and check mass of ionX quickly:

MarkTOF:

We have mass and tof of ref after sampling. Using following functions to mark tof on pad1

1. void MarkTof(double mass\_xx, int \_q\_x, const char\* IonName, int \_laps\_x = 0, int Tag = 0, bool renew\_current = false, bool renew\_all = false)
2. void MarkTof(int Anum, const char\* EleName, int \_q\_x, int \_laps\_x = 0, int Tag = 0, bool renew\_current = false, bool renew\_all = false)
3. void MarkTof(string formula, int \_q\_x, int \_laps\_x = 0, int Tag = 0, bool renew\_current = false, bool renew\_all = false)

2 and 3 more convenient when ID of ion is known or expected:

2 for single isotope:

91Rb =>

MarkTof(91,”Rb”,1);

// denote (Anum, “element name; with space for single letter cases”, charge , [default 0=> same laps as reference ion; -1=> determined by program; other=> specified by user] , [tag0=>draw on pad1;tag1=>draw on pad3], true=>remark current marker, true=> erase all marker to start from beginning)

3 for molecule:

Water =>

MarkTof(“1H;2 16O;1”, 1, -1); =>(“water’s formula”, single charge, laps determined by program)

Trick:

formula 3 work for element and molecule case;

using -1 as \_laps\_x input when you don’t know what the laps number should be. But bref is needed in this case.

Check mass candidate:

Double click at the peak on pad1 => list of candidates at different laps and charge state will be shown.

Set ROI to get counts:

Choice One: double click the peak on pad1 then press ‘r’ ; to correct current marker => press ‘R’;

set **ROI\_WIDTH** to change width of ROI when necessary

set **ROI\_INDEX=1**; and **ROI\_initial**=false to restart index counter;

choice Two: using function, if identification or mass is known, that will be more convenient because commands can be temporarily written to another txt file; copy and paste to use for another tof file. Especially for degrader scan.

void SetROI(double TCento, double ROI\_width = 30, int ROI\_index = 1, double TOffset = 0, bool showcount = false,int Tag=0);

//(peak center, ROI width is defaulted to 30 ns , index maximum is 40, offset from peak center, whether to shown how many counts in the ROI, Which tag)

void SetMassROI(double \_massxx, int \_q\_x = 1, int \_laps\_x = -1, int ROI\_index = 1, double TOffset = 0, double ROI\_width = 30, bool showcount = false, int Tag=0);

(mass, charge, \_laps\_x is defaulted to determine by program, ……same meaning as above)

void SetEleROI(int Anum, const char\* element, int \_q\_x = 1, int \_laps\_x = -1, int ROI\_index = 1, double TOffset = 0, double ROI\_width = 30, bool showcount = false, int Tag=0);

// single isotope case

void SetMoleculeROI(string formula, int \_q\_x = 1, int \_laps\_x = -1, int ROI\_index = 1, double TOffset = 0, double ROI\_width = 30, bool showcount = false, int Tag=0);

// molecule case

void ROIClear(); 🡺 to clear all ROI setting

Beta-tof coincidence:

Important containers:

int syn\_CH = 4; // beta-tof synchronizer channel

Long64\_t RIHalflive; // in nano-second

double Times2Halflive = 4; // looking for coincident event in 4times halflive

RejectionList rejectlist; //reject repeated records

TCutG \*mycut[5];

syn\_CH => change synchronizer channel in USB-MCS; directly set by giving value

Times2Halflive => default as 4 times RI halflife as window for beta-tof coincidence; set by giving value directly

Double detector\_effi =0.33; // default detector efficiency

Double Beta\_bg\_rate; // beta background rate

TH1D\* h\_beta\_accumulate = NULL; // histogram of beta decay time

Set and show beta-beta conditions:

void SetBeta\_E\_Calibrate\_para(double \_slope1=1,double \_intercept1=0,double \_slope2=1,double \_intercept2=0);

// calibration parameters; default: set Energy = ADC

void GetBeta\_E\_Calibrate\_para();

void SetCoinCondition(Long64\_t \_gate\_time=500, int \_gate\_adc\_low=0, int \_gate\_adc\_hi=1000);

// ( |beta1-beta2| window as coincident event, E window adc Min, E window adc Max);

void ShowCoinCondition();

void SetRIHalflife(double InMilliSecond = 150); // halflife time unit in ms

void SetRIHalflife\_byLifeTime(double Inlivetime\_ms=150) // unit is ms

procedures:

1. In principle, the beta.lst is detected when execute autopreview(); if preview() is used instead, **Run** LoadNewBetaFile("name of beta file with .lst"); to input the name of the first corresponding beta.lst file
2. Set key configuration in “BetaCoin.confg” file**(Remember to save the file after change):**

{

cout<<"\e[1;33m"<<"Loading setting parameters from BetaCoin.confg"<<"\e[0m"<<endl;

**ShapingTime[0] = 500; ShapingTime[1] = 500;** // shaping time in unit ns

**MCAThreshold[0] = 500; MCAThreshold[1] = 110;** // MCA threshold setting [0] -> ch 1; [1] -> ch2

**ADC\_max\_ch=16384;** // MCA maximum channel

**MakeBetaRawTree=true;** // Make beta raw tree or not. --> consume hard disk space, but save RAM and time; NOTE: ONLY available in !BetaFastMode !!!!!

**BetaFastMode=false;** // false -> load all beta.lst at once; true ->load only relevant beta.lst

**alphaMode=false;** // false -> beta-tof mode; true -> alpha tof mode

//Path\_beta\_file=FilePath+"../";

**ReadBetaOnline=false;** // false -> load data in LST as tof.lst ; true->data from the following path

**OnlineBetaFilePath="/mnt/beta\_driver/F11\_MRTOF/240619MT/usb/";**

**RecordOnly\_1st\_beta=false;** // just keep first positive beta corresponding to TOF.

**RecordOnly\_1st\_beta\_neg = false;** // by default(no matter set to true or false): using the average interval of the closet negative beta to Tof to evaluate the beta background rate

// set to "false" enable second method: beta counts / total observation time to evaluate the beta background rate; This result replaces the result by method 1

**Times2Halflive=10;** // get coincidence in time window [tof- Times2Halflive x Halflife, tof+ Times2Halflive x Halflife]

**SetRIHalflife(1180);** // expected half life of RI in unit ms

**SetBeta\_E\_Calibrate\_para(0.2159461,-24.7021,0.360014,-6.56918);** // Energy calibration parameters for Si1 and Si2; (slop1,offset1,slop2,offset2)

**b2tob1\_standard = 20;** // standard value of b2\_time - b1\_time depend on shaping time if shape t1 > shape t2 ==> t2-t1 <0

**SetCoinCondition(5000,0,16384,0,16384,b2tob1\_standard);** // beta-beta coincident condiction (Abs(time2-time1) window, ADC low 1, ADC high1, ADC low2, ADC high 2, center of histogram of time2 – time1); // time 1 -> Si1; time2 -> Si2

//\*\*\*\*\*\*\* display for beta raw histograms \*\*\*\*\*\*\*\*\*\*\*\*\*

**DownScale=1;** // for beta raw histogram display; get data per "DownScale" events --> set a larger value to speed up; "DownScale" ==0; --> no draw raw beta histogram

//\*\*\*\*\*\*\*\*\*\* Set ADC threshold to filter raw data from each Si \*\*\*\*\*\*\*\*\*\*\*

// data below threshold will be skiped and will not be in RAM

**PreSetADC1\_low\_Thres\_ch=1100;** //500; //1200

**PreSetADC2\_low\_Thres\_ch=200;** //190;//200

**binwidth\_fraction\_halflife=10;** // binwidth = halflife/10; for binwith of beta decay time spectrum; 🡪 larger value to get finer bins in beta decay time histogram

**BetaBetaFilter=true;** // false-> keep all beta-beta coin. event in time gate (Abs(time2-time1)); true-> keep only the beta-beta combination with time different approaching given time different value "b2tob1\_standard"

}

1. Giving the range of peak by click on pad1 or pad2 ( better to use ‘space’ to create a new histogram on pad2 first)
2. Press ‘c’

It will run automatically the following function in sequence:

Read\_Beta\_lst\_batch(xxxxx); 🡪 read data from .lst or tree

Compare\_Beta\_Beta(); 🡪 find beta-beta coincident candidates as coincident conditions

ShowBeta\_Beta\_Coin();🡪 how raw beta and beta-beta coincidence histograms

FindBetaTof\_Coin(xxxxx); 🡪 matching tof and beta

**It is possible to execute any of these functions independently as needed.**

**Trick:**

If thebeta-beta coincident condition, cut condition, or beta.lst file does not change,

**Compare\_Beta\_Beta() will refused to run again**. If need to execute it again, **using SetCoinCondition() to change the threshold a little bit**. ThenCompare\_Beta\_Beta() becomes executable.

**Old version procedures:**

1. ShowEjectionBeta(EJE0,EJE1)
2. MakeCut(); or LoadCut() [ can be skipped if “cut” is made already when analyze other peak; or use LoadCut() to load Cut history.

cut at beta E1 vs E2 2D histogram

cut at red points region on c\_eject canvas ( usually on pad1 and pad3) // use ‘n’ to skip pad2 and pad4

1. ShowDecayHisto(mycut, &rejectlist) // beta decay half life shown in pad4 of c\_beta\_beta [just an option to run here]
2. **Option:**

ScanBetaTof\_tree(TCutG\*\* \_usecut=NULL, string speciesname="", RejectionList\* \_rejectlist=NULL,bool IsSave=false, bool IsAdd=true);

//(mycut, ”anyname”, &rejectlist, true to save to beta\_tof.log, true to recreate beta\_tof.log)

If save to file, easy to find any repeated event belonged to different ion. To exclude these events, add them to rejectlist by🡺

rejectlist.Add( global time stamp TOF, global time stamp time of Si\_1, and that of Si\_2)

1. GenerateRejectList(tbeta\_tof,mycut,&rejectlist); //=>copy this to terminal to generate reject list
2. ShowDecayHisto(mycut, &rejectlist); // => refresh after new reject list is created

New version procedures:

1. Make cut to remove ejection noise from canvas “c\_beta\_coin\_and\_raw” at pad1 and pad3 by:

MakeBetaCut\_Or\_FindSignal(Long64\_t \_TimeRelative = -1000, double \_EkeV = -1000, int WhichChannel = 1, bool GetCutStatus = false, TFile\* RootfilePtr = nullptr, int Clear0\_Recreate1\_Add2 = 1)

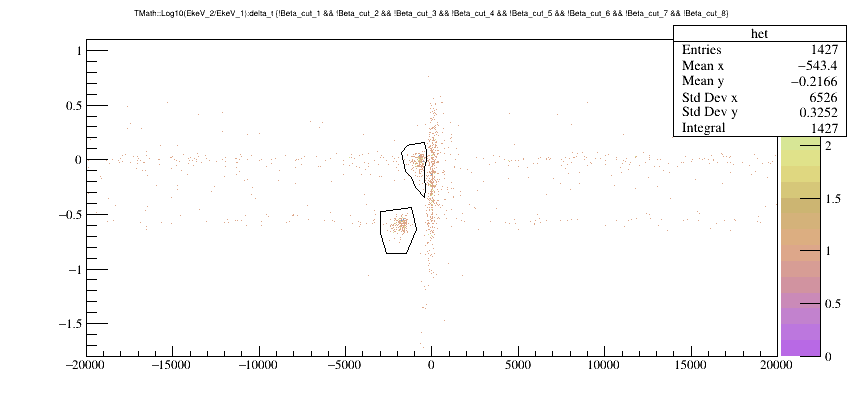
MakeBetaCut\_Or\_FindSignal(-1000, -1000,1,false,fin); 🡪 for recreating and make first cut

MakeBetaCut\_Or\_FindSignal(-1000, -1000,1,false,fin,2); 🡪 for second cut and after

MakeBetaCut\_Or\_FindSignal(-1000, -1000,1,false,fin,0); 🡪 remove cut information

**Using: LoadBetaCut(fin)** to **load cut exist already**

1. Repeat step 3 and step 4
2. Make cut manually at Log10(EkeV\_2/EkeV\_1) vs delta\_t histogram



cut1

cut2

And rename the cut such as “cut1” , “cut2”,

Change the name of x axis and y axis of cut by:

Like: cut1->SetVarX(“delta\_t”); cut1->SetVarY(“Tmath::Log10(EkeV\_2/EkeV\_1)”);

1. **Run:** ShowDecayHisto(NULL,NULL,"IonA;index IonB;index", **"!cut1 && !cut2 && delta\_t>-1000 && delta\_t<500”**); // "!cut1 && !cut2 && delta\_t>-1000 && delta\_t<500” 🡪 is cut condiction to remove extract noise.

**Decay spectrum and fitting of it will show.**

1. Option: FitBetaOption(bool \_fit\_vb, bool \_fit\_efficiency, bool \_fit\_tao = true); // to determine whether to fit beta background rate, whether to fit detector efficiency, and whether to fit beta decay constant (half life related)
2. Option:

void showEnergyVsDecayTime(string GeneralSelection = "", int Draw1Si1\_2Si2\_3All = 1);

// GeneralSelection <-- put the same cutting conditions as that put in ShowDecayHisto()

// Draw1Si1\_2Si2\_3All : 1 -> draw Si1 only; 2 -> draw Si2 only; 3 -> draw Si1 and Si2

This equivalent to scan the threshold of Energy. But the low limit of the threshold seen here is limited by

**PreSetADC1\_low\_Thres\_ch=1100;** //500; //1200

**PreSetADC2\_low\_Thres\_ch=200;** //190;//200

Specify in “BetaCoin.confg”

Set smaller value to allow more low energy data but consume more RAM;

**Repeat step 3 and step 4 if BetaCoin.confg file is changed**

1. **Option:**

**Old version**

Use: **ScanBetaTof\_tree(mycut,”name such as 84Ge”,&rejectlist)**; again to confirm whether the repeated event is excluded; Go further: enable “IsSave” to save it to log

**Note: if rejectlist has too many inputs; this method may not work properly.**

**Save it to a tree file (no limit by the size of rejectlist):**

void SaveBeta\_tof\_tree(TCutG\*\* \_usecut = \_\_null, string speciesname = "", RejectionList\* \_rejectlist = \_\_null, bool IsAdd = true); void SaveBeta\_tof\_tree(TCutG\*\* \_usecut = \_\_null, string speciesname = "", RejectionList\* \_rejectlist = \_\_null, string GeneralSelection = "", bool IsAdd = true)

**SaveBeta\_tof\_tree(mycut,”ionName”,&rejectlist, [true to update; false to recreate]);**

**Note: true or false of IsAdd can update or recreate file on hard disk**. So if there are different ion. Use true to save them to the same file. Even change coincident condition to get a different result of the same species. Use this method to save to the same file. Different input can be read out by specify the corresponding “IonFillIndex” number*(refer to the following part)*.

New version

**Save it to a tree file (no limit by the size of rejectlist):**

void SaveBeta\_tof\_tree(TCutG\*\* \_usecut = \_\_null, string speciesname = "", RejectionList\* \_rejectlist = \_\_null, string GeneralSelection = "", bool IsAdd = true)

**SaveBeta\_tof\_tree(NULL,”ionName”,NULL, Cutting condition , [true to update; false to recreate]);**

**Note: true or false of IsAdd can update or recreate file on hard disk**. So if there are different ion. Use true to save them to the same file. Even change coincident condition to get a different result of the same species. Use this method to save to the same file. Different input can be read out by specify the corresponding “IonFillIndex” number*(refer to the following part)*.

How to read out results saved in hard disk and further eliminate repeated beta events from different species:

**An update version of “ShowDecayHisto”;**

void ShowDecayHisto(TCutG\*\* usecut = \_\_null, RejectionList\* \_rejectlist = \_\_null, string SelectIonList = "IonA;index IonB;index", string GeneralSelection = "", string DrawIonName = "", bool refinalltree = false)

**For example: 84Ge was saved three times; 81Ga was saved one time; and 83As was save two times.**

**Firstly use**：ShowDecayHisto(NULL,NULL,””); //=>copy to terminal

to show the list of result stored in file;

Something like this is shown on screen.

Ion name index1 index2 ….

84Ge 1 2 3

81Ga 1

83As 1 2

**Then Specify in “SelectIonList” parameter** to read out corresponding results and **also ” DrawIonName” which ion to show** in tof histogram and bete decay time distribution graph:

ShowDecayHisto(**NULL,NULL,”84Ge;2 81Ga;1 83As;1”, ”other condition if have, otherwise leave it empty”, “81Ga”, [true** to enable repeat beta event exclude from different species; default as false] );

**Thus:**

ShowDecayHisto(NULL,NULL,”84Ge;2 81Ga;1 83As;1”,””,” 81Ga”);

OR:

**Trick:**

**Tree of beta-tof history can be accessed globally after ShowDecayHisto() run successfully by**

**tbeta\_tof\_history;**

**Assign it to:**

**tbeta\_tof = tbeta\_tof\_history;**

**Run ScanBetaTof\_tree();**

**to peek what is inside quickly**

1. Fit and calculate the mass of peak after coincidence as introduced in corresponding parts here.
2. **Show 2D histogram** of decay time in second VS tof

ShowBeta\_Tof\_2D(bool \_makecut = false)

Set \_makecut to true to draw a cut in this 2D plot; then using ShowDecayHisto(xxx, xxx, xxx, **"my2d"**) to show plots with cut. Rerun ShowBeta\_Tof\_2D() to show 2D plot with cut

1. **Show histogram of all events selected and events without coincidence** after beta-Tof measurement.

Showhisto\_except\_coin(int which\_all1\_nocoin2\_both3)

0: remove all extra histogram from cd(2);

1: only add histogram of all events selected to cd(2)

2: only add histogram of events without coincidence to cd(2)

3: add both histogram of all selected and no-coincidence events to cd(2)

1. **Trick:**

Beta\_Beta\_coin.cc can be used independently. Using

Read\_Beta\_lst\_batch(xxxx); //can read and generate raw beta tree.

Compare\_Beta\_Beta(); // to get beta\_beta coincident events

ShowBeta\_Beta\_Coin(); // to draw histograms of raw and coincident events

Rate meter:

Which is used to show the counter rate of ROI;

Depend on the setting of ROI;

Global variable (refer to ROI introduction above):

ROI\_WIDTH;

ROI\_INDEX;

To set ROI at tag 1; No hot key can be used; Must use method directly. And suggest to set as last one.

**SetROI(12627300, ROI\_WIDTH,++ROI\_INDEX,0,true,1)**; //

（put TOF here, ROI\_WIDTH, ++ROI\_INDEX, offset to TOF, true to show counts, tag1）

Then:

**RefreshRateMeter(int rate\_in\_second = 1)** // default is rate in one second

If set more ROI, just use the same method to refresh.

To clear all setting of ROI. 🡺 ROIClear();

**How to see the rate of events with and without coincidence🡺 using true option**

**RefreshRateMeter**(100,true);

**How to see the beta-beta rate changes with Run time**

**Trick:**

**First using the following codes to load beta-beta events in “**sweep\_glo\_Beta\_Beta”

Firstly:

tbeta\_beta->Draw("sweeps\_gclock\_1","**cutting condition**","goff"); // replace “**cutting condition”** accordingly

sweep\_glo\_Beta\_Beta.clear();

double\* getsweep;

getsweep = tbeta\_beta->GetV1();

for(unsigned long index=0;index<tbeta\_beta->GetSelectedRows();index++){

sweep\_glo\_Beta\_Beta.push\_back((Long64\_t)getsweep[index]);

}

Second:

Execute:

**RefreshRateMeter**(100,false, true);