

# Muon induced secondary electrons at the KATRIN experiment Detector installation and setup and data analysis

Diploma Thesis of

**Philipp Rovedo**

At the Department of Informatics  
IKP - Institut fuer Kernphysik - KIT Karlsruhe

Reviewer:	Prof. Dr. Guido Drexlin
Second reviewer:	Prof. Dr. Ulrich Husemann
Advisor:	Nancy Wandkowsky
Second advisor:	?

Duration:: September 27th 2012 – September 27th 2013

---

I declare that I have developed and written the enclosed thesis completely by myself, and have not used sources or means without declaration in the text.

**PLACE, DATE**

.....  
**(YOUR NAME)**

# Contents

<b>1. Introduction</b>	<b>1</b>
1.1. The standard model . . . . .	1
1.1.1. The Standard Model Neutrino . . . . .	1
1.2. Massive neutrinos . . . . .	1
1.2.1. Neutrino Oscillations . . . . .	1
1.2.2. Measuring Neutrino Mass . . . . .	1
1.3. Cosmic air showers . . . . .	1
1.4. Muon interaction with matter . . . . .	1
<b>2. The KATRIN experiment</b>	<b>3</b>
2.1. Source Side and Transport Section . . . . .	3
2.1.1. MAC-E Filter . . . . .	3
2.2. Pre-Spectrometer . . . . .	3
2.3. Main Spectrometer . . . . .	3
2.4. Focal Plane Detector System . . . . .	3
2.5. Solenoids . . . . .	3
2.6. Air coil system . . . . .	3
<b>3. The muon detection system</b>	<b>5</b>
3.1. Scintillator modules . . . . .	5
3.2. Photomultipliers . . . . .	5
3.3. Gains, Thresholds and Acceleration Voltages . . . . .	6
<b>4. Data aquisition crate</b>	<b>7</b>
4.1. First level trigger cards . . . . .	7
4.2. Second level trigger cards . . . . .	7
<b>5. Orca control</b>	<b>9</b>
5.1. Software Gains and Thresholds . . . . .	9
5.2. Run control . . . . .	9
5.3. File handling . . . . .	9
5.4. Orca Fit . . . . .	9
<b>6. Analysis software</b>	<b>11</b>
6.1. Data structure . . . . .	11
6.2. Search Algorithms . . . . .	12
6.2.1. Frequency Search . . . . .	12
6.2.2. Incremental Search . . . . .	13
6.3. Member Functions of the class <b>run</b> . . . . .	13
6.3.1. Constructor <b>run()</b> . . . . .	13
6.3.2. Destructor <b>run()</b> . . . . .	13
6.3.3. <b>getRun()</b> . . . . .	13

6.3.4. getLocalRun()	13
6.3.5. detectCoincidences()	14
6.3.6. channelCoincidences()	14
6.3.7. loopOverSides()	14
6.3.8. panelCoincidences()	14
6.3.9. findSmallest()	14
6.3.10. TOFHist()	14
6.3.11. TOFMuonDet()	15
6.3.12. determineEfficiency	15
6.3.13. getSize()	15
6.3.14. readVetoEventData(), readDetectorData() and readMOSDetector-Data()	15
<b>7. Analysis</b>	<b>17</b>
7.1. Gain-, Threshold and Acceleration Voltage Settings	17
7.2. Finding the best filter settings	17
7.3. Modules in high magnetic fields	18
7.4. Module Stability	18
7.5. Module Efficiency	18
7.6. Photo Multiplier Tube Test with $^{60}\text{Co}$ source	18
7.7. Coincidence Search between Muon- and Detector Events	19
7.7.1. Monitor Spectrometer	19
7.7.2. Main Spectrometer	20
<b>8. Simulation software</b>	<b>21</b>
8.1. Geant4	21
8.2. Geometry setup	21
8.3. Muon generator	22
8.4. Hit counter	22
<b>9. Conclusion</b>	<b>23</b>
<b>10. Outlook</b>	<b>25</b>
<b>Bibliography</b>	<b>27</b>
<b>Appendix</b>	<b>29</b>
A. First Appendix Section	29

# **1. Introduction**

## **1.1. The standard model**

During the second part of the 20th century, a model has been developed to describe a huge portion of phenomena stating 16 particle, that is six quarks, six leptons (both made up of three particle generations) and the four Gauge Bosons. The latter are carriers of the standard models interactions of the former particles, meaning all interactions of matter are based on the exchange of one or more of the Gauge Bosons.

### **1.1.1. The Standard Model Neutrino**

## **1.2. Massive neutrinos**

### **1.2.1. Neutrino Oscillations**

### **1.2.2. Measuring Neutrino Mass**

## **1.3. Cosmic air showers**

## **1.4. Muon interaction with matter**



## **2. The KATRIN experiment**

### **2.1. Source Side and Transport Section**

#### **2.1.1. MAC-E Filter**

### **2.2. Pre-Spectrometer**

### **2.3. Main Spectrometer**

### **2.4. Focal Plane Detector System**

### **2.5. Solenoids**

### **2.6. Air coil system**





## 3. The muon detection system

The need for low background rates at the main detector requires for a good knowledge of background sources. Despite magnetic reflection and wire electrodes, cosmic ray and particularly cosmic muon induced background may be an issue for the KATRIN experiment. To gather and assess muon related data, scintillator modules have been installed at both the monitor spectrometer and the main spectrometer. While the monitor spectrometer is equipped with only two rather small modules, at the larger main spectrometer, 8 modules have been installed at different positions enabling the user to cover different regions of the vessel. This freedom is enlarged by installing the detection system on three independently movable trolleys.

### 3.1. Scintillator modules

The central part of the detection system are the eight scintillator modules. They are made of the synthetic material BC-412 which is utilized in applications requiring large area coverage [Cry05]. These have previously been used at **(From Where?)**. Every scintillator cuboid is read out by two sets of four photomultiplier tubes. Photons arriving at the short ends of the module are guided to the photomultiplier tubes via non-scintillating material which, away from that, exhibits similar optical properties. All other sides of the scintillator are covered in reflective foil to push detection efficiency to the maximum. Of the eight photomultiplier tubes per scintillator module installed, 4 are read out via one FLT channel. The background of low energy events can be reduced significantly by recording only events occurring on both sides of the module at once. Only coincident signals should be recorded by the DAQ, though, on some occasions, quite a lot of single side signals occur. To account for those, every dataset is first analysed by a search algorithm to filter them. **(reference search algorithms)**

**ToDo**

**ToDo**

### 3.2. Photomultipliers

Each Photomultiplier tube is made up of a layer of **(of what)** where photons from scintillation ionize the layers' atoms leaving electrons with their initial Energy less the ionization energy

**ToDo**

$$E_{e^-} = E_{phot} - E_{ion}$$

---

---

---

The electron is then accelerated and guided by the electric field from dynode to dynode, cascading to more and more electrons, as each electron's energy rises by  $e \cdot U_{acc}$  between each pair of dynodes.

### 3.3. Gains, Thresholds and Acceleration Voltages

To achieve the best possible event detection, the photomultipliers' acceleration voltages as well as the software gains and thresholds in Orca had to be adjusted. The focus here was to obtain Landau peaks with equal height and width, as the rates throughout the modules can be considered equal. At first, the acceleration voltages were kept low to limit the signal peaks' heights to around 2 V. Carefully setting the mentioned parameters, one achieved the following, well aligned curves:

Later in the commissioning process, it got clear from the handbooks that the photomultiplier tubes had to be operated at acceleration voltages of 1.5 kV and above. To keep the signals' height as small as possible, most of the tubes were limited to this minimal voltage, whereas the sides (**which ones**) were set to 1.6 kV over showing lower rates than the others. Following this procedure, the tubes seemed much more stable and comparable, as all the gains and thresholds could now be set to the same values while still showing aligned peak positions:

**ToDo**

## 4. Data aquisition crate

### 4.1. First level trigger cards

asdf

### 4.2. Second level triger cards



## 5. Orca control

### 5.1. Software Gains and Thresholds

### 5.2. Run control

### 5.3. File handling

### 5.4. Orca Fit



## 6. Analysis software

To analyse the data recorded by DAQ and ORCA software, completely new data structures fit to the needs of muon detection and coincidence analysis have been created. Methods have been implemented to further investigate data stored inside those structures.

### 6.1. Data structure

All data from the IPE-servers arrives converted from ORCA-specific formatting to .root files. Hence, ROOT Methods are used to extract data from these structures, while most of these methods are implemented as part of the KaLi package in Kasper, which constitutes for a complete and closed data transfer protocol. Through those structures, data will be cached locally and can be analysed.

Before heading to actual analysis, all data is stored in the runtime structures. Here, the newly written class **event** with the following members comes into play. For each member, corresponding set- and get-methods have been implemented. Furthermore, the operators "<", "<=", ">", ">=", "==", and "-" have been overloaded to compare the timestamps of the event class. This was useful especially since ADCValues are merely used for plausibility checking the data but not for quantitative analysis. Doing so, events and the classes derived from them can easily be compared and searching becomes cleaner and clearer.

Derived from the base event class are two more storage classes:

**panelEvent** storing the second ADCValue

and the common timestamp of events activating both panel sides and

**coincidentEvent** storing ADCValues of simultaneous events in multiple modules and the

**event** class members

- fADCValue
- fTimeSec
- fTimeSubSec
- fPanel
- fSide

**coincidentEvent** additional member

- fADCValue2

**coincidentEvent** additional members

- std::vector fADCValues
- fnPanels

number of modules involved: Every ORCA-run then utilizes the class **run** storing the data of the .root files in vectors of events. Recorded events should already be filtered - only simultaneously occurring events on the two sides of the same module should be recorded. As, under conditions not known, single sided events are recorded as well, a software workaround is needed. All events of one side of the modules are scanned to find whether a corresponding event with the same timestamp exists on the other side. If so, a coincidentEvent is created and pushed back into the run's vector of coincident events corresponding to the module it occurred in. Now, the user can decide on which modules to analyse with the setPanels() function. This can be done sequentially for multiple sets of modules without newly reading the run's data, as all the primary data is stored inside the event vector.

**run** class members

- std::vector events
- std::vector detectorEvents
- std::vector eventsByPanels
- std::vector coincidentEvents
- std::vector selectedPanels

## 6.2. Search Algorithms

To analyse data, at various points searches for events with a particular timestamp have to be performed. This was simplified by the time-sorted recording of events. A first implementation to search for coincident events was done on the base of average frequency and its standard deviation. This algorithm proved as fast and stable, though well applicable only for two sets of timed events. That is why an advanced incremental method has been created. The number of modules is now limited only by the physically available memory and the speed is even higher.

### 6.2.1. Frequency Search

As this algorithm was built to run on only two sets of data, it simply walks through one set incrementally and looks for corresponding data in the other. Latter is not done in a "dumb" way by incrementing through the second set as well, but by calculating the



---



---

average frequency of events inside the set and performing an intelligent guess on that basis. If the guessed event shows a different timestamp, the algorithm will keep going forward or backward in time in steps of the frequency's standard deviation until the timestamp searched for is in between two steps. Lastly, simple incrementation is used to find out whether an event at the desired point in time exists or not.

### 6.2.2. Incremental Search

While the frequency search increments solely one dataset, the incremental search steps through all the event tress, incrementing the one with the smallest timestamp. It then compares all events to each other, writes out the coincident ones, if any, and goes on incrementing the next smallest stamp. This assures the finding of all coincident events while keeping the speed very high.

## 6.3. Member Functions of the class `run`

### 6.3.1. Constructor `run()`

Whenever a new instance of "run" is created, the constructor is called. Arguments to be passed are a `KaLi::KLRunIdentifier`, basically a string distinctively naming the run to be analysed, such as "myo00000001", `KaLi::KLDataManager`, a class handling the download of the Files from IPE-servers and a toggle variable telling the constructor which data to read via the member function `getRun()` and what member functions to call afterwards:

### 6.3.2. Destructor `run()`

The destructor deletes all the contents of the vectors of events and inherited classes and clears them afterwards before deleting the member `RUN` which in fact frees all the memory reserved by the `KaLi` classes.

### 6.3.3. `getRun()`

This sets the member `KaLi::KLRun` through the `KaLi::KLDataManager` and then returns its `KaLi::KLRunEvents` - these include all recorded events meaning also both the relevant `KaLi::KLEnergyEvents` and `KaLi::KLVetoEvents`. The `getRun()` function is used for example in the constructor to read the run's data.

### 6.3.4. `getLocalRun()`

It is not always possible to read data from the file servers, example given were files too big leading to timeouts at least in older `KaLi` versions. That is why the `getLocalRun()` function was introduced reading data from the local filesystem via the `KaLi::KLRunIdentifier`. The path to the files needs to be adapted in the source code (**environment variable?**). ToDo

### Toggle Choices

- **0:** Data is downloaded and both muon data and detector data are stored
- **1:** Data is downloaded and only detector data is stored
- **2:** Data is downloaded and only muon data is stored
- **3:** Data is read from local file system, only muon data is stored

### 6.3.5. detectCoincidences()

After calling the member function `channelCoincidences()`, `panelCoincidences(nPanels)` is returned where `nPanels` defines, how many modules have to show coincidences for the counter to increment.

### 6.3.6. channelCoincidences()

This **always** clears the vector `eventsByPanels` before filling it according to the current `selectedPanels` settings. To do so, it loops over all entries of `selectedPanels`, calling `loopOverSides()` of the current module.

### 6.3.7. loopOverSides()

Analysing only one of the modules for coincident events between the two sides, the function runs through all the events of one panel side using the operators "`<`" and "`==`" overloaded for the class `run` to compare event times. For the search itself, the "A" side's index is incremented step by step while the "B" side's index is pushed up as long as its event time is smaller than A's. Every time that condition changes, it checks whether the events occurred at the same time - pushing a `coincidentEvent` with both the events `ADCValues` and the timestamp into the vector for the corresponding module if so - and then going on incrementing A's index.

### 6.3.8. panelCoincidences()

**ToDo** As mentioned above, the first algorithm to search for coincidences between different panels was based on the average event frequency and its standard deviation, soon being replaced by a simpler, more efficient incremental algorithm: This features a storage for the smallest timestamp in a group of events. **(change code to overl.ops)** This is set to the smallest timestamp of the first event of all the modules analysed. Now, all the events are compared to the smallest one. This has the advantage, that one does not need to cross check every event with every other one but can simply compare every event to the smallest in a linear way. If simultaneous events are found, they are pushed back into the `coincidentEvents` vector together with the timestamp and their `ADC` values, `nPanels` is risen by one. Subsequently, the index of the smallest event storage is incremented and the new smallest event in the changed pool is searched for via the member function `findSmallest()`. This is repeated until all the event storages have reached their last entry. The return value is the number of events fulfilling the requirement passed through `nPanels` to `panelCoincidences`: if it is zero, every coincident event with two or more modules involved is counted, for every other number, only the number of event with exactly this number of modules is counted.

### 6.3.9. findSmallest()

Setting the smallest event's timestamp through references, the `findSmallest` function requires only the steppoints of the different modules and returns the one with the smallest timestamp.

### 6.3.10. TOFHist()

Setting the modules to be analysed to one and two, this function was designed to analyse monitor spectrometer data. This also reflects in the fact, that both muon data and detector data are expected to be stored within the same `mosxxxxxxxxx` run file. The function then runs `channelCoincidences()` and `panelCoincidences()` before shifting through all the muon events searching for following detector events in a certain time interval.

- **default/1:** Size of events returned
- **2:** Size of eventsByPanels returned
- **3:** Size of coincidentEvents returned
- **4:** Size of detectorEvents returned

### 6.3.11. TOFMuonDet()

In contrary to the TOFHist function, this one reads muon and detector data from different files as it is designed for the needs of the main spectrometer. Here, two DAQs record muon and electron detections to myoxxxxxxxxx and fpxxxxxxxxxx files respectively. That is why the function reads a muon run and requires a guess as to where corresponding detector data is located. It then loads on detector run after the other to check whether there's events inside a time window after each muon event. If so, a histogram is filled with the data acquired to inspect it for cumulation at particular times.

### 6.3.12. determineEfficiency

Efficiencies of modules can be determined through three of them located coextensively in front of each other. Then, all events recognised in both the uppermost and the lowest module have to - leaving geometrical inaccuracies unaccounted for - pass the middle module as well. By comparing the counts one can determine an efficiency for the module:

$$\%_{eff} = \frac{\wedge_{68}}{\wedge_{678}} \quad (6.1)$$

### 6.3.13. getSize()

The getSize() function returns the size of one of the vectors storing events or one of the inherited classes depending on the passed integer "what":

**(default nonsense, reimplement)**

**ToDo**

### 6.3.14. readVetoEventData(), readDetectorData() and readMOSDetectorData()

Depending on the toggle choice in the constructor, either one of the two or both of the functions are called. While the readDetectorData() function reads all recorded KaLi::KLEnergyEvents (only the FPD records those), the readVetoEventData() function reads all the KaLi::KL vetoEvents from cards three, six and nine. This can never interfere with veto data recorded directly around the FPD vor active vetoing of detector signals, as cards 15 and 16 are used here. For analysis of monitor spectrometer data, a function readMOS-DetectorData() has been implemented reading all energy events of card one independent of channel, while of course single channels can easily be excluded. The pulser usually active at the monitor spectrometer creating KaLi::KLEnergyEvents at constant frequency is standardly excluded from analysis. All the member functions reading data require the passage of an instance of the KaLi::KLRunEvents, usually the member of the same class set in the getRun() function.



## 7. Analysis

Using data obtained by the muon modules and the detector as well as other subsystems' data, the muon induced background rates and both spatial and energy distribution can be obtained. Before actual measurements, the modules had to be set up and calibrated. Comparable rates for all modules and

### 7.1. Gain-, Threshold and Acceleration Voltage Settings

A first amplification - linear with acceleration voltage - of the scintillation photons occurs in the photomultiplier tubes. As signals at first seemed too high for the DAQ to handle at nominal values of the datasheet, these were reduced to around 1200 V. Now, the software gains and thresholds in the ORCA software needed to be set carefully to make

### 7.2. Finding the best filter settings

As the PMT tubes are directly, without any preamplifiers, connected to the DAQ, the signal lengths arriving at the latter are in the order of 20 ns. This poses a problem for filters as the sampling rates need to be high and anti-aliasing is inevitable. To find the best settings, a pulser has been set up to create events at known frequency and peak height. The signal form (**what form**) was chosen as closely to the actual shape as possible

**ToDo**

Now, to evaluate filter goodness, the width of the resulting energy histogram, which should, assuming perfect pulser signals and perfect filters, be monoenergetic, was analysed for each filter setting. This resulted in the following set of data:

On average, the boxcar filter at shaping lengths (**was it shaping?**) of 150 ns shows the most promising results. This concurs with the settings chosen for the active fpd veto; here slightly longer (around 30 ns) but comparable signals occur, showing best results at the same filter settings[Wie13].

**ToDo**

Figure 7.1.: Pulser shape compared to actual signal shape

---

Table 7.1.: Energy resolution at different filter settings

50 ns gap, 0 s shaping time			
standard filter	1	2	3
	1	2	3
	1	2	3
	1	2	3
boxcar filter	1	2	3
	1	2	3
	1	2	3
	1	2	3

---

### 7.3. Modules in high magnetic fields

### 7.4. Module Stability

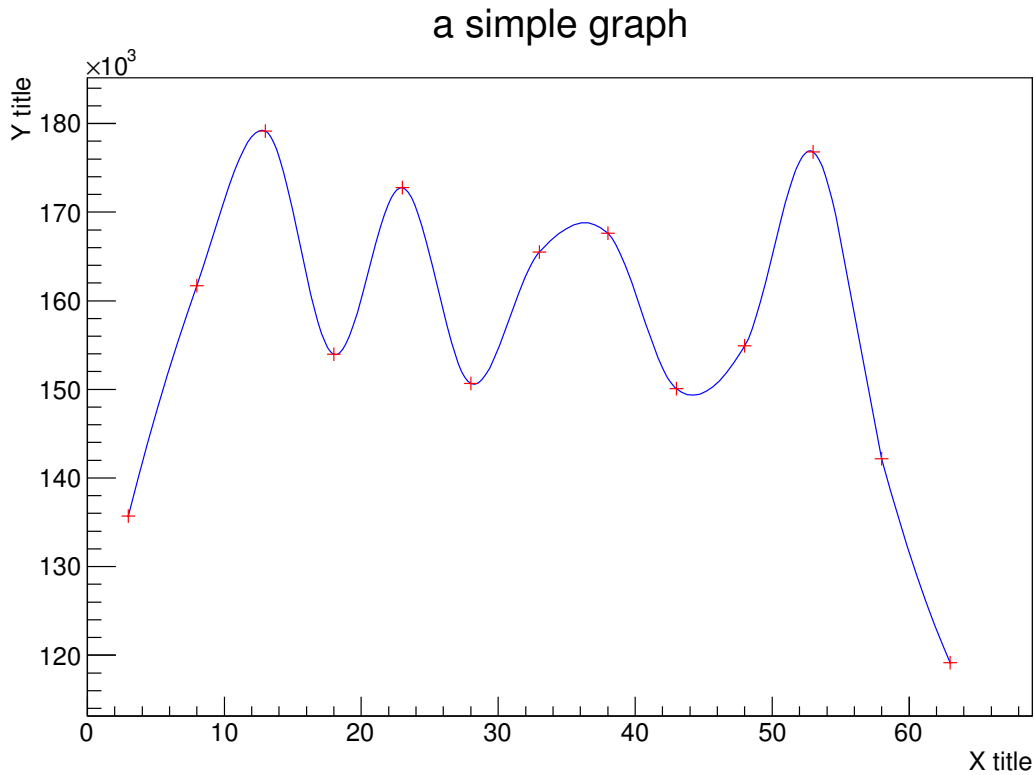
If consistent on muon induced background is to be made, the modules need to work stable over the course of days, as rates are supposed to be comparable. For this reason, over the Christmas time 2012, a two weekly measurement has been taken. The timeslot was chosen for the lowly frequented spectrometer hall to exclude outer (**D: einflusses**). For analysis, a simple program to count events in variable time bins was written, creating a rate histogram for all the runs in the measurement period. The following fluctuation could be observed: These are closely related to the fluctuations in atmospheric density, pressure and temperature. If one looks at the data from (**find weather station data, see if it fits fluctuations**)(**maybe cascade data to back it up?**)

### 7.5. Module Efficiency

The runs used for stability measurements, as well as any other including modules six, seven and eight, can be used to check muon module seven for efficiency. For every other module, the geometry would need to be changed so that the one to be checked is in between at least two other modules. For analysis, the function determineEfficiency() 6.3.12 has been written. It shows that during the measurement period end of 2012, the efficiencies were at (**rerun**) 94 % which is less than one would expect at a thickness of 5 cm. For that reason, the filter settings were checked and changed to the boxcar filter with a gap of 150 ns from the before used (**exact name**) filter. This lead to an increase in efficiency of about (**x percent**). As the efficiency is still below expected values, a possible improvement would be to use preamplifiers before signals arrive at the DAQ, which would then need to have a higher sampling frequency and widen the signal timewise.

### 7.6. Photo Multiplier Tube Test with $^{60}\text{Co}$ source

With sets of four photomultiplier tubes being read out over one cable, and, consequently, via one channel, the test of individual PMTs is not trivial. Nevertheless, a method using a MBq  $^{60}\text{Co}$  source to trigger events was used to check functionality. Of course, all tubes were able to see the source at any position but rates were expected to rise as the distance to any of the tubes shrank. A source holder was constructed from acrylic glass to shield the user from radiation and to attach the source to the modules, as a large dependence of rate on the position was found when the source was simply duct taped to the modules. As the foil mantling the modules absorbs a non negligible part of the radiation emitted from the source, it had to be ensured that the number of layers was equal for all measurements.



This was given only below the modules as the foil has been folded around them at the ends in a gift wrapping way. Thus, the source was pretty far away from the photomultiplier tubes making it more difficult to distinguish between them. A first measurement was then to check for exactly that distinguishability. (insert measurement for small steps over PMTs positions). As one can see a rise in rate at the positions the tubes are located at, it was decided that four measurements per module and side were sufficient, especially as all measurements can afterwards be compared to each other. The tube positions at (n,n,n,n cm) were used as measurement positions as well. For each side, a run has been taken containing five minute subruns for every position. Figure ?? shows the result of these measurements. One can see that the general shape of each side compares very well to the others. Exceptions are (which ones different?) showing slight, but acceptable deviations off the norm.

ToDo

ToDo

ToDo

## 7.7. Coincidence Search between Muon- and Detector Events

If one wants to actually detect background induced by myonic events registered by the muon modules, those events need to be correlated to detector events timewise. For this purpose, the analysis code's class run was extended by the member functions TOFHist()6.3.10 and TOFMuonDet()6.3.11, where the former is used for monitor spectrometer analysis and the latter for the main spectrometer.

### 7.7.1. Monitor Spectrometer

Measurements at the monitor spectrometer are easily manageable due to the fast accessibility of all the components and the collection of data in a single run file through the mini crate. Several hourly runs have been taken under different magnetic field compositions. Both asymmetric magnetic (ref to figure of field lines) and non-axially-symmetric field configurations have been investigated. The TOFHist6.3.10 function has been used to analyse the data. In all of the settings, a clear peak is visible at around 7  $\mu$ s. As for count rates, they are a lot higher in the asymmetric magnetic field setup as secondary electrons

ToDo

---

are guided from their point of origin to the detector instead of mostly being magnetically shielded. In this setup, only the reflection through the rise in magnetic field takes its toll on the rate.

### 7.7.2. Main Spectrometer

Monitor spectrometer results suggested that the time of flight was well measurable, even if on bigger scale, at the main spectrometer. So, during first light measurements, already during measurement M1, some runs with asymmetric magnetic field have been taken with switched polarity or turned off pre spectrometer magnets compared to standard setup. The data was analysed for each single ring of the FPD, though remains inconclusive at the moment. This might be due to the combination of muon module position and the magnetic field setup as the wall area covered by the flux tubes and the volume surveilled by the muon modules did not overlap very much.

Following that presumption, for M6 measurements, the setup was changed. The fluxtube was widened for its outer parts to hit the spectrometer wall in regions around combs n and n (**which combs**). This raises the probability of the muons inducing secondary electrons being detected by the modules as well. The resulting flux tube is shown in figure ?? (**flux tube including modules for horizontal and vertical view**) (**If data will be taken add to analysis.**)

ToDo  
ToDo  
ToDo

Figure 7.2.: new flux tube

---



## 8. Simulation software

To compare the data acquired to theoretically expected values, a Geant4 simulation of cosmic showers has been set up including the geometry of the main spectrometer as well as the muon modules.

### 8.1. Geant4

The Geant4 package is a powerful tool for simulation of particles. It has loads of possible interactions already integrated making it easy for the user to set up and run simulations. After setting up geometry and detectors, the user starts a run. Each run may consist of one or more events. During a single run, a loop of processes is called, i.e.:

- Primary Generator Action
- Run action
- Event action
- Stacking action
- Tracking action
- Stepping action

For the user, for each action, there are classes called user action called before or after the standard action class. Through those, it is possible to change behaviour of the simulation or extract data needed.

Running the simulation, one can either interactively enter commands or write those to a .mac file, by default the vis.mac file, which are then sequentially executed.

### 8.2. Geometry setup

The main spectrometer geometry was already (**written by who?**) existant but had to be modified as many of the border volumes implemented here were not needed for this simulation. Additionally, the muon modules have been added as sensitive volume.

**ToDo**

### 8.3. Muon generator

Muon generation was realised through the primary generator action. The angular distribution suggested by Henrik Arlinghaus (**cite**) was implemented. The angular rate dependence is shown in (**ref figure**). The energy was set to (**reasonable value**) disregarding the actual anergy distribution as this was mainly about flight paths that are not strongly dependant on energy at high energies. Starting positions were spherically distributed, with the direction towards the origin, which is in the center of the main spectrometer. Positions were then randomly moved in a volume surrounding the spectrometer to account for the non-point like structure of the detection system as a whole.

**ToDo**  
**ToDo**  
**ToDo**

### 8.4. Hit counter

For comparing the simulation to real data, of all the events generated, those hitting modules were counted. Each event containing at least one hit is written into a file for further analysis. This made it possible to compare the rates of the single modules, showing that the generator works fine. furthermore, it allowed for a estimation of muons hitting the modules compared to the total of inciding muons.

## 9. Conclusion



## 10. Outlook

...



# Bibliography

- [Cry05] S.-G. Crystals, “Premium plastic scintillators,” [http://www.detectors.saint-gobain.com/uploadedFiles/SGdetectors/Documents/Product\\_Data\\_Sheets/BC400-404-408-412-416-Data-Sheet.pdf](http://www.detectors.saint-gobain.com/uploadedFiles/SGdetectors/Documents/Product_Data_Sheets/BC400-404-408-412-416-Data-Sheet.pdf), 2005, [Online; accessed 11-07-2013].
- [Wie13] K. Wierman, personal conversation, 2013.





# Appendix

## A. First Appendix Section

ein Bild

Figure A.1.: A figure

...