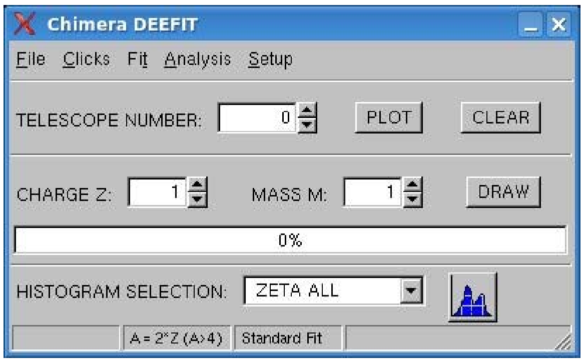
DEEFIT: A TOOL FOR CHARGE AND MASS IDENTIFICATION IN Si-CsI MATRICES OF THE CHIMERA DETECTOR

The DEEFIT program collects in a single program many different procedures that, starting from 2002 have been developed for the Si-Csi identification in the Chimera detector. These procedures generally were based on different steps: 1) Obtain a set of raw data for a set of telescopes with a well defined matching between low gain and high gain for Si energy signal and Fast CsI component; 2) Draw a set of sampling clicks for selected Z and M in the Si-CsI matrix. 3) Fit the data (click points) with a multi-dimensional functional mainly based on formulas developed by L. Tassan-Got (NIM B194, 503, 2002) and extended by the Chimera group for mass and charge identification in the Chimera Si-CsI (N. Le Neindre et al., NIM A490, 251, 2002); 3) Get charge and mass identification by an iterative procedure based on the previous fit results; 4) Check the goodness of the results on charge and mass spectra. Most of these procedures were written in different programming or macro languages, divided in tenth of separate files and input data for various analysis frameworks, making difficult to improve, modify or just to start using them.



The DEEFIT program has the goal to simplify and improve all these procedures in a single graphical application developed for the ROOT framework in order to make easier the Si-CsI analysis for the 2008 or future experiment campaign or progress in the analysis of the previous campaigns. DEEFIT is an object oriented application written in C++, made by about 2000 lines source code. Deefit runs both as a "standalone" executable program and as a "shared library" that can be executed within the ROOT program. The graphical interface is mainly based upon "menus", buttons and input widgets.

WHERE IS *DEEFIT*
The "official" version of deefit can be found in

CC01 linux cluster (sezione):
/home/farmcc/chimera_world/ANALISI/PROGRAMS_OFFICIAL/DEEFIT
The last stable version is 1.1 (January 2010).

CHINAF01 linux (LNS): /home/chimera/ANALISI/DEEFIT/DEEXX

HOWTO RUN *DEEFIT* ON LINUX:
1) Standalone version:
From the command line console in Linux give the command "**deefit_S"**, eventually with the complete path if you are not running in the directory where deefit is.

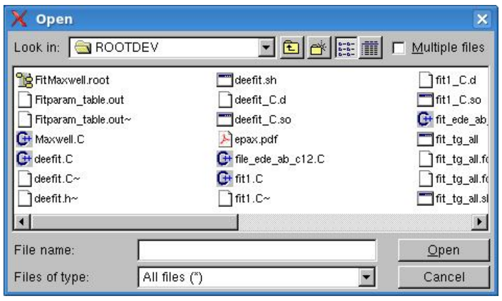
2) Shared library version:
In the DEEFIT directory the library *deefit_S.so* should be present.
Start ROOT (better if from version 5.18 or greater) from the directory in which the program deefit is. Inside Root:
Give command: *gSystem->Load("deefit_S.so")* to load the deefit_S library.
Give the command: *.L deefit.C+* in order to compile the little main program macro before to execute it.
Execute deefit with the command: *deefit()* without arguments inside parentheses.

HOWTO RUN *DEEFIT* ON WINDOWS (XP,Vista):
Note: version 0.99 only. Version 1.1 is not tested on Windows
Start ROOT (better if from version 5.18 or greater). Inside Root:
Give command: gSystem->cd("c:/users/…..") to change directory where the program deefit is.
Give command: *gSystem->Load("deefit_C.dll")* to load the deefit library if present.
Execute deefit with the command: *deefit()*

Alternatively: give command ".L deefit.C" and after *deefit()* to run deefit as a macro if windows installation has no available C++ compiler.
Warning: if you want to modify and create a .dll library from the deefit source file on Windows a complete C++ development environment (Visual C++ .net) have to be installed with appropriate setup (see the Root web page for information).

REFERENCE MANUAL

Menu File: This menu manages the Root Trees containing the raw data to analyse.



Open File: Open a window in which the name of a root file containing the raw data in a "tree" format can be opened. Generally the root tree has to contains almost the variables: numtel, desilpgf (the floating variable containing both the high gain and low gain for silicon), and fastpg as in the original 2003 Paw Nuple (see E. Geraci instructions), but is possible to work with a more general Root TREE containing all the raw variables created with the Isospin20 program. The deefit program declares the following structure to hold variables having the same order and number of variables of the original tree:

```
struct MIdent {
 Int_t numtel;
// raw parameters
 Int_t desilpg,desilgg;              //desilpg/desilgg
 Float_t desipgf;                    //float desilpg
 Int_t time,time80;                  //Tof 30%, 80%
 Int_t fastpg,slowpg;                //fast/slow (pg)
 Int_t fastgg,slowgg;                //fast/slow (gg)
};
```

and the **Mtree** class in order to manage raw data in tree format. With the *InitTree method* we will tell the tree to populate only these variables, when reading an entry, that are interesting for us:

```
void MTree::InitTree()
{
 if(IsOpen()) {   //if root file is opened
  OpenTree();     //assign ftr
  fevt = new MIdent;
  ftr->SetBranchAddress("numtel", &fevt->numtel);
  ftr->SetBranchAddress("desipgf",&fevt->desipgf);
  ftr->SetBranchAddress("fastpg", &fevt->fastpg);
  . . . . . . . . . . . . . . . . . . . . . . . . .
 }
}
```

Warning: if the root file is generated by the h2root program from a hbook original one the variables have to declarad Ushort_t in the structure like in this example:

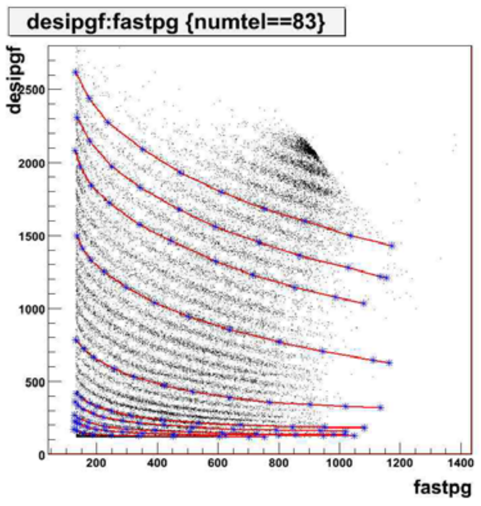
```
struct MIdent {
 UShort_t numtel;                    //numtel
 Float_t desipgf;                    //float desilpg
 UShort_t fastpg                     //fastpg
};
```

Close File: Close a Root file tree, previous opened with the Open command.

Quit Root: Quit the deefit program and the Root program. It is possible to quit the deefit program without closing Root clicking on the closing cross on the right of the deefit window.

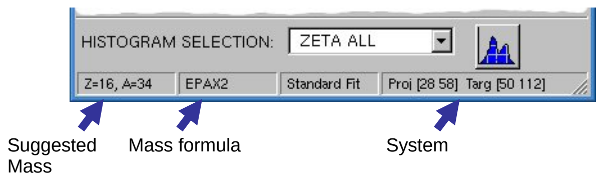
In order to plot the Si-Csi matrix after having opened a root file is possible to choose a telescope in the entry box (TELESCOPE NUMBER) and press the **PLOT** button. Use the **CLEAR** button to draw again a matrix in a clean state.

Menu Clicks: This menu manages the "clicks" i.e. the sampling points in the isotopes lines or charge Z lines on the Si-CsI matrix.



Once drawn the matrix select a charge Z and a mass M and push the **DRAW** button. With the mouse "click" several points along the line for the chosen isotope. To finish over a line double click the left mouse button.

If mass resolution is present on light charges ($Z \leq 9$) click on isotopes for example following the sequence ^{1}H, ^{3}H, ^{4}He, ^{7}Li, ^{7}Be, ^{11}B, ^{12}C, etc. For charges for which no mass resolution is present choose the mass following the most appropriate mass law. Default is A=2*Z. Other available formulas are the Charity mass formula and EPAX2 calculation, which can be selected in the menu Fit. When you select a mass formula the suggested result for the mass corresponding to a given Z is always indicated on the status bar on the lower part of the deefit window. The suggested mass value is **not automatically** copied in the MASS input.



Z=16, A=34 | EPAX2 | Standard Fit | Proj [28 58] Targ [50 112]

In the example on the left the EPAX2 (see menu fit for explications and references) calculation is used for a neutron poor system ^{58}Ni + ^{112}Sn. When selecting Z=16 charge the mass A=34 is suggested for the click line. This value of mass will be saved in the clicks database and used in the fit procedure. Calculated masses with a chosen law will be used in the all subsequent data analysis procedures (for charge lookup) and in the procedures to draw the fit results (also for charges Z for which no click lines were drawn). Note that this is a change respect to the original "Isospin 2003" charge lookup routines where A=2*Z law was always used. Results and goodness of the fit procedure can depend from the click sample selection.

How to modify graphically the clicks: clicks lines can be graphically modified with the mouse. Click points can be deplaced (left mouse button), deleted or added (Use context menu with the right mouse button). A whole line can also be deplaced. Changes are automatically saved in the clicks list database in memory. *Warning*: do not use the context menu to delete a whole click's line, use the deefit click menu instead as described below.

Menu Clicks reference:

Load File: Open a window asking to load a file containing saved clicks. The clicks are saved as a Root objects file. Clicks objects are put in a "list" container. The default file name for a click file is "graphcut.root". If you want to load in memory multiple clicks files check the appropriate check box in the window and select the files pushing the "shift" key + the left mouse button.

Save: Save in a file with name "graphcut.root" all click lines actually stored in memory for all telescopes for which clicks were defined. *Warning*: if the file with the same name exists in the current directory it is rewritten and old data are lost.

Save As: Open a window asking to save a clicks file. User can choose a particular name and navigate among directories.

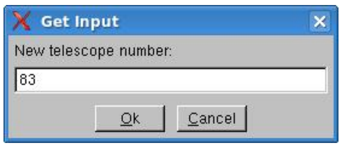
Spline: After selecting a telescope for which a set of clicks exists do an hyperbolic fit of all click lines using the "fusp" object instance of a TF1 class:

```
TF1 fusp("fusp","[0]/(x**3)+[1]/(x**2)+[2]/x+[3]");
```

where [0]... [3] are the fit parameters. Fit results are displayed as a blue line and the clicked points are adjusted following the TF1 "fusp" function for each click line of the current telescopes. The new points can be saved permanently in a file using the Save or Save As menu item (*The idea of a spline command is taken from a A. Benitz macro program*).

Graph: For the current selected telescope (and if the Si-Csi matrix canvas is opened) draw all the clicks defined for this telescope (if any in the database) as a set of points joined by a red line to guide the eyes.

Copy: After selecting a telescope for which no click line is yet defined the copy command asks for a telescope number (see input window on the left) for which clicks exist (for example if your current telescope is n° 84, you ask to copy all clicks of telescope n° 83 upon the telescope 84. The clicks are copied in the current telescope and their names changed accordingly.



The clicks are added to the current list database in memory. After they can be graphically adjusted to adapt them to the new telescope.

List: List to the standard output (the Root console) all clicks in the database. Information contains the name of the click (ex. "Z5A11_t83"), the title ("ex. t83"), the address in memory. From the list you can deduce that clicks belongs to the class TGraph. The clicks are organized in a dynamic list (TList class) container. Each TGraph object stores the x-y array values (points) relative to the click line.

```
OBJ: TGraph     Z2A4_t83        t83 : 0 at: 0x8f01a68
OBJ: TGraph     Z3A7_t83        t83 : 0 at: 0x8f01970
. . . . . . . . . . . . . . . . . . . . . . . . . . .
OBJ: TGraph     Z22A44_t83      t83 : 0 at: 0x8ed5818
OBJ: TGraph     Z24A48_t83      t83 : 0 at: 0x8ed59c0
```

Warning: In the current version few care has been taken of list indexing for easier lookup. This is the goal of a next version.

Delete All: Delete all clicks in the database for all telescopes.

Delete Curr: Delete all clicks lines for the current telescope

Delete One: Request to delete one particular click line of the current telescope. A window is opened asking for the click line name. Answer giving a valid name (Ex. “Z3A7_t83”). The cut is deleted from database and graphically disappears from the matrix.

Other ROOT applications: The Tiger program by M. Houck (Rochester) is based on automatic search of Z lines in Si-CsI matrices, in order to substitute the “clicks” procedure and refers to Bologna fortran routines for fitting. In contrast, the DEEFIT program is constructed around the fitting procedures (see below) and make no trying to insert total automatic tasks for Z line identification.

Menu Fit: This menu manages the fitting procedures. It uses *chi-square* method minimizing over the sampled click points and the user fitting function(s). It is based on TMinuit minimization class.

The identification function of the Chimera Si-Csi is based on the Bethe-Block formula used to fit correlations between ΔE (Si) and E (Fast) as a function of Z and A. As stated in N. Le Neindre et al. NIM A490, 251, 2002, this correlation, taking into account of various correction can be parametrized by a 7 parameters functional of the form:

$$\Delta E = [(gE)^{\mu+\nu+1} + (\lambda Z^{\alpha} A^{\beta})^{\mu+\nu+1} + \xi Z^{2} A^{\mu} (gE)^{\nu}]^{1/(\mu+\nu+1)} - gE$$

A multiparameter fit where ΔE = *f*(E,Z,A) have to be performed simultaneously over a sample of (Z,A) isotopes of the ΔE-E matrix. The standard fit has a total of 9 (7+2) parameters to take into account an offset upon the fast (X = fast – offset_f) and an offset upon the Si energy (Y= ΔE + offset_s).

The TMinuit package acts on a function (fcn) defined via the SetFCN member function called the “fitter”. The fitter defines a method (chi-square in our case) to minimize the distance between the user function and the data points. This is the “fitter” implementation in the DEEFIT program returning the chisq value:

```
void fcn(Int_t &npar, Double_t *gin, Double_t &f, Double_t *par, Int_t flag)
{
 Double_t chisq = 0.0;
 Double_t fret;
 vector<MPart> data(MDEEFrame::fparray);
 FitPtr fp = gfit->GetFPtr(); //get the address of the fitting function fptr

 gfit->SetOffset(kTRUE);
 Int_t n=data.size();
 for(Int_t i=0; i<n; i++) {
  fret = data[i].y - (gfit->*fp)(data[i],par);  //calculate chisquare
  chisq += fret*fret / (data[i].e*data[i].e);
 }
 f = chisq;
}
```

For chisquare minimization: $\chi^2 = \sum_{i=1}^{N} \left[\frac{y(i) - f(x(i), A(i), Z(i), P_1 ... P_k)}{e(i)} \right]^2$

Here data[i] is a vector of Mpart structure containing data information for each click point:

```
struct MPart {
 Double_t x,y;  // x,y value of the clicks
 Double_t Z,A;  // charge and mass associated to the x,y point
 Double_t e;    // error on y value
};
```

and *gfit->*fp(data[i],par)* is a pointer (this pointer contains the address of a fitting function) to one fitting multi-parametric functional that receives as input the *data[i]* structure and *par* (a vector of the fitting parameters). This construction is needed to maintain the highest degree of abstraction in the fitter function because we want to choose between more than one functional (a standard Tassan-Got one with 9 parameters as described above, or an enhanced fitting functional with 14 parameters for example) without modifications in the fitter function code.

The fitting functions defined in DEEFIT are:

```
Double_t MDEEFrame::func(MPart &p, Double_t *par) // Standard Tassan-Got 9 param
Double_t MDEEFrame::func14(MPart &p, Double_t *par)    // Enhanced 14 fit params
```

Func14 is a 14 parameters functional, proposed by A. Pop in order to fit with a unique set of parameters the ΔE-E forward rings matrices for the reactions ^{124}Sn+^{64}Ni and similar reactions with a Z=50 projectile (mainly ISOSPIN campaign). In this functional the fast component *L* is parametrized following the Gawlikowicz law (NIM A491, 181, 2002) in the form adopted for Si-Csi telescopes (J. Blicharska et al., LNS report 2005):

$$E = f_2(L, Z, A) = aW(Z)L + bZ\sqrt{A}\ln(1 + cW(Z)L)$$

$$W(Z) = p_1 \frac{\ln(1 + p_2 Z)}{Z}(1 - \exp - p_3 Z^{p_4})$$

where a,b,c,p_1,p_2,p_3,p_4 are free parameters, and to compose this relation with the Tassan-Got one.

The InitFitFunction `MDEEFrame` class member is the place in the program where the pointer (*fptr*) to the fitting member function is initialized (i.e is assigned a function to the pointer) and the parameter's number fixed. Thus deefit is easily opened to modifications in which several new fitting functions can be selected without modification to the “core” routines for fitting and data analysis.

```
void MDEEFrame::InitFitFunction(Int_t id)
{
 if(id==kFitStandard) {
  fNpar = 9;
  fptr = &MDEEFrame::func;
  ffitkind = kstandard;
 }
 else if(id==kFitEnhanced) {
  fNpar = 14;
  fptr = &MDEEFrame::func14;
  ffitkind = kenhanced;
 }
}
```

Menu Fit reference:

Init Fit: Initialize the fit parameter following the chosen fit functional. Also set the minimum and maximum values among which the parameters of the fit can fluctuate during chi-square minimization procedure and the variation step. Here are the output values for the 9 parameter fit (standard fit) The values set are almost the same chosen in the Bologna fit procedure (N. Le Neindre, E. Geraci) with the Fortran Minuit routines (*fit_tg_all.for program*):

```
PARAMETER DEFINITIONS:
NO.   NAME         VALUE      STEP SIZE      LIMITS
 1 alfa        1.00000e+00  1.00000e-03    0.00000e+00  2.00000e+00
 2 beta        5.00000e-01  1.00000e-03    2.00000e-01  1.00000e+00
 3 mu          1.00000e+00  1.00000e-03    2.50000e-01  2.00000e+00
 4 nu          5.00000e-01  1.00000e-03   -1.00000e+00  2.00000e+00
 5 csi         2.00000e+02  1.00000e-01    1.00000e+01  4.00000e+03
 6 g           2.00000e+01  5.00000e-01    1.00000e-01  4.00000e+01
 7 lambda      5.00000e+02  1.00000e+00    0.00000e+00  2.00000e+03
 8 of_s        5.00000e+01  1.00000e+00    0.00000e+00  2.00000e+03
 9 of_f        1.22000e+02  1.00000e+00    1.00000e+01  2.00000e+03
```

The parameter 9 in fact (offset_f) is determined automatically looking at the minimum value of fast component channel in the clicks points. This starting value will be adjusted by the fit procedures.

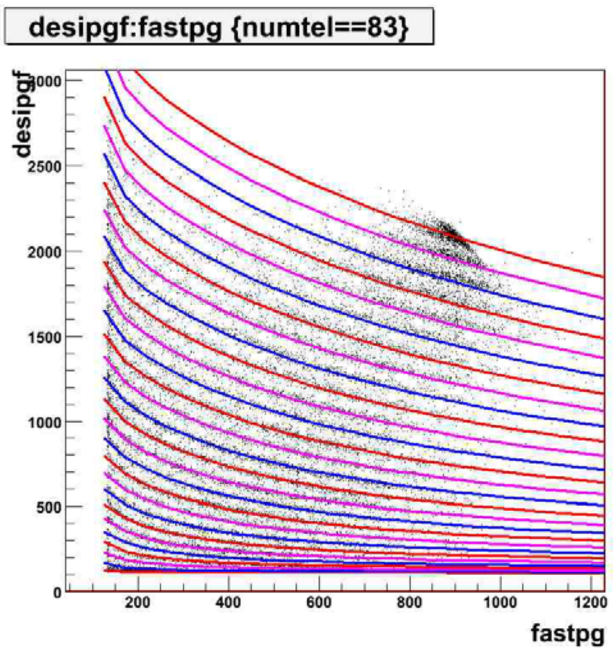
The initialization command do also a first step Montecarlo minimization upon the starting values of the parameters. In order to complete this command a valid set of data points (clicks) have to be present in the clicks database list for the selected telescope. During this phase the offset_f parameter is fixed to the startup value and can not be modified by the procedure.

Fit: Perform a chi-square minimization fit following the chosen fit functional using the TMinuit class. Gives fit statistics and chi-square results. Create a new instance of the MFitParam class in order to store the results of the fit for the current telescope:

```
fvpar[ftel_curr] = new MFitParam(ftel_curr, fNpar, param, chisquare);
```

These results, on request (see below), will be stored on disk as a plain ascii file. This file is one of the input file for the isospin general analysis program.

Graph One: For the current telescope, and selecting Z and A in the deefit CHARGE and MASS selection input boxes, draws a graphical result of the fit.



Graph All: For the current telescope, draw the graphical fit result for all charges up to a maximum charge defined by the user. The mass used to draw Z lines is the one defined in the fit menu (A=2*Z, Charity or EPAX2, see below). For coherent results it should be the same used to draw the clicks (this affect only visualization indeed). The picture shows the fit result for the ^{58}Ni + ^{112}Sn (35 A.MeV, Timescale exp., tel. 83, ring 3I) with A=2*Z for Z>5 charges and standard (9 parameters) fit.

Standard (9 params): Set the standard fit function for Chimera Si-CsI with 9 parameters as defined in NIM A490, 251, 2002. This function will be used in all successive requests.

Enhanced (14 params): Set an enhanced fit function for Chimera Si-Csi with 14 parameters and a parametrized fast component light emission as defined above.

Mass = Charity: Given a charge Z (Z>4) return the mass A as defined by the Charity formula (Phys. Rev. C58, 1073, 1998). For Z≤4 the following constant values contained in the array *gLight* are used:

```
Double_t gLight[] = {1,4,7,9};
```

The GetMassCharity member function is defined below:

```
Double_t MDEEFrame::GetMassCharity(Double_t Z)
{
 Double_t A;

 if(Z<=4) {
  A = gLight[(Int_t)Z-1];
  return A;
 }
 else
  return (UInt_t)(2.072*Z + 2.32E-03 * Z*Z) + 1;
}
```

Mass = 2*Zeta: For Z>4 always define A=2*Z. For Z≤4 the mass is defined by the *gLight* array. This is the default.

Mass = EPAX2: For Z>4 calculate a mass following the EPAX2 systematic. For Z≤4 the mass is defined by the *gLight* array.

EPAX is an empirical parametrization based on fragmentation cross section, revised in the year 2000 (version 2) (Phys. Rev C61, 034607, 2000). For a given a projectile (Z_P,A_P) and target (Z_T,A_T) system the EPAX parametrization calculates several quantities with the final goal to estimate the production cross sections for projectile fragmentation reaction products. Among these quantities is Z_P, the most probable charge for a given *A* corrected by various factors that take into accounts corrections for nuclei far from beta stability line and neutron richness of the projectile.

The deefit program defines the class MEpax that implements the EPAX2 parametrization in order to calculate for a given Z the most probable mass A by means of an iterative procedure. This is implemented by the GetMassEpax function:

```
Double_t MEpax::GetMassEpax(Double_t Z) { . . .}
```

For a given system data are stored to an array for fast use when required, as in the data analysis calculations for charge lookup.

Save Param: Save the fit parameters in a plain ascii file for all telescopes for which a fit has be given during a work session. The file has the standard name "**Fitparam_table.out**". The current file format is the following:

```
*Fit parameters table created by DEEFIT
*09| ntel  alfa   mu  nu  csi  g  lambda  of_s  of_f  chi2
*14| ntel alfa   mu  nu  lambda csi gap gb cp zl zi ze of_s of_f chi2
* third column: 0=2*Z  1=Charity  2=Epax
  83 9 0 0.88394 0.40514 0.63914 -0.07777 24.74365 1.99503 17.01960 112.20597 121.10890 5.88848
. . . . . . . . . . . . . . . . . .
```

The example shows the current version file format very similar to the 2003 standard one (defined by the Bologna group). As a difference the file contains after the telescope number (83) the number of fit parameters (9 in this case), and the mass formula used (0=2*Z, 1= Charity, 2= Epax), followed by the parameters value. The last number is the chi-square.
Warning: The isospin program require also four final numbers after the chi-square (quality identification codes).

Load Param: Load the fit parameters written in the file "**Fitparam_table.out**" in memory. In the current version the fit parameters loaded from a file cannot be used with commands "Graph one" or "Graph all", but can be used for data analysis (charge and mass lookup).

Menu Analysis: This menu manages the charge and mass identification procedures.

This menu performs an event-by-event identification: firstly the charge Z is found in a iterative process where the found Z value is the minimum distance between the experimental ΔE point and the result of fit functional calculation at position E. Once found the charge Z, in a second step the mass A is looked for. The mass value is retained and booked in histograms for *Z≤MAXMAS*, where maxmas is a parameter fixed to 14 in version 1.1. The third step is calculation of dispersion around mean charge and mass value for a better evaluation of the identification quality.

The routines used to perform these tasks are mainly a C++ translation of the Fortran95 routines in the isospin analysis program with some modifications. The main modification is the fact that the actual mass calculation setting (used in doing clicks) is taken into account while in the original routines only the relation A=2*Z is considered. A second modification is the fact that the identification routines are able to compare experimental data with different fitting functions (func, func14) following the actual setting in the fit menu. These modifications should be reported in a new version of the isospin fortran analysis program.

Menu Analysis reference:

Identification: This command start the event-by-event identification procedure calling the:

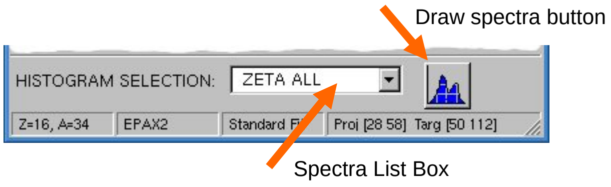
```
 Int_t MDEEFrame::Si_CsiIdentification(Int_t ntel)
```

method. The identification starts if a valid root tree file is opened, and for the current telescope the fit parameters are present. A check is done to establish if the current setting for the functional to use in the calculation (func standard, func14 enhanced) are coherent with the stored fit parameters, by means of the method GetAnaCoherence( ) of MDEEFrame class:

```
. . . . . . . . . . . . . . . . . .
Bool_t IsCoherent = GetAnaCoherence(ntel);
if(!IsCoherent) {
 cout<<"Tel "<<ntel<<":Possible mismatch between current fit setting and data"<<endl;
 cout<<"Please change fit setting (Standard or Enhanced) in fit menu"<<endl;
 return -1;
}
. . . . . . . . . . . . . . . . .
```

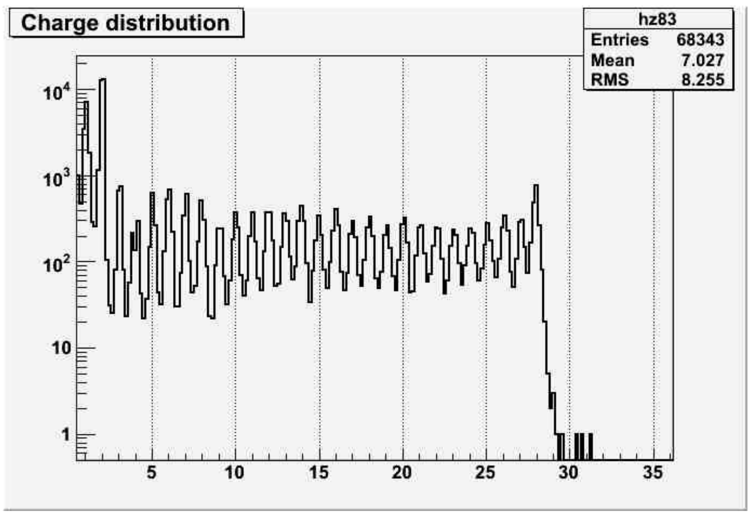
Warning: No check is done in the current version to evaluate coherence in the mass calculation in the routine (that should be the same used for clicks) with the actual mass setting in the fit menu when the charge identification step is performed. This will be done in a next version.

Draw: Draw the selected spectra (in the spectra list box). The defined spectra are, the charge distribution (ZETA ALL) or the mass spectra up to Z=MaxMass. The same effect is obtained pushing the spectra icon on the right.



HISTOGRAM SELECTION: ZETA ALL

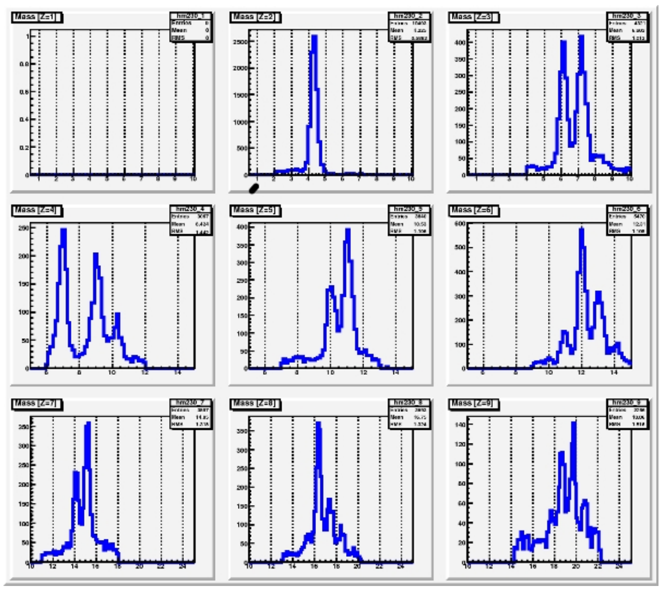
Z=16, A=34 | EPAX2 | Standard F | Proj [28 58] Targ [50 112]



The picture shows the charge distribution for the telescope 83 in the ^{58}Ni + ^{112}Sn reaction obtained with standard fit and A=2*Z mass parametrization in log scale (*Timescale exp*).

Some statistical check and graphical tools to verify the goodness of the identification have been included in the current version of the program (see below).

Draw All Masses: Draw all the mass distributions from Z=1 to Z=9 in a window with 9 panels.



Check Z Identification: When identification is done a root tree file (treeide.root) is saved on disk. This file contains a branch with the charge Z identification variable for each event of the current telescope. The command open this file as a “tree friend” of the raw data root input file and event by event construct a bidimensional Si-Fast matrix for a selected Z choosen by the user. Automatically the data for the selected charge is compared with all other charges in different colour points. The treeide.root file is temporary and can be overwritten, but can be saved (copying it with another name) and used for more specific analysis or checks inside Root if useful. It should be linked to the main root data tree as "tree friend" with the *AddFriend* method of the class TTree:

```
  t1->AddFriend("treeide","./treeide.root");
```

where t1 is a TTree instance of the raw data tree.

Menu Setup: The setup menu set parameters and conditions for the program.

System: This command open a window in which the reaction system (Z_P, A_P, Z_T, A_T) can be defined. This is used by the EPAX2 calculation and in the definition of the maximum charge that have to be calculated in the identification procedure ($Z_{MAX} = Z_{PROJ} + 2$).

E. De Filippo (January 2010)