GTSort; the GRETINA ON–LINE Sorter

torben@anl.gov

June 18, 2008

Contents

1 Introduction

GTSort is a sorting code based on GSSort, the on–line sorting code used at Gammasphere. It shares the look and feel of GSSort as well as a good deal of the source codes. Just like GSSort, GTSort is primarily meant for on–line sorting, to check that your experiment is working. You may well be able to use GTSort for your off–line needs as well, but you will probably have to add specific features that you need.

Early versions of GTSort may not have the full functionality of GSSort – so that we can develop the basic GRETINA sorting engine with less complexity. E.g., GTSort will not have external detector processing in early stage of developments.

As GSSort, GTSort is made to be user friendly – at the expense of speedy code. It is the goal that you should be able to monitor your data on–line without ever having to compile any code or even know how write or compile C++ code. Being able to sort 100% of your data on-line is not part of the design goal.

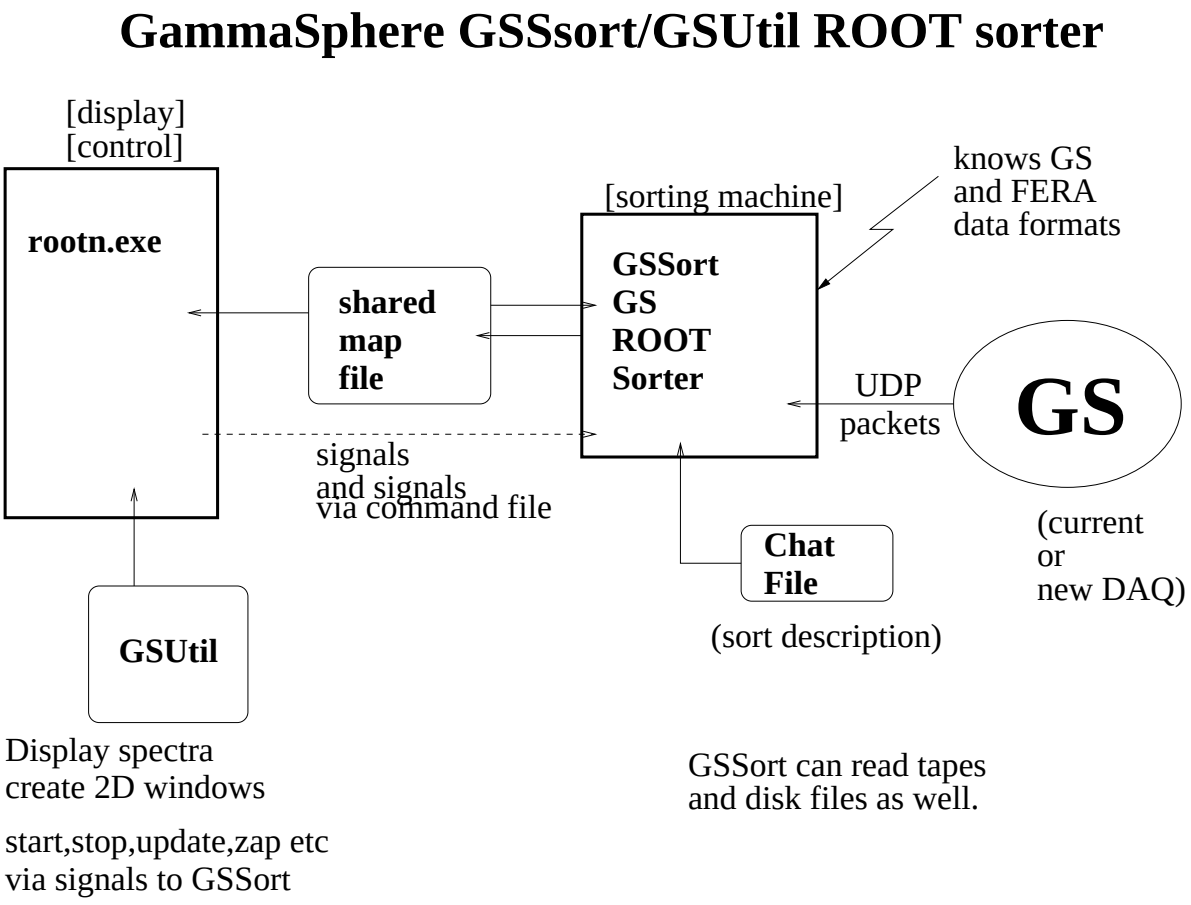

Figure 1: Schematics of the GSSort and GTSort on–line sorter. GSSort and GTSort works the same way with respect to Gammasphere and GRETINA. The data formats and some details are of course different; but the basic ideas are the same.

The GTSort documentation and source codes can be fetched from

```
http://sun0.phy.anl.gov/gretina
```

Figure 1 shows how the sorting engine (the GTSort program itself) interacts with the display program, rootn.exe. From inside rootn.exe you can communicate with the sorting engine as well as display your spectra.

2 Setting up root

GTSort is using the ROOT system from CERN, however, is is not true to say that we 'sort our data in ROOT'. Rather, we sort the data in our own code (sorting engine) and store the resulting histograms in the ROOT system format (in a root file or map file), so we can use 'root' (or 'rootn.exe' in the case of map file) to display our data.

To use GTSort you need to obtain and install a version of ROOT from "http://root.cern.ch/". When you have installed it, you should be able to start root with 'rootn.exe'. Here is an example of how we usually compile and install ROOT:

```
cd /usr/local/src/root_v5.18.00
resource (so there is no hint of old version)
mkdir /usr/local/root_v5.18.00
setenv ROOTSYS /usr/local/root_v5.18.00
./configure --prefix=/usr/local/root_v5.18.00 --etcdir=/usr/local/root_v5.18.00/etc
make  (takes a long time)
make install
```

To get the path right, both for rootn.exe and GTSort, the following setup scripts (usually called 'setuproot') might be useful:

```
#=============================================================================
#!/bin/csh
# setup script for ROOT for a SUN
# select the proper version of root

#setenv ROOTSYS /usr/local/root_v5.13.02
#setenv ROOTSYS /usr/local/root_v5.16.00
setenv ROOTSYS /usr/local/root_v5.18.00

#-----------------------------------------
# you should not have to change anything below
```

```
set path = ( $path $ROOTSYS/bin )
setenv LD_LIBRARY_PATH {$LD_LIBRARY_PATH}:‘root-config --libdir‘
exit
#============================================================================


#============================================================================
#!/bin/bash
# setup script for a LINUX machine
# select your version of ROOT below

#ROOTSYS=/usr/local/root_v5.13.02
#ROOTSYS=/usr/local/root_v5.16.00
ROOTSYS=/usr/local/root_v5.18.00
export ROOTSYS

#-----------------------------------------
# you should not have to change anything below

PATH=$PATH:$HOME/bin:$ROOTSYS/bin
export PATH

LD_LIBRARY_PATH=‘root-config --libdir‘
export LD_LIBRARY_PATH
#============================================================================
```

ROOTSYS points to the *specific* version of ROOT that we will use.

It is recommended that you keep a specific 'setuproot' script in the directory you work in since root and map files are not always backward compatible. I.e., the version of root that you used when you worked on your data last should be used again when you come back later and want to look at your data again.

3 Getting the GTSort/GSUtil codes and compiling them

To get the GTSort and GSUtil code, create a working directory, cd to it and [1]

```
mkdir bin src
cd src
wget http://sun0.phy.anl.gov/gretina/gretina.tgz
```

[1] if you do not have wget, just download gretina.tgz by hand

```
tar -zxvf gretina.tgz
source ../setuproot
make -f MakefileGTSort
```

You may have to touch up the 'Makefile.Sun' or 'Makefile.Linux' for your particular machine (a Makefile.Mac might be available some day). The main MakefileGTSort is generic and you should not have to change anything here. This step created GTSort in the ../bin directory if all went well.

To control the optimization, you can set the CCENV environment variable as one of these

```
setenv CCENV -fast
setenv CCENV -g
```

on the suns or

```
export CCENV=-O
export CCENV -g
```

on a Linux machine. On the suns it is very important to compile with -fast to make the code efficient.

Now you need to generate the GSUtil (which is shared with GSSort) utility. The procedure is simply

```
rootn.exe
.L GSUtil.cc++
.q
```

That should generate the GSUtil_cc.so sharable object file here in your src dir. On some installations, this compilation is a bit 'noisy', but if it compiles, you are OK (that goes for GTSort as well). You should now cd back to your working directory (cd ../).

4 UDP receiver and firewalls

On the fedora core Linux machines, the UDP receiver in GTSort/GSSort will not work until you enable UDP traffic trough the port we use for sending (traditionally 1101). Here is a terse writeup of what you have to do on a Fedora Core 8 Linux box to make it work: Select "System", "Administration", "Firewall", "Other Ports". Then click on "User Defined" and add port 1101 with UDP protocol. Finally implement the new rules by clicking on "apply".

If you still do not receive UDP datagrams from the GRETINA sender, you may have another firewall that will not let the 1101 port UDP traffic through.

5 Starting root for on–line sorting

From your working directory, start up rootn.exe [2]. Once in, start up the GRETINA menu bar (it should be in your src dir by now, just copy it up to the working directory) as

```
.x gtbar.cc
```

That should bring up a list of buttons that should be useful. You can easily add your own favourite commands. Load SGUtil by clicking the first button (or execute ".L src/GSUtil_cc.so" by hand). Once this utility has been loaded, you have the commands listed in section 9 at your disposal. However, before you can start a sort, you need to create a chat file, which is described below in the next section.

6 The chatscript file

GTSort will read a chatscript which describes what you want to do in the sort [3]. Here is an example of a very simple chatscript:

```
#-----------------------------------------
# input (network or disk)

;input disk /home/tl/d6/gtdata/Co60_10usec.r30.dat
input net 1101

#-----------------------------------------
# output to root file or map file

;output test.root
;rootfileoption RECREATE
;rootfileoption UPDATE

sharedmem c1.map  200000000
# NOTE: use 'sdummyload(200000000)' to find startmapaddress
startmapaddress   0xac08e000

#-----------------------------------------
# basic sorting parameters
```

[2] Do not start root - since 'root' does not know about shared map files!. you must start rootn.exe.

[3] It also conveniently documents what you are doing – which comes in handy later on when you are trying to figure what you *were* doing some time ago – especially when you have external data.

```
nevents          2000000000
printevents      20
hiresdatamult    0.0055
egemin           10
```

This example is close to the gtsort.chat file that you should have in your src directory (you may want to 'cp src/simple.chat' and use it as a starting point). All the commands that are available are documented in section 8. Comments are specified by '#' or ';' at the start of the line.

In the chatfile, 'input' specifies where the data comes from [4]. Next, you specify the output – which can either be a root file or a map file. A root file is usually used for off–line sorts and a map file for on–line sorts. In the latter case we can *see the data as it come in*, but you have to find the startmapaddress as described in section 8.

Here we will assume you want to use the map file, so, according to the instructions in section 8, first you need to execute this command in rootn.exe

```
sdummyload(200000000)
```

to find this address. If the result that comes back is

```
Memory mapped file:   dummy.map
Title:
Option:               CREATE
Mapped Memory region: 0xac08e000 - 0xb7f4b000 (190.74 MB)
Current breakval:     0xac095000
```

then specify '0xac08e000' as the address [5]. If you have this address wrong, you may see a message like

```
Error in <TMapFile::TMapFile>: mapped file not in mmalloc format or
already open in RW mode by another process
```

Use

[4] We may not implement tape reading since it is not used much anymore.

[5] Be aware that this address is sometimes a moving target. It might depends on what is going on on the machine and patches you have installed.

```
sload("c1.map")
```

to attach the rootn.exe display session with the map file you specified in the chat file. Figure 2 shows an example of a the spectrum you would see by issuing the command 'd1("ehi017")'

7 Running GTSort for off–line sorting

This is simpler than the mapfile sort in section 5. Comment out the mapfile output option in the example in section 5 and enable the rootfile option. The root files are much smaller than the mapfiles (they are automatically compressed), but you cannot look at the data as it comes in.

To start the sort, type

```
bin/GTSort -chat gtsort.chat > gtsort.log
```

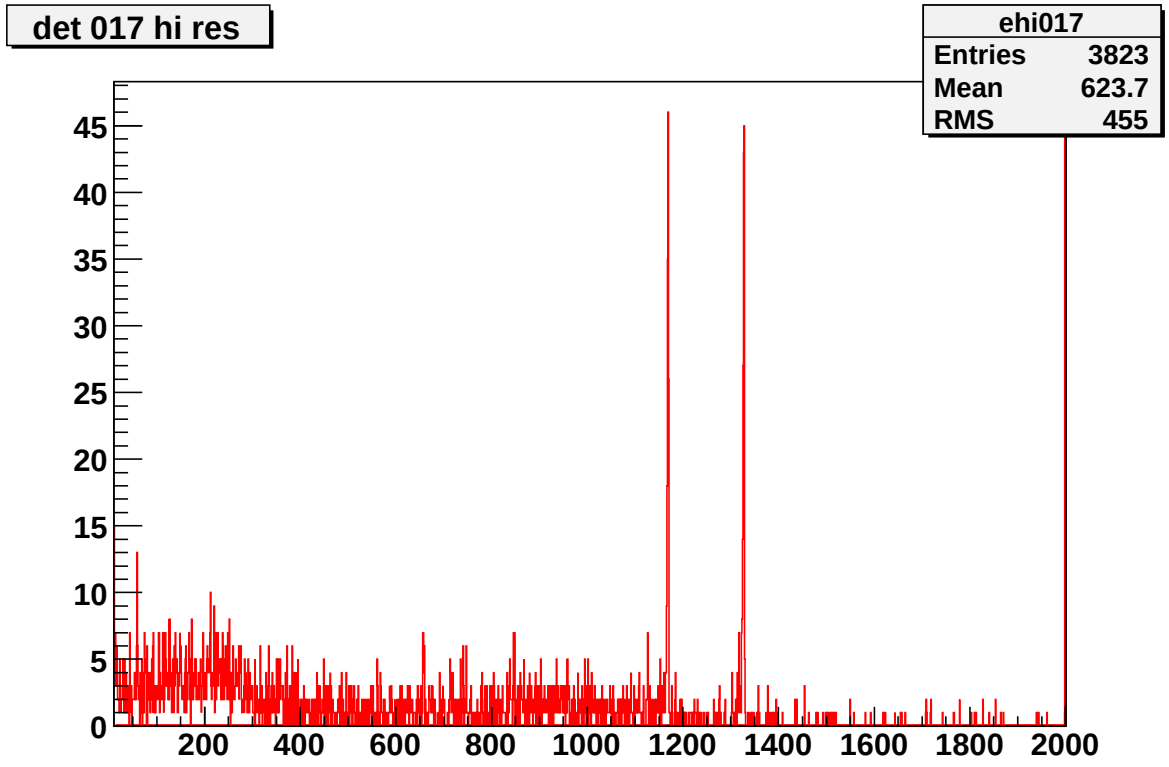

Figure 2: Example of a ^{60}Co spectrum from 'det17' (which is the digitized output from digitizer 1, channel 7).

You may already have this command in the gtsort.cc bar which you loaded by ".x gtsort.cc". When done, you load the root file in rootn.exe by the command

```
dload("test.root")
```

You should now be able to display spectra as e.g.,

```
d1("ehi017")
d1("hitpat")
d2("arrayhit")
etc
```

type "ls()" to see what spectra you have.

8 List of chatscript commands (GTSort)

Note: some commands and options are not (yet) implemented in GTSort.

- input source device—port

  Specify where the data is read from. Options for source is 'tape', 'net' or 'disk'. If tape is specified you must also specify the tape drive. For the 'net' option you must specify the port to listen to (Gammasphere uses port 1101). For 'disk', specify the diskfile to sort from

  examples:

  input tape /dev/rmt/1mbn
  input net 1101
  input disk /home/expdata/run2.data

  Note: on the Solaris machines, the tape drives are usually named something like:

  /dev/rmt/1mbn - stacker
  /dev/rmt/0mbn - single side drive
  /dev/rmt/2mbn - DLT drive (sun1 only)

  ...to make sure, put a tape in and probe it as e.g.,

  mt -f /dev/rmt/0mbn status

  Note: you can use 'GSudpSender' to simulate the Gammasphere sender

  Note: Use 'tape2disk' to dump data from a tape to disk

- output file

  specify the root disk output file

  example

  output data.root

- rootfileoption s

  specify how the root file (specified with "output file") should be opened. s can be:

  RECREATE or UPDATE

  RECREATE: create a new file, if the file already exists it will be overwritten.

  UPDATE: open an existing root file and sort more data into it.

  the default is RECREATE

  example

  rootfileoption RECREATE

  create a new root file at the beginning of a sort

  example

  rootfileoption UPDATE

  read in the content of the root file specified with the 'output' chat script option before the sort start. I.e start the sort with the current content of the output root file.

  Note: At the moment you cannot add to a shared memory file. Always use 'RECREATE' with shared memory files.

- sharedmem s #

  Use roots shared memory implementation. Second argument should be the name of the shared memory map and the third argument should be the size of the memory map in bytes.

  example

  sharedmem gs.map 200000000

  ..... note1: You MUST specify a shared memory file on a local disk on the machine you are working on. Otherwise you will get extremely poor performance! You should only use shared memory on machines that have a lot of memory.

  ..... note2: if you make the shared memory map too small you will get an error message as:

  updating empty shared mem file... Segmentation fault (core dumped)

  In that case, just increase the size and try again.

- startmapaddress #

  Set the start address of the root shared map file.

  example

  startmapaddress 0xe2000000

  note: To figure out what address to use, follow these steps:

  rootn.exe
  .L src/GSUtil_cc.so
  sdummyload(200000000)

  see what start address rootn.exe chose. Set the argument to sdummyload above to the shared memory size you chose with the 'sharedmem' chatscript option. This start address will in general be different on different machines.

- dumpevery #

  write all spectra to the map file every so many minutes. (Note: you can always dump spectra using 'pkill -HUP GSSort' or update() from inside rootn.exe)

  example

  dumpevery 10

  warning: program may only respond if it has data flowing to it.

- printevents #

  number of GammaSphere events to print out in glory details (including FERA interpretation if any). Good for debugging the electronics... and seeing conditions and Pseudo event vector etc.

  example

  print 200

- egemin e0

  set the minimum energy of germaniums before they are accepted in the sort. The cut is applied to the raw data before Doppler corrections of compression with the option "hiresdatamult". Thus, it will influence the multiplicity cuts (cleanmultlim dirtymultlim totalmultlim). E.g, this option could be used to cut x-rays.

  example

  egemin 20

- beta #

  specify the beta=v/c of the recoil residues.

  example

  beta 0.02650

- echo

  if set, the program will echo the chatscript instructions from the point where echo is mentioned

  example

  echo

- nevents #

  specify the max numbers of events to sort

  example

  nevents 500000

9 GSUtil commands

This is a generic listing of commands in GSUtil. Not all of them the are relevant for GTSort at this point.

A number of commands are available to 'control' the GSSort ot GTSort sorting program from inside rootn.exe without having to go into another session window. They are typically accomplished by signalling GSSort using the unix utility "pkill".

Here is a list of the commands that are available when you load GSUtil (.L GSUtil_cc.so):

- startsort("sortengine","chatfile", "logfile");

  start GSSort (= specified version of GSSort) with chatfile "chatfile". A window will pop up and show the output from GSSort. The output will also be logged to "logfile". Note: YOU MUST RELOAD THE SHARED MEMORY FILE AGAIN AFTER A RESTART OF THE GSSORT SORTER TO SEE THE NEW DATA.

  Example

  startsort("GSSort_3.10.02","gsfma137.chat","gsfma137.log")

- stopsort()

  Stops GSSort gracefully, with a data dump, sorting statics etc

- killsort()

  Stops GSSort cold (KILL -9)

- update()

  Signals GSSort to update the shared memory. ROOT does not constantly update the map file, so you need to ask the sorting engine to update the map file so that we can display the latest data in rootn.exe. See setdumpinterval for automatic updates.

- setdumpinterval(dt)

  Instructs GSSort to set the time between automatically updating the shared memory (or diskfile) to dt minutes.

  Example

  setdumpinterval(5)

  ask GTSort/GSSort to update the map file every 5 minutes.

- zapall()

  zaps all spectra in the shared memory. This command has no effect if you are just looking at an old map file, i.e., not sorting.

- zap("spname")

  zaps spectrum "spname" in shared memory. This command has no effect if you are just looking at an old map file, i.e., not sorting.

- printevents(numberOfEvents)

  print events in the sorting window. Good for debugging electronics or just see that the events are ok.

- setbeta(beta)

  change the beta value in the sort (on the fly). After the change, you will want to use the zap command.

- zapcounters()

  Tells GSSort to zap its counters so it is easier to see if there are problems with the data as it is coming in right now.

- ;

- ;

- sload("name")

  load (or rather attach) the shared memory "name" to the rootn.exe session. It will do a close() first. Set "name" to whatever you called the shared memory map using the 'sharedmem' chatscript option.

  NOTE: whenever you restart your sorting program [stopsort(), startsort("","","")], you must do an sload again or you will be looking at old stuff...

- dload("name")

  read the root disk file with called "name" into the rootn.exe session. It will do a close() first.

- dmake("name")

  create root disk file "name", It will do a close() first.

- close()

  will close the shared memory or diskfile

- mkcanvas()

  set up "c1" canvas with useful properties. If an old one exists, it will be deleted before a new canvas is set up. The name of the canvas will be "c1".

- ls()

  list the spectra available in the rootn.exe shared memory or disk file.

- d1("name",lo,hi)

  display 1D spectrum between lo and hi channels. Destroys old displayed spectrum.

- d1o("name",lo,hi)

  as d1() but does not destroy old displayed spectrum. I.e., it will overlay the spectrum.

- md1("List","GenNam",xmin,xmax[,wait])

  utility to overlay many spectra. "List" contains a list of the module numbers to display in the format shown in the example below. "GenNam" is the generic name of the spectra, I.e., in the example below, the spectra will be: ehi005,ehi007,ehi008,ehi009, etc. xmin and xmax are the display limits to use.

  Example

  md1("7-9,11-52,54-57,59-109","tge",3700,4200,0);

  The y scale is set by the first spectrum (so make sure that one has counts). ¡br¿ The meaning of wait is wait=0: don't wait; true overlay; wait=1: overlay, but you must hit return for every spectrum; wait=2: display each spectrum clearing every time (movie like display); wait=3: as 2, but you hit return for each spectrum;

  Note: the overlay options: wait=0 and wait=1 can be quite slow if you display a lot of spectra. The 'movie' versions of the example above:

  md1("7-9,11-52,54-57,59-109","tge",3700,4200,2);

  is much faster.

- showlimits

  shows that x and y limits are set for 1D displays

- setxlimits(lo,hi)

  set the x limits from lo to hi

- unsetxlimits

  set x limits to 'auto'

- setylimits(lo,hi)

  set the y limits from lo to hi

- unsetylimits

  set y limits to 'auto'

- d2("name",xlo,xhi,ylo,yhi)

  display 2D spectrum between lo and hi channels indicated.

- ft1("name",lo,hi,p1)

  fit a single peak, at channel p1, with a quadratic background between the limits of lo and hi. You must display the spectrum first using the d1() command. This function is not much more than a prototype at the moment, it needs more work to be really useful. We also need to be able to fit more than one peak.

- pjx("2Dname","proname",lo,hi)

  project on the x-axis from 2D histogram "2Dname" with y-axis gate between lo and hi. Final spectrum is called "proname"

- pjy("2Dname","proname",lo,hi)

  project on the y-axis from 2D histogram "2Dname" with x-axis gate between lo and hi. Final spectrum is called "proname"

- gate("name","pname",p1,w1,bf)

  extract a gate from lo=p1-w1 to hi=p1+w1 on the gamma-gamma matrix called "name" and subtract "bf" fraction of the total projection. Write the result into spectrum "pname". If bf=1.0, the resulting spectrum will have no net counts. If bf=0.0, the resulting spectrum will not have any background subtracted.

- wrspe("root name","spe name")

  write 1D root spectrum spectrum "root name" out in Radford spe format as spectrum "spe name".

- rdspe("spe name","root name")

  read "spe name" 1D spectrum in Radford spe format into root as spectrum "root name".

- rdmat("name")

  read a matrix in Torben mat format into root.

- tgealign("old","new",pos,w1)

  find the germanium time offsets to line up all peaks in position "pos". "old" is the name of the file that contains the current offsets and "new" is the name of the new offset file. "w1" is the number of channels around the maximum channel used to find a mean position of the time peak. If "old" is set to "DUMMY", no old file is read in and the offsets are assumed to be zero.

- tbgoalign("old","new",pos,w1)

  find the BGO time offsets to line up all peaks in position "pos". "old" is the name of the file that contains the current offsets and "new" is the name of the new offset file. "w1" is the number of channels around the maximum channel used to find a mean position of the time peak. If "old" is set to "DUMMY", no old file is read in and the offsets are assumed to be zero.

- ehialign(oldf, newf, kevch, sgain, w1, source)

  find the germanium hires energy calibration parameters using a 207Bi source for calibration (Energies: 569.702 keV and 1063.662 keV). "oldf" is the old calibration parameter file and "newf" is the file to which the new calibration parameters are written. "kevch" is the desired gain in kev/ch. "sgain" is the gain that you specified in the GSSort chatfile with "hiresdatamult". "w1" is the number of channels around the max channel of the peaks that you want to use to define a mean position of the two source peaks (typically 2-3). Source must be one of "207Bi", "60Co" or "88Y".

  examples

  ehialign("DUMMY","x.cal",0.666666666, 0.66666666, 6, "207Bi")
  ehialign("/sga/config/current_ehi.cal","x.cal",0.666666666, 0.66666666, 6, "207Bi")
  ehialign("/sga/config/current_ehi.cal","/sga/config/current_ehi.cal",0.666666666, 0.66666666, 6, "207Bi")

- eloalign(oldf, newf, kevch, w1, source)

  find the germanium lores energy calibration parameters using a 207Bi source for calibration (Energies: 569.702 keV and 1063.662 keV). "oldf" is the old calibration parameter file and "newf" is the file to which the new calibration parameters are written. "kevch" is the desired gain in kev/ch. "w1" is the number of channels around the max channel of the peaks that you want to use to define a mean position of the two source peaks (typically 2-3). Source must be one of "207Bi", "60Co" or "88Y".

- esidealign(oldf, newf, kevch, w1, source)

  find the germanium side channel energy calibration parameters using a 207Bi source for calibration (Energies: 569.702 keV and 1063.662 keV). "oldf" is the

old calibration parameter file and "newf" is the file to which the new calibration parameters are written. "kevch" is the desired gain in kev/ch. "w1" is the number of channels around the max channel of the peaks that you want to use to define a mean position of the two source peaks (typically 2-3). Source must be one of "207Bi", "60Co" or "88Y".

examples

esidealign("DUMMY","eside.cal",2.5, 1.0, 6, "207Bi")

- del("name")

  delete a TH1D (1 dimensional) spectrum. If it resides in shared memory the utility quietly ignores you since you cannot delete shared memory spectra from inside rootn.exe.

- add1("h1","h2",af)

  add TH1D (1 dimensional) spectrum "h2" to "h1". If "h1" does not exist, it will be created with the same x-axis as "h2". Optionally 'af' can be specified as the factor to multiply "h2" with before addition (this allows for subtraction of spectra using negative 'af' values). Use del("h1") to delete "h1" first if you want to add up from scratch.

- version()

  display rcs version number of src file

- mk2dwin(histname)

  display the 2D histogram "histname" and allow the user to create a 2Dwindow (bananagate) on it. Use "save2dwin" to save the 2dwin.

- save2dwin(winname)

  save the 2Dwindow (bananagate) created with "mk2dwin" to a file so that it can be read by GSSort and used to gate spectra. "winname" is both the diskfile name as well as the 2dwin name.

- d2dwin(winname,hist)

  utility to display the 2dwin "winname" on top of the 2D histogram "hist". The 2dwin is read from the diskfile "winname".

- lsrootfile(histname)

  list the content of a rootfile. Will also list the content of a 2dwin file; but it cannot list the content of a shared memory map file.

- confirm()

  function used to temporary halt execution in a script file. E.g., to make a number of 2d windows and write them out you can have a sequence of commands like:¡br¿ mk2dwin(histname);confirm();save2dwin(winname);..... ....oops, this doesn't work yet. Not sure why!

- ;
- ;
- ;
- ;

10 User include files

To facilitate the user being able to add their own sorting code to GTSort/GSSort, a number of include files are embedded in the GTSort code. If you adhere to the practice of keeping your own code additions in these file you will be able to fetch and use a new version of the official GTSort code (with bug fixes and improvements) without much trouble.

Search for include points as

```
grep -n "#include " G?Sort.cxx | grep User
```

11 Todo list

1. add external data processing
2. add pseudo vector processing (wait for external data)
3. coincidence events!?: done in GTSort or processor farm?? need better data for development
4.
5.
6.

12 Misc.

Index