Readout of Data from an Advanced Detector System in Nuclear Physics

Mikael O. Bonnier

Department of Physics
LUND 2003

Readout of Data from an Advanced Detector System in Nuclear Physics

Master of Science Thesis

By due permission of the Faculty of Science (Naturvetenskapliga
fakulteten) at Lund University to be publicly presented in
room C366 at the Department of Physics
on October 7th, 2003, at 13:15
by

Mikael O. Bonnier

LUND
UNIVERSITY

DEPARTMENT OF PHYSICS
LUND, 2003

Supervisor: Professor Bo Jakobsson
Division of Nuclear Physics
Lund University
P.O. Box 118
221 00 LUND
Sweden

Web: http://www.df.lth.se/~mikaelb/
E-mail: mikaelb@df.lth.se
ICQ # 114635318

Printed in Sweden
Lund 2003
1st Edition
4th printing

Latest edition can be downloaded from: http://www.df.lth.se/~mikaelb/msc/

Organization
LUND UNIVERSITY
Division of nuclear physics
Department of Physics
Lund University
P.O.Box 118
SE-221 00 Lund, Sweden

Document name
MASTER THESIS

Date of issue
February 13, 2004

CODEN

Author(s)
Mikael O. Bonnier

Title and subtitle
Readout of Data from an Advanced Detector System in Nuclear Physics

Abstract

An experiment at the CELSIUS storage and accelerator ring in Uppsala, where a Ne-beam collides with an Ar-target at 200 MeV/nucleon, is described.

The advanced nuclear physics $\Delta E - E$ detector system CHICSi is described. The full CHICSi consists of several hundred telescopes, covering 3π sr, each equipped with Si and GSO/PD detectors, pointing toward the intersection of the accelerator beam and the gas-jet target flow. The particles from the nuclear reactions are detected.

The readout system for the telescopes is described and ROOT C++ softwares for converting and analyzing data from CHICSi is developed and used.

A code is developed and used to visualize data from all telescopes, and to observe structures in the images.

A method for calibrating the detectors using the stopping power code TRIM98 is explained and used for some detectors.

Three isotopes of Be are identified and the angular distribution for certain intervals of energy is investigated by developing and using a program that counts events inside polygons.

In the introduction some topics of heavy ion theory is presented, and a way to compute the results of the experiments from theory is indicated.

Key words:
Multi-detector system, ultra-high vacuum compatibility, VLSI electronics, ultra-thin detectors, heavy ion reactions, data acquisition, data analysis, ROOT, C++

Classification system and/or index terms (if any)

Supplementary bibliographical information:

Language
English

ISSN and key title:

ISBN

Recipient's notes

Number of pages
65

Price

Security classification

Distribution by (name and address)
Mikael O. Bonnier
Östen Undéns gata 88, SE-227 62 LUND, Sweden

Signature **Date**

Contents

List of Tables

List of Figures

Abstract

An experiment at the CELSIUS storage and accelerator ring in Uppsala, where a Ne-beam collides with an Ar-target at 200 MeV/nucleon, is described.

The advanced nuclear physics $\Delta E - E$ detector system CHICSi is described. The full CHICSi consists of several hundred telescopes, covering 3π sr, each equipped with Si and GSO/PD detectors, pointing toward the intersection of the accelerator beam and the gas-jet target flow. The particles from the nuclear reactions are detected.

The readout system for the telescopes is described and ROOT C++ softwares for converting and analyzing data from CHICSi is developed and used.

A code is developed and used to visualize data from all telescopes, and to observe structures in the images.

A method for calibrating the detectors using the stopping power code TRIM98 is explained and used for some detectors.

Three isotopes of Be are identified and the angular distribution for certain intervals of energy is investigated by developing and using a program that counts events inside polygons.

In the introduction some topics of heavy ion theory is presented, and a way to compute the results of the experiments from theory is indicated.

Chapter 1

Introduction

There is a discussion in nuclear physics that concerns how nuclear matter behaves under different pressure and temperature. Even though in this experiment only 60 nucleons are involved, one often considers theoretical models with an infinite number of nucleons.

A specific goal in the CHIC (CELSIUS Heavy Ion Collaboration) program at CELSIUS is to prove or disprove the predicted liquid-gas phase transition in nuclear matter, a question of great importance for understanding the astrophysics of supernovae and neutron stars.

The ordinary way to handle collisions in quantum mechanics, using time dependent perturbation theory, is not suitable for heavy ion collisions. In the perturbation theory a perturbing potential causes a flow of particles to change direction. This is similar to diffraction of light due to a small grain. In our experiment 2 heavy ions of comparable weight collides at relativistic speeds causing some nuclear reaction. A method to compute the results is to regard this as a collision between individual nucleons in the field of all the others.

The ions in this experiment are ^{20}Ne in the accelerator beam at 200 MeV/u (A MeV) and ^{40}Ar in a gas-jet target. The gas-jet, also called cluster-jet, is essentially fixed compared to the accelerator beam. In order to compare the measured data to theory one should transform the laboratory system to the center of mass system using the Lorentz-transformation, since the theory is most easily worked out in the center of mass system.

The experiment is conducted in the CELSIUS storage ring at The Svedberg Laboratory in Uppsala, Sweden. A feature of the CELSIUS storage ring is its slow ramping operation, *i.e.* it can slowly increase the energy of the beam. This can be used to scan over interesting energy intervals to study production rates of particles as a function of collision energy. However, this feature was not used in this experiment.

The purpose of the CHIC program is to completely characterize the collision events. The component in the CHIC program tested in this experiment is CHICSi – a cylindrical multi-detector system using Si-detectors and GSO scintillators with photo diodes.

In the old days one would use cloud- and bubble-chambers for the job that CHICSi is designed to do. The advantage with CHICSi over these chambers is that it has better time resolution to separate events, and that it is possible to analyze huge amounts of data in a fully automatic way. It is of course the computer revolution and the possibility to custom design VLSI chips, together with developments of better solid state detectors, that has made CHICSi possible.

The CHICSi detectors are placed inside the CELSIUS storage ring in ultra high vacuum (UHV) and this is unique and has been a tremendous challenge. Therefore one can also expect unique results. A spin-off of the CHICSi development could be new space technology.

In the CHIC team there are also other groups with other detector systems that work in unison with CHICSi. One is the PF-WALL, which detects projectile fragments using phoswich detectors in the most forward direction that CHICSi does not cover, and the other is HR-TOF, which detects slowly moving fragments using a time-of-flight technique.

The experiments in this report are mostly for testing the detectors, electronics, and low level software together with the data acquisition system. My work has been to develop low level software to convert the raw data into a format that can be analyzed, and higher level software to view and analyze the data from CHICSi. The purpose of this is to discover and solve problems and to prepare for future physical discoveries.

Chapter 2

The Experiment

In this chapter I say something about how the Ne-beam and the Ar-target were introduced. Here is also some information about the experimental setup, the electronics, and the low-level data acquisition system. My specific tasks in the experiment are discussed both here and in later chapters. However, since I only saw a limited part of the whole experiment there can be things which are important, but which I missed.

2.1 Beam

The Svedberg Laboratory (named after the Swedish chemist Theodore Svedberg, Nobel Prize 1926) in Uppsala, Sweden, houses the accelerator facility used in this experiment. This contains an external Electron Cyclotron Resonance (ECR) ion source, the Gustaf Werner Cyclotron, the CELSIUS storage and acceleration ring with electron cooler, and the cluster-jet target.

2.1.1 The Gustaf Werner Cyclotron

The ions are produced in the ion source building. For heavier ions, from ^{12}C to ^{129}Xe, the ECR ion source is used and the ions are injected in the center of the cyclotron.

A cyclotron is circularly shaped and consists of two hollow D-shaped parts, see figure 2.1. The ions are accelerated across the gap between the "dees" by an alternating voltage but are forced to move in a circle by the external magnetic field. As the energy of the particle increases, the radius increases, and the ion spirals outward. For non-relativistic speeds the time for each turn is the same, and this makes it easier to accelerate the ions since one can use the same constant cyclotron frequency to accelerate all ions

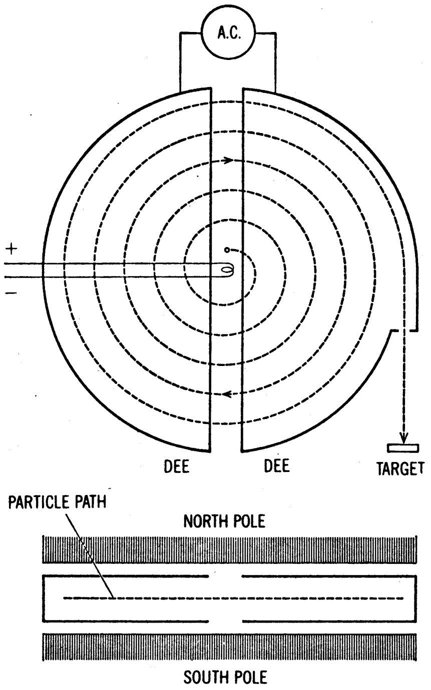


Figure 2.1: *The principle of the cyclotron. Top view (above) and side view (below) (from [1]).*

across the gap. If the ion speed becomes relativistic one needs to operate the cyclotron in synchrocyclotron mode and this requires more sophisticated control of the frequency.

The Cyclotron Control Room

A few pictures of the cyclotron control room (see figure 2.2) and the important high frequency equipment (see figure 2.3) are shown in this section.

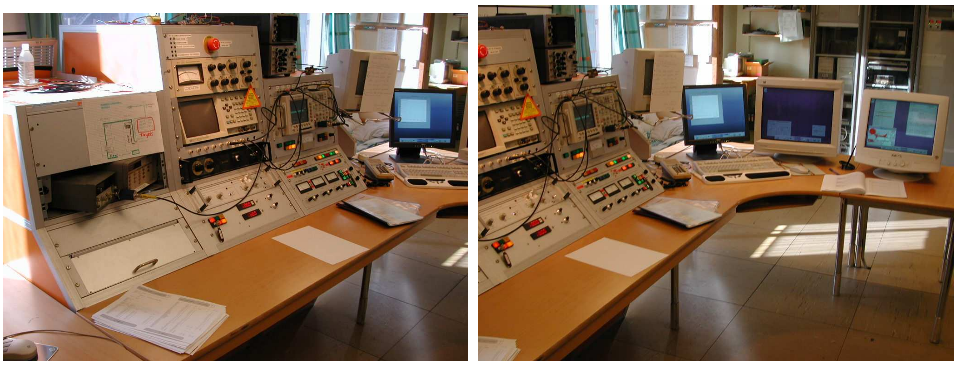


Figure 2.2: *The Gustaf Werner cyclotron control room.*

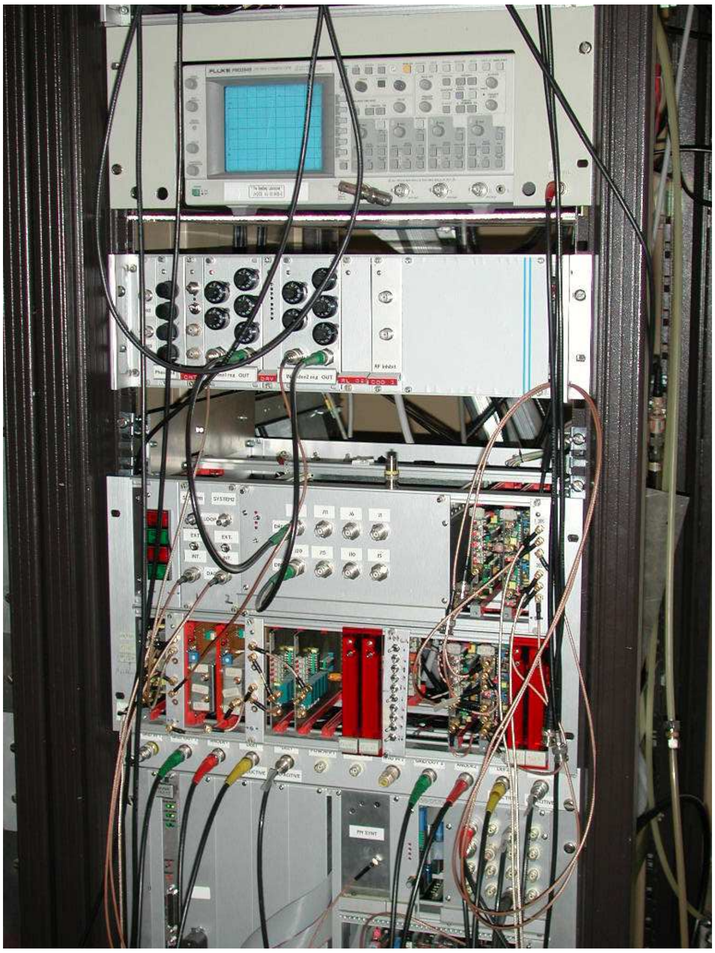


Figure 2.3: *The high frequency equipment.*

2.1.2 The CELSIUS Storage Ring

The CELSIUS storage ring consists of 4 straight sections and 4 curved sections in the shape of a rounded square. Magnets bend the beam in the rounded corners. At the 1st straight section the ion beam from the cyclotron is injected, at the 2nd there is an experimental station with a pellet target that operates with frozen hydrogen and deuterium, and at the 3rd there is an HF-voltage accelerating the beam and an electron cooling system (explained below). The 4th section is where the cluster-jet target and CHICSi are placed.

Electron cooling

An interesting part of CELSIUS is the system for cooling the beam using electrons. Cooled electrons are injected and travel along the beam and picks up heat from the beam. The electrons are extracted a few meters down beam stream.

In figure 2.4 is an example of the beam current cycle of a cooled ^{20}Ne beam, and in figure 2.5 the beam is uncooled. Here the target (^{40}Ar gas-jet) is on. The beam current can vary from cycle to cycle. Some ^{20}Ne nuclei are removed from the beam due to collisions with the target. Figure 2.5 is an example of a beam cycle during the data recording of the experiment.

The Storage Ring Control Room

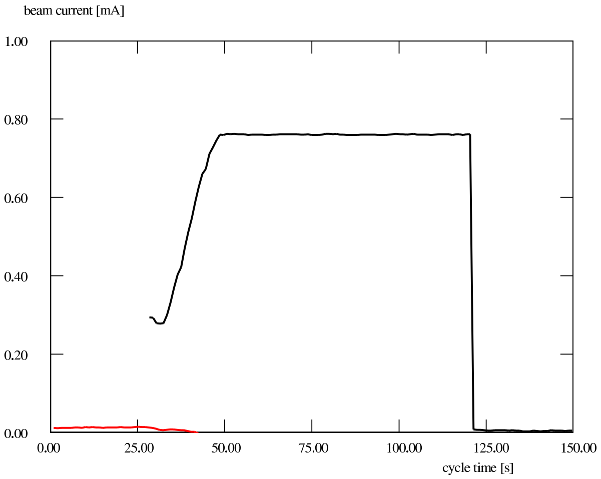
During our experiment on Saturday 14 June I visited the storage ring control room and took some pictures (see figure 2.6).

I also checked the pressure meter at the ion pump control which showed $4.0 \cdot 10^{-7}$ mbar ($4.0 \cdot 10^{-5}$ Pa) at 18:43. The ion pumps were not on at this time.

Later on Saturday evening we made a short routine break (semi) in CELSIUS operation to go in and start and check some pumps and other things.

2.2 The Cluster-jet Target

The cluster-jet target station produces a gas-jet going through the beam from top to bottom (see figure 2.7). In our experiment the gas was ^{40}Ar.



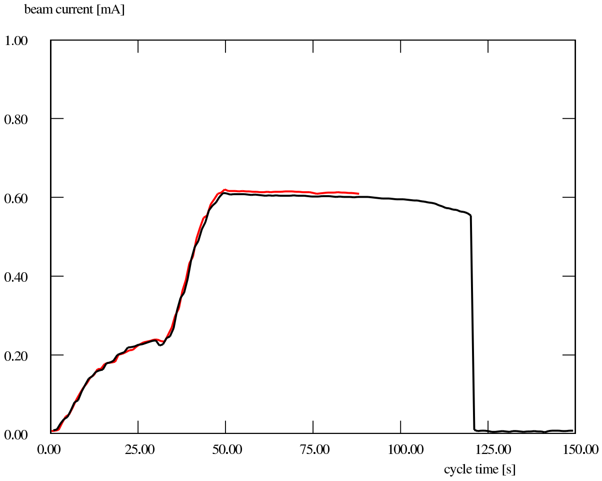


Figure 2.4: *Top: Beam of 4 GeV* $^{20}Ne^{10+}$ *current versus time for the intense electron cooled beam. Without target jet.*

Figure 2.5: *Bottom: Beam 2003-jun-15, 01:45. Without electron cooling. With target jet on.*

Figure 2.6: *The CELSIUS storage ring control room.*

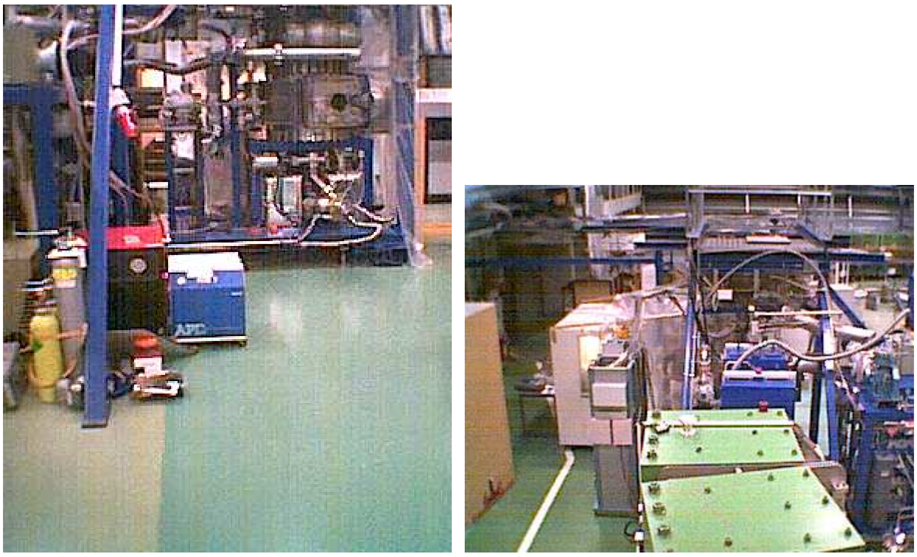


Figure 2.7: *Argon cluster-jet target (left). Target chamber from above (right).*

2.3 How the Beam was Developed and Tuned

The plan in tables 2.1 and 2.2 was followed except for some delays due to problems. It was a warm week in June and the cooling system for the vacuum pumps broke down. At one time the cooling water was 50°C. But they were repaired. Also some tuning of the beam took longer time than expected. The operation of TSL was prolonged to Sunday June 15 and was stopped at 11 a.m.

2.4 How the Experimental Setup was Introduced

CHICSi uses the method of $\Delta E - E$ to determine the charge and mass of a particle and its energy within a wide range. This is explained in chapter 4. In order to use this method 3 detectors are mounted on mother boards which are placed in small steel boxes – these units are called telescopes. The telescopes are described in section 2.4.1.

The telescopes are mounted on Grand Mother Boards (GMB) which are mounted on the CHICSi skeleton. There are 12 short telescopes and 6 long telescopes on each GMB.

In total there can be 28 GMBs in CHICSi. One can thus describe CHICSi as a cylinder with 18 rings of 28 telescopes each, in total 504 telescopes pointing toward the crossing of the accelerator beam and the cluster-jet target. In this experiment we used the 2 GMBs produced to date, called GMB I and GMB II. In the November 2002 experiment they were mounted at angles $\phi = -20°$ and $\phi = 90°$, and in the current June 2003 experiment they were mounted at $\phi = 77.1°$ and $\phi = 38.6°$, respectively ($\phi = 0°$ is up). The positions should be physically interesting but also possible with the current cabling.

On Friday June 6 we mounted the GMBs and the cables that connect the GMBs to a connector on the inside of the cluster jet target station. On the outside this is connected to the Power Distribution Board (PDB) which in turn is connected to the low level data acquisition electronics. The PDB had wrong connectors and these had to be replaced. I helped in checking the new connectors for the PDB.

In this experiment we used new UHV compatible cables, that was made by a company in Malmö, Sweden. The new cable had lower out-gassing than the old. I also helped in checking the new cable.

Then the accelerator was closed and pumping started, and the beam and target was switched on and we started to take data. We then discovered that

Time	*Activity*
Tue 10 June 14	• 17.28 MeV/u ^{20}Ne^{6+} from the cyclotron. • Transport the beam to CELSIUS. • Cooler fields off. • Find ^{20}Ne^{10+} beam in CELSIUS and optimize. • Turn on cooler fields. • Find electron cooling. • Optimize accumulation. • Get 300 s cycle with acceleration to 200 MeV/u. • Turn on argon target, verify overlap.
02	• Un-cooled beam of 200 MeV/u ^{20}Ne^{10+} for CA47.
Wed 11 June 06	• Un-cooled beam of 200 MeV/u ^{20}Ne^{10+} for CA47.
10	• Get electron cooling at flat top, optimize carefully. • Turn on argon target, verify overlap. • Measure beam size with Mg-jet beam profile monitor and beam energy spread with Schottky noise, try to quantify halo.

Table 2.1: *Plan for CA47 in June 2003/*preliminary.

Time	*Activity*
Wed 11 June 18	• Cooled beam of 200 MeV/u $^{20}\mathrm{Ne}^{10+}$ for CA47.
Thu 12 June 06	• Cooled beam of 200 MeV/u $^{20}\mathrm{Ne}^{10+}$ for CA47.
Fri 13 June 06	• Cooled beam of 200 MeV/u $^{20}\mathrm{Ne}^{10+}$ for CA47.
Sat 14 June 06	• Cooled beam of 200 MeV/u $^{20}\mathrm{Ne}^{10+}$ for CA47.
18	• Stop.

Table 2.2: *Plan for CA47 in June 2003/*preliminary *(continued).*

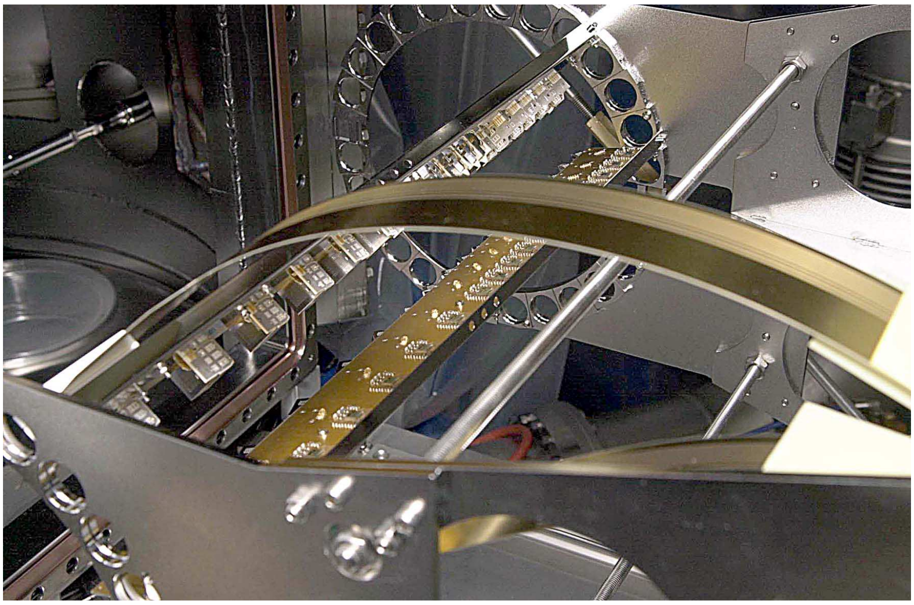


Figure 2.8: *CHICSi before mounted in chamber (photographer: Teddy Thörnlund).*

communication with the GMBs didn't work.

It turned out that the new cable was broken. Apparently the cable wasn't possible to bend in the way seen in figure 2.8, without destroying the electrical lines.

The cable consists of several layers and my guess is that the outer layers in a bend stretched and broke. The old type of single wire cable was reintroduced and this operated well for the remaining experiment.

2.4.1 Telescopes

The small stainless steel boxes with the detectors are called telescopes. There are 2 types of telescopes. The 1st shorter type has 3 Si-detectors where the 1st is about 12 μm thick, the 2nd is 300 μm thick, and the 3rd is also 300 μm thick. In the 2nd longer type of telescope the 1st and 2nd detector is similar to the 1st type of telescope, but the 3rd is a 6 mm GSO scintillation crystal with a photo-diode.

Before the first detector there is a thin 1 μm Ni-foil. The detectors are surrounded by metal in order to shield from electromagnetic fields. The steel can also shield against stray radiation not entering the telescope from the opening.

Figure 2.9: *Outside target chamber: Pavel Golubev and Lars Carlén are setting the detector thresholds, behind the plastic coverage needed to have a clean environment.*

2.4.2 Signal processing

There are specially made VLSI chips that preprocesses the signals from the detectors in each telescope. The chips are mounted on the GMB close to each telescope.

The signals from the chips are read out by the ADC boards. One ADC board can read out 2 GMBs. The ADC boards is read by Combox, which sends the data to the master module, which sends the data to the high level data acquisition system. This system is described in detail in [19] and in paper III in [5].

One problem that was discovered was that some trigger pulse was too low, due to a overheated power regulator that sunk the in voltage. Another problem was that a MAX120 ADC broke and had to be replaced.

Some problems with the individual threshold settings was discovered and later on solved (see figure 2.9). The detection thresholds are set via Combox that is connected to a PC via a serial RS232 port. The Combox communicates with the VLSI chips on the GMBs via an I2C bus. Both the ADC boards and Combox use Microchip PIC micro-controllers and these are described in [15], this book also explains the I2C protocol. An example of a pedagogical PIC project is in [2].

Another problem was then discovered: the zero levels of the detectors changed due to disturbances. It was, as seen in the data in chapter 5, apparently not possible to reduce all disturbances.

One way to reduce disturbances is to make shielded circuit boards and reduce the cabling, but I don't know if this would have helped in this case.

2.4.3 Low-level Data Acquisition

An overview of the low-level data acquisition system is in figure 2.10. There is also a higher level data acquisition system which stores incoming data to tape. This is treated in chapter 3.

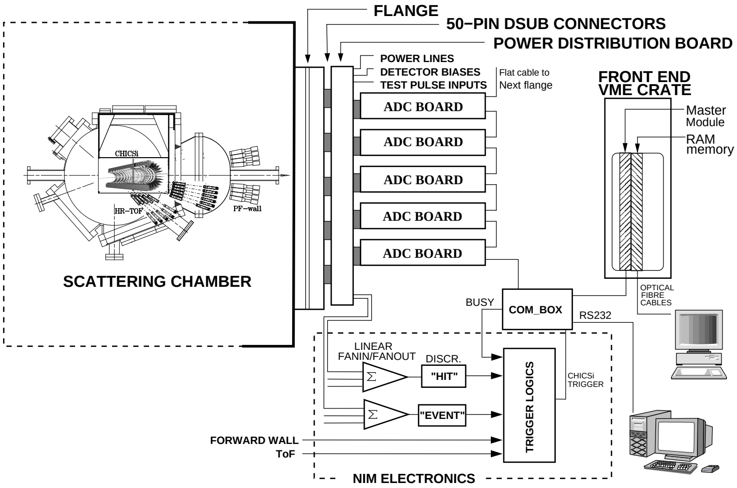


Figure 2.10: *Overview of the low-level data acquisition system for CHICSi (from Paper III in [5]).*

Some of the software at the low level had to be adapted to the higher level data acquisition system, and vice versa. If we had our own data acquisition system for CHICSi that we could use in Lund, where we develop the electronics and software, then we could test CHICSi together with data acquisition more easily. But in slow ramping experiments information about the beam cycle needs to be stored synchronized with event data and this needs to be tested at the accelerator.

The Combox sends data from the ADCs to a master module which is connected to the high level data acquisition system. We have a data acquisition system called R-Quant that uses ROOT but it was not used in our experiments at TSL. In order to use R-Quant one needs a different master module that sends the events as TCP/IP datagrams directly to an Ethernet network.

In our experiment we used an IndustryPack master module inside a VME-crate which communicated with the TSL high level data acquisition system using a fiber optical cable to the CELSIUS counting room a few stores higher up. In the counting room the fiber optical cable enters another VME-crate that shares memory with the one close to CHICSi.

Chapter 3

Data Acquisition and Conversion

In this chapter I present how the data acquisition (DAQ) is constructed and how the data conversion software for this experiment was developed.

3.1 Data Acquisition

The local SVEDAQ DAQ system was used in this particular experiment [17]. An alternative for us is R-Quant, or to use ROOT directly (ROOT is presented in section 3.2).

SVEDAQ uses VME with a Motorola 68040 processor and the VxWORKS real time operating system (RTOS). SVEDAQ sends the data to a local Ethernet network using TCP/IP (see figure 3.1).

There is an interesting discussion about the pros and cons of VxWORKS in [21], which *e.g.* has been used in the successful PathFinder Mission on Mars. An alternative to VxWORKS is Lynx or Embedded Linux.

In order to modify SVEDAQ you need to know 68k assembly, and one way to learn it is to use a TI-89 graphing calculator [20].

The lower level SVEDAQ system is connected to a Sun workstation running the SVEDAQ user interface under Solaris 9 operating system (see figure 3.2).

I used SVEDAQ as a user when I was collecting the data at the night between Saturday and Sunday on June 14 to 15. It was good that I had previous experience with Sun work stations and Solaris, because I had to learn SVEDAQ the same evening [11].

The user interface is rather sensitive and can crash if you don't do things in the right way. However a crash of the user interface doesn't affect data

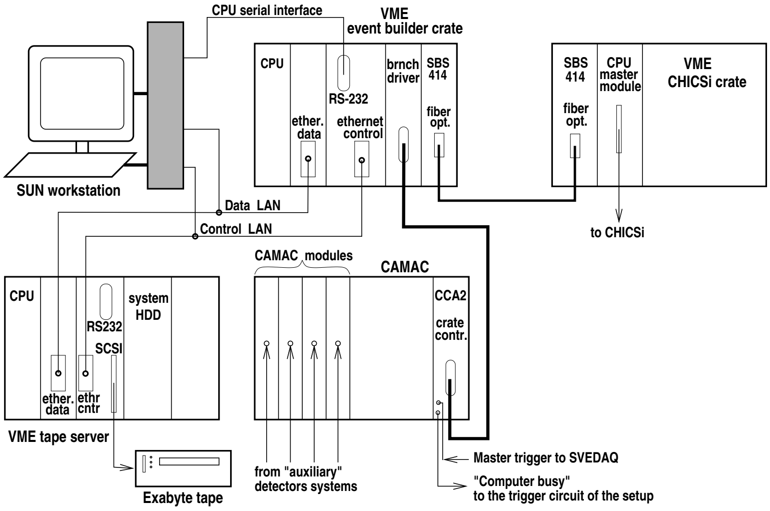


Figure 3.1: *Overview of the SVEDAQ data acquisition system for CHICSi (illustrator: Anatoly Kolozhvari).*

Figure 3.2: *The main SVEDAQ work station in the CELSIUS counting room.*

storage to tape, and there is plenty of time to restart the system.

3.2 Data Conversion

SVEDAQ is the data acquisition system for the CELSIUS storage ring and it ultimately stores its data to tape. These tape files are transfered to a hard drive where they are accessible from Internet. The tape file format is based on on a standard binary file format for nuclear physics: EUROGAM, see [9], [16], and [14]. The EUROGAM format has many features to make it possible to restore data from corrupt files.

We use the ROOT framework from CERN to analyze data. ROOT is primarily a C++ class library, where some of the classes have a graphical user interface. ROOT also has a development environment using interpreted C++ called CINT. Most of the programs I developed are to be compiled using a system called ACLiC, but some are macros run from CINT. It is also possible to compile programs using makefiles. ROOT also has its own native format for storing experimental data. The best way to learn ROOT is to run it at the same time as one is reading [3] on screen. Then one can copy and paste examples of code to see the effect immediately.

I developed a software for converting these EUROGAM files for CHICSi to ROOT format. It was quite difficult to figure out the data format from these documents. Initially I had got an obsolete version of [14], and this explained some contradictions between the documents. I used some code from earlier tape file checking and conversion programs.

I used data from test runs to test my program and since SVEDAQ and CHICSi were being adapted to each other at the same time there were of course errors during the development, and I believe I found some. However all errors where finally corrected.

3.2.1 The SVEDAQ data format for CHICSi

As I was developing the data conversion software I made notes in the form of XML-style diagrams (see figure 3.3 and 3.4), though the data files are not XML-files. In the diagrams a box on the right is built up of the box or boxes on the left. A $*$ before a box indicates that it can be repeated 0 or many times, a + that it can be repeated 1 or many times, and a **?** that it can be repeated 0 or 1 time. **B** means byte, and **b** means bit.

The analog signals from the VLSI chip of a telescope are converted using the ADC-board, and the digital are stored unprocessed (see table 3.1).

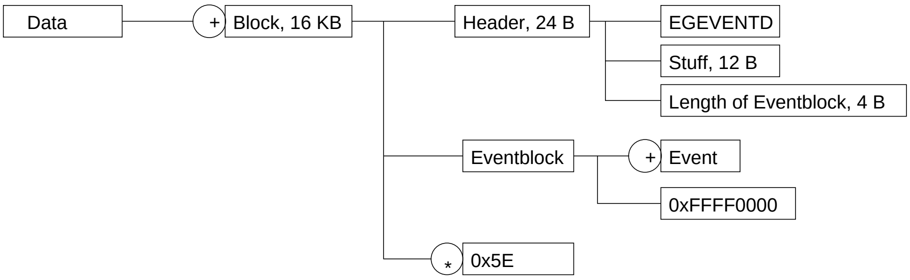


Figure 3.3: *The EUROGAM data format in XML-style. The symbols are explained in the text.*

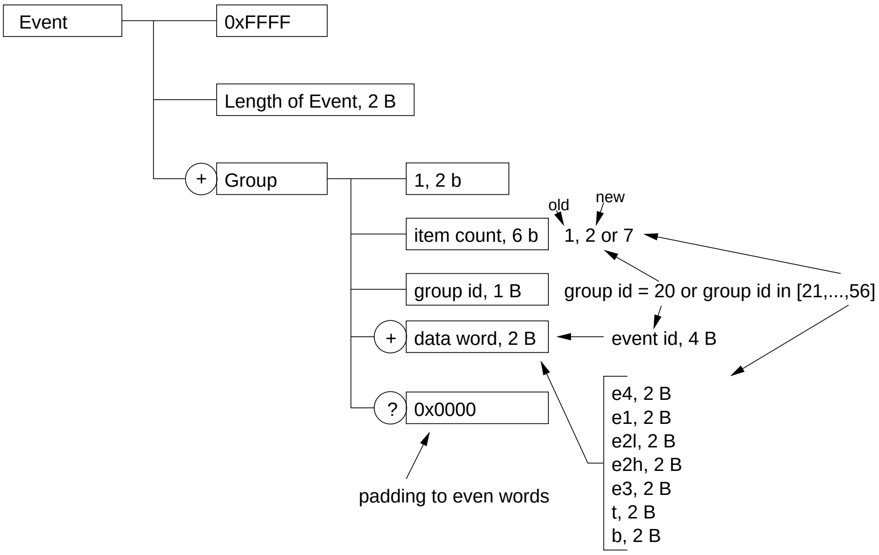


Figure 3.4: *The EUROGAM event format in XML-style. Event is a part of Data in figure 3.3.*

3.2.2 Conversion

A file containing all the raw data was created by simply concatenating the 3 raw data files. This total data file was then converted to ROOT format and used in the further analysis. I also converted each of the data files to check them for errors (see table 3.2). The number of blocks is equal to those we noted from the SVEDAQ user interface. In the first recorded file, RN0104, there were some errors. The errors in data were rare and periodic and probably comes from SVEDAQ, as we noted by studying hex dumps. However, after stopping and restarting the recording, all errors disappeared.

Signal	*Short telescope*	*Long telescope*	*Analog/Digital*
e4	300 μm Si, 3rd	—	A
e1	10/12 μm Si, 1st	10/12 μm Si	A
e2l	300 μm Si, 2nd, low gain	300 μm Si, low gain	A
e2h	high gain	high gain	A
e3	—	6 mm GSO/PD	A
t	time	time	A
b	bits	bits	D

Table 3.1: *Signals in the order they are readout for the short (12 pcs/GMB) and long (6 pcs/GMB) telescopes. e2l and e2h are connected to the same Si detector but with different amplifications. t is duration between telescope and GMB trigger, and b are flags.*

Run	*Time*	*Blocks*	*Block err.*	*Events*	*Event err.*	*Empty ev.*
RN0104	20:35 – 23:27	2859	86	91102	108	18902
RN0105	23:30 – 02:11	2920	0	95932	0	20013
RN0106	02:12 – 10:09	8383	0	275141	0	57208
Total	**20:35 – 10:09**	**14162**	**86**	**462175**	**108**	**96123**

Table 3.2: *Result of the data conversion of the data recorded 2003-06-14 – 2003-06-15.*

The empty events are generated by the CHICSi electronics or low level DAQ software. They are not stored in the resulting ROOT-file.

One could count the number of events at the lowest level, in order to know the number of lost events, if any.

3.2.3 The ConvSVEDAQ2ROOT4CHICSi ROOT Software

In the software below all data, also digital, are stored as floats, and this was done to be compatible with the old files. This program is only tested with the new data files.

```
#include <stdio.h>
// #include <string.h>
#include <TFile.h>
#include <TNtuple.h>

/* --------------------------------------------------------
convSVEDAQ2ROOT4CHICSi.C by Mikael Bonnier 2003-jun-12
```

```
program to check integrity of the SVEDAQ tape data and
convert it to ROOT file. Used some code from check_tape.c
by Anatoly Kolozhvari and read_wach.cpp by Artur Siwek.
The SVEDAQ data format is decribed in "Event-by-Event data
tapes from EUROGAM" (1997) and "EDOC073 Output Data Format,
Edition 2.1" (2001).
Usage: run from ROOT, the data file should be in argument.
The created file will have the provided name appended
with ".root".
Example:
.L convSVEDAQ2ROOT4CHICSi.C++
convSVEDAQ2ROOT4CHICSi("jun2003/data")
   --------------------------------------------------------- */

Bool_t readBlock();
Bool_t readEvent();
void readGroup();
UShort_t searchNextEvent();
UShort_t ADCFix(UInt_t i);
void printBlockErrors(UInt_t blockErrorCounter, UInt_t blockCounter);
void printEventErrors();

const UInt_t HDR_LENGTH = 24;
const UInt_t BUFLEN = 16 * 1024;
const Int_t NO_OF_DETECTORS = 7 * 36;
FILE *inpfile;
UInt_t isbuf[4096];             /* input buffer */
UChar_t *chwp = (UChar_t *) isbuf;
UInt_t idx;
UInt_t idlng;
UInt_t blockErrorCounter;       // Increased for each error
UInt_t blockCounter;
Bool_t blockNoIsSet;
UInt_t blockNo;
UInt_t eventErrorCounter;       // Increased for each error
UInt_t emptyEventCounter;
UInt_t eventCounter;

TFile *froot;
TNtuple *N;
Float_t A[NO_OF_DETECTORS];

void convSVEDAQ2ROOT4CHICSi(const char *fname)
{
   blockErrorCounter = 0;
   blockCounter = 0;
   eventErrorCounter = 0;
   emptyEventCounter = 0;
   eventCounter = 0;
   blockNoIsSet = 0;
   blockNo = 0;

//    ---- open new ROOT file ----------
   TString *s = new TString(fname);
   *s += ".root";
   froot = new TFile(s->Data(), "recreate");

//   -------- define NTUPLE variable  --------
   TString BranchNames;

   BranchNames = "e4_1:e1_1:e2l_1:e2h_1:e3_1:t_1:b_1:";
   BranchNames += "e4_2:e1_2:e2l_2:e2h_2:e3_2:t_2:b_2:";
   BranchNames += "e4_3:e1_3:e2l_3:e2h_3:e3_3:t_3:b_3:";
   BranchNames += "e4_4:e1_4:e2l_4:e2h_4:e3_4:t_4:b_4:";
   BranchNames += "e4_5:e1_5:e2l_5:e2h_5:e3_5:t_5:b_5:";
   BranchNames += "e4_6:e1_6:e2l_6:e2h_6:e3_6:t_6:b_6:";
   BranchNames += "e4_7:e1_7:e2l_7:e2h_7:e3_7:t_7:b_7:";
   BranchNames += "e4_8:e1_8:e2l_8:e2h_8:e3_8:t_8:b_8:";
   BranchNames += "e4_9:e1_9:e2l_9:e2h_9:e3_9:t_9:b_9:";
   BranchNames += "e4_10:e1_10:e2l_10:e2h_10:e3_10:t_10:b_10:";
   BranchNames += "e4_11:e1_11:e2l_11:e2h_11:e3_11:t_11:b_11:";
   BranchNames += "e4_12:e1_12:e2l_12:e2h_12:e3_12:t_12:b_12:";
   BranchNames += "e4_13:e1_13:e2l_13:e2h_13:e3_13:t_13:b_13:";
   BranchNames += "e4_14:e1_14:e2l_14:e2h_14:e3_14:t_14:b_14:";
   BranchNames += "e4_15:e1_15:e2l_15:e2h_15:e3_15:t_15:b_15:";
   BranchNames += "e4_16:e1_16:e2l_16:e2h_16:e3_16:t_16:b_16:";
   BranchNames += "e4_17:e1_17:e2l_17:e2h_17:e3_17:t_17:b_17:";
   BranchNames += "e4_18:e1_18:e2l_18:e2h_18:e3_18:t_18:b_18:";
   BranchNames += "e4_19:e1_19:e2l_19:e2h_19:e3_19:t_19:b_19:";
   BranchNames += "e4_20:e1_20:e2l_20:e2h_20:e3_20:t_20:b_20:";
   BranchNames += "e4_21:e1_21:e2l_21:e2h_21:e3_21:t_21:b_21:";
   BranchNames += "e4_22:e1_22:e2l_22:e2h_22:e3_22:t_22:b_22:";
   BranchNames += "e4_23:e1_23:e2l_23:e2h_23:e3_23:t_23:b_23:";
   BranchNames += "e4_24:e1_24:e2l_24:e2h_24:e3_24:t_24:b_24:";
   BranchNames += "e4_25:e1_25:e2l_25:e2h_25:e3_25:t_25:b_25:";
   BranchNames += "e4_26:e1_26:e2l_26:e2h_26:e3_26:t_26:b_26:";
   BranchNames += "e4_27:e1_27:e2l_27:e2h_27:e3_27:t_27:b_27:";
   BranchNames += "e4_28:e1_28:e2l_28:e2h_28:e3_28:t_28:b_28:";
   BranchNames += "e4_29:e1_29:e2l_29:e2h_29:e3_29:t_29:b_29:";
   BranchNames += "e4_30:e1_30:e2l_30:e2h_30:e3_30:t_30:b_30:";
   BranchNames += "e4_31:e1_31:e2l_31:e2h_31:e3_31:t_31:b_31:";
   BranchNames += "e4_32:e1_32:e2l_32:e2h_32:e3_32:t_32:b_32:";
   BranchNames += "e4_33:e1_33:e2l_33:e2h_33:e3_33:t_33:b_33:";
   BranchNames += "e4_34:e1_34:e2l_34:e2h_34:e3_34:t_34:b_34:";
   BranchNames += "e4_35:e1_35:e2l_35:e2h_35:e3_35:t_35:b_35:";
   BranchNames += "e4_36:e1_36:e2l_36:e2h_36:e3_36:t_36:b_36:";

   N = new TNtuple("N", "ntuple", BranchNames);

   inpfile = fopen(fname, "r");
   if (inpfile == NULL) {
      printf("File open problem; job terminated\n");
      exit(1);
   }
   while (readBlock()) {
      while (readEvent()) {
      }
   }
   fclose(inpfile);

   froot->Write();
   froot->Close();
}

Bool_t readBlock()
{
   const UInt_t L_OF_EB = 20;
   const UInt_t BLOCK_NO = 8;
   const UInt_t END_OF_BLOCK = 0xFFFF0000;
   ++blockCounter;
   int lread = fread(isbuf, 4, 4096, inpfile); /* read the data buffer */
   if (lread <= 0) {
      printf("EOF\n");
      printBlockErrors(blockErrorCounter, blockCounter);
      printEventErrors();
      return 0;
   }
   if (strncmp((char *) chwp, "EGEVENTD", 8) != 0) {
      printf("MAGIC:%8s : EGEVENTD???\n", chwp);
      ++blockErrorCounter;
   }
   if (lread < 6) {
      printf("Too short input buffer lng=%d 4-byte words\n", lread);
      printBlockErrors(blockErrorCounter, blockCounter);
      printEventErrors();
      return 0;
   }
   UInt_t iblcnt = chwp[BLOCK_NO] << 3 * 8 | chwp[BLOCK_NO + 1] << 2 * 8
| chwp[BLOCK_NO + 2] << 8 | chwp[BLOCK_NO + 3]; // Due to Intel byte order
   /* it should be id of the 16K block */
   if (!blockNoIsSet) {
      blockNoIsSet = 1;
      blockNo = iblcnt;
   } else {
      if (iblcnt != ++blockNo) {
         printf("Missing blocks: %u-%u\n", blockNo + 1, iblcnt - 1);
         ++blockErrorCounter;
         blockNo = iblcnt;
      }
   }
   idlng = chwp[L_OF_EB] << 3 * 8 | chwp[L_OF_EB + 1] << 2 * 8
       | chwp[L_OF_EB + 2] << 8 | chwp[L_OF_EB + 3];
   /* data length in byte beginning from isbuf[6] */

   if (idlng + HDR_LENGTH >= BUFLEN) {
      printf("Eventblock too long in block %u (#%u), length=%u bytes\n",
             iblcnt, blockCounter, idlng);
      ++blockErrorCounter;
   } else {
      idx = HDR_LENGTH + idlng - sizeof(UInt_t);
      UInt_t eofb = chwp[idx] << 3 * 8 | chwp[idx + 1] << 2 * 8
          | chwp[idx + 2] << 8 | chwp[idx + 3];
      idx = HDR_LENGTH;
      if (eofb != END_OF_BLOCK) {
         printf("No END_OF_BLOCK token in block %u (#%u)\n", iblcnt,
                blockCounter);
         ++blockErrorCounter;
      }
   }
   return 1;
}

Bool_t readEvent()
{
   const UShort_t START_OF_EVENT = 0xFFFF;
   if ((idx + sizeof(UInt_t) >= BUFLEN) || (idx >= idlng + HDR_LENGTH)) {
      return 0;                  // end of block
   }
   ++eventCounter;
   if (eventCounter % 100 == 0) {
      printf(".");
   }
   UShort_t eventFrame = chwp[idx] << 8 | chwp[idx + 1];
   if (eventFrame != START_OF_EVENT) {
      printf("No START_OF_EVENT token in event #%u\n", eventCounter);
      ++eventErrorCounter;
      return 0;
   }
   UShort_t lengthOfEvent = chwp[idx + 2] << 8 | chwp[idx + 3];
   UShort_t nextEvent = idx + lengthOfEvent;
   if (nextEvent + sizeof(UInt_t) >= BUFLEN) {
      printf("Event #%u is too long @0o%o, block #%u, length=%u bytes",
             eventCounter, BUFLEN * (blockCounter - 1) + idx, blockCounter,
             lengthOfEvent);
      ++eventErrorCounter;
      lengthOfEvent = searchNextEvent();
      printf(" (%u bytes)\n", lengthOfEvent);
      if (lengthOfEvent == 0) { // START_OF_EVENT not found
         return 0;
      }
      nextEvent += lengthOfEvent;
   }
   eventFrame = chwp[nextEvent] << 8 | chwp[nextEvent + 1];
   if (eventFrame != START_OF_EVENT) {
      printf("No END_OF_EVENT token in event #%u\n", eventCounter);
      ++eventErrorCounter;
   }
   idx += 2 * sizeof(UShort_t);
   if (idx >= nextEvent) {
      ++emptyEventCounter;
      return 1;
   }
   for (int i = 0; i < NO_OF_DETECTORS; ++i) {
      A[i] = 0.0;
   }
   while (idx < nextEvent) {
      readGroup();
   }
   N->Fill(A);
   return 1;
}

void readGroup()
{
   UShort_t groupId = 0;
   UShort_t itemCount = 0;
   static UInt_t eventId;
   UChar_t itemType = (chwp[idx] >> 6) & 0x03;
   switch (itemType) {
   case 0x00:                  // Simple Data Word Item
      idx += 2 * sizeof(UShort_t);
      return;
      break;
   case 0x01:                  // Group Data Item
      itemCount = chwp[idx] & 0x3F;
      idx += sizeof(UChar_t);
      groupId = chwp[idx];      // Group id
      idx += sizeof(UChar_t);
      break;
   case 0x10:                  // Extended Group Data Item (not used)
      itemCount = chwp[idx] & 0x3F << 8 | chwp[idx + 1];
      idx += sizeof(UShort_t);
      groupId = chwp[idx] << 8 | chwp[idx + 1]; // Group id
      idx += sizeof(UShort_t);
      break;
   }
   if (groupId == 20) {
      switch (itemCount) {
```

```
      case 1:
         eventId = chwp[idx] << 8 | chwp[idx + 1];
         break;
      case 2:
         eventId = chwp[idx] << 3 * 8 | chwp[idx + 1] << 2 * 8
             | chwp[idx + 2] << 8 | chwp[idx + 3];
         break;
      }
   } else if (21 <= groupId && groupId <= 56) {
      UShort_t e4 = ADCFix(idx); // 6k GSO
      UShort_t e1 = ADCFix(idx + sizeof(UShort_t)); // 10 Si
      UShort_t e2l = ADCFix(idx + 2 * sizeof(UShort_t)); // 300 Si low gain
      UShort_t e2h = ADCFix(idx + 3 * sizeof(UShort_t)); //        high gain
      UShort_t e3 = ADCFix(idx + 4 * sizeof(UShort_t)); // 300 Si
      UShort_t t = ADCFix(idx + 5 * sizeof(UShort_t)); // time
      UShort_t b = chwp[idx + 6 * sizeof(UShort_t)] << 8
         | chwp[idx + 6 * sizeof(UShort_t) + 1]; // bits or flags
      // printf("eventId=%u,groupId=%u,e4=%u,e1=%u,e2l=%u,e2h=%u,e3=%u,"
      // "t=%u,b=0x%X\n",
      //
      UShort_t telescope = groupId - 21;
      A[7 * (telescope)] = e4;
      A[7 * (telescope) + 1] = e1;
      A[7 * (telescope) + 2] = e2l;
      A[7 * (telescope) + 3] = e2h;
      A[7 * (telescope) + 4] = e3;
      A[7 * (telescope) + 5] = t;
      A[7 * (telescope) + 6] = b;
      // A[7*(telescope)+7] = eventId;
   } else {
      // Not CHICSi event
   }
   idx += itemCount * sizeof(UShort_t);
}

UShort_t ADCFix(UInt_t i)
{
   // Fix for Maxim MAX120 ADC
   UShort_t iw1 = chwp[i] << 8 | chwp[i + 1];
   iw1 += 0x0800;
   iw1 &= 0x0FFF;
   return iw1;
}

UShort_t searchNextEvent()
{
   UChar_t *pNextEvent;
   UShort_t nextEvent = idx + 2;
   for (;;) {
      pNextEvent = (UChar_t *) memchr(&chwp[nextEvent], '\xFF',
                                      BUFLEN - nextEvent);
      if (pNextEvent == NULL) {
         return 0;
      }
      nextEvent = pNextEvent - chwp;
      if ((char) chwp[nextEvent + 1] == '\xFF') {
         break;
      } else {
         ++nextEvent;
      }
   }
   return nextEvent - idx;      // lengthOfEvent
```

```
}

void printBlockErrors(UInt_t blockErrorCounter, UInt_t blockCounter)
{
   printf("%u block errors in %u blocks\n", blockErrorCounter,
          blockCounter-1);
}

void printEventErrors()
{
   printf("%u event errors and %u empty events in %u events\n",
          eventErrorCounter, emptyEventCounter, eventCounter);
}
```

Chapter 4

Conversion of Detector Signal to Particle Energy

In this chapter I explain how to calibrate the detector signals to measure particle energy.

4.1 Energy-loss and Energy-range Relation

An ion is stopped in the detector mostly due to interactions with the electrons in the material. The Si detectors are operated in full depletion mode. The mean rate of energy loss (or stopping power) of ions is given by the Bethe-Bloch equation in [12],

$$-\frac{dE}{dx} = Kz^2 \frac{Z}{A} \frac{1}{\beta^2} \left[\frac{1}{2} \ln \frac{2m_e c^2 \beta^2 \gamma^2 T_{max}}{I^2} - \beta^2 - \frac{\delta}{2} \right] . \tag{4.1}$$

The stopping and range of various radiation types is also explained in [13].

If we consider the first two Si-detectors in a telescope we can have 3 situations when a particle enters:

1. it stops in the 1st detector, leaving all the energy there
2. it stops in the 2nd detector, leaving some energy in the 1st and the rest in the 2nd
3. it goes through both detectors, leaving some energy in the 1st, some in the 2nd, and the rest when it collides with other things.

Since these phenomena are quantum mechanical the discussion below concerns average values.

If each particle is plotted in a coordinate system, where the energy deposited in the 1st detector (ΔE) is the ordinate (y-coordinate), and the energy deposited in the 2nd detector (E) is the abscissa (x-coordinate), we get a curve for each particle type. This is because the measured energy varies continuously, while the dependence on charge and mass varies discretely. Thus we get different curves for different ions.

In the 1st situation the abscissa would represent 0 energy for all types of particles.

In the 2nd situation we would get a downward sloping curve since the energy deposited decreases in the 1st detector and increases in the 2nd as the energy of the incoming particle increases. This is because a fast moving particle interacts less with matter (see equation 4.1).

In the 3rd situation we would start from the position where the last curve ended. If we now add some energy to the incoming particle, less energy will be deposited in both the 1st and 2nd detector, and it will proceed through both.

The point in the coordinate system where both curves for the 2nd and 3rd situation intersect is called the back-bending point. The measurements are not in energy units, but rather in some arbitrary unit called channel numbers. For our Si-detectors the channel number should depend linearly on the deposited energy [8], i.e. the response function is linear.

If we can connect certain points on the curve with known energies then we can calibrate the detector. We know that particles of a certain type barely entering the 2nd detector will be at the upper left corner of the curve. And particles almost or barely leaving the 2nd detector will be at the back-bending point. We know how far the particles have traveled through Si to reach these points.

4.2 The TRIM98 Program Package

Instead of using equation 4.1 I used the program SR (Stopping and Range) in the TRIM98 package to compute the range in Si as a function of energy for various isotopes of Be.

In the MS-DOS TRIM98 package there is a more advanced program called TRIM (Transport and Range of Ions in Matter) that uses simulations to determine particle statistics through complex detectors. This could be used for our telescopes with Ni-Si-Si-Si (short type) or Ni-Si-Si-GSO-PD (long type), it is however more difficult to get results from this automatically for many energies of the incoming particle. The method TRIM uses is quite interesting: it's like 3D quantum billiard, and source code in BASIC is included in the TRIM98 package. The algorithms are explained in [26]. I used DOSEMU on Linux Red Hat 9 to run the programs from the TRIM98 package.

4.3 Energy-range Results

I used a special compound.dat file with Si data installed. In SR I entered the Be isotope weights from [10]. For the results see table 4.1.

Ion Energy (MeV)	*Range* ^{7}Be (μm)	*Range* ^{9}Be (μm)	*Range* ^{10}Be (μm)
65	277.33	234.74	219.33
70	314.20	265.42	247.74
80	393.70	331.63	309.04
90	480.89	404.28	376.25
100	575.90	483.00	449.23
110	678.62	567.67	527.65
120	788.96	658.46	611.41

Table 4.1: *Range of ^{7}Be, ^{9}Be, and ^{10}Be in Si as a function of particle energy. The table was computed by SR, a program in the TRIM98 package [25].*

Then a 2nd-degree polynomial of the form,

$$y = ax^2 + bx + c, \qquad (4.2)$$

was fitted to the data for each isotope using the least squares method MATLAB macro in section 4.5. The coefficients are displayed in table 4.2.

A for Be	*a*	*b*	*c*
7	0.038627	2.156887	-26.063705
9	0.030722	2.023704	-26.719896
10	0.028285	1.901100	-23.883049

Table 4.2: *Coefficients for the fitted polynomial in equation 4.2.*

The graphs in figure 4.1 was drawn using the same MATLAB macro in section 4.5. ^{8}Be disintegrates too quickly into 2 alpha particles and is therefore never measured in our experiments.

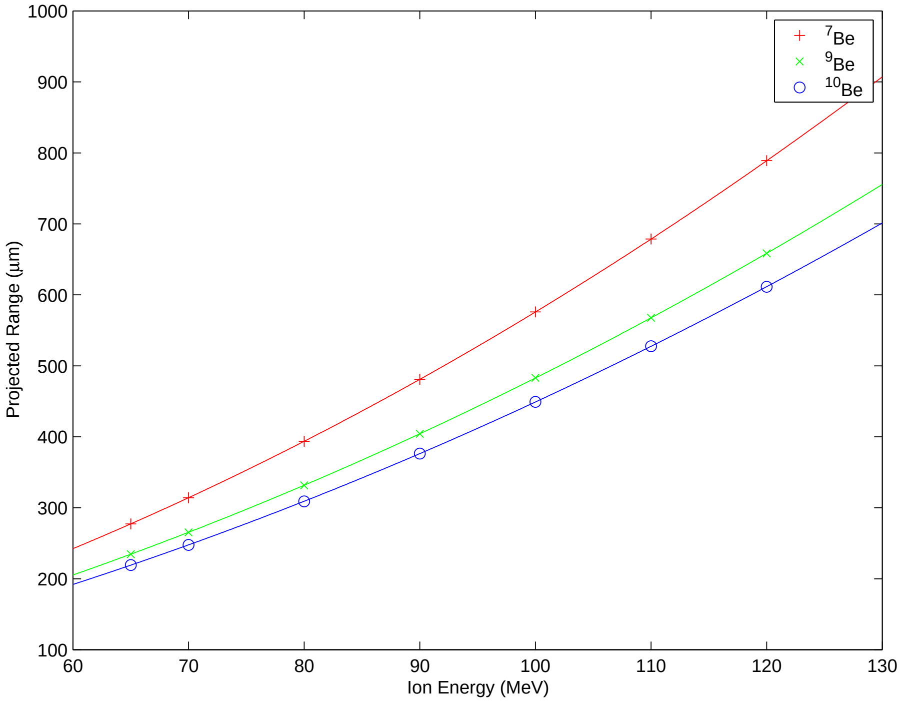


Figure 4.1: *Graph of the range in Si as a function of energy for* ^{7}Be*,* ^{9}Be *and* ^{10}Be*.*

4.4 Energy Calibration

In this case, since $x > 0$, according to the graph in figure 4.1, we have,

$$x = \frac{\sqrt{4ay - 4ac + b^2} - b}{2a}. \tag{4.3}$$

Using this formula we can compute table 4.3. The 1st detector can be 10 or 12 μm, the 2nd 300 μm, and the 3rd 300 μm. In this table we have the energy for the starting and back-bending points for two $\Delta E - E$ detector combinations: 1st versus 2nd, and 2nd versus 3rd.

Range (μm)	*Energy* ^{7}Be (MeV)	*Energy* ^{9}Be (MeV)	*Energy* ^{10}Be (MeV)
10	13.47	14.81	14.64
12	14.09	15.49	15.36
310	69.44	76.81	80.12
312	69.71	77.11	80.43
610	103.40	114.75	119.82
612	103.60	114.97	120.05

Table 4.3: *Ion energy of incoming Be isotopes for different ranges.*

4.5 The ErgRange MATLAB Macro

Here is the least squares 2nd degree polynomial fitting and graphing MATLAB macro.

```
energy = [65 70 80 90 100 110 120]';
rangeBe7 = [277.33 314.20 393.70 480.89 575.90 678.62 788.96]';
rangeBe9 = [234.74 265.42 331.63 404.28 483.00 567.67 658.46]';
rangeBe10 = [219.33 247.74 309.04 376.25 449.23 527.65 611.41]';
plot(energy, rangeBe7, 'r+')
hold on
plot(energy, rangeBe9, 'gx')
plot(energy, rangeBe10, 'bo')
A = [energy.^0 energy.^1 energy.^2];
cBe7 = A\rangeBe7;
cBe9 = A\rangeBe9;
cBe10 = A\rangeBe10;
e = [60:.1:130]';
plot(e,cBe7(1)+cBe7(2)*e+cBe7(3)*e.^2,'r')
plot(e,cBe9(1)+cBe9(2)*e+cBe9(3)*e.^2,'g')
plot(e,cBe10(1)+cBe10(2)*e+cBe10(3)*e.^2,'b')
xlabel('Ion Energy (MeV)')
ylabel('Projected Range (\mum)')
legend('^7Be', '^9Be', '^{10}Be')
hold off
```

```
cBe7
cBe9
cBe10
```

Chapter 5

Particle Identification

In this chapter I compare the data from 2 experiments – in November 2002 and in June 2003, respectively. Both experiments were conducted in the CELSIUS storage ring at The Svedberg Laboratory in Uppsala, Sweden.

In both experiments 2 GMBs with 18 telescopes each was used – of these 12 of each GMB were short. The first produced board is GMB I and the second GMB II. To simplify comparison I have renumbered the telescopes: 19–36 ↦ 1–18. Telescope 1 is the most forward angle, and 18 is the most backward angle, see figure 5.1.

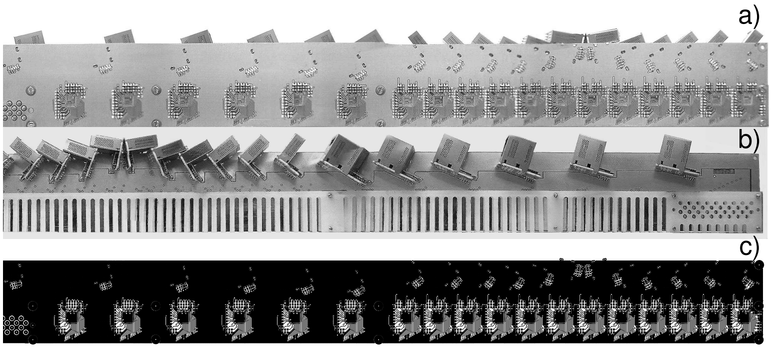


Figure 5.1: *Telescopes on GMB. Photo b has telescope 18 to the left and 1 to the right. 7 is the most forward angle of the short telescopes. (From Paper III in [5].)*

A confusion arises in the numbering since the readout positions of the GMBs were switched between the two experiments, and this affects the numbering in softwares. However, in the figure texts and tables I have compensated for this.

A difference between the 1st and the 2nd experiment is that in the 1st the thresholds for the 2nd Si detector is set too high for all the telescopes, therefore a few[1] events are detected below that level (channel number 1000). Another difference is that in the 2nd experiment some disturbances have caused the details to be smeared out (see section 2.4.2). That is why I will use data from the 1st experiment in the further analysis.

The 2D-histograms were created using the ROOT software called PlotCHICSi in section 5.3 p. 40. The program generates histograms for all telescopes but in this report I only give some interesting examples. The coloring is made so that 0 events is white, 1 is violet and 20 and above is red, and the colors in between are spectral colors with a unique color for each number of events.

5.1 Particle Identification in 2nd * 3rd Detector

The 2nd and 3rd detectors of the short telescopes in the 2 GMBs of the CHICSi detector system are compared.

5.1.1 GMB I Telescopes

Here are 2D-histograms of the data for GMB I. On the x-axis is energy in 3rd detector and on the y-axis is energy in the 2nd.

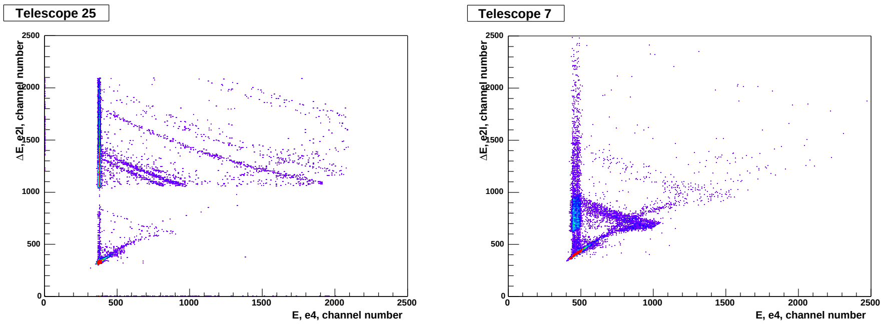


Figure 5.2: *A forward angle example. Telescope I-7, November 2002 (left), June 2003 (right).*

[1] A few events are observed below the threshold because of false triggers (from another telescope).

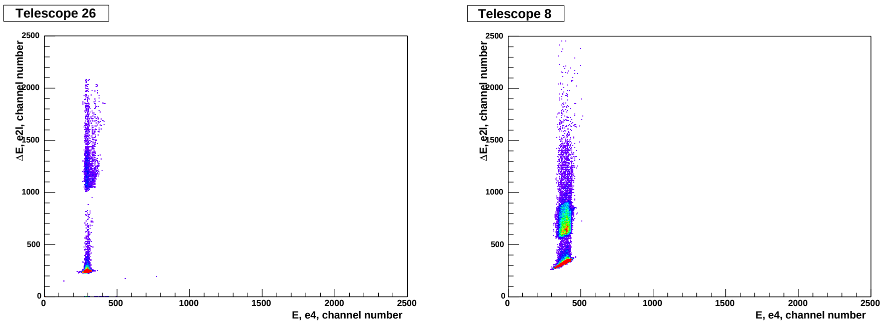


Figure 5.3: *An example with a broken chip. Telescope I-8, November 2002 (left), June 2003 (right).*

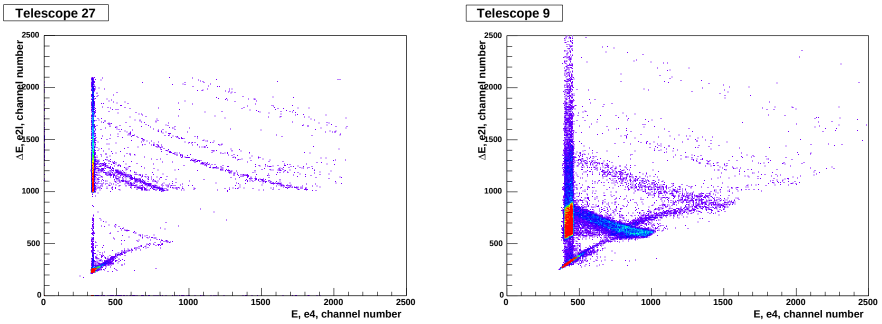


Figure 5.4: *Telescope I-9, November 2002 (left), June 2003 (right).*

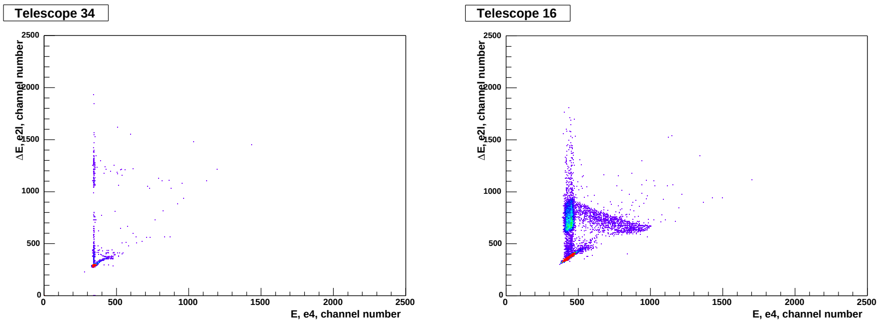


Figure 5.5: *A backward angle example. Telescope I-16, November 2002 (left), June 2003 (right).*

5.1.2 GMB II Telescopes

Here are 2D-histograms of the data for GMB II. On the x-axis is energy in 3rd detector and on the y-axis is energy in the 2nd.

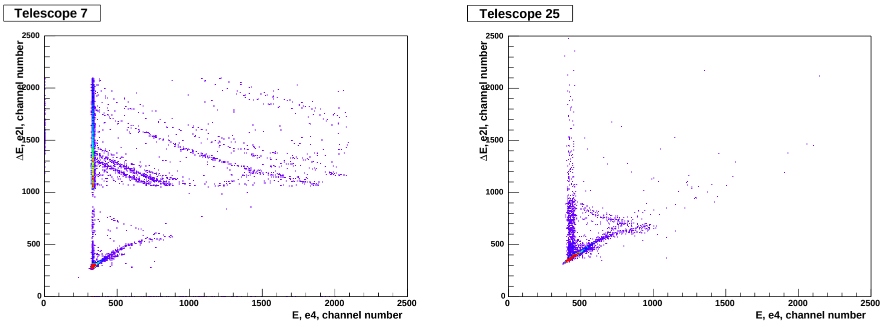


Figure 5.6: *A forward angle example. Telescope II-7, November 2002 (left), June 2003 (right).*

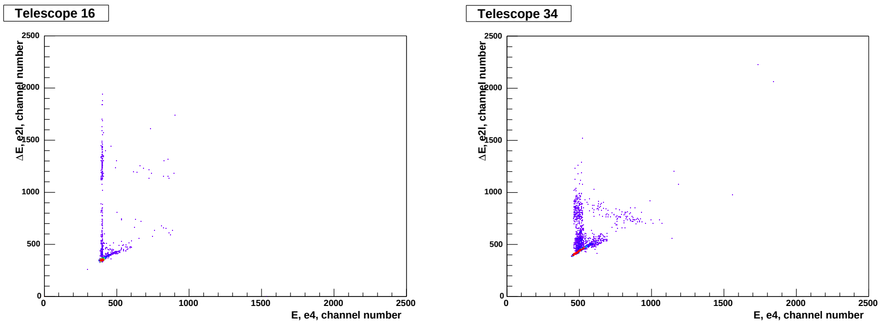


Figure 5.7: *A backward angle example. Telescope II-16, November 2002 (left), June 2003 (right).*

5.2 Particle Identification in 1st * 2nd Detector

The 1st and 2nd detectors of all the telescopes in the 2 GMBs of the CHICSi detector system are compared.

5.2.1 GMB I Telescopes

Here are 2D-histograms of the data for GMB I. On the x-axis is energy in the 2nd detector and on the y-axis is energy in the 1st.

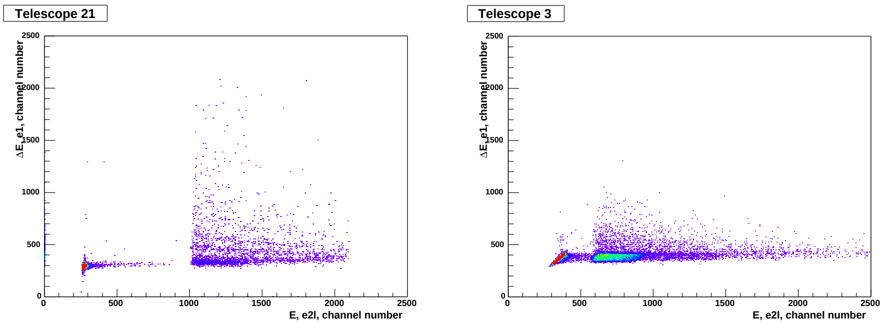


Figure 5.8: *Telescope I-3, November 2002 (left), June 2003 (right).*

5.2.2 GMB II Telescopes

Here are 2D-histograms of the data for GMB II. On the x-axis is energy in the 2nd detector and on the y-axis is energy in the 1st.

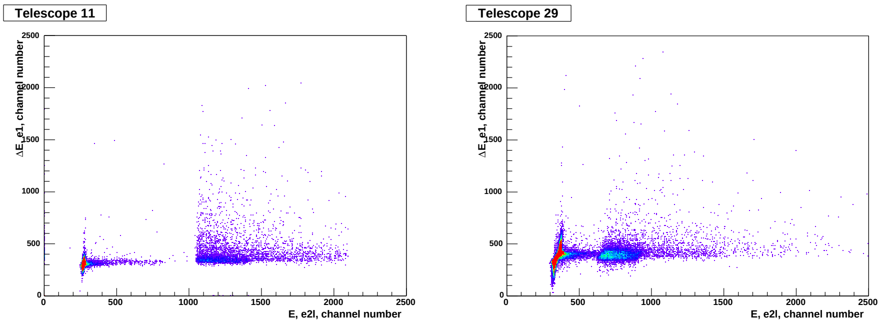


Figure 5.9: *Telescope II-11, November 2002 (left), June 2003 (right).*

5.3 The PlotCHICSi ROOT Software

Here is the program to plot histograms over all the telescopes.

```
#include <Riostream.h>
#include <TStyle.h>
#include <TFile.h>
#include <TNtuple.h>
#include <TH2.h>
#include <TCanvas.h>

/* -------------------------------------------------------
plotCHICSi.C by Mikael Bonnier 2003-jul-11
   A program to plot many telescopes. Used some code from
   chicsi.C by Pavel Golubev.

Usage: load this file, run function from ROOT, the filename
should be in argument, the others are optional.
Example:
.L PlotCHICSi.C++
PlotCHICSi("wach01/wach12.root", "e21", "e1", 500)
   ------------------------------------------------------- */

void Prepare(TTree * N, TH2F * h[], float x[], float y[], Int_t tel,
             const char *xdet, const char *ydet, Int_t nbins);
void Plot(TH2F * h[], Int_t tel, const char *xdet, const char *ydet);

void PlotCHICSi(const char *fname, const char *xdet = "e4",
                const char *ydet = "e21", Int_t nbins = 300)
{
   const Int_t BOARD_A_LO = 7;  // 7, 1
   const Int_t BOARD_A_HI = 18; // 18
   const Int_t BOARD_B_LO = 25; // 25, 19
   const Int_t BOARD_B_HI = 36; // 36

   gROOT->Reset();
   gROOT->SetStyle("Plain");
   TFile *f = new TFile(fname); // "data.root" "wach12.root"
   TTree *N = (TTree *) f->Get("N");

   TH2F *h[36];
   float x[36], y[36];

   for (Int_t tel = BOARD_A_HI; tel >= BOARD_A_LO; --tel) {
      Prepare(N, h, x, y, tel, xdet, ydet, nbins);
   }
   for (Int_t tel = BOARD_B_HI; tel >= BOARD_B_LO; --tel) {
      Prepare(N, h, x, y, tel, xdet, ydet, nbins);
   }
   Int_t nentries = (Int_t) N->GetEntries();
   cout << nentries << " events" << endl;
   for (Int_t i = 0; i < nentries; ++i) {
      N->GetEntry(i);
      if (i % 1000 == 0)
         cout << i << endl;
      for (Int_t tel = BOARD_A_HI; tel >= BOARD_A_LO; --tel) {
         h[tel - 1]->Fill(x[tel - 1], y[tel - 1], 1);
      }
      for (Int_t tel = BOARD_B_HI; tel >= BOARD_B_LO; --tel) {
         h[tel - 1]->Fill(x[tel - 1], y[tel - 1], 1);
      }
   }
   cout << endl << "plotting" << endl;
   for (Int_t tel = BOARD_A_HI; tel >= BOARD_A_LO; --tel) {
      Plot(h, tel, xdet, ydet);
   }
   for (Int_t tel = BOARD_B_HI; tel >= BOARD_B_LO; --tel) {
      Plot(h, tel, xdet, ydet);
   }
}

void Prepare(TTree * N, TH2F * h[], float x[], float y[], Int_t tel,
             const char *xdet, const char *ydet, Int_t nbins)
{
   TString s("Tel_");
   s += tel;
   h[tel - 1] = new TH2F(s, s, nbins, 0, 2500, nbins, 0, 2500);
   s = xdet;
   s += "_";
   s += tel;
   N->SetBranchAddress(s, &x[tel - 1]);
   s = ydet;
   s += "_";
   s += tel;
   N->SetBranchAddress(s, &y[tel - 1]);
}

void Plot(TH2F * h[], Int_t tel, const char *xdet, const char *ydet)
{
   static TCanvas *c[36];

   TString s("c");
   s += tel;
   s += xdet;
   s += ydet;
   TString s2("CHICSi/Uppsala "); // "CHICSi/Uppsala 11-2002 "
       // "CHICSi/Uppsala 06-2003 "
   s2 += s;

   c[tel - 1] = new TCanvas(s, s2, 200, 10, 800, 600);

   c[tel - 1]->cd(1);

   h[tel - 1]->SetStats(kFALSE);
   s = "E, ";
   s += xdet;
   s += ", channel number";
   h[tel - 1]->GetXaxis()->SetTitle(s);
   s = "#DeltaE, ";
   s += ydet;
   s += ", channel number";
   h[tel - 1]->GetYaxis()->SetTitle(s);

   h[tel - 1]->GetXaxis()->SetTitleSize(0.035);
   h[tel - 1]->GetXaxis()->SetTitleOffset(0.035);
   h[tel - 1]->GetXaxis()->SetLabelSize(0.03);
   h[tel - 1]->GetYaxis()->SetTitleSize(0.035);
   h[tel - 1]->GetYaxis()->SetTitleOffset(0.035);
   h[tel - 1]->GetYaxis()->SetLabelSize(0.028);
   // h[0]->SetMarkerSize(1.0);
   s = "Telescope ";
   s += tel;
   h[tel - 1]->SetTitle(s);

   Double_t levels[50];
   for (Int_t i = 0; i < 50; ++i)
      levels[i] = i;
   h[tel - 1]->SetContour(50, levels);
   gStyle->SetPalette(1);    // Pretty Palette with a Spectrum Violet->Red
   h[tel - 1]->Draw("COL");
}
```

Chapter 6

Physics Results – The Angular Dependence of Be Emission in ^{20}Ne + ^{40}Ar Reactions at 200A MeV

^{20}Ne and ^{40}Ar nuclei collide and produce other particles. The number of detections of Be isotopes as a function of telescope angle is determined. We have determined the energy interval that corresponds to the particles that stop in the 3rd detector (section 4.4 p. 31).

The data analyzed in this chapter is from the November 2002 experiment.

6.1 Identification of the ^{7}Be Isotope

The events belonging to an isotope is scattered along a curve as explained in section 4.1 p. 27. The method is to count the events in a curve by drawing a polygon and then count the events within that, see figure 6.1. In this figure the events are due to ^{7}Be, and the 2 curves just above this is ^{9}Be and ^{10}Be, respectively [8]. All in all I had to draw 72 polygons by hand.

If you regard the polygon as a series of vectors one starting where the other ends, you can determine if a point is inside the polygon, by taking the vector product between the side and the vector that you get by subtracting the origin of the side vector from the point, and then doing this for all side vectors, and if all product vectors point in the same direction the point is inside the polygon. However, this only works for convex polygons. In my program I use the ROOT function `TCutG::IsInside(x, y)` [4]. It is based on an algorithm developed by R. Nierhaus and it handles any polygon.

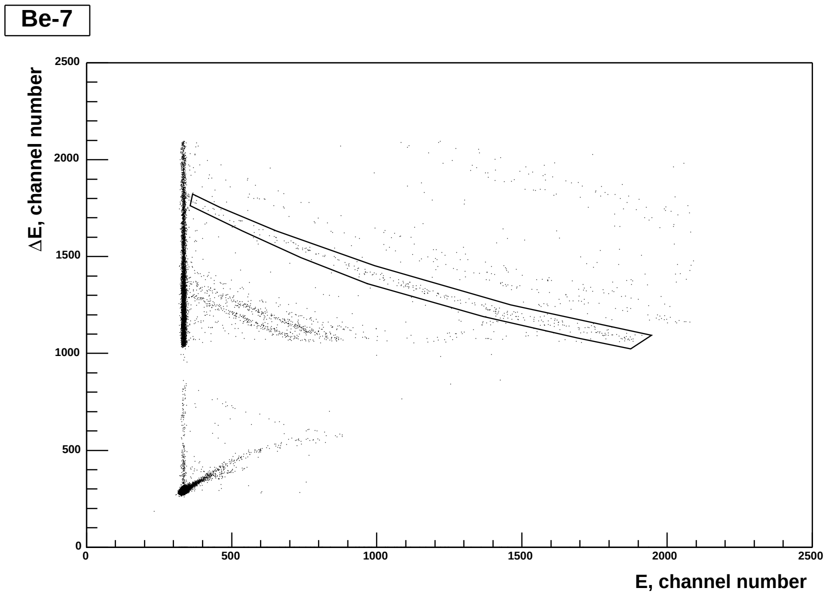


Figure 6.1: *Histogram over telescope II-7 from November 2002 with graphical cut polygon around* ^{7}*Be. There are 305 events inside the polygon.*

A disadvantage with the polygon method is that it is not entirely objective, since it depends on a person drawing the polygons. If there are many events and no empty space between the curves, it can be sensitive to how one draws the polygons.

6.2 Angular Dependence in Be Emission

The number of ^{7}Be, ^{9}Be, and ^{10}Be isotopes for each telescope is shown in table 6.1. Actually it is the ratios of the isotopes that is interesting in order to compare this to theory [8].

In table 6.1 the energy ranges was 69.71–103.60 MeV for ^{7}Be, 77.11–114.97 MeV for ^{9}Be, and 80.43–120.05 MeV for ^{10}Be. I have assumed that the thickness of the 1st detector is 12 μm for all telescopes.

I don't have any results of the standard deviation in data or other error statistics yet. It is possible to get this by dividing the events into groups, and then do the calculations for each group, and then compute standard deviations and error bars from the group results. To make this credibly would require more data.

In the November 2002 experiment the azimuthal position of the GMBs

Telescope	$^{7}Be\ I$	$^{7}Be\ II$	$^{9}Be\ I$	$^{9}Be\ II$	$^{10}Be\ I$	$^{10}Be\ II$
18	1	0	1	0	0	0
17	0	2	0	0	0	1
16	3	1	0	0	0	0
15	5	1	0	1	0	1
14	11	8	0	1	0	0
13	26	26	1	0	1	0
12	—	111	—	31	—	12
11	252	151	78	31	35	21
10	364	264	93	65	47	24
9	393	275	142	105	88	49
8	—	269	—	101	—	71
7	394	305	140	107	79	66

Table 6.1: *Number of detections of ^{7}Be, ^{9}Be, and ^{10}Be in each short telescope for GMB I and GMB II.*

where $\phi = -20°$ for GMB I and $\phi = 90°$ for GMB II as in section 2.4 p. 9. In order to know the angular position and solid angle of each telescope I developed the ROOT macro in section 6.4; the result is in table 6.2. Figure 6.2 summarizes table 6.1 and 6.2. It is interesting to note that the curves for $\phi = 90°$ are lower than $\phi = -20°$ for each isotope of Be. If this is physically significant or if this has to do with the electronics, I can not say, but it would be interesting to investigate further.

Actually one should transform the telescope position into the relativistic center of mass system in order to simplify comparison with theory.

The aim of this work was mostly to prove that it is possible to use CHICSi data and ROOT analysis software to get from raw experimental data to something that is comparable to theory.

6.3 The CutGFileCHICSi ROOT Software

This ROOT code computes the number of events inside each polygon for all telescopes.

```
#include <Riostream.h>
#include <TFile.h>
#include <TNtuple.h>
#include <TCanvas.h>
#include <TCutG.h>

/* -------------------------------------------------------
```

Telescope	θ (°)	$d\Omega$ (msr)
18	160.2	3.18
17	156.3	4.47
16	149.0	7.35
15	141.7	10.67
14	125.0	18.64
13	104.0	26.14
12	76.0	26.14
11	61.2	21.33
10	51.3	16.94
9	42.7	12.78
8	35.9	9.53
7	31.2	7.46
6	27.6	5.94
5	24.4	4.76
4	21.9	3.88
3	20.0	3.24
2	18.2	2.70
1	16.8	2.31

Table 6.2: *The angular position and solid angle for each telescope.*

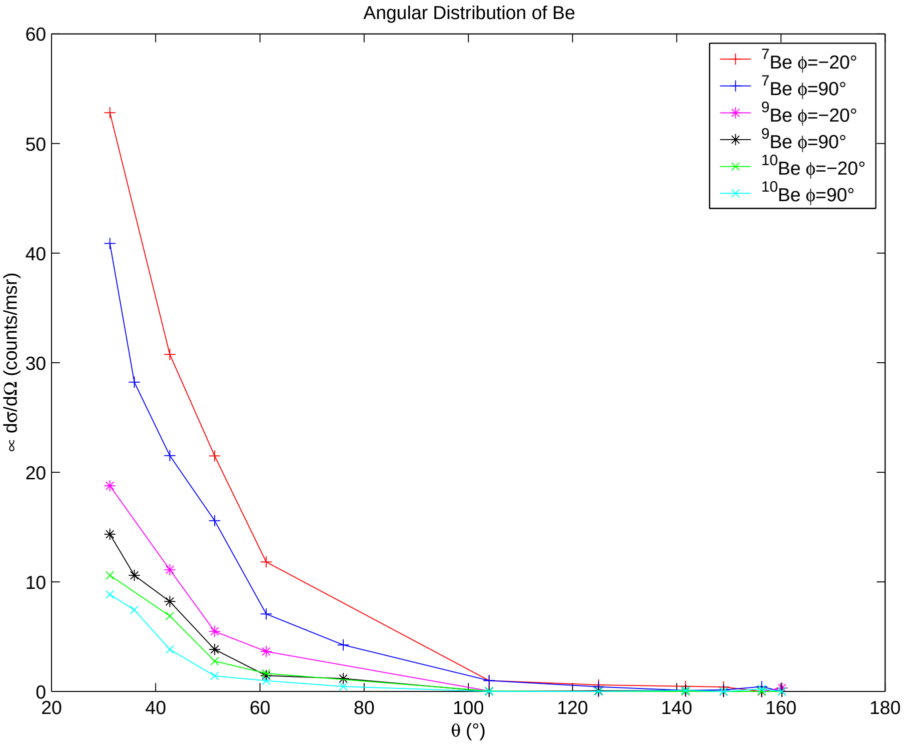


Figure 6.2: *Angular distribution of Be isotopes in the November 2002 experiment. $\phi = -20^\circ$ and $\phi = 90^\circ$ corresponds to GMB I and II, respectively.*

```
CutGFileCHICSi.C by Mikael Bonnier 2003-aug-11
A program to count the events that are inside the
graphical cut polygon. Used together with PlotCHICSi.C.

Usage: In all root histograms over a file and telescopes,
draw a graphical cut polygon, save as canvas.root, load
this file, run function from ROOT, the filename and the
detector names should be in arguments.
Example:
In histogram, chose: Edit/Editor/<...Graphical Cut...>,
click polygon corners, double click on last point
to close the polygon. Then chose File/Save As
canvas.root. Do this for all histograms.
.L CutGFileCHICSi.C++
CutGFileCHICSi("wach01/wach12.root", "e4", "e2l")
   --------------------------------------------------------- */

Int_t CountEventsInTele(const char *fname, Int_t tel,
                        const char *xdet, const char *ydet);
void PrepareCut(TTree * N, float x[], float y[], Int_t tel,
                const char *xdet, const char *ydet);

void CutGFileCHICSi(const char *fname, const char *xdet =
                    "e4", const char *ydet = "e2l")
{
   const Int_t BOARD_A_LO = 7;  // 7, 1
   const Int_t BOARD_A_HI = 18; // 18
   const Int_t BOARD_B_LO = 25; // 25, 19
   const Int_t BOARD_B_HI = 36; // 36

   Int_t eventsInTele[36];

   for (Int_t tel = BOARD_A_HI; tel >= BOARD_A_LO; --tel) {
      eventsInTele[tel - 1] = CountEventsInTele(fname, tel, xdet, ydet);
   }
   for (Int_t tel = BOARD_B_HI; tel >= BOARD_B_LO; --tel) {
      eventsInTele[tel - 1] = CountEventsInTele(fname, tel, xdet, ydet);
   }
   cout << "Telescope" << '\t' << "No of detections" << '\a' << endl;
   for (Int_t tel = BOARD_A_HI; tel >= BOARD_A_LO; --tel) {
      cout << tel << '\t' << eventsInTele[tel - 1] << endl;
   }
   for (Int_t tel = BOARD_B_HI; tel >= BOARD_B_LO; --tel) {
      cout << tel << '\t' << eventsInTele[tel - 1] << endl;
   }
   cout << "\\emph{Telescope} & \\emph{Events, GMB~I} & "
       << "\\emph{Events, GMB~II}\\\\ \\hline" << endl;
   for (Int_t tel = BOARD_B_HI; tel >= BOARD_B_LO; --tel) { // For wash12.root
      Int_t gmbi = eventsInTele[tel - 1];
      Int_t gmbii = eventsInTele[tel - 1 - 18];
      cout << (tel-18) << " & " << gmbi << " & " << gmbii << " \\\\"
         << endl;
   }
   cout << "\\hline" << endl;
}

Int_t CountEventsInTele(const char *fname, Int_t tel,
                        const char *xdet, const char *ydet)
{
   float x[1], y[1];
   gROOT->Reset();
   TFile *f = new TFile(fname); // e.g. "data.root" or "wach12.root"
   TTree *N = (TTree *) f->Get("N");

   Int_t eventsInside = 0;
   TString s("c");
   s += tel;
   s += xdet;
   s += ydet;
   TString s2(s + ".root");
   TFile *fc = new TFile(s2.Data());
   cout << s2 << ' ';
   TPad *pad = (TPad *) fc->Get(s.Data());
   cout << s << endl;
   pad->cd();
   TCutG *cutG = (TCutG *) pad->GetPrimitive("CUTG");
   PrepareCut(N, x, y, tel, xdet, ydet);
   Int_t nentries = (Int_t) N->GetEntries();
   cout << nentries << " events" << endl;
   for (Int_t i = 0; i < nentries; ++i) {
      N->GetEntry(i);
      if (cutG->IsInside(x[0], y[0]))
         ++eventsInside;
      if (i % 1000 == 0)
         cout << tel << ' ' << i << ' ' << eventsInside << endl;
   }
   cout << endl << eventsInside
       << " events inside the graphical cut polygon for telescope "
       << tel << '.' << endl;
   // Alternative faster method, but doesn't work.
   /*
      eventsInside = N->Draw("e4_7:e21_7", "cutg2", "goff");
      cout << endl << eventsInside << " events inside the graphical cut polygon."
         << endl;
    */
   /* Results in strange errors:
    *ERROR 26 :
    Unknown name : "Be"

    365966 events inside the graphical cut polygon. [Should be 2 in my test]
    */
   delete cutG;
   delete N;
   pad->Clear();
   pad->Close();
   delete pad;
   fc->Close();
   delete fc;
   f->Close();
   delete f;
   return eventsInside;
}

void PrepareCut(TTree * N, float x[], float y[], Int_t tel,
                const char *xdet, const char *ydet)
{
   TString s;
   s = xdet;
   s += "_";
   s += tel;
   N->SetBranchAddress(s, &x[0]);
   s = ydet;
   s += "_";
   s += tel;
   N->SetBranchAddress(s, &y[0]);
}
```

6.4 The Angle ROOT Macro

This ROOT macro computes the angular position and solid angle for each telescope. This macro could be developed using MATLAB instead.

```
{
   Double_t meas[] = {199, 183, 165, 149, 132, 115, 99, 83, 65, 48,
      33, 15, -15, -42, -76, -100, -137, -167};
   for (Int_t pos = 18; pos >= 1; --pos) {
      Double_t theta = TMath::Nint(10.0 * TMath::ATan2(60.0, meas[pos - 1])
                                   * 180.0 / TMath::Pi()) / 10.0;
      Double_t omega = TMath::Nint(100.0 * 1000.0 * (10.0 * 10.0)
                                   / (60.0 * 60.0 +
                                      meas[pos - 1] * meas[pos -
                                                           1])) / 100.0;
      printf("%3d %8.1f %8.2f\n", pos, theta, omega);
   }
}
```

Chapter 7

Concluding Remarks

I got into this diploma work mostly because I was interested in PIC microcontroller programming. However, it turned out that this work for CHICSi was already made. Data analysis using ROOT is almost as exciting.

Here are some suggestions based on this report and my experiences:

- In order to suppress disturbances we might make shielded circuit boards.
- We could use R-Quant or ROOT for DAQ to simplify development.
- We can use TRIM98 in a more exact way by using TRIM instead of SR.
- We might use function graphs instead of polygons to count the particles.
- We can compute 1D energy histograms for each particle type, by projecting the data in a polygon on the x-axis. However, this requires some preprocessing to get a calibrated energy scale.

I believe some useful software for converting and analyzing CHICSi data has been developed. And we are able to produce data that after analysis with this software can be compared with theory.

Chapter 8

Acknowledgments

I would like to thank these people, helpful in this diploma work, in order of appearance: Martin Karlsson, who I met a few years ago, and who is always helpful, and who introduced me to ROOT-programming and also helped me getting started with Red Hat Linux (before I had mostly used Linux as a server OS). Lars Carlén, who was one of the teachers on the excellent course Microcomputer Technology, and who introduced me to doing diploma work with the CHIC team. Richard du Rietz, for giving me many time-saving Linux-tips. Bo Jakobsson, my supervisor, for answering my questions, explaining some of the physics, and good suggestions for improving the text in this report. Pavel Golubev, for introducing me the ROOT, and kindly borrowing me his LaTeX templates, and letting me use some pictures. Vladimir Avdeichikov, for introducing me to TRIM98. Torbjörn Hartman, for showing me around the TSL. Anatoly Kolozhvari, for explaining the EUROGAM format, SVEDAQ, and kindly letting me use an illustration. Tord Peterson, for borrowing me a digital camera that was used for some of the pictures in this report.

I would like to thank the good people on the SSLUG, DF Aktiva, and ROOT-talk mailing lists for help when I needed it.

And thanks to all those other helpful and friendly people which I didn't have space enough to mention.

Appendix A

List of Abbreviations

ACLiC	Automatic Compiler of Libraries for CINT
ADC	Analog to Digital Converter
A MeV	MeV/nucleon = MeV/u
CELSIUS	Celsius
CINT	C++ Interpreter
CHIC	CELSIUS Heavy Ion Collaboration
DAQ	Data Acquisition
ECR	Electron Cyclotron Resonance (the external ion source)
GMB	Grand Mother Board
GSO	Gd_2SiO_5, a scintillation crystal
OS	Operating System
PD	Photo-Diode
PDB	Power Distribution Board
ROOT	Root
RTOS	Real Time Operating System
TCP/IP	Transfer Control Protocol/Internet Protocol
TSL	The Svedberg Laboratory
UHV	Ultrahigh Vacuum (below 10^{-7} Pa)

Bibliography

[1] Isaac Asimov, *Asimov's New Guide to Science,* Penguin (1987).

[2] Mikael O. Bonnier, *Projektrapport för Mikrodatorteknik HT02,* http://www.df.lth.se/˜mikaelb/micro/mbproj.html (2002).

[3] Rene Brun *et al.*, *The ROOT User's Guide,* CERN (2003).

[4] Rene Brun & Fons Rademakers, *The ROOT System Home Page,* http://root.cern.ch/ (2003).

[5] Pavel Golubev, *Development of Detector Systems for Internal and Fixed Target Heavy Ion Physics Experiments,* Thesis for the Ph.D. Degree, Department of Physics, Lund (2003).

[6] A. Ingemarsson (editor), *The Svedberg Laboratory, Progress Reports,* The Svedberg Laboratory (1987–2001).

[7] Bo Jakobsson, *CHICSi at CELSIUS – a New Generation of Nuclear Reaction Experiments,* Nuclear Physics News Int., 9:2, 22 (1999).

[8] Bo Jakobsson, *private communication* (2003).

[9] S. Kossionides, *EUROGAM Project: Event-by-Event Data Formats, Ed. 2.0,* Institute of Nuclear Physics, N.C.S.R. Demokritos (1990).

[10] Kenneth S. Krane, *Introductory Nuclear Physics,* John Wiley & Sons (1988).

[11] Mikko Lahtinen & Johan Nyberg, *SVEDAQ User's Guide,* The Svedberg Laboratory, Uppsala (1997).

[12] Particle Data Group, *Particle Physics Booklet,* LBNL & CERN (2002).

[13] W. R. Leo, *Techniques for Nuclear and Particle Physics Experiments,* Springer-Verlag (1992).

[14] Nuclear Physics Software Group *Output Data Format, Ed. 2.1,* Liverpool University (2001).

[15] John B. Peatman, *Design with PIC Microcontrollers,* Prentice Hall (1998).

[16] Johan Nyberg, *Event-by-Event data tapes from EUROGAM,* The Svedberg Laboratory, Uppsala (1997).

[17] Johan Nyberg, *SVEDAQ – A Short Description,* The Svedberg Laboratory, Uppsala (1997).

[18] *Plan for CA47/preliminary,* The Svedberg Laboratory, Uppsala (June 2003).

[19] Richard du Rietz, *The Analogue-to-Digital Conversion for the Readout of the CHICSi detector,* Department of Physics, Lund (2000).

[20] *Assembly Programming for TI-68k,* http://www.technoplaza.net/tigcc-asm/ (2002).

[21] *Vx Works,* http://c2.com/cgi/wiki?VxWorks (2003).

[22] *The CELSIUS / WASA Facility,* The Svedberg Laboratory, Uppsala (1999).

[23] *The Svedberg Laboratory,* The Svedberg Laboratory (1999).

[24] *TSL Home Page,* http://www4.tsl.uu.se/tsl/tsl/ (2003).

[25] J. F. Ziegler, *SRIM Version 96.xx, Instruction Manual,* http://www.srim.org/ (1996).

[26] J. F. Ziegler, J. P. Biersack & U. Littmark, *The Stopping and Range of Ions in Solids,* http://www.srim.org/ (1996).