# GRESS User's Guide

Revision 7
November 17, 2005

Andrew S. Hoover
Los Alamos National Laboratory

Contents

1. Software Package Overview
    1.1 Architecture
    1.2 Software Dependancies
2. Geometry Model Input
3. Using gbmsim
    3.1 Example Macros
    3.2 Initialization, Visualization, and /gps Commands
    3.3 /gbm commands
    3.4 OutputFile
4. Using calsim
    4.1 calsim Commands
    4.2 Calibration Parameters
    4.3 Output File
5. Summary of Commands
    5.1 GDML File Flags
    5.2 gbmsim Commands
    5.3 calsim Commands
        5.3.1 calsim Macro Commands
        5.3.2 calsim Calibration Input File Commands

# 1. Software Package Overview

The GBM REsponse Simulation System (GRESS) code uses monte carlo simulation to determine the GBM detector response to source photons. The code records energy deposits within the detector and applies calibration effects to the data. It is designed to provide flexibility in the choice of source position and input energy spectra.

## 1.1 Architecture

The general architecture of the GRESS code is shown in the diagram below. At this point, gbmsim includes the instrument simulation; the spacecraft simulation is in development.

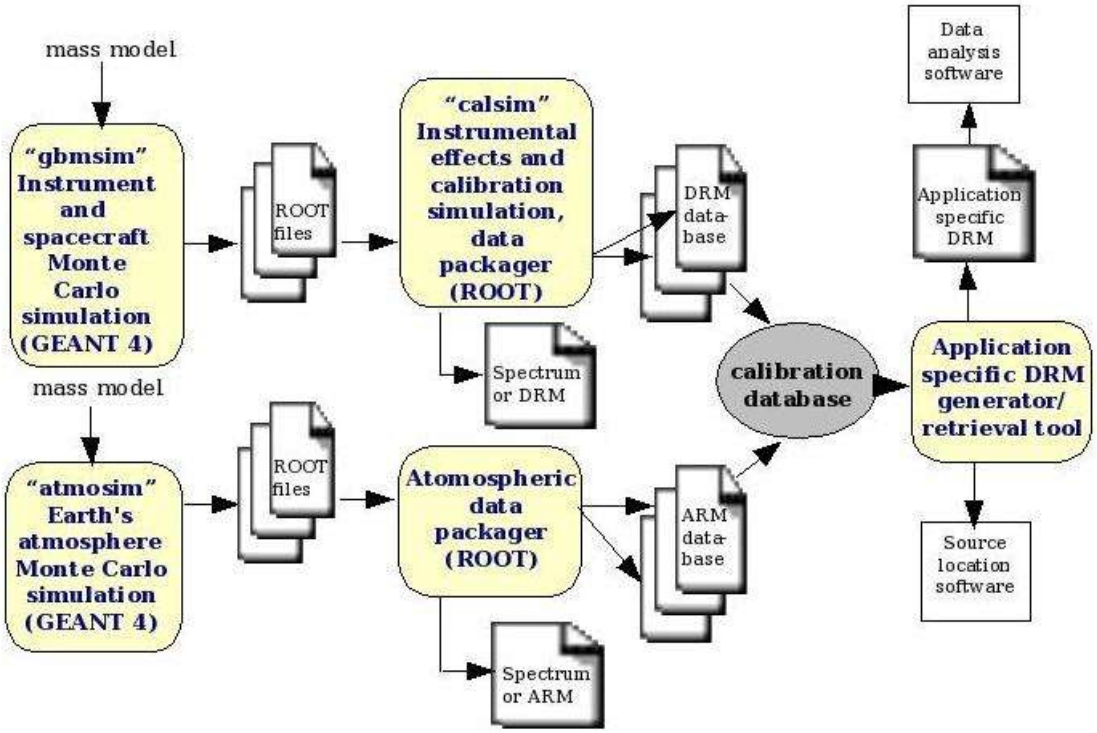


*Figure 1- Architecture of the GRESS code.*

The GBM simulation code is contained within several directories: /gbmsim, /calsim, /ggdml, /g4lecs, and /atmosim. /gbmsim contains detector models and code to run the GEANT4 monte carlo. /calsim contains code to read in the output from gbmsim, apply calibration effects like energy resolution and thresholds, and generate spectra or DRMs. /ggdml has source code for the geometry description language. /g4lecs contains a low-energy compton scattering enhancement to the GEANT4 code, called G4LECS (http://public.lanl.gov/mkippen/actsim/g4lecs/). /atmosim is a preliminary version of the earth atmosphere scattering simulation, which uses data about the earth's atmosphere from the /nrlmsise00 directory. This guide currently explains the use of /gbmsim and /calsim, and will expand as the code is developed.

### 1.2 Software Dependancies

The software has several external dependancies. It requires the GEANT4 simulation toolkit (http://geant4.web.cern.ch/geant4/). ROOT (http://root.cern.ch/) and FITS (http://heasarc.gsfc.nasa.gov/docs/heasarc/fits.html) files are used to store simulation data. We use CCfits (http://heasarc.gsfc.nasa.gov/docs/software/fitsio/CCfits/) , a c++ interface for writing FITS files. CCfits is a wrapper for cfitsio (http://heasarc.gsfc.nasa.gov/docs/software/fitsio/fitsio.html) routines. Correct compilation of the code requires that several environment variables have been set in addition to the usual GEANT4 variables:

$GRESS_DIR: top level directory for GBM simulation code
$ROOTSYS: ROOT location
$CCFITS_DIR: CCfits location
$CFITSIO_DIR: cfitsio location
$XERCESCROOT: xerces-c location
$CLHEP_BASE_DIR: clhep location

The code has been developed on a Linux platform using gcc 3.2.3, ROOT version 4.04/02, CCfits version 1.3, and cfitsio version 2.47. It has also been shown to work on MacOS X 10.3.5.

Many users may not be familiar with ROOT. It is a well-supported object-oriented software package from CERN, popular in the high-energy physics community. It contains many types of data structures for storing, analyzing, and presenting data, as well as a flexible file format which can contain directory structures.

## 2. Geometry Model Input

The Geometry Description Markup Language (GDML) is used to define the simulation models. GDML is an XML-style language that interfaces to GEANT4. Documentation can be found at http://savannah.cern.ch/projects/gdml/ . Several enhancements have been made to the standard GDML software, and we have used the extension "ggdml" as a reminder that we are using modified code. The modified GDML software is contained in the /ggdml directory. Great care has been taken to create accurate and detailed models of the GBM detectors. Currently, two models are provided in the /gbmsim/Models directory: NaiModel_v1.0.ggdml allows one to use the NaI detector, and BgoModel_v1.0.ggdml allows one to use the BGO detector. The user will generally be concerned with several tags belonging to the "volume" object. These tags are an enhancement to the standard GDML and allow the user to input information concerning triggering, sensitive volumes, and visualization attributes. One should look into the .ggdml files to see how they are used. These tags are described below:

sdflag : Sensitive detector flag (boolean); the simulation code will store energy deposits for all detector volumes that have sdflag set to true. For the default models, the NaI crystal and BGO crystal are set as sensitive volumes.

trigflag: Trigger flag (boolean); the simulation code can be made to accept only events where energy is deposited in a specific detector or detectors. When trigflag is set to true, the volume will be added to the list of possible triggering volumes. A triggering volume must also be declared as a sensitive volume. For the default models, the NaI crystal and BGO crystal are set as triggering volumes.

ethresh: Energy threshold (double); for volumes which have been declared triggering volumes via trigflag, the energy threshold is set with the ethresh tag, where the energy is to be given in keV. If an energy is deposited in a triggering volume which is greater than the threshold for that volume, the event will be accepted. For the default models, the energy thresholds for the NaI and BGO crystals are set at 0 keV.

visible: (boolean) If set to true, the object is visible when visualization is used, if set to false, the object is not visible. Some visualizers, like DAWNFILE, might override this.

solid: (boolean) If set to true, the object is solid when visualized, if set to false it is wireframe. Some visualizers, like DAWNFILE, might override this.

color: (3 doubles) Determines the color of the volume when it is visualized.

## 3. Using gbmsim

After compiling the software, an executable file called "gbmsim" will be created in the /bin/Linux-g++ directory. Making the simulation software do something can be accomplished in several ways. If one simply executes the gbmsim binary file, the GEANT "PreInit>" prompt will appear and wait for the next command. This mode of operation will probably not be useful except for users with detailed understanding of how the GEANT user interface commands work. Most of the time, simulation jobs will be controlled with an input macro which contains a list of commands to set up the source distribution, control visualization, and control the run parameters. Several example macros are provided in the /gbmsim/Macros directory.

Macros can be run in one of two ways: run the "gbmsim" executable and enter "/control/execute <macroname>" at the Preinit> prompt, or give the macro name as a command line input when running gbmsim (e.g. - "gbmsim <macroname>"). Make sure to give the correct path names to your executable or macro, depending on what directory you are running in (or add /bin/Linux-g++ to your $PATH). The main difference between the two methods of running is that the visualization window will remain displayed and you will be returned to the GEANT "Idle>" prompt awaiting the next command in the former case, whereas the program will simply terminate, destroy any visualization windows, and return to the system shell prompt in the latter case. Both methods are useful depending on what exactly you are trying to do.

As can be seen in the run1.mac file, the commands are in a directory style structure. Many of the commands are standard GEANT4 options, and many of them are custom commands which have been added for the GBM simulation.

Some enhancements were made to GEANT4's general particle source commands, contained in the /gps directory. The /gbm directory contains more commands that were added specifically for the GBM simulation. Usually, information about the command and what its arguments are can be obtained by using the help command, for example "help /gps/particle".

### 3.1 Example Macros

Several example macros are provided in the gbmsim/Macros directory to help the user understand how to run the code. A brief description of what each macro does is given below, followed by an explanation of the commands used in the macros in sections 3.2 and 3.3.

PlaneWaveSource.mac: Sets up a plane-wave source.
PointSource.mac: Sets up a point source.
run1.mac: Throw 100 photons at the NaI detector.
run2.mac: Throw 100 photons at the BGO detector.
run3.mac: Collect 100 triggered events for the NaI detector. Only events which deposit energy in the NaI crystal are accepted.
run4.mac: Use a Band's function spectrum. The number of events is determined via the "flux" event type by the Band function normalization and the source area.
run5.mac: Source energy varies from event to event and is defined by a user-input histogram.
run6.mac: Have multiple runs with different energies and put them in the same output file.
run7.mac: Translate a plane-wave source without affecting its rotation to irradiate a small portion of detector.
NaiDrm.mac: Simulate multiple energies to make a DRM for the NaI detector.
BgoDrm.mac: Simulate multiple energies to make a DRM for the BGO detector.

### 3.2 Initialization, Visualization, and /gps commands

/run/initialize

This command initializes GEANT and loads the geometry.

```
/vis/open OGLIX
/vis/viewer/set/viewpointThetaPhi 90 180 deg
/vis/viewer/zoom 0.5
/vis/viewer/set/lineSegmentsPerCircle 100
/vis/drawVolume
/vis/viewer/flush
/tracking/storeTrajectory 1
/vis/scene/endOfEventAction accumulate
```

This set of commands controls the visualization of the simulation. Users should consult the GEANT4 manual for details. Users may prefer the DAWNFILE visualization tool over OGLIX in certain situations. These lines can be removed or commented out with the # character if

visualization is not desired.

```
/gps/particle gamma
/gps/pos/type Plane
/gps/pos/shape Circle
/gps/pos/radius 10.0 cm
/gps/ang/type iso
```

These commands define the source particle type, shape, and angular distribution. Here we have defined a gamma source in the shape of a plane circle with a radius of 10 cm. The angular distribution has been defined as isotropic.

```
/gps/pos/srcdistance 60. cm
/gps/pos/srctheta 0. deg
/gps/pos/srcphi 0. deg
```

These commands define the position of the center of the source distribution. When using these commands, the source will automatically be rotated such that its face is orthogonal to the vector joining the center of the source to the origin (this really only makes sense for Plane sources). The /gps/position command can also be used if one prefers to use x,y,z coordinates instead of spherical coordinates, but the source will not be automatically rotated in that case.

```
/gps/ang/mintheta 0.0 deg
/gps/ang/maxtheta 0.0 deg
/gps/ang/minphi 0.0 deg
/gps/ang/maxphi 0.0 deg
```

These commands confine the photons emitted from the source to a range of theta and phi. This is useful to avoid throwing photons which will not hit the detector. In this case, the angles have been set such that the photons are emitted orthogonally to the face of the plane circular source (e.g. - a plane wave).

```
/gps/ene/type Mono
```

This command defines a monoenergetic source. Several options for the energy type in addition to Mono are provided as part of the standard /gps package. Two additional options have been added for GBM: "DiscUser" and "Band".

The "DiscUser" option allows the user to define an arbitrary number of energies with a corresponding probability. For example, if one wanted the source to be 100 keV 70% of the time, and 200 keV 30% of the time, this can be accomplished via:

```
/gps/ene/type DiscUser
/gps/hist/type energy
/gps/hist/point 0.1 0.7
```

/gps/hist/point 0.2 0.3

The first command specifies the DiscUser energy type. The second line indicates we are using a histogram with energy values and not momentum values. The third and fourth lines give the energy in MeV (first number) and the probability for the source to have that energy (second number). If the probabilties do not add to 1, they will be renormalized by the sum.

The "Band" option allows the user to set up a source with a spectrum from the GRB Band function.

$$N_E(E) = AE^{\alpha}\exp\left(-\frac{E}{E_o}\right), \quad (\alpha-\beta)E_o \geq E$$

$$= A[(\alpha-\beta)E_o]^{\alpha-\beta}\exp(\beta-\alpha)E^{\beta}, \quad (\alpha-\beta)E_o \leq E$$

A Band function source can be setup as follows:

```
/gps/ene/type Band
/gps/ene/bandalpha -1
/gps/ene/bandbeta  -2.3
/gps/ene/bandezero 230.0 keV
/gps/ene/min 40 keV
/gps/ene/max 10 MeV
```

Here, a Band function is set up with $\alpha$=-1, $\beta$=-2.3, $E_o$=230 keV, with a range from 40 keV to 10MeV. If the user runs in "/gbm/evttype flux" mode (see below) with a differential spectrum, the normalization parameter A should also be defined as follows:

```
/gps/ene/diffspec true
/gps/ene/bandnorm  <normalization constant>
```

The first command indicates that you are assuming a differential spectrum, and the second command sets the normalization constant A.

/gps/energy 200.0 keV

This command sets the energy for the monoenergetic source to 200 keV.

/run/beamOn 100

This command instructs the simulation to run for 100 events. If the "thrown" event type is used, 100 events will be thrown. If the "triggered" event type is used, events will be thrown until 100 events have satisfied the trigger conditions. If "flux" is used, the numerical argument to /run/beamOn is not necessary since the number of events to throw is calculated based on the energy spectrum integral if /gps/ene/diffspec true is used, or from the flux input by the user via /gps/setflux if

/gps/ene/diffspec false is used.

### 3.3 /gbm commands

/gbm/ggdmlfile <filename>

This command specifies the geometry input file to use. It is assumed the file is located in the /gbmsim/Models directory.

/gbm/rootfile <filename>

This command specifies the name of the ROOT file that will store the output data. ROOT files usually have the .root suffix.

/gbm/rndmseed <number>

This command specifies the random number seed (unsigned long) to be used.

/gbm/evttype <option>

This command has three options: "thrown", "triggered", and "flux". If "thrown" is selected, all events will be processed and stored in the output file. If "triggered" is selected, only events which have energy in a triggering detector above the set threshold well be stored in the output file. For Plane source types there is also a "flux" option. The "flux" option behaves in two different ways depending on whether a differential spectrum is used. For example,

```
/gps/ene/diffspec false
/gbm/evttype flux
/gps/setflux 1000
```

will assume an integral spectrum and the number of events will be the source area times the flux the user has entered in #events/mm$^2$ (1000 events/mm$^2$ in this example). If a differential spectrum is to be used, the commands

```
/gps/ene/diffspec true
/gbm/evttype flux
```

will cause the number of events to be computed as the area of the source times the integral of the energy spectrum.

/gbm/multiE <energy> <#events>

This command allows the user to have multiple runs at different energies stored in the same output file. For example, the following series of commands:

```
/gbm/multiE 0.1 100
/gbm/multiE 0.2 100
```

```
/gbm/multiE 0.5 100
/gbm/evttype triggered
/run/beamOn
```

will cause the simulation to execute three seperate runs at 100, 200, and 500 keV, with 100 triggered events for each. The first argument is the energy in MeV, the second is the number of events at that energy. The random number seed will be set to the value input by the user with the /gbm/rndmseed command for the first run, and will be be incremented by 1 between each subsequent run.

/gbm/pointsource <dist> <unit> <theta> <phi> <mintheta> <maxtheta><unit> <x> <y> <z> <unit>

It is anticipated that users will frequently desire to use a simple point photon source. This command simplifies setup of this source. A disc of diameter 1 mm is actually used instead of a point source for technical reasons. The required arguments to this command are the distance to the source, the theta and phi values defining the position of the source, and the min/max theta angles which allow the user to limit the photons to a cone. The min/max theta angles are defined relative to the vector connecting the world origin to the location of the point source (before any translation, see below...). If the user enters a value of -1 for the distance, then the distance to the source will be taken from a pre-defined variable in the GDML file. The user still needs to define the source energy distribution type when using this command. Optional arguments x,y,z can be used to translate the source location without changing the direction of emitted photons.

/gbm/pointtargetsource <dist> <unit> <theta> <phi> <mintheta><maxtheta> <unit><x> <y> <z> <unit>

Same as /gbm/pointsource, except that x,y,z specify an arbitrary target point where the source will be aimed.

/gbm/planewavesource <radius> <distance> <unit> <theta> <phi> <unit> <x> <y> <z> <unit>

It is anticipated that users will frequently desire to use a simple plane wave photon source. This command simplifies setup of this source by automatically setting the particle type, source shape, source radius, source angular limits, and source rotation in the correct way to achieve a photon source eminating from a Plane Circular distribution.
The required arguments to this command are the radius of the source distribution, the distance to the source distribution, and the theta and phi values defining the location of the center of the source, each with a unit. If the user enters a value of -1 for the radius and/or distance, then the radius and/or distance of the source will be taken from a pre-defined variable in the GDML file (the pre-defined radius is sufficient to irradiatate the entire detector from any angle). The source distribution will always be automatically rotated such that the incident photons are in the direction of the vector connecting the world origin to the center of

the source distribution. The user still needs to define the source energy distribution type when using this command.
Optional arguments x,y,z can be used to translate the source such that the emitted photons will be centered on this point (the rotation of the source will not be affected).

/gbm/planewavetargetsource <radius> <dist> <unit> <theta> <phi> <unit> <x> <y> <z> <unit>

Same as /gbm/planewavesource, except that x,y,z specify an arbitrary target point where the source will be aimed.

### 3.4 Output File

The output from gbmsim is a ROOT file. Several ROOT data structures are contained in the file. The structure called "Header" contains information about the run. The Header is a TTree (a data structure similar to an Ntuple) and has the following branches which are filled once at the end of the run:

| | |
|---|---|
| ggdmlfile: | name of GDML file used |
| macrofile: | name of macro file used |
| codever: | code version |
| codedate: | code date |
| sysinfo: | system information |
| time: | time job was run |
| elaptime: | elapsed time for job |
| user: | user who ran the job |
| rndmseed: | random number seed used |
| mode: | thrown, triggered, or flux mode |
| nthrown: | number of thrown events |
| naccept: | number of accepted events |
| srcarea: | area of source ($mm^2$) |
| r: | radius of source (mm) |
| theta: | theta angle of source (rad) |
| phi: | phi angle of source (rad) |

A TObjArray structure called "CommandHistory" contains a list of the commands contained in the macro used to execute the simulation run.

Another tree called "EventInfo" is written to the output file. It has the following branches which are filled for each event:

| | |
|---|---|
| incidentE: | incident particle energy |
| incidentx: | incident particle start position x |
| incidenty: | incident particle start position y |
| incidentz: | incident particle start position z |
| incidentpx: | incident particle start $p_x$ |
| incidentpy: | incident particle start $p_y$ |
| incidentpz: | incident particle start $p_z$ |
| firstinttype: | first interaction type (e.g. - compton, photoelectric, etc...) |
| firstintvol: | first interaction volume |

Finally, a TTree containing energy deposit information is written to the output file for each volume which is declared sensitive via the "sdflag" parameter in the GDML file. These trees have the following branches, filled for each event:

etot: total energy deposit for the volume
ex: $\Sigma E_i x_i$ ,sum of energy times x-position for each step in the volume
ey: $\Sigma E_i y_i$ ,sum of energy times y-position for each step in the volume
ez: $\Sigma E_i z_i$ ,sum of energy times z-position for each step in the volume
et: $\Sigma E_i t_i$ ,sum of energy times time for each step in the volume

The calsim code is designed to read and analyze output files from gbmsim, but a sample ROOT macro is provided in /gbmsim/analysis/readdata.C, which demonstrates how to read data from the TTree structures. The macro can be run from the ROOT command prompt and requires a filename as input, e.g.-

```
analysis% root
root [0] .x readdata.C("../data.root")
```

where the input file is a file generated as output from gbmsim. The ROOT macro will dump the command history to the screen and histogram the energy deposits in the NaI crystal volume.

## 4. Using calsim

calsim code is located in the /calsim directory. The purpose of this code is to read in data files produced by gbmsim, apply calibration effects, and generate energy spectra or DRMs. The usage of calsim is similar to gbmsim: it can run in command line mode or macro mode. When calsim is compiled an executable file will appear in /calsim/bin. Executing this file will start calsim in "interactive mode", where commands are entered one at a time by the user. Usually it is easier to run calsim by specifying a macro which contains several commands. The input to calsim is the ROOT file generated by gbmsim. The output is a ROOT file and a FITS file, both containing the energy spectrum or DRM. The commands used by calsim are described in section 4.1. Parameters describing the energy resolution and threshold are contained in a simple text file, explained in section 4.2.

### 4.1 calsim Commands

A list of calsim commands can be displayed by starting calsim in interactive mode and typing "help":

```
> help

AnalyzeData      : Do the analysis.
CalibFileName    : Specify the calibration parameter input file.
Draw     : Flag to turn graph drawing on
DumpCmdHistory   : Dump the command history from the ROOT input file to a text
                   file.
EpeakBins        : Plot Emeasured/Epeak instead of Emeasured on x-axis
```

```
FitsFileName     : Specify the name of the FITS output file.
HistogramBins    : Define measured energy (x-axis) histogram bins (define
                   min,max and #bins)
HistogramBins2   : Define measured energy (x-axis) histogram bins (define min,
                   bin width, and #bins).
HistogramType    : Define histogram binning type for x-axis and y-axis (Lin or Log)
InputFileName    : Specify the name of the input data file (ROOT format)
ListVolumes      : List names of volumes contained in the input file.
MakeResponseMatrix      : A flag to indicate you want to make a response matrix
                          from the ROOT input file.
RandomSeed : Set the random number generator seed for calsim.
ReadAppHeader    : Print the application specific header from the ROOT input file.
RootFileName     : Specify the name of the ROOT output file.
ReadROOTHeader   : Print information about the ROOT input file.
Volume   : Volume you want to analyze data for.
exit     : Exit interactive mode.
help     : Print this information.
macro    : Run a macro.
```

The commands are described below:

AnalyzeData

Command to open the input data file and perform the analysis. This should typically be the last command in a macro file.

CalibFileName <filename>

Specifies the name of the file containing the calibration parameters.

Draw

Command to enable drawing of graphs. Graphs are not drawn by default.

DumpCmdHistory <filename>

Dumps the command history from the gbmsim file to the specified text file.

EpeakBins

Plots $E_{measured}/E_{incident}$ on x-axis instead of $E_{measured}$.

FitsFileName <filename>

Name of the output FITS file.

HistogramBins <min> <max> <#bins>

Defines the binning used for measured energies (x-axis). min and max are in keV. y-axis bins are determined from incident energies used in the simulation. WARNING: If using log binning, don't use 0 as the minimum

value.

HistogramBins2 <min> <binwidth> <#bins>

Defines the binning used for measured energies (x-axis). min and binwidth are in keV. y-axis bins are determined from incident energies used in the simulation. WARNING: If using log binning, don't use 0 as the minimum value.

HistogramType  <xaxis type> <yaxis type>

Indicates whether to use linear ("Lin") or logarithmic ("Log") binning for the x and y axes.

InputFileName <filename>

Specifiy the name of the file containing the gbmsim data.

ListVolumes

List the names of volumes for which this file contains data.

MakeResponseMatrix <true/false>

A flag to indicate if you want to make a response matrix from the input file data.

ReadAppHeader

Displays the contents of the Header tree on the screen for each directory in the input file.

ListRootFile

Displays the contents of the input file top directory.

RandomSeed <seed>

Set the random number generator seed.

ReadAppHeader

Read the header containing information about the gbmsim run and dump it to the screen.

RootFileName <filename>

Name of the output ROOT file.

Volume <name>

The name of the volume you want to analyze data for. This name should

be the GEANT4 physical volume name assigned by gbmsim. The volume names for the NaI and BGO crystals are "pv_NaiCrystal_0" and "pv_BgoCrystal_0". You can display a list of names of the volumes in the input file with the "ListVolumes" command.

exit

Exit the session.

help

Print the help information.

macro <filename>

Run a macro file containing commands.

A typical macro to make a DRM from input file "naidrm.root", using calibration constants in "calib.in" for volume "pv_Crystal_0" might look like this:

```
#----Setup----
InputFileName naidrm.root
CalibFileName   calib.in
RootFileName   out.root
FitsFileName     out.drm
HistogramType Log Log
HistogramBins 0 500 250
Volume pv_NaiCrystal_0
Draw
MakeResponseMatrix

#----Execution----
ListRootFile
ReadAppHeader
DumpCmdHistory  cmdhist.txt
AnalyzeData
```

Any line in the macro file beginning with # will be ignored. Do not put tab characters in the commands.

### 4.2 Calibration Parameters

Instrumental effects are applied to the simulation data according to parameters described in the file specified by the "CalibFileName" command.
Energy resolution effects and an energy threshold can be defined by placing the following commands in the calibration input file:

EresParam 42.6 1 0

This command will cause the energy deposit values to be smeared by a value picked at random from a Gaussian distribution where the sigma

of the Gaussian function is defined as

$$\sigma = \sqrt{p_0^2 + p_1^2 E + p_2^2 E^2}$$

The three arguments to the command are $p_0$, $p_1$, and $p_2$.

```
EresPoint 100 10
EresPoint 200 7
EresPoint 400 5
EresPoint 500 4
```

These commands will cause the energy deposit values to be smeared by a value picked at random from a Gaussian distribution where the sigma of the Gaussian function is determined from a user input pointwise histogram of values. The first argument to the command is the energy in keV, the second is the sigma value of the energy resolution in keV at that energy. A linear interpolation is used to determine the value of sigma for each event.

```
Ethresh 50.0 1.0
```

A lower energy threshold can be applied with these commands. A value for the threshold is selected at random from a Gaussian function having its mean at the energy specified with the first argument of the "Ethresh" command and a sigma as specified with the second argument of the "Ethesh" command. Both numbers are expected to be in keV. If the unsmeared energy is greater than the smeared threshold, the smeared energy value is accepted.

### 4.3 Output File

The energy spectrum or DRM generated by calsim is written to a ROOT file and also to a FITS file. The FITS file format generated by calsim is currently evolving but the intention is to make it the same as the real data format. Currently, the FITS file contains three tables. For single energy spectra (PHA files), the first table contains the energy bounds, and the second table contains the bin contents. For DRM files, the first table contains the incident energy bin information, and the second table contains a zero-supressed DRM. In both cases, the third table contains a history of gbmsim and calsim commands.

## 5. Summary of Commands

### 5.1 GDML File Flags

| Flag | Arguments | Description |
|---|---|---|
| sdflag = bool | Boolean bool | Designates a volume as being sensitive. Data will be stored in the output file for sensitive volumes. |
| trigflag = bool | Boolean bool | Designates a triggering volume (must also be declared sensitive). |
| ethresh = energy | Double energy, in keV | Energy threshold for triggering volume. |
| visible = bool | Boolean bool | Determines whether a volume is visible when visualization is used. |
| solid = bool | Boolean bool | Determines whether a volume is solid or wire frame when visualization is used. |
| color = col1, col2, col3 | Float col1, col2, col3 | Determines the color of an object when it is visualized. |

### 5.2 gbmsim Commands

Note: this table contains only commands which have been added to the standard GEANT4 commands or GEANT4 commands that have been modified. The user should consult the GEANT4 manual for complete descriptions of the multitude of possible commands.

| Command | Arguments | Description |
|---|---|---|
| /gps/pos/srcdistance <dist> <unit> | Double dist and string unit. The units can be nm, um, mm, cm, m, pc. | Sets the distance to the center of the source distribution. |
| /gps/pos/srctheta <theta> <unit> | Double theta and string unit. Units can be deg or rad. | Sets the theta angle to the center of the source distribution, and automatically rotates the source to point at the world origin. |
| /gps/pos/srcphi <phi> <unit> | Double phi and string unit. Units can be deg or rad. | Sets the phi angle to the center of the source distribution, and automatically rotates the source to point at the world origin. |
| /gps/ene/type <EnergyDis> | String EnergyDis. Valid arguments are Mono, Lin, Pow, Exp, Gauss, Brem, Bbody, Cdg, User, DiscUser, Arb, Epn, and Band | Sets the energy distribution type. DiscUser (user specified discrete histogram) and Band function have been added to the standard GPS options. |

| ***Command*** | ***Arguments*** | ***Description*** |
|---|---|---|
| /gps/ene/bandalpha <alpha> | Double alpha | Sets alpha for the Band function. |
| /gps/ene/bandbeta <beta> | Double beta | Sets beta for the Band function. |
| /gps/ene/ezero <energy> <unit> | Double energy and string unit. Unit can be eV, keV, MeV, GeV, TeV, or PeV | Sets $E_o$ for the Band function. |
| /gps/ene/bandnorm <norm> | Double norm | Sets the normalization constant for the Band function. |
| /gbm/ggdmlfile <file> | String file | Name of geometry file. |
| /gbm/rootfile <file> | String file | Name of ROOT output file. |
| /gbm/rndmseed <seed> | Long seed | Random number seed. |
| /gbm/evttype <type> | String type. Valid arguments are thrown, triggered, and flux | All events are accepted for option "thrown". Only events which satisfy trigger conditions are accepted for option "triggered". User entered flux or energy spectrum integral is used to determine number of events for "flux" option. |
| /gps/setflux <flux> | Double flux in units (events/$mm^2$) | When /gps/diffspec is false and "/gbm/evttype flux" is used, the number of events to throw is this user defined flux times the source area. |
| /gbm/multiE <energy > <nevt> | Double energy in MeV. Int nevt. | Use to place multiple runs with different source energies in the same output file. |
| /gbm/pointsource <dist> <Runit> <theta> <phi> <mintheta> <maxtheta> <unit> <x> <y> <z> <Tunit> | Double dist, theta, phi, mintheta, maxtheta. String Runit (cm, mm, etc...), unit (deg or rad), and Tunit (cm, mm, etc...). | Set up a point source. The user can enter a distance or enter -1 for a pre-defined distance. Source can be translated in space with x,y,z arguments. |
| /gbm/planewavesource <rad> <dist> <Runit> <theta> <phi> <unit> <x> <y> <z> <Tunit> | Double rad, dist, theta, phi, String Runit (cm, mm, etc...), unit (deg or rad), and Tunit (cm,mm, etc...). | Set up a planewave source. The user can enter a radius and distance or enter -1 for a pre-defined radius and/or distance. Source can be translated in space with x,y,z arguments. |

| Command | Arguments | Description |
|---|---|---|
| /gbm/pointtargetsource <dist> <Runit> <theta> <phi> <mintheta> <maxtheta> <unit> <x> <y> <z> <Tunit> | Double dist, theta, phi, mintheta, maxtheta. String Runit (cm, mm, etc...), unit (deg or rad), and Tunit (cm, mm, etc...). | Set up a point source with a specified target location. The user can enter a distance or enter -1 for a pre-defined distance. Source photons will be aimed at a point in space with x,y,z coordinates. |
| /gbm/planewavetargetsource <rad> <dist> <Runit> <theta> <phi> <unit> <x> <y> <z> <Tunit> | Double rad, dist, theta, phi, String Runit (cm, mm, etc...), unit (deg or rad), and Tunit (cm,mm, etc...). | Set up a planewave source with a specified target location. The user can enter a radius and distance or enter -1 for a pre-defined radius and/or distance. Source will be aimed at a point in space with x,y,z coordinates. |

## 5.3 calsim Commands

### 5.3.1 calsim Macro Commands

| Command | Arguments | Description |
|---|---|---|
| AnalyzeData | none | Perform data analysis. |
| CalibFileName <file> | String file. | Name of file containing calibration parameters. |
| Draw | none | Enable drawing of graphs. |
| DumpCmdHistory <file> | String file. | Dumps the command history from the data file to the specified text file. |
| EpeakBins | none | Plots $E_{measured}/E_{peak}$ instead of $E_{measured}$ on x axis. |
| FitsFileName <file> | String file. | Name of FITS output file. |
| HistogramBins <low> <high> <nbin> | Double low, high. int nbin. | Define the measured energy bins. |
| HistogramBins2 <low> <binwidth> <nbin> | Double low, binwidth. Int nbin. | Define the measured energy bins. |
| HistogramType <xtype> <ytype> | String xtype, String ytype | Indicated whether to use linear ("Lin") or logarithmic ("Log") binning for the x and y axes. |
| InputFileName <file> | String file. | Name of input data file. |
| ListVolumes | none | List the names of volumes for which the input file contains data. |

| Command | Arguments | Description |
|---|---|---|
| MakeResponseMatrix | none | Instructs the code to attempt to make a DRM from the input data. |
| ReadAppHeader | none | Dump the contents of the Header tree to the screen for each directory |
| ListRootFile | none | Display contents of the input file top directory. |
| RootFileName <file> | String file | Name of ROOT output file. |
| RandomSeed <seed> | Unsigned int seed. | Sets calsim random number generator seed. |
| Volume <vol> | String vol. | Specifies the volume for which data is to be analyzed (must appear as one of the volumes returned by ListVolumes) |
| exit | none | Exit the session. |
| help | none | Print the list of possible commands. |
| macro <file> | String file. | Execute a macro file. |

### 5.3.2 calsim Calibration Input File Commands

| Command | Arguments | Description |
|---|---|---|
| EresParamZero <par0> <par1> <par2> | Double par0, par1, par2. | Specifies parameters when using functional form for energy resolution. |
| EresPoint <energy> <res> | Double energy in keV. Double res in keV. | Define an energy resolution point when using point-wise linear interpolation for energy resolution. |
| Ethresh <thresh> <sig> | Double thresh, sig in keV. | Define a low-energy threshold and sigma value for the threshold. |