Geomega

Geometry for MEGAlib

Version of 2010-12-16

1. Prelude

What is Geomega?

Geomega provides a uniform geometry and detector setup description for MEGAlib. Starting from a geometry file, which includes the description of all materials, volumes, detectors, trigger criteria, etc., the geometry is built and can be viewed with Geomega. The underlying geomega library is in turn used by all other programs of MEGAlib to access this geometry information. For example:

- The simulation tool Cosima is using Geomega to import the geometry into its own Geant4 format.
- When the simulation file is read by e.g. the event reconstruction tool Revan, then the ideal simulation data is noised according to the detector description of geomega
- Revan & Mimrec uses Geomega to calculate absorption probabilities, check where the hits occurred etc.

Installation

Since Geomega is part of MEGAlib, please see the MEGAlib installation instructions for a complete step by step installation guide.

Bug reports

If you find a bug or other problem, please email it to me: Andreas Zoglauer, zog@ssl.berkeley.edu

2. Invocation

```
geomega <options> -g <geometry file>
```

Geomega can be started with a variety of command line options. The geometry file is always given via the "-g" parameter.

The other options are:

`-g/--geometry <file name>` or `-f/--filename <file name>`	Give the geometry file name usually ending with *.geo.setup
`-d/--debug`	Activate debug mode for more detailed command line output
`-c/--configuration <file name>`	Use this Geomega configuration file containing a previously stored GUI configuration. Using the option –g (or –f) overwrites the geometry file name stored in the configuration file.
`-s/--startvolume <volume name>`	Use the given volume as world volume. If multiple copies of this volume exist, use one of the copies (instead of the template used to generate the copies)
`–create-mggpod <filename suffix>`	Create mggpod files with the given file name suffix

`–create-mggpod-default`	Create mggpod files with the default file names (setup.geo, media. med, materials.mat)

3. The geometry file format

The central input to Geomega is its geometry setup file. Its characteristic suffix is"*.geo.setup". It has an object oriented, keyword based style. For example a volume is declared and initialized the following way:

```
Volume MyVolume
MyVolume.Material Germanium
MyVolume.Shape BRIK 5.0 4.0 0.5
MyVolume.Position 0.0 1.0 2.0
MyVolume.Mother WorldVolume
```

A complete list of all keywords can be found in the following sections.
To ensure that the sequence of keywords can be arbitrary the setup file is scanned several times:

- Include all "Include"-files
- Handle "For" loops
- Handle constants
- Handle "If" statements
- Scan for other command keywords and object keywords
- Scan for Clones of the object keywords
- Assign all other parameters
- Validate the input

All units of all values are in cm, keV, g, deg – the same is in the rest of MEGAlib.
There are several important limitations with this geometry and detector description format compared to Geant3 and Geant4:

- All daughter volumes have to be completely contained in their mother volumes -- No overlapping volumes are allowed. An exception is virtual volumes which are removed from the later geometry – but if possible try to avoid virtual volumes.
- There is only a limited number of volume types implemented so far. If you need more, let me know, but make sure, that they exist in Geant3, Geant4 and Root.
- It is not possible to divide one volume in sub volumes.
- All detector volumes need to be boxes (with exception of the Scintillator- and Anger Camera-type detectors)
- It is very time-consuming to protect against each possible input error in an open file format like this. So do not expect to get a warning/error message for each time you make a mistake – in most cases the program will warn you with an error message, but sometimes it will simply accept the error and produce false results and sometimes it will simply crash. Whenever you come across such a problem, let me know and I will implement a protection.

3.1. Writing a good geometry

Let's start with some practices for writing a good geometry

- Before you start coding a geometry, take a look at the examples in the directory $(MEGAlib)/resources/examples/geomega. Try to write your code as close as possible to the examples, in order to avoid any trouble.
- Use meaningful descriptions, e.g. WorldVolume instead of VAC0

- Don’t make your geometry flat! The more daughters a volume has, the more volumes have to be searched when a particle moves from one volume to the next! It i significantly better to have a good balance between flatness and steepness geometry: The world volume contains a few daughters, which contain a few daughter volumes, which contain a few daughter volumes, etc.
- Try to use constants frequently.
- If you have to make multiple copies, try to use the “For” … “Done” loop.
- Use multiple files representing different objects: If you have distinguishable objects such as individual detectors with all their surround electronics and mounting, describe them in an individual file! Simply “Include” this file in the file where its mother volume is described. This makes your geometry much more clearly laid out.
- Geomega comes with a checker for overlaps. Do not use a geometry until you have verified that it does not contain overlaps!
- Check that your world volume is large enough to be able to contain your geometry and your surrounding sphere – but don’t make it too large or your simulations will take too long!

3.2. Global keywords

Key: **Name**

Parameters: 1: Name of the geometry

Description: Gives the setup a distinguishable name

Example `Name MegaPrototype`

Key: **Version**

Parameters: 1: Version number

Description: Version of the current setup

Example `Version 1.1`

Key: **Include**

Parameters: 1: Name of the file name to be included

Description: Include the given file into the setup-file. This is very useful when splitting a large setup-file into smaller parts, e.g. defining an own file for each detector or a file containing the specific materials, etc. The file name is allowed to be a relative path and might contain wildcards.

Example `Include Materials.geo`

Key: **SurroundingSphere**

Parameters: 1: Radius in cm
2: x-Position of the center of the sphere in cm
3: y-Position of the center of the sphere in cm
4: z-Position of the center of the sphere in cm
5: Distance of the disk to the center in cm – should be in all cases the same as radius – unless you understand the consequences

Description: A surrounding sphere is a mandatory requirement for the simulations with GMega or Cosima. If you do simulations in the far-field, the photons are simulated from a disk with

radius pointing from a distance towards point. This feigns a real far-field. Therefore, make sure you define a surrounding sphere which is as small as possible but encloses the complete detector geometry, without intersecting any volume.

Example `SurroundingSphere 25.0 0.0 10.0 15.0 25.0`

Key: ***//*** *or* ***#***

Parameters: 1: Comment

Description: Allows commenting the code

Example
```
# This is a comment …
Include Materials.geo // … and this too
```

Key: **BeginComment** *together with* **EndComment**

Parameters: -

Description: Those two keywords allow to comment out a larger block of text

Example
```
BeginComment
Include Materials.geo
// More blabla
EndComment
```

Key: **Echo**

Parameters: 1: Text

Description: During the final stage of the parsing of the geometry file print the given text on the screen.

Example `Echo Hello World!`

Other global keywords are Volume, Material, Detector and Trigger. Since those have sub-keywords they are described in the following sections in more detail.

3.3. Materials

Keyword: **Material**

Parameters: 1: Name of the material (no spaces allowed!)

Description: Defines a material. The name must be unique.

Example `Material CsI`

Sub-Keyword: **<Material>.Density**

Parameters: 1: Density of the material in g/cm3

Description: Defines the density of the material

Example `CsI.Density 4.5`

Sub-Keyword: **<Material>.ComponentByAtoms**

Parameters: 1: A – the mass number of the atoms
2: Z – the atomic number – the number of protons
3: Number of these atoms in the material (integer)

Description: Declares a component of a material. For example if the material is a mixture of the two components Cs and I, they have to be declared as in the example below. The number of atoms needs to be an integer!

Example
```
CsI.Component 132.9 55 1 // Cs
CsI.Component 126.7 53 1 // I
```

Sub-Keyword: **<Material>.ComponentByMass**

Parameters: 1: A – the mass number of the atoms
2: Z – the atomic number – the number of protons
3: Fractional mass of the component in the material

Description: Description Declares a component of a material. For example if the material is a mixture of the two components N and O, they have to be declared as in the example below. The total fractional mass needs to be 1!

Example
```
Air.Component 14.0 7 0.7 // N
Air.Component 16.0 8 0.3 // O
```

The directory $(MEGALIB)/resource/examples/geomega/materials contains a general file “Materials.geo”, which includes all used materials for the MEGA geometries. Take a look for more examples.

One important feature of Geomega is that it provides absorption probabilities (cross-sections for photo effect, Rayleigh scattering, Compton scattering, pair creation, and total). Those cross sections are automatically determined via Geant4 whenever a material is created or changed.

Keyword: **AbsorptionFileDirectory**

Parameters: 1: Name of the directory

Description: Gives a directory where to store & search the absorption probability files. The default directory (when this keyword is not given) is “absorptions” in the directory where the geometry is stored.

Example
```
AbsorptionFileDirectory MyAbsorptions
```

Obsolete material sub-keywords which are no longer used are: Sensitivity & RadiationLength (automatically calculated in Geant4).

3.4. Volumes

Please take notice of all distances except radii being half distances!

Keyword: **Volume**

Parameters: 1: Name of the volume (no spaces allowed!)

Description: Defines the volume. The name must be unique.

Example
```
Volume Wafer
```

Sub-Keyword: **<Volume>.Density**

Parameters: 1: Density of the material in g/cm3

Description: Defines the density of the material

Example `CsI.Density 4.5`

Sub-Keyword: **<Volume>.Shape**

Parameters: 1: Keyword describing the shape
2-N: Parameters of the shape

Description: Detailed descriptions of the objects can be found in the Geant3, Geant4 or ROOT manual (ROOT comes with pictures!). The argument list corresponds to that of ROOT. If you want new volumes to be added, let me know, but make sure such a volume exists in Geant3, Geant4 and ROOT, and provide an equation for its volume! One limitation for the current Geant4 implementation is that TRAP and GTRA are not allowed to be triangles (i.e. non of the values height, bottom and top length is allowed to be zero). So if you need triangles make sure, they contain very small values.

BRIK or BOX (a box)	2: half-size x in cm 3: half-size y in cm 4: half-size z in cm
SPHE (a sphere, which can be hollow, or a segment of it)	2: inner radius in cm 3: outer radius in cm 4: theta min in deg 5: theta max in deg 6: phi min in deg 7: phi max in deg
TUBS (a cylinder, which can be hollow, or a section of it)	2: inner radius in cm 3: outer radius in cm 4: half height in cm 5: phi min in deg 6: phi max in deg
CONE (a cone)	2: half height in cm 3: inner bottom radius in cm 4: outer bottom radius in cm 5: inner top radius in cm 6: outer top radius in cm
TRD1 (a trapezoid)	2: half distance x1 3: half distance x2 4: half distance y 5: half distance z
TRD2 (a trapezoid)	2: half distance x1 3: half distance x2 4: half distance y1 5: half distance y2 6: half distance z
TRAP	2: half distance in z

(a general trapezoid) **Avoid if a simpler trapezoid can be used!**	3: theta 4: phi 5: half height trapezium bottom 6: half bottom length trapezium bottom 7: half top length trapezium bottom 8: alpha trapezium bottom 9: half height trapezium top 10: half bottom length trapezium top 11: half top length trapezium top 12: alpha trapezium top
GTRA **(a general trapezoid, which can be twisted)** **Avoid if a simpler trapezoid can be used!**	2: half distance in z 3: theta 4: phi 5: twist 6: half height bottom trapezium bottom 7: half bottom length trapezium bottom 8: half top length trapezium bottom 9: alpha trapezium bottom 10: half height trapezium top 11: half bottom length trapezium top 12: half top length trapezium top 13: alpha trapezium top>
PCON **(a polycone – round corners)**	2: the azimuthal angle phi at which the volume begins (angles are counted counterclockwise) 3: opening angle of the volume 4: number of sections (number of time the following three arguments are repeated), the number should be at least 2 *The following three arguments are repeated accordingly:* M_1: height (full not half) M_2: inner radius M_3: outer radius
PGON **(a polygon)**	2: the azimuthal angle phi at which the volume begins (angles are counted counterclockwise) 3: opening angle of the volume 4: number of sides of the cross section between the given phi limits 5: number of sections (number of time the following three arguments are repeated), number should be at least 2 *The following three arguments are repeated accordingly* M_1: height (full not half) M_2: inner radius M_3: outer radius
Example	`Wafer.Shape BRIK 5.0 10.0 0.01`

Sub-Keyword: **<Volume>.Material**

Parameters: 1: Name of an existing material

Description: Material of the volume. It has to be defined somewhere else in the setup file.

Example `Wafer.Material Silicon`

Sub-Keyword: **<Volume>.Visibility**

Parameters: 1: Visibility value (0, 1)

Description: If the value is zero, then the volume is not visible, if it is 1, the volume is visible in the geomega viewer.

Example
```
Wafer.Visibility 1
```

Sub-Keyword: **<Volume>.Color**

Parameters: 1: Color as defined in ROOT

Description: The color of the volume when display in the viewer. The numbers correspond to the ROOT color IDs.

Example
```
Wafer.Color 5
```

Sub-Keyword: **<Volume>.Mother**

Parameters: 1: Name of an existing volume, or 0 in case of the world volume

Description: In this mother volume the volume will be placed. Attention: the volume must be fully contained in her mother and the mother must be defined somewhere else in the setup file. For the world volume, use 0.

Example
```
Wafer.Mother Tracker
WorldVolume.Mother 0
```

Sub-Keyword: **<Volume>.Position**

Parameters: 1: x position in the mother volume in cm
2: y position in the mother volume in cm
3: z position in the mother volume in cm

Description: Position of the volume within its mother's coordinate system. Position is the center of the volume. Attention the volume has to be fully contained in its mother volume!

Example
```
Wafer.Position 0.0 10.0 30.0
```

Sub-Keyword: **<Volume>.Rotation**

Parameters: Type A:
1: Counterclockwise rotation around x-axis in the mother coordinate system
2: Counterclockwise rotation around y-axis in the mother coordinate system
3: Counterclockwise rotation around z-axis in the mother coordinate system
Type B:
1: Theta1 in deg: the polar angle of the x-axis in the mother reference system
2: Phi1 in deg: the azimuthal angle of the x- axis in the mother reference system
3: Theta2 in deg: the polar angle of the y-axis in the mother reference system
4: Phi2 in deg: the azimuthal angle of the y-axis in the mother reference system
5: Theta3 in deg: the polar angle of the z-axis in the mother reference system
6: Phi3 in deg: the azimuthal angle of the z-axis in the mother reference system

Description: Defines the rotation of the volume

Example
```
Wafer.Rotation 90 45 0 45 90 0
```

Sub-Keyword: **<Volume>.Scale**

Parameters: 1: Scaler

Description: Scale (shrink or enlarge) a volume and all sub volumes. Useful if one has a large volume tree and wants to modify its size. This keyword can not be applied to Copies/Clones.

Example `SpaceCraftBody.Scale 0.5`

In usual detector geometry some volumes will appear several times. To avoid any copy and paste, the keyword Copy has been introduced, which copies the characteristics of one volume to another. Normally the base volume is defined as template, i.e. it does not have a mother and it is not positioned.

```
Volume GeWafer
GeWafer.Material Germanium
GeWafer.Visibility 1
GeWafer.Color 6
GeWafer.Shape BRIK 4.0 3.0 1.0
// Arrange the Ge-detectors
GeWafer.Copy GeE1N001
GeE1N001.Position 0.0 10.0 3.75
GeE1N001.Mother WorldVolume
GeWafer.Copy GeE1N002
GeE1N002.Position 0.0 10.0 1.25
GeE1N002.Mother WorldVolume
```

Up to now, the keyword Copy is only implemented for volumes!

Sub-Keyword: **<Volume>.Copy**

Parameters: 1: Name of a new volume

Description: Create the new volume and copy all characteristics of <Volume> to the new volume.

Attention:

1. **Make sure you do not place the original volume itself, just it copies, i.e. the original volume should have neither a Position nor a Mother (keyword defined).**
2. **The keyword cannot be used for sensitive detector volumes (see keyword SensitiveDetector)**

Example `SiLayer.Copy bachus`

One major restriction of the geomega format (which is actually a restriction of Geant4) is that no overlapping volumes are allowed. But sometimes it is useful to have a larger volume, in which several other volumes are grouped, to use as template. In the case this larger volume has no real meaning other than being a container for other volumes. In those cases it can be declared as virtual. Those virtual volumes are allowed to overlap with other volumes – as long as their content does not overlap with any other volume! During the creation of the geometry, virtual volumes are removed from the volume tree! As consequence, the volumes need to be renamed (format: "VolumeName_VirtualVolumeName"). Otherwise, since a virtual volume can have copies and its daughter volumes are placed in its mother volumes, multiple volumes with the same name in the same volume could exist.

Sub-Keyword: **<Volume>.Virtual**

Parameters: 1: true or false (default for a volume is always false)

Description: Declare a volume as virtual. Try to avoid!

Attention: Do not use for world, sensitive or detector volumes!

Example `TrackerContainer.Virtual true`

However, try to avoid virtual volumes! They only make things much more complicated for the simulation, because they result in more flat volume hierarchies and thus much slower simulations!

3.5. Detectors

There exist 7 different detector types:

- A 2D strip detector like the MEGA Silicon wafers (Strip2D)
- A 3D strip detector (Strip3D) like the NCT Germanium detectors
- A directional 3D strip detector, where some information of the electron direction is retained (*)
- A Drift Chamber detector including capabilities for light sensing – needed for liquid Xe and gas microwell detectors (*)
- A calorimeter like in MEGA: Many CsI bars separated by passive material are sitting in a housing (*)
- A one-volume-type detector, which can be used as MEGA ACS as well as SPI Germanium detector, or any other thumb detector, which can only measure energy information
- A 3D voxel detector

The detectors marked with (*) are rather specialized and most likely not needed in your setup.

Sub-Keyword: **Strip2D**

Parameters: 1: Detector name (must be unique)

Description: Declares a 2d strip detector which can have a guard ring. The strips are always oriented in x and y direction. If you want another orientation, simple rotate the detector volume.

Example
```
Strip2D bachus
```

Sub-Keyword: **Strip3D**

Parameters: 1: Detector name (must be unique)

Description: Declares a 3D strip detector which can have a guard ring. It inherits all capabilities from Strip2D. In addition it has a depth resolution, which is always the z direction.

Example
```
Strip3D nct
```

Sub-Keyword: **Strip3DDirectional**

Parameters: 1: Detector name (must be unique)

Description: Declares a 3D strip detector. It inherits all capabilities from Strip3D. In addition it has a directional resolution: It can detect all directions of electrons originating from Compton interaction. Due to limitations in the simulation, it is currently not possible to detect the direction of electrons simply passing through the silicon layer. If you want to use this detector, you need full “IA” information from the simulation. Thus if you use mggpod, make sure to use INIT2 and ACT2 options.

Example
```
Strip3DDirectional SiWafer
```

Sub-Keyword: **DriftChamber**

Parameters: 1: Detector name (must be unique)

Description: Declares a drift chamber detector. It inherits all capabilities from Strip3D. In addition,

specific information like the (optical) light speed in this material (”LightSpeed”), the light sensitive detector side (”LightDetectorPosition”) and its energy resolution (”LightEnergyResolution”) as well as the electron drift parameters (”DriftConstant”, ”EnergyPerElectron”) can be set.

Example `DriftChamber Chamber`

Sub-Keyword: **Calorimeter**

Parameters: 1: Detector name (must be unique)

Description: Declares a MEGA-style calorimeter e.g. consisting of individual CsI bars surrounded by passive (reflective) material in a common housing.

Example `Calorimeter Fortuna`

Sub-Keyword: **Scintillator**

Parameters: 1: Detector name (must be unique)

Description: Declares a large, one channel and non position sensitive detector. Since those are mostly scintillator it is called that way. But you can of course use it also for non-scintillator detectors, such as e.g. the SPI Germanium detectors. The special feature of this volume is that it does not need to be box-like and that it can consist of multiple volumes (see keyword SensitiveVolume). The latter allows building more complex detector shapes. Hits in this detector are always centered.

Example `Scintillator SPI21`

Sub-Keyword: **Voxel3D**

Parameters: 1: Detector name (must be unique)

Description: Declares a box-shaped volume consisting of voxels in all three dimensions. This detector has no known real-world counterpart, as it has no passive material (electronics, connectors, etc.) between the voxels. Its primary function is currently simulation diagnostics. Hits within the voxels are always centered

Example `Voxel3D NuSTAR_CZT_Detector`

3.5.1. Common keywords

Sub-Keyword: **<Detector>.SensitiveVolume**

Parameters: 1: Name of an existing volume

Description: Volume, in which positions and energies of interactions are measured. Typical examples are one CsI-crystal or one single Si-wafer of the MEGA prototype.
Attention: This volume cannot have been generated via the “Copy” keyword – but it is OK if you use a volume from which you generate copies. The reasoning is that all copies of a volume must have the same status, either be sensitive or not. It is also OK (actually normal) when the sensitive volume is part of a super-volume structure which is copied, or – and this is also a usual case – that many copies of the sensitive volume exist, which then have the same detector properties.

Multiple sensitive volumes:

The only detector type which supports multiple, different sensitive detectors is the scintillator detector type. The different sensitive volumes are handled as they were one volume, i.e. all energy deposits are summed together (Of course this can be avoided by using the "DiscretizeHits false" keyword in cosima and if you write your own detector effects engine).
Having multiple different sensitive volumes in one detector comes with the following restrictions to uniquely identify which volumes belong together:

- The sensitive volume must neither be a "Copy" of a volume, nor be used to generate Copies, i.e. it must be unique (although a volume up in its tree might be a Copy, thus indirectly you might have multiple copies)
- The volumes must have a common mother volume, but which must not be the direct mother
- There are no copies allowed in the common mother volumes and no other sensitive volumes, i.e. the common volumes contains exactly one of the sensitive volumes

However, the common volume can be copied.

Example `MEGACal.SensitiveVolume CsICrystal`

Sub-Keyword: **<Detector>.DetectorVolume**

Parameters: 1: Name of an existing volume

Description: A larger volume which contains several evenly spaced sensitive volume. The position the sensitive volumes are specified in the "Structural" parameters. They must be identical with the positions given in the volume description! Typical examples are the MEGA calorimeter, which consists of 120 CsI crystals or one layer of the MEGA tracker, which consists of 9 Si-wafers. The sensitive volume is either the same as the detector volume or entirely and unrotated contained in the detector volume!

Attention: Simular restrictions as for sensitive volumes apply: This volume cannot have been generated via the "Copy" keyword – but it is OK if you use a volume from which you generate copies. The reasoning is that all copies of a volume must have the same status, either be a detector or not. It is also OK (actually normal) when the detector volume is part of a super-volume structure which is copied, or – and this is also a usual case – that many copies of the detector volume exist, which then have the same detector properties.

If the detector volume is not given, and you only have one sensitive volume, then the detector volume is the sensitive volume.

See Figure 1 for an illustration.

Example `MEGACal.DetectorVolume CsIDetector`

Sub-Keyword: **<Detector>.StructuralPitch**

Parameters: 1: x spacing in cm
2: y spacing in cm
3: z spacing in cm

Description: Spacing between the sensitive volumes (distance between the end of last sensitive volume to the start of the next sensitive volume). The detectors need to be evenly spaced. This is not the pitch between the individual strips of a strip detector! If you have a tracker, then also add the distance between the layers as z-component!
This keyword is always required for the Calorimeter detector. For strip/voxel detectors

you can use it, if you want to join several sensitive detectors into one logical detector. You do not need it, if you just have a simple, one-volume strip or voxel detector. The scintillator and Anger camera detector types don't use this keyword.
See Figure 1 for an illustration.

Example `MEGACal.StructuralPitch 0.07 0.03 0.0`

Sub-Keyword: **<Detector>.StructuralOffset**

Parameters: 1: x spacing in cm
2: y spacing in cm
3: z spacing in cm

Description: Distance between the edge of the detector volume to the beginning of the first sensitive volume. Calculated from negative to positive axis!
This keyword is always required for the Calorimeter detector. For strip/voxel detectors you can use it, if you want to join several sensitive detectors into one logical detector. You do not need it, if you just have a simple, one-volume strip or voxel detector. The scintillator and Anger camera detector types don't use this keyword.
See Figure 1 for an illustration.

Example `MEGACal.StructuralOffset 0.235 0.47 0.185`

Sub-Keyword: **<Detector>.NoiseThreshold**

Parameters: 1: Energy in keV

Description: All hits in one voxel of the detector which are below this energy (in keV) are assumed not to be measured.

Example `MEGACal.NoiseThreshold 50`

Sub-Keyword: **<Detector>.TriggerThreshold**

Parameters: 1: Energy in keV

Description: A hit need to deposit at least this energy (in keV) to raise a trigger signal.
The difference between noise and trigger threshold: Each strip has a certain amount of electronics noise. So when one reads it out, one gets a signal in ADC counts, which only reflects the noise of the electronics. So normally one only uses hits which are well above the noise, e.g. 5 sigmas above it. Trigger threshold is something different. A hit needs to produce a certain voltage (i.e. has to deposit a certain energy) to initiate the read-out of the detector.

Example `MEGACal.TriggerThreshold 100`

Sub-Keyword: **<Detector>.EnergyResolution**

Parameters: *Old format:*
1: Input energy in keV
2: 1 sigma with of a Gauss distribution at this energy in keV
New format:
1: Resolution type: Ideal, Gauss, Lorentz, GaussLandau
2+: Parameters

For case *"Ideal"*:
No more parameters necessary

For the case *"Gauss"*:
2: Input Energy in keV
3: Peak energy of Gauss distribution in keV
4: One sigma width of Gauss distribution in keV

For the case *"Lorentz"*:
2: Input Energy in keV
3: Peak energy of Lorentz distribution in keV
4: Width of Lorentz distribution in keV as 2/2.35 times the "scale parameter" which defines the half-width half maximum of the distribution

For the case *"GaussLandau"*:
2: Input Energy in keV
3: Peak energy of Gauss distribution in keV
4: One sigma width of Gauss distribution in keV
5: Peak energy of Landau distribution in keV
6: Width of the Landau distribution in keV given as its "scale parameter"
7: Contribution (value between]0..1[) of the Gauss part

Description: Energy resolution information at the given energy in keV (one sigma)
See the file EnergyResolutionTester.geo,setup for an example.

Example
```
PerfectDet.EnergyResolution Ideal
MEGACal.EnergyResolution Gauss 662 662 20
```

Sub-Keyword: **<Detector>.EnergyLossMap**

Parameters: 1: file name

Description: The file given contains a 3D matrix of detector positions (a detector "map") and an associated energy loss, which describes e.g. energy loss through charge trapping.
The file format is identical to the 2D-Function described in the Cosima – just add another dimension.
Attention: In contrast to all other options, this energy loss is applied during the simulation in Cosima, since only there the detailed positions of all energy deposits are known. Thus, if you change the file, you have to redo your simulations.
See the file EnergyResolutionTester.geo,setup for an example.

Example
```
MEGACal.EnergyLossMap EnergyLoss.dat
```

Sub-Keyword: **<Detector>.EnergyCalibration**

Parameters: 1: file name

Description: If an energy loss file is used or the peak energy is not the input energy of the energy resolution, then the "measured" energy is not equal the input energy and an energy calibration is need. Otherwise this option is NOT required.
The given file describes a simple input-energy-to-detected-energy ratio, which is used to calibrate the energy.
The file format is identical to the 1D-Function described in the Cosima manual.
See the file EnergyResolutionTester.geo.setup for an example.

Example
```
MEGACal.EnergyCalibration Calibration.dat
```

Sub-Keyword: **<Detector>.TimeResolution**

Parameters: 1: Deposited energy in keV
2: 1 sigma with of a time distribution at this energy in seconds

Description: Gaussian (!) time resolution information of the detector at the given energy in seconds (one sigma).

Example
```
StripDetector.TimeResolution   1.0 1.0E-9
StripDetector.TimeResolution 100.0 0.8E-9
```

Sub-Keyword: **<Detector>.FailureRate**

Parameters: 1: failure rate [0..1]

Description: Random failure rate – values are between 0 (no failures) and 1 (complete loss of detector)

Example
```
MEGACal.FailureRate 0.01
```

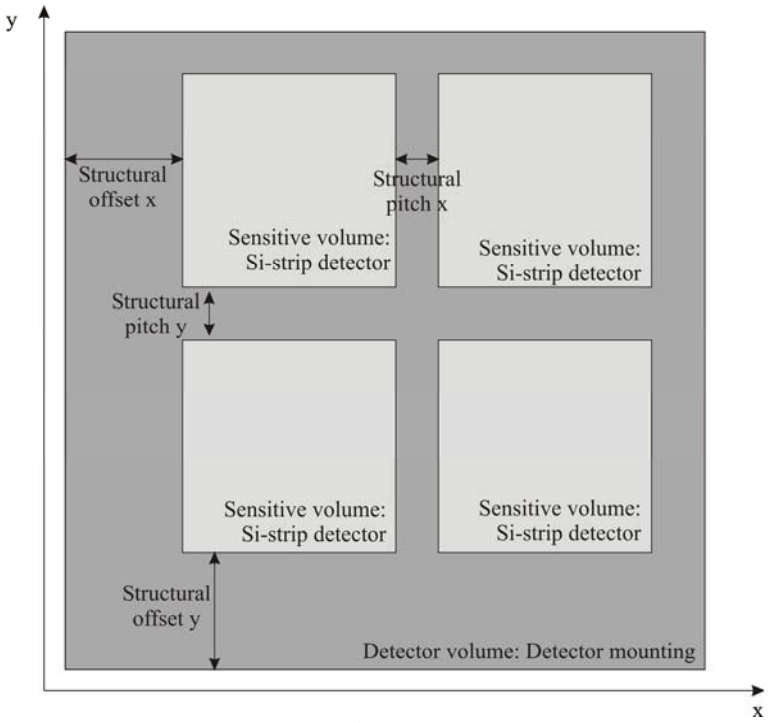

Figure 1

Illustration of the StructuralOffset, StructuralPitch, SensitiveVolume, and DetectorVolume keywords: Complex detectors may consist of many sensitive detectors arranged in a regular way. In the above example we have one detector volume, the detector mounting, which has several daughter volumes, in this case silicon strip detectors, which are the actual sensitive (i.e. measure energy) volumes. In order for MEGAlib to find the strips in the detectors, the first detectors start at distance of structural offset measured from the negative axis, and the distance between the detectors is the structural pitch.
For all simple detectors the sensitive volume should be the detector volume and structural offsets and pitches should therefore be zero.

3.5.2. Keywords specific to calorimeters

Sub-Keyword: **<Detector>.DepthResolution**

Parameters: 1: Energy in keV
2: Depth resolution in cm (one sigma)

Description: Give the depth resolution at the given energy in cm (one sigma). If this keyword is not given, then no depth resolution is assumed, i.e. it behaves like a Strip2D detector.

Example `MEGACal.DepthResolution 500 1.2`

3.5.3. Keywords specific to strip/voxel detectors

This includes Strip2D, Strip3D, Voxel3D, DriftChamber detectors.

Sub-Keyword: **<Detector>.Offset**

Parameters: 1: x Offset in cm
2: y Offset in cm
3: z Offset in cm (Voxel3D only)

Description: Distance between the edge of the sensitive detector to the beginning of the first strip

Example `WaferDetector.Offset 0.142 0.071`

Sub-Keyword: **<Detector>.StripNumber**

Parameters: 1: Number of strips/voxels in x direction
2: Number of strips/voxels in y direction
3: Number of strips/voxels in z direction

Description: Gives the number of strips/voxels in each dimension

Example `WaferDetector.StripNumber 128 64`

Sub-Keyword: **<Detector>.GuardringTriggerThreshold**

Parameters: 1: Energy in keV

Description: A hit in the guard ring needs to deposit at least this energy (in keV) to raise a trigger (veto) signal.

Example `WaferDetector.GuardringTriggerThreshold 100`

Sub-Keyword: **<Detector>.GuardringEnergyResolution**

Parameters: 1: Energy in keV
2: Measurement uncertainty at this energy in keV (one sigma)

Description: Energy resolution at the given energy in keV (one sigma) for the guard ring

Example `WaferDetector.EnergyResolution 100 10`

For 3D strip detectors (with and without electron direction resolution) and DriftChambers there is another special keyword:

Sub-Keyword: **<Detector>.DepthResolution**

Parameters: 1: Energy in keV

2: Depth uncertainty at this energy in cm 9one sigma)

Description: Set the one sigma depth resolution. If this keyword is not given, then no depth resolution is assumed.

Example `GeStrip.DepthResolution 200 0.2`

Sub-Keyword: **<Detector>.DepthResolutionThreshold**

Parameters: 1: Energy in keV

Description: Below this threshold no depth resolution can be measured, i.e. the center (z-axis) of the detector is used as z-position.

Example GeStrip.DepthResolutionThreshold 25

For 3D strip detectors with directional resolution exists another special keyword:

Sub-Keyword: **<Detector>.DirectionalResolution**

Parameters: 1: Energy in keV
2: Resolution in degree (one sigma)

Description: Set the one sigma directional resolution

Example `SiStrip.DirectionalResolution 200 30`

DriftChambers have additional keywords

Sub-Keyword: **<Detector>.LightSpeed**

Parameters: 1: Light speed in cm/s

Description: Light speed of the scintillation light in the drift chamber

Example `Chamber.LightSpeed 1.8E+9`

Sub-Keyword: **<Detector>.LightDetectorPosition**

Parameters: 1: 0: none; 1: +x; -1:-x; 2: +y; -2 -y; 3: +z; -3:-z

Description: Represents the side of the detector which is light sensitive (i.e. equipped with PMTs or diodes). If this value is zero, then no light detector is assumed.

Example `Chamber.LightDetectorPosition 3`

Sub-Keyword: **<Detector>.DriftConstant**

Parameters: 1: Drift constant in cm

Description: One sigma of the opening cone of the drift in the E-field. If this is zero, a simple and fast projection is used.

Example `Chamber.DriftConstant 0.01`

Sub-Keyword: **<Detector>.EnergyPerElectron**

Parameters: 1: Energy in keV per electron

Description: Energy of one drifting electron

Example `Chamber.DriftConstant 0.01`

Sub-Keyword: **<Detector>.LightEnergyResolutionAt**

Parameters: 1: Energy in keV
2: Energy resolution in keV (one sigma)

Description: One sigma energy resolution of the detected light.

Example `Chamber.LightEnergyResolutionAt 1000 5.0`

3.5.4. Keywords specific to Scintillator-like detectors

The special feature of this detector type is that it may contain several sensitive volumes of different shapes.

Sub-Keyword: **<Detector>.HitPosition**

Parameters: 1: Volume name 1
2: Volume name 2
3: x-position in cm
4: y-position in cm
5: z-position in cm

Description: A hit in volume 1 is moved into position (x, y, z) in volume 2. All sensitive volumes of the detector have to be covered!

Example
```
GeDet.HitPosition GeCentral GeCentral 0 0 0
GeDet.HitPosition GeLeft GeCentral 0 -2.59807621135 0
GeDet.HitPosition GeRight GeCentral 0 -2.59807621135 0
```

3.5.5. Multiple detectors

Detectors always relate to volumes not created by the Copy-keyword (see keyword SensitiveVolume and DetectorVolume). The detector parameters are then passed on to all volumes created via the copy keyword (see example A below). If you want to have detectors which have the same volume (e.g. shape) but different detector parameters, you also have to create different volumes (see example B below).

```
# Example A: multiple volumes with identical detector parameters

Volume SensV
SensV.Material Germanium
SensV.Shape BOX 1.0 1.0 0.1

SensV.Copy SensV1
SensV1.Position 0.0 0.0 1.0
SensV1.Mother World

SensV.Copy SensV2
SensV2.Position 0.0 0.0 1.0
SensV2.Mother World

Scintillator Det
```

```
Det.SensitiveVolume SensV
Det.DetectorVolume SensV
Det.TriggerThreshold 50
Det.EnergyResolution Gauss 50 50 5
Det.EnergyResolution Gauss 500 500 50


# Example B: multiple volumes with different detector parameters

Volume SensV1
SensV1.Material Germanium
SensV1.Shape BOX 1.0 1.0 0.1
SensV1.Position 0.0 0.0 1.0
SensV1.Mother World

Volume SensV2
SensV2.Material Germanium
SensV2.Shape BOX 1.0 1.0 0.1
SensV2.Position 0.0 0.0 1.0
SensV2.Mother World

Scintillator Det1
Det1.SensitiveVolume SensV1
Det1.DetectorVolume SensV1
Det1.TriggerThreshold 50
Det1.EnergyResolution Gauss 50 50 5
Det1.EnergyResolution Gauss 500 500 50

Scintillator Det2
Det2.SensitiveVolume SensV2
Det2.DetectorVolume SensV2
Det2.TriggerThreshold 40
Det2.EnergyResolution Gauss 40 40 4
Det2.EnergyResolution Gauss 400 400 40
```

3.6. Triggers

The current implementation of triggers is not completely unique through out the package. The Cosima simulations perform a pre-trigger, which currently has to be defined in the Cosima setup file.
The final trigger is then applied during reading the sim file with e.g. Sivan or revan.

The approach is the following:
First define if this trigger is a veto or not. Then define if the triggering happens by channel or by detector. "TriggerByChannel" means that the channels are counted for the trigger, "TriggerByDetector" means that the detectors are counted (irrelevant of how many channels in this detector have trigger - if at least one is above the trigger threshold). Then define the triggering detectors (by type or name) and the number of hits they must accumulate. Pay attention, you can only have one type of detector keyword per trigger, i.e. one trigger is allowed to contain only "DetectorType" keywords or only "Detector" keywords or only "GuardringDetectorType" keywords or only "GuardringDetector" keywords!

Keyword: **Trigger**

Parameters: 1: Name of a trigger (must be unique)

Description: Declares a trigger (or veto) condition

Example
```
Trigger D1D2
```

Sub-Keyword: **<Trigger>.Veto**

Parameters: 1: true/false

Description: Determines if this is a real trigger (acceptance) or a veto trigger (rejection). If this option is not given then veto is false!

Example
```
ACSVeto.Veto true
```

Sub-Keyword: **<Trigger>.TriggerByChannel**

Parameters: 1: true/false

Description: Determines that the hits are accumulated channel wise. This is the default.
Attention: A trigger can be either by channel or by detector, not both!

Example
```
D1D2.TriggerByChannel true
```

Sub-Keyword: **<Trigger>.TriggerByDetector**

Parameters: 1: true/false

Description: Determines that the hits are accumulated detector wise. TriggerByChannel is the default!
Attention: A trigger can be either by channel or by detector, not both!

Example
```
D1D2.TriggerByChannel true
```

Sub-Keyword: **<Trigger>.DetectorType**

Parameters: 1: Detector type name (e.g. Strip2D, Strip3D, etc.)
2: Number of hits required to raise the trigger or veto

Description: Number of hits are necessary in the given detector type to raise a trigger. If this keyword occurs multiple times all conditions have to be fulfilled. The "Detector type name" follows the MEGAlib convention: Strip2D is a 2D strip detector, Calorimeter is a MEGA calorimeter, Strip3D is a 3D strip detector, Scintillator is a scintillator/ACS type detector and DriftChamber is of course the drift chamber.
Attention: A trigger can either be defined by DetectorType or by Detector, not both!

Example
```
D1D2.DetectorType Strip2D 1
D1D2.DetectorType Calorimeter 1
```

Sub-Keyword: **<Trigger>.Detector**

Parameters: 1: Detector name (not type)
2: number of hits required in this detector to raise a trigger/veto

Description: Number of hits are necessary in the given detector to raise a trigger/veto. If this keyword occurs multiple times all conditions have to be fulfilled.
Attention: A trigger can either be defined by DetectorType or by Detector, not both!

Example
```
D1D2.Detector WaferDetector 4
```

Sub-Keyword: **<Trigger>.GuardringDetectorType**

Parameters: 1: Detector type name (e.g. Strip2D, Strip3D, etc.)
2: Number of hits required to raise the trigger or veto

Description: Number of hits are necessary in the given detector type (which is required to have guard ring) to raise a trigger. If this keyword occurs multiple times all conditions have to be fulfilled. The "Detector type name" follows the MEGAlib convention: Strip2D is a 2D strip detector, Strip3D is a 3D strip detector.
Attention: A guard ring trigger can either be defined by GuardringDetectorType or by GuardringDetector, not both!

Example `D3.GuardringDetectorType Strip3D 1`

Sub-Keyword: **<Trigger>.GuardringDetector**

Parameters: 1: Detector name (not type)
2: number of hits required in this detector to raise a trigger/veto

Description: This number of hits (usually only one is reasonable) is necessary in the guard ring of the given detector to raise a trigger.
If this keyword occurs multiple times all conditions have to be fulfilled.
Attention: A guard ring trigger can either be defined by GuardringDetectorType or by GuardringDetector, not both!

Example `D3.GuardringDetector MyStrip3D 1`

Here are more examples:
The MEGA prototype has an electron tracker and a calorimeter. Thus, a reasonable trigger condition would require at least two layers of the tracker and one calorimeter triggering. In addition any events with hits in the veto dome should be rejected. Thus we define two trigger conditions:

```
Trigger Main
Main.Veto false
Main.TriggerByDetector true
Main.Detector MyD1 2
Main.Detector MyD2 1

Trigger AntiCoincidence
AntiCoincidence.Veto true
AntiCoincidence.TriggerByDetector true
AntiCoincidence.Detector MyAnticoidence 1
```

A thick Germanium detector might require at least three hits for Triple Compton coincidence. Since the detector is thick, those hits do not need to be in different detectors but only different channels need to trigger. In addition, events which deposit energy in the guard rings of the detector are going to be rejected:

```
Trigger Main
Main.Veto false
Main.TriggerByChannel true
Main.DetectorType Strip3D 3

Trigger Guardring
Guardring.Veto true
Guardring.TriggerByDetector true
Guardring.GuardringDetectorType Strip3D 1
```

The triggering is performed during reading the events from the *.sim file into revan or Sivan. No triggers are tested for the *.evta files, since those events represent detector data. A trigger is raised, when the deposited energy in the given channel is above the trigger threshold of the detector or guard ring.

3.7. System

You only need to define a system if you want to noise the event time (not the hit time).

The "System" is a special object as it describes overall detector characteristics common to all detectors. For the time being, the only characteristic is the noising of the event time.

Sub-Keyword: **<System>.TimeResolution**

Parameters: 1: Resolution type: Ideal, Gauss
2+: Parameters

For case *"Ideal"*:
No more parameters necessary

For the case *"Gauss"*:
2: One sigma time resolution in seconds

Description: Time resolution of the event (not the individual hit) given as one sigma in seconds in the case of Gaussian noising.

Example
```
PerfectSystem.TimeResolution Ideal
MEGA.TimeResolution Gauss 1E-7
```

3.8. Additional features

3.8.1. Constants

A very important feature is constants. They are useful if you develop your geometry in a way that allows for easy modification in combination with the math environment and for-loops. Examples are:
Your geometry has a variable number of detector layers. You define the number of layers as a constant, and generate the layer copies in a for-loop. Then you just have to change the number-of-layers constant to modify your geometry.

Keyword: **Constant**

Parameters: 1: String
2: Number

Description: Replace all occurrences of String with number in the whole geometry (even if it is spanned over different files). Attention: It is not checked if the string is a keyword!

Example
```
Constant Size 2.0
Wafer.Shape BRIK Size Size 0.025
```

3.8.2. The math environment

Another very important feature is that one can do very basic mathematical calculations in the setup file. The signs "{" and "}" start and end the math environment. Everything within those brackets is considered as one token:

```
Constant Size 2.0
Wafer.Shape BRIK {1.2*Size} {1.2*Size} {0.5*(log(Size)+1.5)}
```

The math environment relies on the ROOT interpreter. So whatever manipulation or function ROOT knows can be used in the math environment. Thus you can use sin, cos, log, exp, etc.

3.8.3. For loops

Another important feature is for-loops. Many geometries use repeating volumes structures. Those can be easily created with for loops:

```
Constant NLayers   10
Constant ZMax      +4.5
Constant ZDistance +1.0

For Z NLayers ZMax { -ZDistance }
   Layer.Copy Layer_%Z
   Layer_%Z.Position 0.0 1.0 $Z
   Layer_%Z.Mother WorldVolume
Done
```

The loop expands to:

```
Layer.Copy Layer_1
Layer_1.Position 0.0 1.0 +4.5
Layer_1.Mother WorldVolume

Layer.Copy Layer_2
Layer_2.Position 0.0 1.0 +3.5
Layer_2.Mother WorldVolume

Layer.Copy Layer_3
Layer_3.Position 0.0 1.0 +2.5
Layer_3.Mother WorldVolume
```

…

Keyword: **For … Done**

Parameters: 1: String defining the looping variable
2: Number of repeats
3: Start value of the variable
4: Increment of the variable

Description: This is an implementation of a for-loop.
Within the loop the variable value can be a accessed with the "$" command, the loop

index can be accesses via putting “%” in front of the variable name.

Example `see above`

3.8.4. If-conditions

Sometimes it is necessary to generate some geometry code only under certain conditions. For this case the if-condition has been introduced. The segment between the If … EndIf is only generated if the mathematical expression after “If” is true. “Else” is not yet implemented.

Example:

```
Constant UseShield 1

If { UseShield == 1 }
  Volume Shield
  Shield.Material BGO
  Shield.Shape Box 10.0 8.0 3.0
  Shield.Position 0.0 10.0 2.0
  Shield.Mother WorldVolume
EndIf
```

Keyword: **If {} … EndIf**

Parameters: 1: Mathematical expression evaluating to true or false

Description: This is an implementation of an if-endif-condition.

Example `see above`

3.8.5. Echo

Finally in many circumstance when you use constants, for loops, or if-loops you might want some debugging output. For those cases the Echo keyword has been introduced.

Keyword: **Echo**

Parameters: 1: String (may contain constants, etc)

Description: Dump some text to the console

Example `Echo Hello World!`

3.9. Final words

DO NOT FORGET TO CHECK YOUR GEOMETRY FOR OVERLAPS BEFORE STARTING ANY SIMULATIONS!