Thesis for the degree of Licentiate of Engineering

The DAQ always runs

Performing large scale nuclear physics experiments

Håkan T. Johansson

Department of Fundamental Physics
Chalmers University of Technology
Göteborg, Sweden 2006

The DAQ always runs
Performing large scale nuclear physics experiments
Håkan T. Johansson

ISSN 1653-7521
Department of Fundamental Physics
Chalmers University of Technology
SE-412 96 Göteborg
Sweden
Telephone: +46 31 772 10 00

Chalmers Reproservice
Göteborg, Sweden 2006

The DAQ always runs
Performing large scale nuclear physics experiments
Håkan T. Johansson
Department of Fundamental Physics
Chalmers University of Technology, 2006

Abstract

An important part and a growing field of modern subatomic physics is the study of exotic isotopes using radioactive beams. The experiments started out simple. Along with a better and more detailed understanding, together with the availability of new technology, the experiments are becoming more and more advanced. The increased complexity make the use of software at all levels, from data acquisition to analysis, evermore present. However, it is much more the vast amounts of data collected in each experiment, than the sophistication of the calculations needed, that mandates the use of computers.

This thesis is about how to make difficult things simple again. The main thread throughout is that by acknowledging and understanding the complications, they can be used to overcome themselves. To take full advantage of the setups, it is necessary to make close ties between the data acquisition, slow control and analysis programs - they must be able to act as one. At the same time they must be decoupled enough, that any one part can be replaced without severely affecting the other. This is required in order that, while keeping coherence in the analysis procedures - the continuous improvement of the setups can take place without disrupting ongoing analysis work, also making new developments easily applicable to older data.

Keywords:
Data Acquisition, Analysis, Calibrations, Exotic Nuclei, Radioactive Beams

Contents

Chapter 1

Introduction

The world would be dead without interactions between every constituent. There exist only a few types of fundamental interactions, but the resulting forces make the world go round. A fundamental interaction is a mechanism by which objects interact with each other, and which cannot be explained by another more fundamental interaction.

Traditionally, modern physicists have counted four fundamental interactions: gravitation, electromagnetism, the weak interaction, and the strong interaction. These interactions are sometimes called fundamental forces. Their magnitude and regions where some of them are more pronounced than others vary greatly. Gravitation aggregates space dust into stars and stars into galaxies. The variety of all the things around us, from chemical elements and molecules to wild rocks and human beings, are the result of electromagnetic forces. The strong interaction is responsible for binding protons and neutrons into atomic nuclei. The weak interaction is mediating beta decay, which help nuclei that have large excesses of neutrons or protons to reach a more favourable configuration.

The structure of nuclei in the region where they are only weakly bound by the strong interaction and emit electrons or positrons due to weak interaction is the subject of the described experiment. Due to the weakness of the weak interaction, such decays are much slower than strong or electromagnetic decays. The life time of drip-line nuclei varies between milliseconds and years. The most interesting are the nuclei with very large proton-neutron asymmetry at the very edge of the stability line, which means nuclei with very short life time. Traditional targets cannot be prepared and the only possibility to make experiments is with the beams of radioactive short-lived nuclei.

The structural information is accessible only via interaction between nuclei resulting to noticeable changes in their structure. That is why the knowl-

edge of the mechanism of nuclear reaction is of vital importance in understanding of the interplay between the reaction mechanism and the nuclear structure.

At relativistic energies the reaction mechanism is more defined as compared to lower energies, since the time during which the interacting nuclei are close enough for the strong interaction to be at play is short. The reactions can be explained in terms of sudden knock-out, diffractive dissociation, Coulomb excitation and dissociation [1, 2] by virtual photons [3], and described by e.g. the Glauber model [4].

The present work deals with the operation and analysis of nuclear physics experiments, in particular the study of light halo nuclei in complete kinematics using the ALADiN-LAND setup at GSI. The methods described are also suitable for (and applied to) Giant Dipole Resonance experiments at the same setup. One part is also devoted to the application of some of the methods to an experiment performing complementary studies of halo nuclei at REX-ISOLDE, CERN.

1.1 Organisation of the Thesis

Chapter 1 gives a general introduction, and cheats a bit by also giving some more particular observations at the end. In Chapter 2, the experimental setup is briefly described. Chapters 3 and 4 each starts with descriptions of some problems I encountered with data acquisition and analysis, respectively - followed by suggestions for solutions. Chapter 5 touches the important subject of dead-time. The following Chapter 6 reports on my involvements at REX-ISOLDE, partly re-iterating the previous Chapter, but from a different angle. Thoughts about future challenges and needs can be found in Chapter 7 and Appendix F.

Located towards the end is a Glossary, among other things containing all alphabet-soup abbreviations and acronyms used in the text.

I would particularly like to point out Appendix A and B, showing two important principles for data analysis. Appendix D presents a very useful and powerful tool - a software equivalent of a “Gordian Slash”, known to Computing Science for decades.

This thesis is from time to time very harsh. This is intended. It is supposed to be short and no-nonsense. When necessary, it has been attempted to clarify issues, by making them even more explicit, while removing pointless complaints.

1.2 Exotic nuclei

All these experiments are performed with beams of unstable, radioactive nuclei. These nuclei have so short lifetimes, in the order of milliseconds, that they do not occur naturally. They deserve to be called exotic. This means that the ions have to be produced in the laboratory in the same twinkling of an eye that they are used in the experiment, see Section 2.3.

This, however, also means that they (in their respect as matter) have no practical use. They cannot be collected into a jar and put on exposition in a science centre[1]. So why investigate them? Why does Rice play Texas?[2]

The exotic isotopes do carry information, just by being (existing) a short while. By investigating how and into what they break up when challenged (in the same sense as an egg meeting a brick wall) - in this case a head-on or gracing collision with another nucleus, it can be concluded upon how they are held together (but often only via theoretical arguments). Studying these nuclei give further insight into the characteristics of the nuclear force. The leverage these exotic, unstable nuclei have is their unusual proton-to-neutron ratios compared to stable or almost-stable naturally occurring isotopes, probing a different realm of the nuclear interaction.

1.2.1 Few-body quantum systems

To motivate this, let us look at the strong interaction from a calculational point of view.

The Pauli Exclusion Principle forbids two particles with half-integer spin[3] to be in the same quantum state (to be described by the same quantum numbers). This is the reason why matter has size, and do not collapse into one point.

Atomic Physics

When considering the electron cloud around an atom, one may consider as boundary condition (task description) the very small and charged nucleus, together with the fact that a certain number of electrons are to be placed

[1] Museum might be a better word, but sounds so old; past. This is new stuff!

[2] "... But why, some say, the moon? Why choose this as our goal? And they may well ask why climb the highest mountain. Why, 35 years ago, fly the Atlantic? Why does Rice play Texas? We choose to go to the moon. We choose to go to the moon in this decade and do the other things, not because they are easy, but because they are hard, ..." US President John F. Kennedy, Rice University, September 12, 1962, "The Space Effort" [5]

[3] Matter particles like electrons, neutrons and protons have half-integer spin. Force-mediating particles, like photons have integer spin.

in orbit around this nucleus. An additional requirement is to find the set of orbits with least energy, i.e. most bound. A particle (electron in this case) in a certain orbit has a certain set of quantum numbers. A spin ½-particle (like an electron) has two spin states, up or down, which gives one additional quantum number. With this in mind, two electrons can be placed into each orbit in space. Every second additional electron must be placed in orbit a little further away from the centre[4], and is less bound, since the binding energy essentially is given by how deep in the potential well (created by the nucleus) the electron sits[5].

An atom is held together by the electromagnetic force between the positively charged nucleus and the negatively charged electrons, while the electrons are repelling each other. Usually, a little after adding so many electrons that they completely make up for the charge of the nucleus, the next orbit to fill will not provide any binding at all, the state of that electron would be an unbound continuum state and not be bound to the atom at all. Negative ions, with more electrons than central charge exist, as long as it is more favourable to be close to the atom than being free.

Atomic configurations (the electron orbitals) are relatively easy to calculate to high precision starting from basic principles[6]. At least compared to the structure of nuclei[7]...

Nuclear Physics

The structure of nuclei is at a glance quite similar to that of atoms, although smaller: the length scale is fm instead of nm. The major difference - computational horror - is that there is no central potential attracting the constituents. And there are two kinds of building blocks: neutrons and protons. The protons and neutrons must create their own potential. It's more like a party than a rock concert, there is no star - a solar system without a sun.

The neutrons are unaffected by the electromagnetic force while it at these small distances causes very large repelling forces between the protons. Felt equally by protons and neutrons is the strong force, which at large distances (a few times the nucleon radius) is null and void, at very small ranges (fractions of fm) strongly repulsive and in-between attractive. It is attractive

[4]The orbits are not planetary, but wave-functions, better called orbitals. Also with a wave function an average distance to the nucleus can be defined...

[5]The direct connection between distance and binding energy is not really true, but a good approximation.

[6]E.g. Hartree-Fock calculations.

[7]The 'basic principle' calculations for nuclei (like GFMC) are all using effective forces.

enough to be on the same scale as the EM repulsion between protons (particularly when many nuclei are around, as there are more interaction possibilities).

The strong force is not strong enough to bind matter of pure neutrons[8] or protons - which is harder due to the repulsion. But neutrons and protons are different particles, therefore having yet one more quantum number: isospin. Voila - while still obeying the Pauli principle, they can overlap, occupying similar orbitals. And there is the trick! Protons and neutrons can be spatially overlapping, and suddenly we have more interaction possibilities for the strong force, which is able to provide enough binding to make collections of protons and neutrons bound.

This way, we can also with hand-waving arguments explain why stable isotopes have a rather similar amount of protons and neutrons. Whenever one of them become much in majority, that kind of particles need to occupy a much larger space, where the other kind is not present, and then binding the 'outermost' particles of the dominating kind becomes harder, as the interaction possibilities are more scarce. At a certain point, further nucleons of the cannot be added. Halo nuclei represent a limiting case, where the valence nucleons are just barely bound.

1.2.2 Using Approximations

Quantum mechanics only prescribes a model for an entire object as such, one 'large' wave function in the coordinates of all involved particles at the same time. Such that single particle states in principle could be expected to be useless, i.e. separating (or projecting) the wave-function into a product of single particle wave-functions would yield a product having nothing to do with the original.

Nature, however, seem to most often behave such that single particle factorisations give reasonable results. Or at least when factorising the state into a product of shells, each containing a certain number of particles, occupying similar states. This works extremely well for describing the electrons in an atom, and also very well for describing the nucleus as done by the Shell Model, particularly stable nuclei. For nuclei, there are however examples where the wave function must be seen as built in a different way, e.g. the exited triple-alpha clustered state in ^{12}C [7, 8, 9]. Also, when approaching unbound nuclei, often the shell model will predict a different level ordering than observed. This just means that the requirements needed to use the

[8]This statement is currently being challenged by the claimed observation of the tetraneutron [6]. It is hoped that the S245 experiment will be able to shed some more light upon this issue.

approximation used by the simplified model are no longer (so) fulfilled, but it does not necessarily mark any new effects, and is not unsettling at all for quantum mechanics!

The issue is rather that using the language of the shell model (it's states), it is possible to talk about, compare, see similarities and differences between different nuclei and make predictions, which would be very much more complicated (if at all possible) if one always needed to use the complete wavefunction representation of the entire nucleus directly. It just becomes inconvenient when the nuclear state cannot be reasonably described using the words (occupancy numbers) of the shell model.

1.2.3 Halo Nuclei

Experiments have shown that the drip-line nuclear structure differs very much from the structure of stable nuclei. The neutron-rich nuclei with very low binding energy create a phenomenon which is called neutron halo, see [10] for an extensive overview. A neutron halo is a limiting case of fragmentation when the nucleons inside the nucleus are combined in localised objects - clusters.

To behave as a halo nucleus, the valence nucleon(s) must be very loosely bound, usually less than 1 MeV. Only then, a major part of their wavefunction can be located at large distances from the core, in the classically forbidden region - the halo phenomenon can be seen as the halo nucleon(s) tunnelling into this region, where the strong force does not provide binding - making it a quantum effect.

The strong force has a peculiar effect on nuclei - pairing. The total binding energy is larger (by several MeV) when a nucleus has an even number of protons or neutrons. The effect applies to protons and neutrons independently - giving an extra strong effect for even-even nuclei.

A halo nucleus with Borromean structure has a clustered configuration where none of the binary subsystems (of only two clusters) are bound. This is the case for the two-neutron halo systems ^{6}He, ^{11}Li, and ^{14}Be, where the subsystems ^{5}He, ^{10}Li, and ^{13}Be are all unbound, as is the n-n subsystem.

That a two-neutron halo nucleus (Z, N), where the number of neutrons is even (as above), is very likely to be Borromean can be understood as a consequence of the pairing effect. The necessity of it being just barely bound, implies that the subsystem $(Z, N-1)$ is unbound since it would have an odd number of neutrons, and therefore not benefit from a pairing of the last neutron. A similar effect is seen for the neighbouring isotopes ^{11}Be and ^{12}Be. Using the halo nucleus ^{11}Be as basis, having an odd number of neutrons, creating ^{12}Be by adding one neutron gives so much binding that it does not

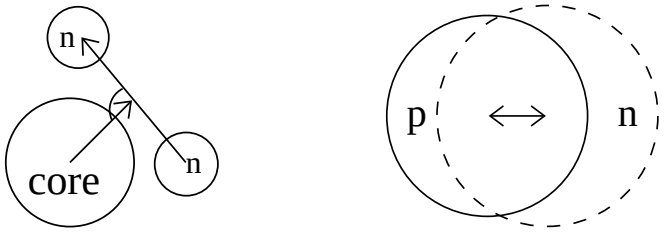

Figure 1.1: Different phenomena investigated with the LAND setup: halo nuclei (left) and giant resonances (right).

qualify as a halo nucleus.

A particularly intriguing approach to further understanding of halo nuclei, and the strong interaction as such, possible with the kinematically complete LAND setup, is the study of e.g. angular correlations between the fragment nucleus and the neutrons after break-up, see Figure 1.1 (left) and [1].

1.2.4 Giant Dipole Resonances

A giant resonance is a collective excitation, where the nucleons can be viewed as having a collective motion. There are several modes of giant resonances. The perhaps most striking one, and easy to depict, is the giant (electrical) dipole resonance, which is viewed as all the protons in the nucleus oscillating against the neutrons, see Figure 1.1 (right).

The kinematically complete ALADiN-LAND setup, with it's emphasis on the detection of neutrons, projectile-like fragments and gammas, is well suited for the investigation of the GDR in stable and neutron rich nuclei. The reactions can schematically be viewed as the isotope of interest being excited by a virtual photon from a high Z reaction target, forming a GDR excited state. The excited nucleus then tries hard to get rid of the surplus energy. As long as the excitation energy is above the particle emission threshold, some nucleons (neutrons in our case) are ejected, largely statistically - like a black body radiator. As the protons has the Coulomb barrier to pass, emitted protons can only be expected if the nucleus is proton rich. After enough energy has been carried away by the escaping nucleons, making the remnant nucleus stable against particle emission, the remaining energy will be expelled by gamma emission.

The interesting quantity in each event is the excitation energy. This would have been the energy of the virtual photon, which we cannot measure[9]. By

[9]Experiments using high energy photons or electrons to do a direct excitation do not

measuring all remnants of the nucleus (fragment, neutrons and gammas) and employing the invariant mass method, the excitation energy of the resonance state can be reconstructed. The giant resonance can then be plotted - the cross section as a function of excitation energy.

1.2.5 Reaction Selection by Hardware and Trigger

Just shooting one nucleus at another one does not automatically make them suffer the reaction type of interest. Most often, nothing happens at all! The target thickness (number of nuclei) is selected such that the probability of multiple interactions is almost negligible, as such events are unanalysable - the problem is that one cannot know which events are affected. A 5% target is usually considered to be thick[10].

The ALADiN-LAND setup is built for investigating peripheral collisions, i.e. reactions where most of the incoming ion remains as a fragment with only a few nucleons lost. In a more central or head-on collision, the incoming ion is likely to disintegrate into many light fragments, not leaving much information at all about the structure of it. Although the incoming ion will be detected also for events where these more violent reaction take place, the outgoing fragments would have too small charge to be detected in the outgoing tracking detectors, and therefore would not satisfy the trigger conditions, thereby evading being recorded, except by a minimum bias trigger.

1.3 Detectors General

The ions in the experiments (outgoing and usually also incoming[11]) need to be detected. In the experiments described in this thesis this is done exclusively by detectors that transform the hits to electrical signals. (We are not involved with (nowadays) exotic detectors using photographic emulsions, bubble chambers, etc. . .)

There are different means to make the passing of an ion produce an electrical signal that can be digitised and recorded by a computer. They usually work by having the charged ion exciting some part of the detector material.

have this problem [11], but they can only investigate stable nuclei, since an appreciable target is needed. This issue is being addressed by the ELISe project, which will investigate exotic nuclei with an precise probe - electrons, in a collider setup.

[10]The probability for an event where something happened to be a multiple interaction event, is about the same as the probability for an event to have a reaction at all.

[11]The incoming ions may only be attempted to be detected if they have so much energy that they can pass through a detector and still have enough energy left to be useful for the experiment afterwards.

This excitation then leads to a small flow of some kind of particles (photons from de-excitation[12], electrons or ions) towards one or several collection regions. Alas, the detection always requires the ion to loose some energy. If the flow is not already consisting of electrons, the collection mechanism also need to do that conversion. In one way or another, an amplification (several orders of magnitude) is also necessary, since the energy deposited by the particle (usually on the order of a few MeV), is so small that it cannot survive the transport in a cable any appreciable distance (a few meters):

- The photons from a **scintillator** are detected and converted into electrons making use of the photo-electric effect in the entrance window of a PM-tube. The knocked-out electrons are accelerated by an electric field towards several stages of dynodes. At each dynode, each incoming electron (on average) knocks several electrons out, creating an avalanche, such that a multiplication of the current takes place, up to a factor 10^6.

  The signal can be directly fed into a QDC, integrating the charge. Common full scale ranges are on the order of a few hundred pC. A pulse 5 ns wide, with a total charge of 100 pC represents a current of 20 mA. The pulse height as measured over a cable impedance of 50 Ω is then 1 V.

- The electron-hole-pairs created by a passing ion (or a photon depositing it's energy) in a **semiconductor** detector (silicon, germanium or diamond) are directly responsible for the current measured between the electrodes. The (reverse) bias voltage applied has the purpose of depleting the entire region of the diode, making the detection volume larger. This also makes it less likely for the created charges to recombine and thereby be lost.

  The signal amplitude from the detector is to small to be handled, such that an additional stage of amplifiers are needed. These must be located close to the detectors, due to the very small signal.

- The signal in a **wire chamber** is created as the passing radiation ionises atoms in the gas of the detector. The separated electrons and ions are accelerated towards the anode and cathode wires respectively by an electric field. When the field is strong enough, then the particles will gain enough energy between collisions such that secondary ionisation takes place, multiplying the current. The major part of the

[12] An exception being the photons from a Čerenkov detector. They are Bremsstrahlung, and not emitted from de-excitation.

current amplification is performed in the vicinity of the read-out wires - by using very thin wires, the electric field strength becomes very large around them, causing a considerable avalanche the last few 100 μm.

For much more information, the reader is referred to [12][13].

1.3.1 Time and Energy (loss)

Each signal from a detector is quantified by it's time of arrival (relative to other signals), t, and amplitude (on an absolute scale). The amplitude usually is proportional to the energy lost by the particle in passing through a detector, Δe. Other quantities, most notably the location of a hit in a detector, x and y, are derived from these during analysis.

The only thing the DAQ need to do is to record these two values for each signal around each accepted event. Simple and painfully straight-forward. So what is the problem - see Section 3.1.

The only other thing being recorded is the values of a few free-running (i.e. not dead-time affected) counters counting the number of raw and accepted triggers. This is done so that cross sections can be determined even though not all events (that actually happen) are being fully recorded, due to them either being minimum bias events (particle comes in, nothing happens in the target) or the DAQ being in dead-time (still processing an earlier event).

1.3.2 Time, Position and Energy (loss)

Taking as example a scintillator paddle (long slab, with read-out at the two short ends, see Figure 1.2(left)).

Based on the assumption of uniform light propagation in a paddle, one may, given the times and amplitudes of the light measured at the both ends calculate the time (T), position (P, measured from the middle of the paddle, c.f. Appendix B) and energy loss (E) of the particle hit in the paddle, see Figure 1.3.

[13] Although the Chapter on digitisation in the "Leo" book is somewhat antiquated, spending much effort on the use of single-channel analysers.

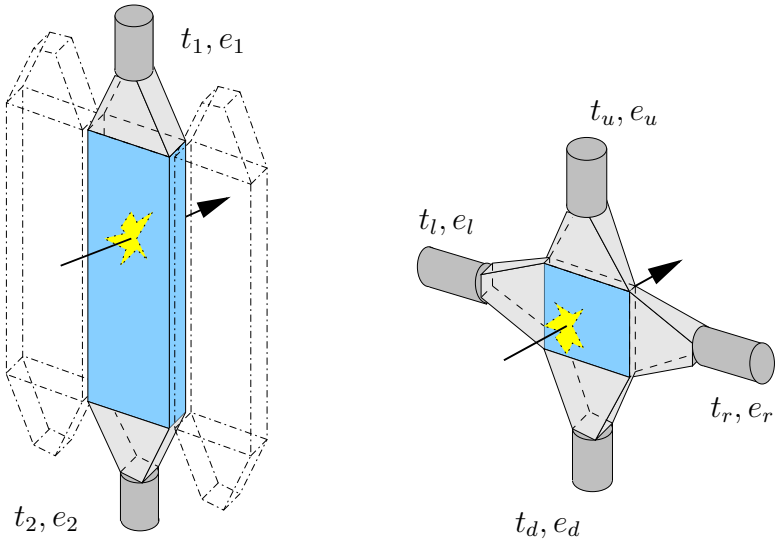

Figure 1.2: Scintillator paddles with PM read-out. To the left a typical TFW paddle. (The LAND paddles are similar, with 10 layers of scintillator, sandwiched with iron converter.) To the right a thin scintillating sheet, used for detecting the incoming beam (POS detector).

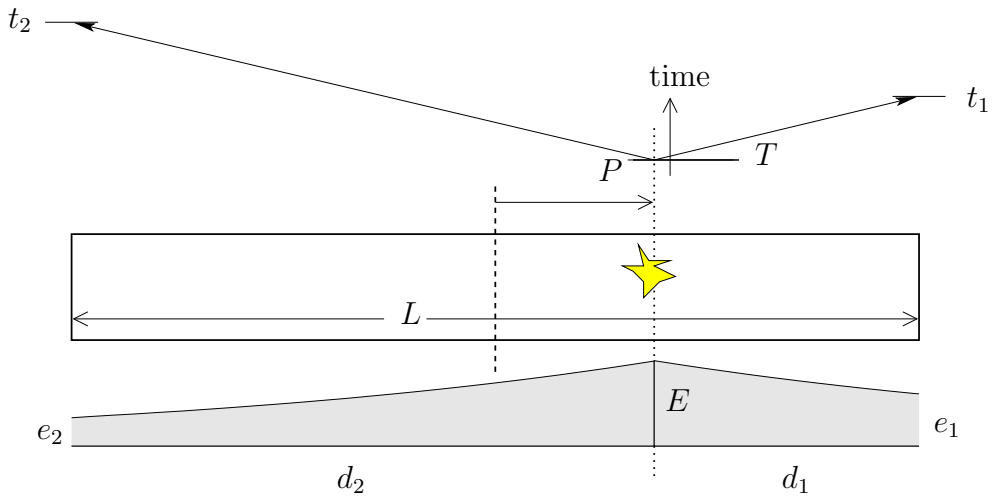

Figure 1.3: Light propagation in a paddle. The time of arrival of the signal (the photons) at the short ends of the paddle depend on the time of the hit and the distance to the hit. The light output also depend on the energy deposited in the paddle, and the attenuation.

Time measurement

With the effective speed of light in the scintillator, v (usually about $0.5c$), the two times measured at the ends of the paddle will be

$$t_1 = T + \frac{1}{v} d_1, \tag{1.1a}$$

$$t_2 = T + \frac{1}{v} d_2. \tag{1.1b}$$

From this, we solve for the time of the hit as an average

$$T = \frac{t_1 + t_2}{2} - \frac{d_1 + d_2}{2v} = \frac{t_1 + t_2}{2} - \frac{L}{2v}. \tag{1.2}$$

The constant length of the paddle, $L = d_1 + d_2$, divided by the effective speed of light, may be folded into the overall time calibration parameters, i.e. just a shift of the time offset for the paddle, leaving us with the more convenient

$$T = \frac{t_1 + t_2}{2}. \tag{1.3}$$

The position is calculated as the difference of the two times, and the paddle length disappear also here from the final formula

$$P = \frac{\left(d_2 - \frac{L}{2}\right) + \left(\frac{L}{2} - d_1\right)}{2} = \frac{d_2 - d_1}{2} = \frac{v(t_1 - T) - v(t_2 - T)}{2} = v\frac{t_1 - t_2}{2}. \tag{1.4}$$

Amplitude measurement

The light attenuation in the paddle can be approximated as an exponential function of the distance between the scintillation location and the PM tube, particularly in cases where the dimension of the paddle along this direction is larger than the other dimensions. When this is not the case, e.g. for the incoming beam detectors (see Figure 1.2, right), quite large abberations may occur, i.e. the formulae below need position dependent corrections, usually done with a second order polynomial in the spatial coordinates.

With the light attenuation length in a paddle λ (for TFW about 1.6 m, for POS around 3 cm), the two recorded energies are

$$e_1 = E \exp\left(-\frac{d_1}{\lambda}\right), \tag{1.5a}$$

$$e_2 = E \exp\left(-\frac{d_2}{\lambda}\right). \tag{1.5b}$$

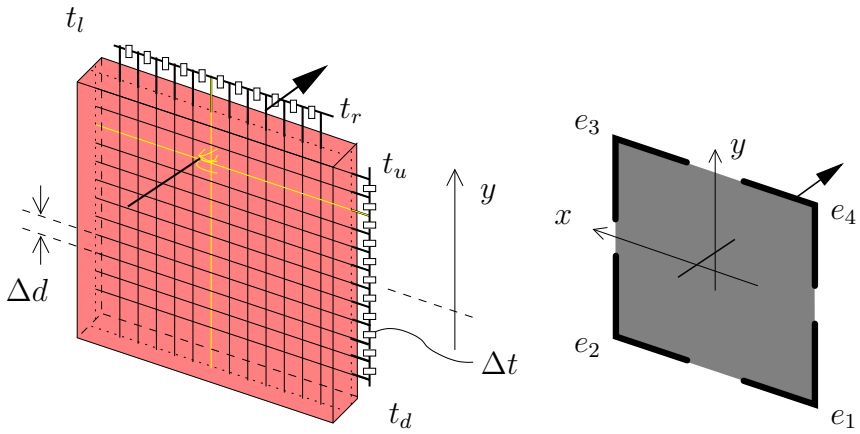

Figure 1.4: Wire chamber with delay line read-out (left). Position sensitive silicon detector, with signal (charge) division between the electrodes in the corners (right).

We solve for

$$E = \sqrt{e_1 e_2 \exp\left(\frac{d_1}{\lambda} + \frac{d_2}{\lambda}\right)} = \exp\frac{L}{2\lambda} \cdot \sqrt{e_1 e_2}, \tag{1.6}$$

$$P = \frac{d_2 - d_1}{2} = \frac{l \ln \frac{E}{e_2} - l \ln \frac{E}{e_1}}{2} = l\frac{\ln e_1 - \ln e_2}{2}. \tag{1.7}$$

Once again, a constant factor, $e^{\frac{L}{2\lambda}}$, may be omitted and this time considered part of the energy gain calibration parameters, giving

$$E = \sqrt{e_1 e_2}. \tag{1.8}$$

Delay line read-out

The read-out of a wire chamber with delay lines (to reduce the number of read-out channels to two per coordinate, x and y, see Figure 1.4(left)) give rise to very similar formulas for the reconstruction of the position from the two times,

$$y = v\frac{t_d - t_u}{2}, \tag{1.9}$$

where the signal propagation speed now is $v = \frac{\Delta d}{\Delta t}$. Δd is the distance between two wires, and Δt is the propagation delay between them.

Silicon detectors

In the particular experiment this thesis is centred around (S245), only two silicon detectors were used. The purpose of the two semiconductor diodes was the determination of the ion charge before and after the target via energy-loss measurements. With a single read-out channel the reconstruction of the total energy deposited is trivial.

For silicon detectors with several read-outs (see Figure 1.4(right)), the deposited energy/charge is shared between the pick-ups (eventually read-out channels) in an additive way. The sharing is usually not affected so much by attenuation depending on the location on the hit. Therefore, for this case, $\ln E$ is of no use.

For such a detector, the deposited energy is then simply

$$E = e_1 + e_2 + e_3 + e_4. \tag{1.10}$$

Assuming a linear dependence of the charge collection on position, the positions are reconstructed as

$$P_x = \frac{E_{left} - E_{right}}{E} = \frac{(e_2 + e_3) - (e_1 + e_4)}{e_1 + e_2 + e_3 + e_4}, \tag{1.11}$$

$$P_y = \frac{E_{up} - E_{down}}{E} = \frac{(e_3 + e_4) - (e_1 + e_2)}{e_1 + e_2 + e_3 + e_4}. \tag{1.12}$$

Just as for the case of amplitudes measured with an exponential relation to the energy loss and position reconstruction using logarithms it is important to here have the energies expressed with a correct absolute zero. For this case it is however also necessary that the four individual energies all have correct gain factors applied before the formulas above are applied.

1.3.3 Tracking the way to Momenta

When the hits in each detector have been reconstructed, they are combined into tracks. The ions, incoming and outgoing, are identified with the characteristics of each track: the measured energy loss in the detectors and the calculated curvature in any magnetic fields passed, and the velocity. The velocity measurement also give the four-momentum of the particles.

At the track level, the detectors are no longer considered on how they function, but what variables $x, y, t, \Delta e$ they measure in the lab system.

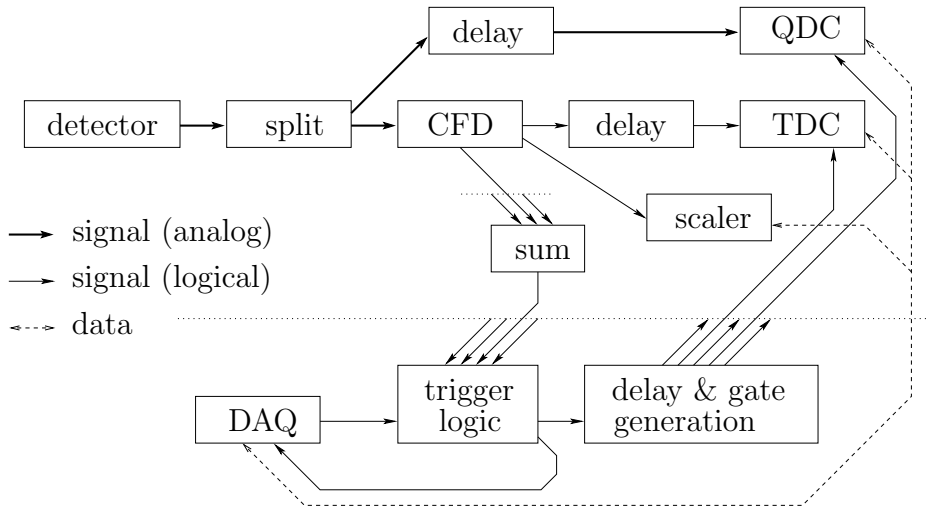

Figure 1.5: Electronics chain for one channel, and it's interaction with the trigger electronics and the DAQ program. (See the Glossary and Figure 7.1 therein for more details.)

1.4 Data Acquisition

The data acquisition system (DAQ) is responsible for the recording of the signals produced by the detectors.

A schematic view of how the information from one detector channel reaches the read-out program in the DAQ is in Figure 1.5. The analog output from the detector channels are continuously processed by the electronics. When both the time and amplitude of a channel are to be recorded, the signal must first be split into two branches. The signal destined for amplitude measurement in a QDC or ADC only need to be delayed, in order to wait for a trigger decision.

Both the trigger decision and time measurement is based on logical signals (i.e. with only timing characteristics being important). The definition of when the analog signal is considered to be a valid signal (large enough), and it's timing, is decided by the constant fraction discriminator (CFD) in the second branch. The total number of valid pulses from each channel, irrespective of any trigger decisions and dead-time, are counted by a scaler.

From each detector that contributes to the trigger decision, the information from the individual channels are combined into a few signals (multiplicity, or and sometimes also an analog sum) continuously measuring the presence and absence of hits. The trigger decision is constructed as the requirement of hits in one or several detectors at the same time (coincidences).

The use of coincidences between several individual channels make it very difficult for random noise that by chance passes the threshold in a discriminator to cause an event read-out when actually no ion has passed through the setup.

Provided the DAQ is not in dead-time (busy processing a previous event), the trigger can be accepted. From the master start, signalling the acceptance of the event, gates and start signals are generated for the Q/ADCs and the TDCs, and the DAQ program is informed of the work to come, asserting the dead-time once again.

In conclusion, at points in time when an event has happened, and the trigger request has been accepted, the processed signals are digitised by the conversion modules. The event data is then read out, dead-time is released and the system is ready for another event.

The time for processing one event is about 350 μs for the LAND electronics. For a smaller setup, it can be ten times shorter, or 50 μs, mostly being limited by the time to notify the read-out processor for each event. By utilising buffering capabilities in the DAMs, this can be squeezed to about 10 μs, being limited by the conversion times. PM tubes are able to handle still higher rates, but e.g. the shaping amplifiers used to read out silicon diodes usually have time constants not much smaller, such that going even further quickly enters the regime of pile-up.

1.4.1 Alternative Schemes

Depending on particular needs in experiments, various adjustments of the arrangement depicted in Figure 1.5 are employed. One example is the deployment of flash ADCs, sampling not only the height or integral of a pulse, but it's entire shape during the gate. This then make it possible to extract more information from each channel, e.g. particle discrimination thanks to their different interaction characteristics with the detector material.

Total Data Read-Out

A large paradigm shift in the way to acquire data is the abolishment of the trigger electronics. By letting each channel "trigger" itself on each signal, all data is collected. One requirement in this scheme is a time distribution/synchronisation system, such that the coincidences between the hits in different channel of the same and different detectors can be determined in software. If this approach is beneficial or not depend on the characteristics of the experiment. When it is hard to define a good minimum bias trigger in real-time with fast electronics (basically incoming ion before reaction tar-

get), e.g. when there are no detectors before the target, the scheme excels. When any detector channel has such a high rate that the digitisation of the signals cannot cope with the rate[14], a triggered system would fare better. This might be the consequence if it is necessary to operate some channel "in the noise", i.e. with a low threshold. If such a channel is not capable on it's own to fire the trigger, it may be harmless to a triggered system, but not to one which attempt to record every pulse. A total data read-out system is suitable for granular detector systems, where the granularity ensures that no single channel see the effect of all ions.

Also, moving the trigger decision from hardware to software does not make it any easier to define and construct the trigger logic. Particularly if the trigger decisions are to be made in software on-line, i.e. as a first stage filter of what data gets permanently stored at all, great caution must be exercised.

1.5 Large Amounts of (raw) Data

The versatility of the ALADiN-LAND experimental setup comes at a price: many (different) detectors are used, and relatively large amounts of raw data is collected in each experiment.

The versatility is shown by the fact that different kinds of physical features are investigated, and among them, many elements have been studied. In common they have that neutron detection is of major importance [also fragment with mass almost the same as the incoming ion (only few nucleons lost)]. Currently a series of experiments are planned, aiming at investigating nuclei where the protons are at large.

This versatility is reached by effectively inserting another layer between the experiment (nature itself) and the physics analysis leading to the final results. The experiment(s) as such are not counting experiments. One does not turn the apparatus on, and after some time a spectroscopic factor[15], cross section, excited state peak position appears on a screen. Neither does an angular correlation spectrum, nor an excitation energy dependent cross section.

[14]One example would be the scintillators far upstream in the FRS that the LAND-experiment uses for TOF measurements. These detectors see a much higher rate of particles, since most are not wanted and will never make it to the experimental area, being rejected by the magnetic rigidity selection of the FRS.

[15]Essentially weights showing how to decompose the wave function into partial waves in another coordinate system: the single particle states).

1.5.1 Advantages

For each event, the incoming and outgoing ions are detected (location, time of passage, energy loss) in several detectors. The physical quantities (like momentum, charge and mass) of the ions can only be determined after these measurements are combined. After this merging of information the physical quantities are used to event-wise see what happened in the target (or at least, what went in, and what came out) and some spectra can be produced. We are still not at the level where the data can be treated as a counting experiment, but a whole lot closer!

The versatility comes from the fact that the detectors as such, their measurements and the reconstruction of the events so far actually can not care less about what happened in the target. Of course, if the ions are lighter (and less charged) and therefore do not produce so large signals as heavier ones, then they will be harder to detect, but this has nothing to do with if the incoming ion got elastically scattered, made a diffraction break-up, got GDR-excited and boiled a neutron off (and some gammas perhaps) or did not undergo any nuclear reaction at all. That is a different issue altogether. *A large part of the analysis can be made truly experiment independent and then re-used many times.*

The large amounts of data comes from the fact that the measurements of all detectors are stored, and later being used as the input of the analysis. This is immediately about an order of magnitude more data than the information about the momenta and particle mass and charge of all observed ions. This approach to performing an experiment is not at all new, but the importance of this fact cannot be overstated. The advantage of recording and storing much more data than immediately necessary[16], is that one may at a later date, after extracting the wanted results, find that also other physical quantities or relations can be extracted from the data collected.

1.6 And They are Us

The tasks of the analysis work are usually separated, or at least viewed as, the on-line and the off-line analysis. On-line is the analysis which is needed as support during data-taking, and off-line the analysis done afterwards (to produce final results) when the information gained to adjust the experimental conditions is no longer available.

[16]What could be recorded, was recorded, with the exception of e.g. the lack of a gamma detector will give no gamma data, and a restrictive trigger logic might completely omit certain events.

This does not mean that two different sets of analysis code need to be developed, but rather that the on-line analysis is a subset of the more accurate off-line analysis. As a consequence a complete off-line analysis tool[17] which should also be used for on-line analysis gets a few extra requirements to be fit for use also under experimental conditions. These demands are chiefly dictated by the extreme time pressure present during an experiment[18]. To cope with that the parts used for on-line analysis need to a large extent be self-adjusting and user-parameter free or easily boot-strappable, such that any needed parameters quickly can be determined. This functionality is of course also useful for an off-line-only analysis, but does for that use not need the stricter designation "necessary", but only "preferable".

I now would like to argue for the case that actually most of the analysis should be done to fulfil the on-line usage requirements.

1.6.1 DAQ and Slow Control

First a slight digression pointing out a difference between the data acquisition (DAQ) and the slow control.

Although having many complications of it's own, and a few setup-parameters controlling it's operation, the DAQ (the running data acquisition program as such) does not have very many soft parameters that need adjustment. The settings are rather of the kind if a certain module is there or not, which is not continuously adjustable like a high voltage or CFD threshold. (The ADC/QDC pedestal cut values used for zero suppression are in one sense soft and an exception, since they are 'continuous' variables. But they are (or at least must, should) not under direct user control - they are a reaction, not an action; a symptom, not a cause - and can easily be automatically handled, see 3.3.) The data acquisition program just runs and collects data and presents this (to the on-line analysis) alongside permanent storage. In this sense, the DAQ is running on it's own, with only one goal, correctly and as quickly as possible transport data from one point (electronic modules) to another (permanent storage).

The slow control, on the other hand, has no such will of it's own. Except for also having a rather hard list of setup parameters, being the mapping of what hardware is connected where, it's setup parameters are of the soft, continuous kind: high voltage settings, thresholds, currents, all to be determined by the user. And of course also an (possibly even larger) set of monitoring

[17] Framework, system, or whatever fancy word comes to mind.

[18] Time is an issue, since waiting for a result, or rather, spending time on coercing the tools to deliver a result equals lost beam time, which is costly - on the order of a few thousand € per hour.

parameters: HV supply currents, magnetic fields, signal rates... The DAQ and the analysis are connected in that the analysis tool need to be able to unpack and use the data produced, but this is only an one-way communication. The slow-control and the analysis however should interact.

1.6.2 Slow Control and Analysis are One

Particularly for large, segmented detectors, most of the slow control parameters are actually not freely adjustable. When one high voltage value has been specified, then all others are (in theory) given due to the preference of running a detector with gain-matched signals, making segments behave as similarly as possible. (For a gamma detector, this might of course be a Doppler corrected correlation between the channels. Still it is such that setting one gain, the gain of all other channels are implied.)

To have different gains is not immediately fatal, since it can be corrected for during analysis by supplying appropriate calibration parameters. There are side-effects however, that are not so easy to compensate for after data has been taken, and they are usually more subtle. A channel with a larger gain will go into overflow for lower actual deposited energy than others. A too high gain make it sooner cross the registration threshold, rendering it more sensitive (and noisy). Even more disturbing, having a smaller gain causes it to be less sensitive and loose information, that other channels would have recorded. The only correction possible is to compare the data collected with simulated data, where the detector segments are simulated with the same non-gain-matched settings. This actually requires that the gains and effective thresholds be determined *at latest* during the off-line analysis and to be used for the simulation.

Having the code ready and available to determine these parameters, already on-line, during the setup phase of an experiment, allowing the correction and matching of the gains before data is taken, relieves much pain during off-line analysis, and also makes on-line analysis simpler, since already raw data will be of much use, reducing the effort needed to calibrate the detectors during experiment.

This argumentation is circular, a "Catch 22". The only way not to need to calibrate the detector on-line is to run code off-line which calibrates them. So the routines must be available in any case! It just shows that it cannot (or at least should not) be avoided on-line, at the peril of the dungeons of off-line analysis of data from half randomly operating equipment.

Therefore the slow control must not only be easy to operate. It is of paramount importance that one can also easily use the on-line analysis to feed calibration/correction values determined by it to the running slow control.

1.6.3 Perpetrating an Experiment

As seen, many parts of the full analysis are useful also for performing an experiment, and by using the off-line analysis tools also during data-taking, the need for special on-line programs is reduced. Also, some of the extra programs needed primarily for monitoring during experiment can be quite helpful also afterwards, while performing the off-line analysis.

The main obstacles most often preventing these synergy effects to come into play are not technical, but management related. In short: to be able to use the off-line analysis during experiment requires it *to exist*. All to often, analysis work is completely postponed until after the experiment. This is to some extent linked to the all to common lack of a full simulation of the performance of the experiment before the hardware setup phase.

It should also be noted, that when "cheating", it usually goes into resolution not into the direct results. The results can most often be verified by checking the outcome of known cases, while the resolution obtainable with a certain setup is an intrinsic property which cannot be checked against outside sources. But cheating may (due to an apparent lack of resolution) prevent some results from being obtained at all! This is however usually very hard to show or prove, since the effects of the hit-and-run approach to performing experiments are mostly subtle and creeping (creepy?). The only effective proof is to actually do an "full" analysis. However, this most often can not be done in scattered pieces, but requires the entire process to be rectified.

Chapter 2

The Setup

The virtues and benefits for physics experiments of the kinematically complete ALADiN-LAND reaction setup in Cave B, during this work dismantled and re-ignited in Cave C, has already been briefly covered in the introduction, and extensively elsewhere [13]. Instead, we will take a look from a DAQ perspective.

2.1 Electronics and Data Acquisition

The electronics for the LAND setup is divided into two parts. The detectors with many channels (segmented detectors located behind the target) have their full read-out (split-card, QDC, CFD, delay, TDC, and scaler) in the cave not too far away from the detectors. All signals from the detectors with few channels (mostly upstream the target) are transported outside the cave to the Messhütte where their full electronics is housed. The only exception is a few pre-amps that must be close to their Si-detectors at the beam line. The trigger logic is housed in the Messhütte. The electronics in the cave provide trigger signals (sum/or) from the large detectors to the trigger electronics, which in return provide the master trigger (also) for the electronics in the cave.

The DAQ/digitisation itself consist of one FASTBUS crate[1] in the cave surrounded by about 10 CAMAC crates of constant fraction discriminators (handled by the slow control[2]), delay modules and scalers. In the Messhütte,

[1]Since the move of the setup to cave C, there are actually two FASTBUS crate - since the one has gotten old and cranky, and hardly is able to work reliably when it is completely filled with modules. It is also mechanically much easier to handle two half-full crates, as with only one the cables to the modules become very densely packed.

[2]In its current incarnation (out-of-date and subject to bit-rot), it is more slow than control...

the DAQ controls two CAMAC crates, one mainly housing TDCs and QDCs (and an ADC for some silicon pin diodes), the other one is used for a pattern unit and a few scalers recording the actual triggers, and two logic matrices controlling the trigger conditions. It also has a programmable time calibrator module, used to gain-match all TDCs in the setup. Some 15 NIM crates are also used.

The data acquisition is running the Multi Branch System, a general DAQ framework actively developed at GSI [14]. It supports the use of several read-out processors taking data, being synchronised via trigger modules interconnected with a trigger bus (cable). The MBS system handles all data transport and file writing after the data has been put into the event buffers in the processor. The read-out functions, "talking" to the actual hardware, is implemented as user functions (as they are experiment specific) and then compiled and linked into one of the programs making up the full MBS system. The user function is not only responsible for the event-wise read-out, but also for initialising the DAMs (digital acquisition module) at startup and to control other periodic tasks, like reacting to the special time calibrator events by doing the programming and firing of the time calibrator module. (And lately also assuring a reliable and responsible handling of the A/QDC pedestals, see Section 3.3).

2.2 Backwards ray-tracing

Backwards in the sense that forwards would be if one knew the initial conditions (particle ID (mass and charge) and momenta at the target).

The basic idea is elegant and simple. Since one can not observe how a reaction happens, measurements must be limited to a before and after view of the involved nuclei. Using the fact the the reactions are here performed at high enough energies[3], enough detectors can be placed such that the incoming (before) and outgoing (after) particles are fully characterised.

Fully described means an event-wise unambiguous determination of the isotope and the momentum vector. It is not including properties whose measurements event-wise only would be statistical (like polarisation) or internal energy (isomers)[4]. The exception being the gamma ray measurement around the target location.

[3]Enough energy (speed) is necessary, to allow for particles being detected passing through a set of detectors while losing energy.

[4]The setup does not have enough resolution to determine the mass of the particles down to isomer excitation energies.

2.3 The S245 Experiment

The S245 experiment, aiming at the investigation of the neutron-rich halo nuclei ^{8}He, ^{11}Li and ^{14}Be, was carried out with the ALADiN-LAND setup in Cave B at GSI in September 2001. The exotic isotopes were produced via fragmentation by letting an ^{18}O beam impinge on a Be target after the SIS, and the isotopes of interest were selected by their magnetic rigidity with the FRS.

The experiment is the third in a series investigating these very light, halo nuclei in complete kinematics. The previous experiments have examined the halo phenomenon using both C and Pb targets, probing with the strong as well as the electromagnetic interaction, respectively. This time a liquid hydrogen target was deployed. The small charge of the target ensures the reactions to be almost exclusively of nuclear origin, i.e. mediated by the strong force. As the target is a nucleon, and not a nucleus, no internal degrees of freedom can be excited (at the energies available in the experiment).

One exciting feature is the possibility of the detection of scattered protons from the target. See Figure 2.1 for more details on the experimental setup.

The remainder of this thesis will only implicitly touch the particular subject of the S245 experiment, as it deals with the use and development of a complete analysis system for the LAND setup, `land02`, re-usable also for future experiments[5]. Starting to work with the analysis, it became clear that some problems have their root in the DAQ...

[5]The calibration and analysis of S245 is still under way, being delayed mostly by the author's insistence on the use of solutions, not hacks, when it comes to software.

Figure 2.1: The S245 experimental setup (September 2001, Cave B, GSI). Not to scale, see Figure 4.5. The incoming beam is identified by means of time-of-flight between two scintillators and energy loss in a silicon detector. The (x,y) position on the target is determined with a MWPC. Outgoing fragments have their angle measured by a MWPC and the velocity by the time of flight to the TFW. The isotope is determined by means of energy loss in a silicon diode (charge) and the track curvature in the magnetic field of ALADiN (mass). Neutrons are detected in LAND giving both the velocities (with a TOF measurement) and the outgoing angles. Target-like protons are detected at angles between $66^\circ - 81^\circ$ in the lab: their direction with two MWPCs, and energy and id via the punch-through pattern in the TOF detector.

Chapter 3

Data Acquisition

A DAQ is like the waist of an hour-glass. All experiment preparations are filtered through it, creating the basis for the analysis and all outcome of the exercises. It is an autonomous conglomerate of analog and digital electronics, computers and software, capable of recording the results of many thousands of events every second. Without rest, sometimes for many months continuously. Reliability and correctness are of paramount importance, and those goals can only be achieved through discipline and hard work.

3.1 Annoyances

After the hardware and software has been designed and proven to assure accurate operation, there are still many things that can cause much grief. Proportional to the number of components in the system, there there are many, although small, technical issues that can disturb the process of data taking:

- Broken equipment. Clear and distinct errors are the best errors, they usually are not shy and make their presence known.

- Semi-functional electronics is much harder to deal with, and may cause much more grief, as it even with extensive monitoring and testing sometimes is detected only when fully calibrating the experiment[1].

[1]Without monitoring, semi-functional equipment can be directly disastrous. As an example, 10 years of collected dust can make a connection, intermittently, e.g. make a TDC unusable by delivering "Christmas Tree" data (all channels firing). This once rendered half the LAND detector blind for one experiment.

3.1.1 Dear Problems

Another nuisance to a stably running[2] DAQ are the Experimentalists themselves. Requests for last-minute changes. Unauthorised spaghetti-cabling. Nurturing of the short-sighted idea that the currently running experiment most likely is the last (as in final) and anyhow the world's most important, such that proper implementation of features needed can wait till after just this run. All this make it virtually impossible to run a DAQ in an efficient manner. But what comes around goes around - any dirty tricks during data collection will spread themselves into an even more unmaintainable analysis code - c.f. Section 4.1.

In setups of this size it is unlikely that everything is working perfectly. It even feels like a miracle each time all components are ready for operation and data-taking can run smoothly. Much moral is required to appreciate that moment, and say: "Keep your fingers away and hold your breath". An even greater amount of courage and discipline must be exercised to keep the factors affecting analysis as similar as possible *between* experiments, while also allowing for the necessary and simplifying developments of the setup to take place.

3.2 Counter-measures

3.2.1 Testing, testing...

The only way to deal with equipment that is not working as expected is to throw it out of the DAQ! This is necessary as basically any single non-working component may render an entire experiment useless, or at least significantly impair it - usually making analysis more complicated, and/or hamper the resolution.

To localise broken equipment, efficient testing is required, particularly to find the problems before the experiment and not afterwards when they find you while analysing. *Therefore the DAQ programs should make use of the built-in testing functions of the electronics and exercise them regularly.* Also, the DAQ and electronics is kept constantly running months before the experiment to weed out also intermittent problems[3].

[2]Within the scope of an experiment, as well as re-usability between different experiments performed with the same setup.

[3]The DAQ always runs.

3.2.2 Cable documentation

As a direct consequence of inevitable human errors[4] a cable documentation was introduced, as a formalised scheme for making sure that cables are connected as one thinks, and not hopes.

To have documentation of the setup is in no way a new invention. Also to have it in digital form is not new. The main difference from the previous documentation at the LAND setup, is that before, the information was *read-out channel* oriented, and kept fairly minimal, although it was (with some gymnastics) possible to figure out what CFD/TDC/QDC/scaler was attached to a particular signal. The new one makes much less attempts to be minimal and instead enforce the documentation of *all* electronics equipment and their interconnections[5]. From this, the same information as before (and much more) can be extracted.

One goal of this more elaborate documentation is to facilitate a more advanced and easier to use on-line analysis. The reason why the elaboration is needed, is that the very nature of doing on-line analysis is characterised by a constant lack of time. It is therefore necessary that the analysis program (framework) easily can adjust itself to the actual setup and provide as complete information as possible[6].

The formalised cable documentation work-flow is then:

1. Setup (parts of) the electronics.
2. Write the cable documentation for that electronics.
3. Run the program, which checks and uses the written documentation, including:
   - Parse and syntax verify the input.
   - Check that the cable connections are consistent (possible since all cables are mentioned twice, once per module they are connected to).

[4] Which is normal, the trick is just to be able to deal with (find) them.

[5] While the old documentation required quite some amount of thinking on the part of the user to locate a module (due to much implicit information), the new system leaves most of the boring thinking to an (actually quite stupid) computer program, freeing up the physicist's mental resources for more productive work (like fixing the actual equipment).

[6] It's like building a (tall house). To get results out of the analysis, if some parts are missing, fudge parameters or the equivalent are needed, which actually can be used quite successfully. When too many levels are fudged, it's no longer possible to add yet another one level. Therefore, a sound base (and first floors) are very helpful.

- Generate code and/or tables for the read-out in the DAQ.
- Generate signal/HV mappings for the slow control and on-line monitoring system.
- Generate signal (unpack) tables for the on/off-line analysis.

If some part of the electronics is then changed (like moving or replacing a CFD in the middle of a chain), it's only necessary to update the documentation accordingly. Finding out which channels are affected is done by just re-running the check/generation program providing new, updated tables and listings.

See Appendix C for an example.

3.3 Continuous pedestal monitoring and adjustment

In addition to the absolute requirement of correctness, a DAQ should be fast. And preferably also collect the data compactly, reducing the size of the data that need to be waded through when analysing the experiment. One popular means to foster both these ends (high speed and small size) is zero-suppression. This Section describes how the necessary run-time parameters can be automatically determined and used.

3.3.1 Zero-suppression

Zero-suppression is the very simplistic but highly effective compression method of leaving all data out which contain no information, i.e. are zero. The additional cost is the requirement to store the channel numbers associated with non-zero data since this is no longer implicitly known as the indices in an array. This is usually much smaller than storing all data when only a few channels out of many fire in each event. This is usually the case for segmented detectors. Zero-suppression is useless (more expensive) for channels which fire in all or almost all events.

Example: LAND has $10 \times 20 \times 2 \times 2$ planes $\times$ paddles $\times$ PM$\times$ (time and energy) = 800 channels. An average cosmic event has ≈ 15 paddles firing, i.e. 60 channels. A physics event ≈ 5 paddles, i.e. 20 channels.

Storing the 800 channels, 12 bits of data directly is 9600 bits = 1200 bytes. Storing them more sanely (byte aligned in 16 bit words) needs 1600 bytes.

Storing 60 channels of data + channel location needs (12+10) (10 is $_2\log 800$) bits per channel, i.e. 1320 bits = 165 bytes. For 20 channels, this is 55 bytes. Storing it more sanely, using 32 bit data words, needs 240 bytes for 60 channels, and 80 bytes for 20 channels. See Table 3.1 for an overview.

The ratios become 165/1200 = 13.7%, 240/1600 = 15%, 55/1200 = 4.5% and 80/1600 = 5%.

Event type	Paddles	Channels	Bits/ch bits	Packed bits (bytes)	16-bit bytes	32-bit bytes
Full	200	800	12	9600 (1200)	1600	3200
Zero-suppr.			12+10			
Physics	5	20		440 (55)	-	80
Cosmics	15	60		1320 (165)	-	240

Table 3.1: Average data sizes for the LAND detector for various event types and storage schemes.

There are two major classes of data, time and amplitude recordings. To achieve zero-suppression for time signals is usually fairly easy. The signal as such is logical, with the recorded value describing when it occurred relative to a common reference time signal. If the logic signal does not occur at all due to not passing the discriminator threshold, the value is zero, and left out. Amplitudes are however always there, as a baseline or 'zero'-noise level, called pedestal. Zero suppression in this case is handled after digitisation and achieved by (per channel) defining a level below which the amplitude is regarded as noise-only, the value as zero, or non-existing, carrying no information.

It is important to note, that although the pedestal (or noise level) as such has a physical meaning, in that the scale for the measured amplitude of a signal does have the origin (zero) at the pedestal; this is not what concerns us here. We only want to avoid storing signals which are anyhow compatible with baseline noise only. Regarding the DAQ correctness issue, this pedestal removal is done behalf of zero-suppression and nothing else.

3.3.2 To cut or subtract the pedestal?

Assuming the pedestal in a single channel (ADC/QDC) spectrum to be a Gaussian, let us characterise it by the location and width μ_{ped} and σ_{ped}, see Figure 3.1. (Where, for a good ADC and analog electronics chain σ_{ped} should

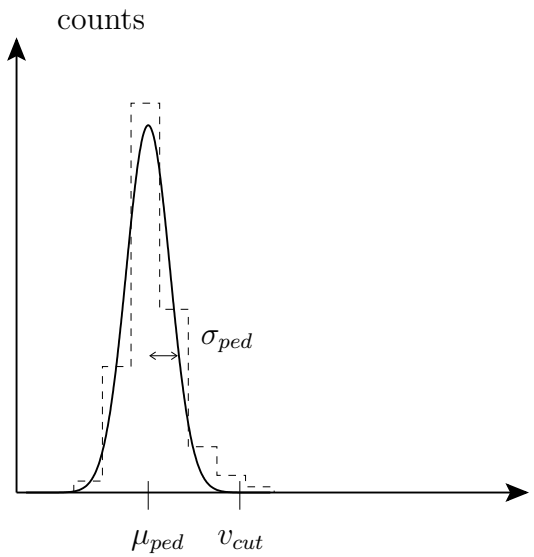

Figure 3.1: Pedestal of one channel. For a good ADC (balanced electronics chain), a peak of width one. The pedestal cut is located above the peak mean value.

be on the order of 1[7], and μ_{ped} not too high, as to make the most out of the range of the ADC, since lower values will never be used).

The DAQ has two ways to handle the removal of the pedestal from the data delivered by the Q/ADC. The noise-compatible values can on one hand be simply discarded (cut), doing the equivalent of

$$v_{stored} = \begin{cases} v_{ADC} & v_{ADC} > v_{cut} \\ 0 & v_{ADC} \leq v_{cut} \end{cases} \tag{3.1}$$

One the other hand, the DAQ can do the subtraction of the pedestal (making the scale of the stored data start at 0 physical amplitude (above baseline)), since the pedestal location is used to do the cut:

$$v_{stored} = \begin{cases} v_{ADC} - v_{subtract} & v_{ADC} > v_{subtract} \\ 0 & v_{ADC} \leq v_{subtract} \end{cases} \tag{3.2}$$

However, (3.2) is fundamentally flawed for even though it makes v_{stored} have 0 at 0 physical amplitude, it does not achieve zero-suppression. In order for the pedestal removal to have an useful effect, $v_{subtract}$ (or v_{cut}) cannot be μ_{ped}, but must be $\mu_{ped} + a\sigma_{ped}$, a being around 2 or so as a margin. Otherwise,

[7]The pedestal width is a measure of the converter resolution. If $\gg 1$, it just means that the least significant bits of the converter are random.

only half of the noise values would be discarded. For the subtraction to be useful for the analysis, the value subtracted must however be μ_{ped}. The DAQ would need to first do a subtraction (needed for the comparison to v_{cut}), and then again add the margin used, $v_{margin} = a\sigma_{ped}$:

$$v_{stored} = \begin{cases} v_{ADC} - v_{subtract} = (v_{ADC} - v_{cut}) + v_{margin} & v_{ADC} > v_{cut} \\ 0 & v_{ADC} \leq v_{cut} \end{cases} \quad (3.3)$$

This leads to unnecessary complication of the DAQ code. Especially as, in the cases where there is any hardware support for pedestal cutting or subtraction, it is always performing (3.1) or (3.2), not (3.3), meaning that the DAQ software would have to rewrite data to do the addition of v_{margin}. It is also needed to store the subtracted values, in case these were wrong.

Discarded data can not be recovered, in any of the cases.

In the opinion of the author, it is therefore strongly recommended to never use pedestal subtraction[8] (even in the presence of Section 3.3.3), but rather do only the cutting. Both operations achieve the same zero-suppressive effect.

3.3.3 Automatic pedestal determination

Either the DAQ performs pedestal cutting or subtraction it must posses the needed parameters (v_{cut} or $v_{subtract}$). This information is obtained by triggering the DAQ with either a regularly occurring pulse[9] or artificially, the important point it being a non-physics-event trigger. Such that the detectors deliver no signals, non-physics = no ions. The ADCs will then digitise the noise they see - the pedestals. It is important that the trigger is external to the modules, using the normal gate generation, since the gate length must be the same as for physics triggers. With the collected data from a few hundred events the needed parameters can be determined and passed to the DAQ program.

Our previous way of doing this was that the data got collected in a special mode of the DAQ (either by stopping it completely, and running a separate program which does artificial triggering and data collection, or by collecting data from the running DAQ, but only for specially triggered events, which are known to be non-physics generated). Another program would then analyse the collected data and deliver the μ_{ped}, σ_{ped} and v_{cut} for each channel. The v_{cut} then need to be fed back into the DAQ, which would involve to storing

[8] The subtraction of v_{cut} might seem useful when doing the on-line analysis, leaving out the requirement for dealing with the pedestals there. It is a Pyrrhic victory when it comes to doing the high resolution off-line analysis.

[9] "Clock" trigger in LAND jargon.

them in a file which is read in the DAQ start-up phase - requiring a reset[10] of the running data acquisition.

In the presence of special pedestal triggers, there is nothing preventing the running DAQ program to 'snoop' on the collected data, and for each such trigger internally keep a copy of the data. When enough such events has been seen (a few hundred), it can do the same (fairly simple) analysis as before to determine the needed v_{cut} parameters, and then put them to immediate use in the running acquisition. This is done repeatedly, i.e. after a new batch of pedestal data is available, the analysis may be done again, and fresh v_{cut} values be put to use. This happens about every 5 minutes.

To avoid the creation of unnecessary dead-time, the DAQ will when it has gotten enough pedestal data, still postpone any analysis until it anyhow is more or less idle, e.g. after receiving an end-of-spill event.

We expect v_{cut} to be stable (otherwise the pedestals are drifting and the analog chain or the ADC modules are broken). The DAQ now also continuously monitors the equipment, issuing warnings whenever the pedestals move suspiciously much.

As a final addition, in case the hardware also is cabled with a possibility of generating an artificial trigger (ADC gate), the DAQ program can upon startup use this facility to make an initial pedestal determination, such that the run also starts with good values and a working zero-suppression, not needing to wait for enough pedestal triggers to occur.

With the LAND FASTBUS, it was noticed that although the artificial trigger generates exactly the same gate as the pedestal trigger, the pedestal values determined were a little different, meaning that optimal pedestals were not achieved until the snooped pedestal triggered data was available to do the analysis. This is attributed to a rate effect in the QDC modules.

3.3.4 Moving pedestals

Although moving pedestals are a sign[11] of broken hardware equipment, the DAQ does it's best to handle the situation. In addition to alerting the user, it adapts to them, such that if they rise, the cut level is increased, keeping zero suppression effective, and when they decrease the cut is lowered such that as little real data as possible get lost.

This behaviour is useful during the setup phase of an experiment, as the pedestals then may have legitimate reasons for varying (changed HV etc).

[10] Almost being a criminal offence (c.f. the title).

[11] Not necessarily, generally impact(?), Could be design.

Clustered hits

For some detectors where each hit is expected to generate signals in several neighbouring channels (clusters) the DAQ also needs to do some kind of peak finding (over several channels) after doing the pedestal subtraction (in each channel). This may be the case when there is an event-by-event baseline shift. It is (in the view of the author) recommended to use a simple difference algorithm, i.e. to find the peaks by locating neighbour channels that after normalisation (equalisation, removing the difference in pedestal between the channels being compared) have a difference larger than a predefined minimum (only a few, since we expect the ADCs to be stable). In this scheme, if the pedestal of one channel turns out to be wrong, the effect it will have is to most of the time give rise to a large difference, i.e. signalling a false peak (which gets stored). It will almost never loose data, since that would require the actual peak to almost exactly compensate the erroneous pedestal value. All further (proper) peak analysis can then be left to the analysis stages.

Furthermore, when each hit can give a signal in several channels, the maximum value obtained in one of the channels can have a large variation due to the exact location of the hit. It can either make the largest energy deposit within one channel with a centered hit, or divide the charge between two neighbour channels when passing through their boundary. In this case, any cut only operating at single channels has very bad efficiency and acceptance characteristics. By instead letting the comparisons operate on running averages of several channels, the threshold effect become more uniform. The effect of single channel noise is also reduced as the size of the average region is increased, although it of course becomes larger on an absolute scale, such that the average region should not be chosen larger than the region the smallest signal peak to be detected is expected to cover.

Chapter 4

Analysis, pre-physics

4.1 Mischief

The formulas used by the reconstruction routines are very seldom complicated[1]. The major hurdle and stumbling block for the analysis code is the necessity to cope with changing conditions:

- Voluntary changes, like adjusting a magnetic field.
- Involuntary changes, like instable power supplies (moving gain factors around), temperature dependent electronics moving timing around, and dust in electronics suddenly disabling entire modules and just as suddenly returning to normal operation.
- Re-cabling. A signal for some reason need to be reconnected, ending up in another DAM. Usually invalidates most of the calibration for that channel.
- Another experiment. The ultimate change. (Handling this is however almost for free when handling the other possible alterations. It must nonetheless be considered when the code should be useful for several experiments.)
- All leftovers from Section 3.1.

This all boils down to the structure of the analysis code being dictated not by what transformations need to be applied to the data, but rather by how to handle all the book-keeping associated with the cataloguing of which parameter to use where for what events.

[1] Most often linear, or in special cases quadratic.

The transformation routines themselves are comparatively simple to invent, describe and develop. The calibration routines require much more work, since they are to provide parameters and not just use them. Artificial intelligence is a too strong word, but good heuristics are needed. The methods also have to be robust.

The challenge is to get them to talk to each other, e.g. to make the output from a calibration routine available to the event reconstruction routine, in an efficient, straight-forward and debuggable way.

4.2 Text files

To ease debugging, all information that is passed among the different stages of the analysis, particularly, in any way passes the file system calls[2] should be in human readable text format.

The use of text files, as opposed to custom binary formats, is an (debatable) design decision - a few reasons:

- They are suitable for handling using standard code version control tools (like CVS). Which also helps to find differences[3] between versions.
- Rollback is easy, particularly for partial operations. Cut and paste with an editor.
- As a reference, I'd point to the `/etc` directory of an UNIX operating system, which contain it's setup parameters. It contains almost exclusively text files.
- It is easy to insert comments into them, describing peculiarities about the information. The documentation is then at the same place as the data itself.

Human read- and editable (text) files have the disadvantage of being more complicated to both generate and particularly parse (read). This disadvantage can actually be turned into an advantage - see Appendix D.

[2]Includes all program I/O, e.g. `read`, `write`, `printf`.

[3]The idea of a code repository with revision control is not just to store the information. One also wants to retrieve it again, and easily see the differences, and then only the differences between different versions, and not two large (binary) files, which need to be compared somehow.

4.2.1 Clarity

The information should also be unambiguous, e.g. unindexed arrays of calibration parameters are prohibited due to their extreme likelihood of confused reconstruction. Each value should also when applicable have units attached.

4.2.2 Exceptions

I case of event-wise data and histograms it is ineffective to store the data in text form.

Also transport of data *within* a program should be in native format, since unnecessary string generation and parsing easily eats a large fraction of the processing time, and also invites to ambiguities, and is error prone.

4.3 Example 1: Time-of-Flight Wall

The charged particle time-of-flight wall (TFW), is a 1.8×1.4m large segmented heavy ion detector. It's main purpose is to measure the time-of-flight of ions relative to some other detector upstream in the setup. It also delivers a rough position measurement $\approx$ 10cm, and energy loss. It consist of two planes of 18×14 paddles of 5 mm thick, 10 cm wide plastic scintillators, read out at both ends with PM tubes. See Figure 1.2 (left).

4.3.1 Internal Detector Calibration

For a crossed (two orthogonal planes) time-of-flight wall, calibration is a straight-forward procedure:

As the detector itself is over-determined, an internal self-calibration can be done. And with real data, so it also may keep track of itself during the experiment. The principle is:

- Using any particles from physics events hitting the wall, select unambiguous events. Unambiguity means that we require to have well separated hits, preferably even single hits (one in the horizontal and one in the vertical plane of paddles), see Figure 4.1. Such a cut here does not mean that events are rejected from the analysis, only from the calibration.

- As the events are unambiguous, the coordinate of the crossing of the two orthogonal paddles are used as an estimate of where the ion passed

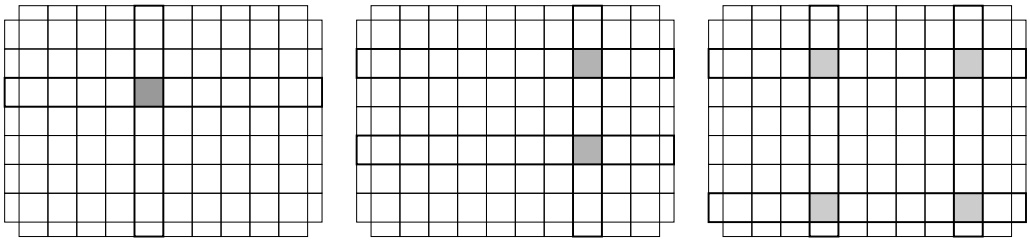

Figure 4.1: Unambiguous (left) and ambiguous (centre and right) hits in the TFW detector.

the detector. The position calculated from the time-difference and energy-log-difference within the paddles can then be calibrated against this position.

- We also know that the ion passed the paddles in the two layers simultaneously[4]. We use this fact to synchronise the mean time calculated for the two paddles, i.e. determine the relative time offset between the pair of horizontal and vertical paddles.

- The same principle is applied to the energies. Assuming a small relative energy loss in the first plane, the lost energy in each plane can be treated to be equal.

- Statistics is collected for these six variables from each accepted event, on a per-crossing horizontal-vertical paddle-pair basis. They should then essentially each show a Gaussian behaviour. To make a further reduction of bad events, if an event makes a contribution far from one of the Gaussians, the event is removed from all distributions.

$$t_{i,j,\text{diffx}} = \frac{t_r - t_l}{2} \tag{4.1a}$$

$$t_{i,j,\text{diffy}} = \frac{t_u - t_d}{2} \tag{4.1b}$$

$$t_{i,j,\text{mean}} = \frac{t_r + t_l}{2} - \frac{t_u + t_d}{2} \tag{4.1c}$$

[4]The time-of-flight difference (1 cm between the two layers) can actually be completely ignored. Since we want the detector to finally deliver one time for each particle, by ignoring this offset, we virtually shift one plane (the back) to have the same z-position as the front plane. This is possible as the velocity of the ions only vary a little around the much larger beam velocity.

$$e_{i,j,\text{diffx}} = \frac{\ln e_r - \ln e_l}{2} \tag{4.1d}$$

$$e_{i,j,\text{diffy}} = \frac{\ln e_u - \ln e_d}{2} \tag{4.1e}$$

$$e_{i,j,\text{mean}} = \frac{\ln e_r - \ln e_l}{2} - \frac{\ln e_u - \ln e_d}{2} \tag{4.1f}$$

- The needed parameters are calculated. For each paddle, the time differences for each crossing are combined into a linear fit, from which a time difference offset is calculated such that a time difference of zero corresponds to a hit in the middle. The slope gives the effective speed of light in the paddle.

  Similarly, a $\ln e_1 - \ln e_2$ offset is calculated such that this is zero when the hit is in the middle. This is equivalent of gain-matching the two energy signals - the corresponding calibration parameter is the gain factor for the energies. The logarithm of the energies comes into play as a consequence of the light attenuation in the scintillator, see Equation 1.7. Also see Appendix A.2.

- The synchronisation of times in the entire detector is calculated at once with an over-determined equation system, where the unknowns are a time offset for each paddle, pairwise involved in one equation per crossing. Since moving all offsets with the same amount also gives a (equally) valid solution, the equation system is singular. To remedy this, one extra equation is added, combining all the unknowns, forcing a weighted average to zero. Still, we cannot say anything about the overall offset, but an internal calibration has been obtained.

  This arbitrary offset is an intrinsic property of the problem and not of this particular solution method[5]. This offset will be fixed when synchronising the detector to the rest of the setup. At which point, the entire setup will have one arbitrary offset.

$$t_i - t_j = t_{i,j,\text{mean}} \tag{4.2}$$

- The overall energy gain-matching is done in the same way as for the times, except that $\ln e$ for each paddle is taken as unknowns instead of t. The offsets of the energies (pedestals) must be determined before the

[5]To e.g. use some (fixed) paddle as reference does not really help, since it only solves part of the problem. What to do when that paddle is broken?

energies can be calibrated at all. Playing the role of floating offset for the time is the absolute scale of the energy. This will be determined when the hits can be associated with different ion species and their (velocity corrected) particular energy losses.

4.3.2 Reconstruction

The Time-of-Flight Wall reconstruction routine is responsible for combining the times and energies of several paddles into one hit (t, ΔE, x, y) describing each particle passing through the detector.

The event reconstruction in a segmented time-of-flight wall proceeds in two steps, paddle hit reconstruction, and combination of paddle hits into physical particles.

- The paddles are first considered individually. Optimally, both times and both energies are available, from which the time of the hit, the position along the paddle and the energy loss are be calculated. If one time or one energy is missing in the event, the position can still be calculated from the quantity which is complete (time or energy), and the time of the hit or the energy deposit is then be recovered using that information. (This becomes especially important for paddles where e.g. the time channel of the PM tube is malfunctioning or for events where the energy deposit is so small (due to low charge) that only one signal make it above the discriminator threshold.)

  The paddle hit reconstruction transforms the two times and energies from the two PM tubes into one mean time (hit time), one energy (loss), and a position along the paddle. The other coordinate is the location of the paddle itself. This way, each paddle delivers a $(t, \Delta e, x, y)$ coordinate within the detector (see 1.3.2).

- Secondly, it is attempted to combine the hits from the paddles into 'particles' by comparing their position (one from paddle location, one along the paddle) and their times. If these are close enough, they are considered as coming from one particle.

  The energies are not used for comparing and thereby combining or separating hits. For good events, we would have the same energy loss in the two planes. Hits just gracing a paddle will have any energy loss between zero and 'full' energy loss (corresponding to passing the full thickness of the paddle). However, it would still most likely have the full energy loss in the crossing paddle, making the difference (or ratio)

arbitrary. The times and positions (even calculated from the energies) will still however be correct.

The combination into physical hits consider all reconstructed hits in the paddle, calculating their relative distance and combines hits deemed to be close enough as being caused by one physical particle.

The distance calculation compares time and distance differences by dividing them by their relative (estimated) accuracies, such that a dimensionless number is obtained, which can be used to declare something to be distant or close. The combination routine most often just need to combine one hit from each plane, but sometimes one particle may affect also two neighbouring paddles in one plane[6].

- Finally, the particle characteristics are calculated as an weighted average from the hits in each cluster. The energies are summed for each plane (vertical and horizontal), and then averaged. These combined quantities will then be used by the next stage analysis, the tracker.

4.3.3 Handling semi-blind paddles

The reconstruction routine event by event tries to recover as much information as possible, and as such also works for semi-blind paddles. It needs the calibration parameters to do this. The calibration routine outlined above can only work with events where the paddles are fully functional (both PM tubes are working). For paddles where no such events are available, the scheme breaks down - since at that stage there are no reference points at all in the variables to be adjusted. The only 'factual' input to the first stage calibrator is the knowledge which two crossing paddles are hit in each event. To be able to adjust the time offsets, the times from both ends of each paddle are needed. These paddles can however be calibrated with a second stage routine, where the fully working (already calibrated) parts of the detector are used as reference. The paddles are individually adjusted such that they fit into the already calibrated (reference) parts of the detector.

4.3.4 Inter-detector Calibration

The calibration and reconstruction so far has only considered the TOF-wall as a stand-alone device. In order to deliver time-of-flight data for particles

[6] An unfortunate side-effect of the current implementation is that a series of hits in neighbouring paddles may pairwise combine. Then also hits far away are declared as belonging to the same particle.

travelling from upstream in the setup, it must have it's time adjusted to the rest of the setup. This is just one parameter (time offset), as the detector is internally synchronised. This is done again with beam, using the particles reconstructed in the TOF-wall for the synchronisation. This accordingly becomes stage 2 (or 3) of the calibration.

4.4 Example 2: LAND, the Large Area Neutron Detector

LAND [15] is a time-of-flight wall for neutrons. The main difference comes from the fact that the neutrons are only indirectly detectable, after they have been "converted" into charged particles. Except for the slightly different active area (2 x 2 m) compared to the TFW, it is in total 1 m thick, half of which is iron in a sandwich structure (each layer .5 cm) between plastic scintillators. The iron acts as a more efficient converter for the neutrons to induce charge particle showers by nuclear reactions. The charged particles are be detected in the scintillator material.

The read-out of the detector is segmented into 10 planes (each 10 cm thick), alternating horizontally and vertically stacked, with 20 paddles in each plane.

4.4.1 Internal Calibration

As the detector is truly 3-dimensional and highly segmented in all directions, it can even without calibrations track (in x, y, z) the unavoidable background radiation of muons (created in the upper atmosphere from cosmic radiation) passing through it, see Figure 4.2. Being minimum ionising, and with a velocity at or above $0.7c$, they usually pass easily through the entire detector in a practically straight track.

The tracks can now be used to for each paddle hit give a reference position, thereby calibrating the effective speed of light in the scintillator and the time difference offset as measured from both ends, then adjustable to be 0 for a hit in the middle of the paddle.

The average time offset of the paddles are calibrated relative to each other, by assuming neighbouring paddles hit by the same muon to have been hit simultaneously. A correction is made for an estimated speed of the muons in case the direction could be determined. The direction can be guessed if the muon has a large vertical component of the trajectory, since they come from above. The uncorrected time-of-flight offsets for horizontal muon tracks will average out, since they horizontally come from all directions, and therefore

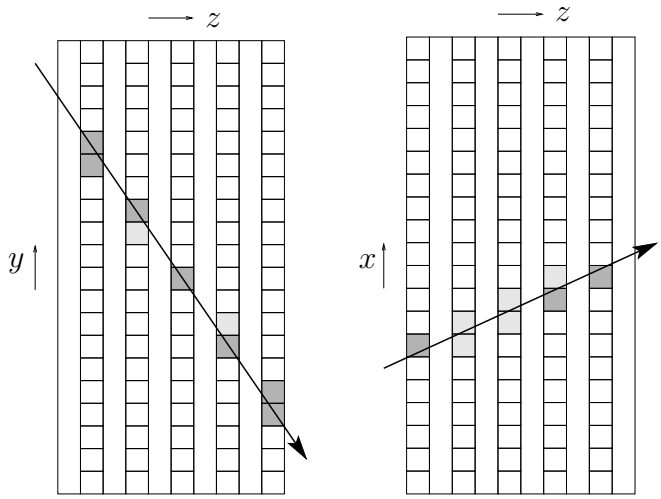

Figure 4.2: Tracking the muon in the zy-plane gives the coordinates in y for vertical paddles, and analogously tracking in the zx-plane gives the horizontal paddles x coordinates.

each pair of neighbouring paddles will get statistics from muons first passing through one and the other.

For further information on the use of this procedure, see [16].

4.4.2 Systematic errors

If not careful, then 'drift' in the times for horizontal paddles depending on height, due to non-c velocity.

The equations are connecting only neighbouring paddles. Thanks to the horizontal/vertical stacking, this will within two equations connect top and bottom horizontal paddles. [7]

4.4.3 Inter-detector Calibration

There are several ways to determine the time offset between LAND and the rest of the setup (target time):

- Using γs coming from the target. The distance is known and the velocity is, by convention, exactly known: c. They are also relatively easy to

[7] Protons with high energy from the target; 1 GeV passes through LAND in also almost straight tracks. These are not suitable for time sync calibration, since they only go in one direction, and the errors in the paddle-to-paddle TOF do not average out.

spot, as they in a velocity spectrum will make a peak around c, while all neutrons are around beam velocity, and the remaining background is relatively flat at larger velocities than beam velocity.

One issue with this method is that while the neutrons have almost the same probability the react anywhere in the depth coordinate in each paddle (due to it's longer mean-free path), the gammas have a mean free path much smaller than the paddle depth, and will therefore mostly react at the front face of the paddles. This causes a systematic offset due to the (on average) less time of flight of at most five 5 cm of the gammas (0.16 ns). This can however be completely cured by determining the gamma average reaction locations, by finding how many gammas react in the second plane relative to the first plane.

- Neutrons from the target, unreacted beam. Assuming beam velocity.
- Cosmics going through both LAND and the TFW, who are quite close. Assuming a correct gamma determination, this might perhaps however be more suited for determining the TFW time offset. (modulo any walk effects in the TFW).

4.5 Data structures

Figure 4.3 takes the TFW as an example through the detector-specific levels.

The data for each event passes through the analysis in several stages, or levels, from unpacking of the data files:

- **raw** data, being the unpacked (integer) data, mapped to the appropriate detector signals.
- **tcal** level data, has the same structure as the raw data, but has undergone a first (linear) transformation, such that the times are in ns, and the energies have 0 at zero. See Appendix A for an analogy/similarity between times and energies. From this level on, the data is floating point.
- **sync** level uses the same data structure as the tcal level. An offset is applied to each time, such that all channels have a common zero, while the energies (amplitudes) get their gain factors applied, such that they are expressed in MeV (according to some definition[8]).

[8]While the times are expressed as real times (with the absolute offset just being fixed relative between all channels), and although an amplitude usually represent the energy

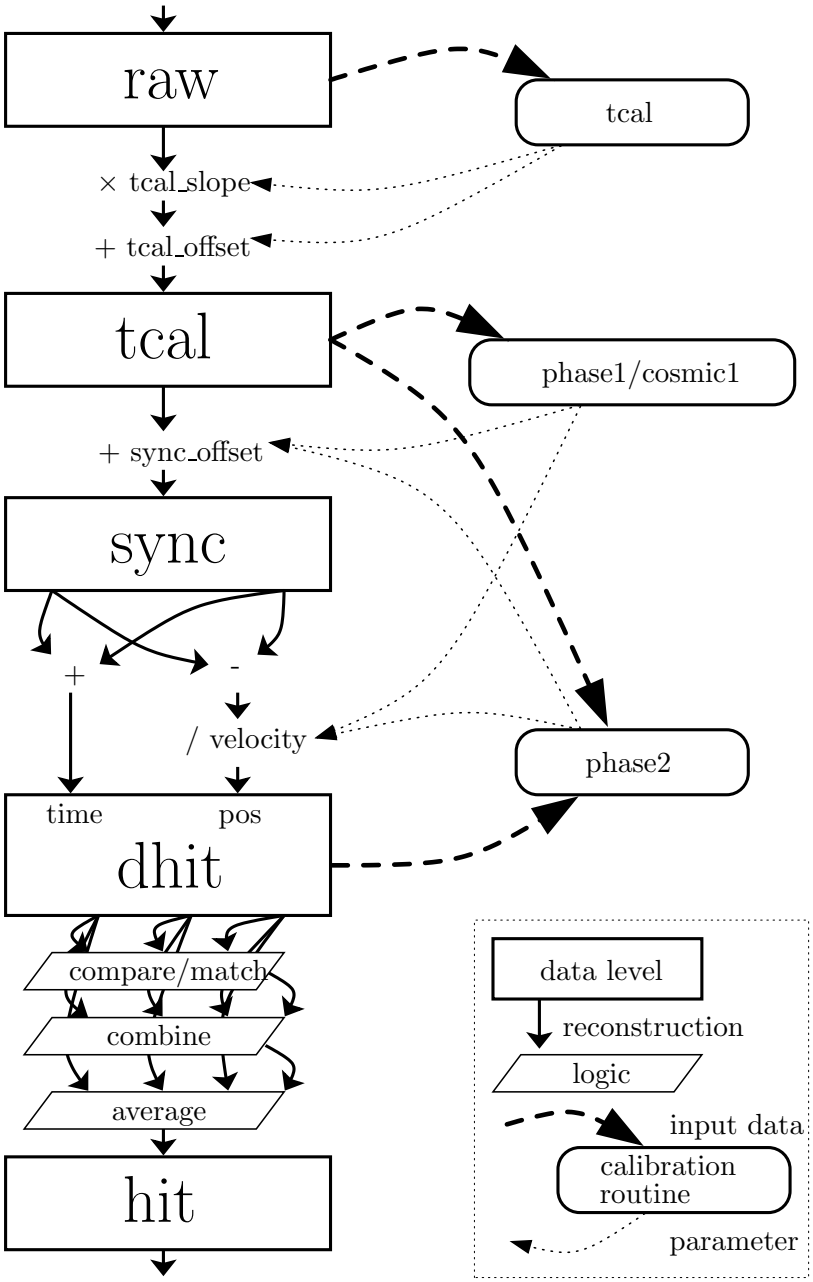

Figure 4.3: Data levels, reconstruction flow and calibration steps. The reconstruction flows from top to bottom through the data levels. The intermediate results are used by various calibration routines that calculate the parameters necessary for the reconstruction to continue to the next level. (The reconstruction and calibration in this example is for the TFW.)

- **dhit** data, describing the hits in a detector internal coordinate system, which usually is not linearly mappable onto the lab coordinate system. This may be a pin-cushion deformation (described as a quadratic polynomial) due to electrical properties of a large silicon diode or light-transport properties in a thin scintillator, or the coordinate on a position sensitive photo-multiplier. It can also be the hits in individual paddles in a segmented detector. From this level on, most quantities have an associated error, stating it's estimated precision.

- **hit** level data, gives the coordinates (in time and space) of the hit in a normal coordinate system with the geometric origin usually at the centre of the active area of the detector, along with the energy deposited in the detector. The units being ns, cm and MeV. This is such that the tracker routines (used to reach the next level) do not need to know what kind of detector delivered the information. A second requisite for the use of generic routines is the uncertainties, as amended and propagated from the dhit level.

- **track** level data connects the information from individual detectors into tracks of particles. This is somewhat separate from the next level, since the track reconstruction need to be done in two stages, first finding out which hits can possibly belong together in a track, and then fitting the data, providing physical quantities around the reaction vertex in the target with:

- **momenta** and particle id of the incoming and outgoing particles (projectile as well as target-like (when possible), and also neutrons and gammas.

4.5.1 Error propagation

Throughout the later stages of the reconstruction code, each value is associated with a precision. One should see them as an approximations, rather than as exact treatment of the uncertainty. They are used to aid the computations, are thought of as the statistical errors, and all supposed (imagined) to be Gaussian. This makes the calculations become manageable and fairly easy.

The errors[9] are stored as the corresponding variance (σ^2 or $1/\sigma^2$). This

loss in a detector, it usually requires a definition to give the signal a scale and relation to the energy loss.

[9]The use of the word "error" is somewhat a misnomer. It is not intended to indicate faults in the data or programs.

is more healthy than storing the bare standard deviation, since it is (almost) always the variance which enter the formulas. This saves many applications of the square and square root operator, making the analysis program both cleaner and faster[10]. A typical example would be the combination of separate measurements, a_i, of one quantity, weighting them with their accuracies, σ_{a_i}, producing one variable, a, accompanied by the combined accuracy, σ_a:

$$\frac{a}{\sigma_a^2} = \frac{a_1}{\sigma_{a_1}^2} + \frac{a_2}{\sigma_{a_2}^2}, \tag{4.3}$$

$$\frac{1}{\sigma_a^2} = \frac{1}{\sigma_{a_1}^2} + \frac{1}{\sigma_{a_2}^2}. \tag{4.4}$$

The application of (4.4) is straightforward when the reciprocal variance is stored. The calculation of (4.3) then also follows as

$$a = \left(a_1 \frac{1}{\sigma_{a_1}^2} + a_2 \frac{1}{\sigma_{a_2}^2} \right) \Big/ \frac{1}{\sigma_a^2}, \tag{4.5}$$

only requiring one division[11].

When measurements are instead to be compared, it is easy to produce dimensionless numbers, s_a, measuring the compatibility, by dividing the difference by the expected accuracy,

$$s_a^2 = \frac{(a_1 - a_2)^2}{\sigma_{a_1}^2 + \sigma_{a_2}^2}. \tag{4.6}$$

These quantities are useful for making decisions, as the threshold has no unit and thus can be rather generally handled. Note how in this case the variance itself is needed in the calculation. To avoid conversions at the end of reconstruction routines, or at the beginning of the next stage, it is necessary to make informed choices of how to store the variance at the various data levels.

These dimensionless numbers for different available quantities (measuring e.g. the compatibility of two hits in the TFW coming from the same particle, comparing their hit times and locations) can then (assuming the errors are independent) be combined into a single dimensionless variable

$$s^2 = s_a^2 + s_b^2 + s_c^2. \tag{4.7}$$

There are several reasons for introducing error handling into the analysis:

[10] One should note, that an increase in computing power available must never be used as an excuse for sloppy programming.

[11] Two multiplications and one addition are very cheap in comparison to the division.

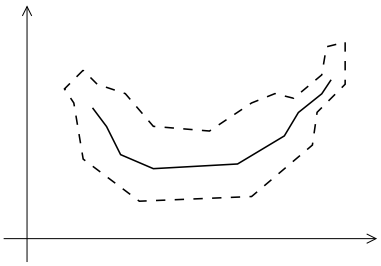

Figure 4.4: A "banana" cut does not explicitly store the expected correlation between variables (solid line), only a limit (dashed line) beyond which it is considered incompatible with the condition requested.

- To avoid (graphical) "banana" cuts. The use of various (most commonly) two-dimensional cuts in correlations between different variables has a few problems, particularly with a general analysis code. The problems have their root in the fact that the cuts do not specify the correlations that are expected, but the area within which they are fulfilled - see Figure 4.4. This makes them hard to transfer between different detectors and experiments, as absolute calibrations then are necessary - the error description is not orthogonal to the location of the values. Worse is that they are hard to determine in any automatic fashion.

- An uncertainty is a continuous variable that directly describe a property of the setup - the resolution or accuracy of a variable. The determination of the uncertainties is almost for free. When calibrating the different parameters of the setup, the variables' accuracies can most often be obtained as the widths of the underlying distributions.

- Information from many quantities can be combined at the same time, by comparing their matching to the nominal values with the respective accuracies into one dimensionless variable, as seen above in (4.6) and (4.7). This single quality quantifier can the used to make decisions.

The uncertainties usually enter the calculations at the dhit level, where they are incorporated as the estimated detector resolutions. They will vary from detector to detector and experiment to experiment, but not event-wise. The event-wise variation then comes from differences of what information is available for each event. The major consumer and beneficiary of the error (or accuracy) estimates are the tracker routines, shown at work in Figure 4.5.

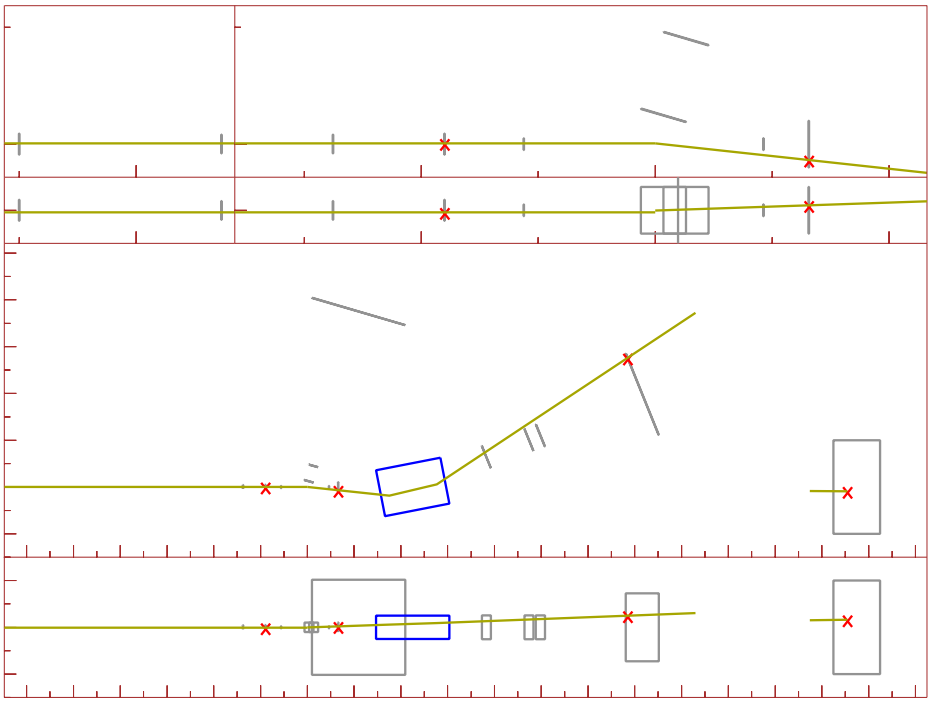

Figure 4.5: View of an incoming and outgoing track in an event from the S245 experiment, with a large outgoing angle from the target. Also a hit in LAND is seen. See Figure 2.1 for a description of the setup.

For the tracker, the use of detector-type unaware[12] data structures with generalised precision specifications is particularly attractive. The reconstruction (hit-matching and fit) routines can be written for a general n-detector setup. Each detector will, per event, deliver the information that is available in the coordinates handled: time (t), position (x, y) and energy loss (Δe). The selection of which detectors to use for events where one measurement of a for the fit over-determined quantity becomes trivial, as it already is part of the handling. Also a selection of an optimal set of detectors to use for the tracking becomes superfluous[13], as an event-by-event optimisation will be done. Detectors with a clearly inferior resolution (for some quantity) will

[12]Independent, unconscious, ignorant, agnostic?

[13]On several occasions when discussing with colleagues the use and inclusion of errors (as opposed to the application of graphical cuts) into the analysis code, quite strong reactions against it have been encountered. The arguments are usually along the lines "Why make it so complicated? Why not just plot a vs. b and have a look? It has not been done like that before." The idea of a general tracker has been met with even more opposition.

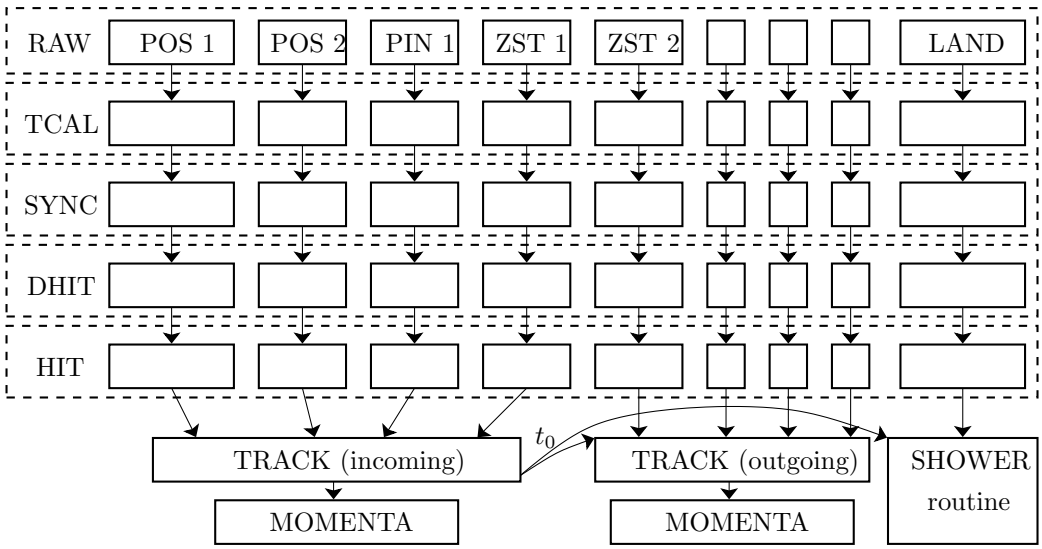

Figure 4.6

not affect the result in an appreciable way, except for events where enough information from precise detectors is missing. The poorer fit quality will be known, and such events can be removed in later analysis stages, when deemed profitable.

4.5.2 Orientation of the enclosing data structure

The event data structures are organised in the levels described above, with each level (raw - hit) having a substructure for the data of each detector, see Figure 4.6. At the track level and beyond, the data is handled per particle (incoming, outgoing (fragment, proton), neutrons, gammas). This leads to the analysis software for each event looping over each pair of subsequent data structures (e.g. raw and tcal) for each detector to do the transformation of the data from the previous to the next level. The track reconstructor is more like an octopus (but still a loop), using as input all detectors at the hit level which are relevant for a certain track (incoming or outgoing) and as output the tracks. The incoming tracker is simple in the sense that all ions do pass all detectors, the only thing to consider is if some detector in an event for some reason does not deliver (complete) information (due to $< 100\%$ efficiency or even being permanently broken). The outgoing track must first identify the candidates for each track.

The neutron detector and gamma detectors are special in the sense that

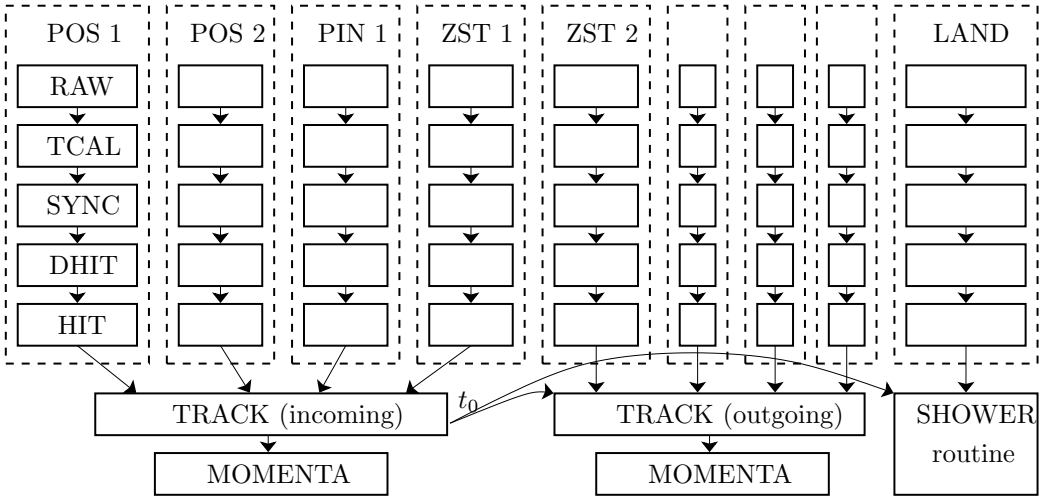

Figure 4.7: The data structures are the same as in Figure 4.6, but their enclosing structure is organised in the other direction, with each detector as a logical unit.

they are not involved in a track together with any other detector, but their 'track' or particle reconstruction requires the time of the reaction in the target as an input parameter, i.e. they must be run after the charged particle trackers.

4.5.3 Re-orienting the data structures

One could also have organised the data in the detector dependent levels (raw - hit) in the other direction, i.e. such that each instance of a detector would have been at the outer level, itself consisting of the data from each level. See Figure 4.7. This would have made the flow of the reconstruction routines up to the hit level more direct. Since some transformations are very similar for many detectors, it would likely have made these synergies less obvious. Also, the octopus behaviour and structure of the track level (and it's reconstruction) can not be changed very much. Since e.g. the neutron reconstruction need to have a reaction time calculated when doing the reconstruction, some parts need to be done after the tracking either way.

Also, running the reconstruction in sync for all detectors in order of refinement levels instead of detectors make much sense for the calibration procedures, since these usually need to run the reconstruction over the full data set, up to one level, and after that, being run on the full set again up till the next level, utilising the parameters produced in the last run for reconstruction

of the next input level.

The effects on the processor should be minimal, since in either case, the same transformations need to be performed. The layout of the data structures can then only (or at least, most importantly) affect the cache characteristics while running the reconstruction. But if all the event structures do not fit into the cache, then also the calibration parameters will be evicted from the cache, which is just as bad, since the calculations use as much data for each event reconstruction from the calibration structures as the event-wise data structures.

The reconstruction methods, calibration routines and data structures are all written such, that although it is not a trivial to re-orient the data structures and associated control flow, only a limited amount of modifications are needed. This comes as a result of the object oriented handling of the data, where the routines only know the minimum needed to perform their work. Hence, only the containing data structures and corresponding loops would need to be changed in order to make the program reconstruct data for one detector at a time from raw to hit level data, in a depth-first approach, as opposed to the current breadth-first operation. The biggest pain would be to adjust the distribution of the calibration parameters.

Chapter 5

Dead-time

When the DAQ system is busy and not able to accept a trigger, it is said to have dead-time. Under the assumption that all events are independent, in particular, that they occur in time randomly, independently of each other, the dead-time will not bias the data collected, only reduce the amount of statistics. This is generally the case for experiments performed at GSI at or after the FRS since the ions for each event comes from stable ions extracted from the SIS. Each secondary ion production reaction is random, i.e. independent, and the time to usage, i.e. the transport time is $< 1\mu$s, such that also any difference (dispersion) is much smaller than the typical half lives of a few ms. Also see Appendix E.

Although one can think of more general setups, where different parts of the setup have independently running DAQs, both the GSI LAND experiments and the REX-ISOLDE reaction experiments run in such a way, that whenever some part cannot acquire data, the entire system is dead and not accepting triggers. This makes perfect sense, since the data is collected for physical events, where if one gets any data, one likes to get all the data from the entire setup. Having events where half the data is missing would just make the analysis uselessly more complicated.

5.1 Sources of evil

Clearly defined dead-time may perhaps be called hard dead-time. It is mechanically/electrically/logically blocking the acquisition of new events, as a veto signal. This would be:

- The time after a common start/stop/gate during which the modules are converting data, and are not able to accept another trigger (common signal). (When running in single-event mode, this seamlessly merge

into the read-out time, but it is a separate quantity in multi-event mode). Time scale: anything between 5-300 μs.

- The read-out time - the time during which the read-out processor(s) are fetching data from the acquisition modules (DAMs). This dead-time must be held until the modules have been cleared of their data, such that they can accept a new trigger. If the data is transported in several steps from the processor, the later steps may be performed while a new event gets accepted and converted, i.e. without causing dead-time. Time scale: from around 20-50 μs to several ms.

  Since the read-out processor may at the trigger time be performing work on behalf of another task (like sending previous data over the network), the response time can for some events be unusually large. This is not necessarily bad, if the occurrences are rare, since the alternative might be a higher *average* response time, due to an increased overall CPU load when chopping the other work up in too small pieces. Time scale: a few ms.

There are also the "soft" sources of dead-time, which are harder to quantify, and therefore not directly included into the DAQ dead-time veto signal. The prime example would be pile-up, i.e. when two events happen too close in time, their signals will overlap, as in Figure 5.1. For time measurements, parts of one event can then not be recorded since it is shadowed by hits from the other event. For pulse heights measurements, the signals add (also non-linearly) to each other such that the data becomes useless. Usually, some (necessary, but) slow shaping amplifier is the cause for the longest overlap time in an experiment.

For events affected by pile-up, although the trigger fires and the DAQ reads the event out as usual, the event will at analysis have to be discarded, since the data is unusable. The unfortunate consequence of this is that even though the events within the overlap window are predestined for rejection, firing an event read-out for any of them usually creates a much larger time window of dead-time than the overlap region, during which perfectly acceptable triggers may occur, which is rejected on behalf of acquiring an unusable event.

5.2 Suppressing pile-up

Their handling of pile-up as far as possible need to go into the trigger logic as such to convert the soft dead-time into hard. Pile-up can be rejected without biasing the data collection provided that:

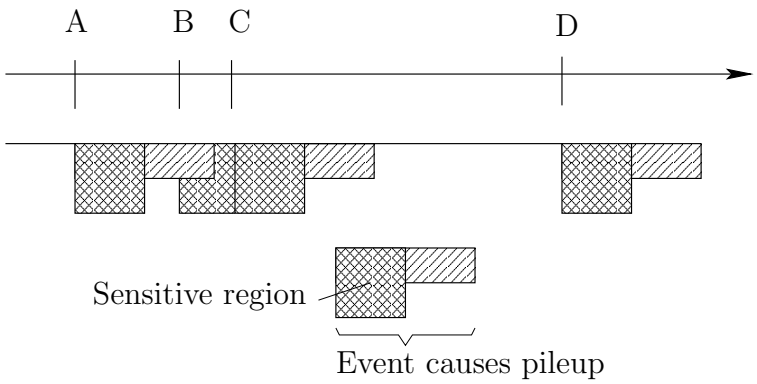

Figure 5.1: Pileup. Event A makes event B useless. B makes C useless, and C make B useless. The measurement time needs to be free of pile-up caused by other events.

- The occurrence of events (actually incoming ions) are independent, coming at random times (but not necessarily uniformly distributed).
- The pile-up can be classified in a neutral way *before* the reaction target.

Then, these events may be removed from the trigger stream. Assuming incoming ions A and B occurring so close that they cause overlap in some of the electronics such that the collected data would be distorted. Note that happening alone, both events A and B would be good, only when occurring together they are useless. Two cases may occur:

Event B would normally fire the trigger. Easy case. Due to A already happening, it make no sense to collect event B. B can then be rejected just with a long veto originating from the occurrence of event A. This would be the equivalent of having a kicker-magnet upstream the beam-pipe, removing ions coming too soon after a previous ones. This is also the same effect as decreasing the beam intensity (which would be the zero order solution to combat pile-up. Effectively, B has never happened.

Event A would normally fire the trigger. This is much harder, since at the time when the trigger decision for A must be made, it is not yet known that B will come[1]. So event A has to fire the trigger and cause signal collection and conversion to be started. Since however B will destroy the data, it would make sense to abort or prevent the conversion (a fast clear) as soon as B is seen. The fast clear however needs cooperation from all involved DAM modules, since all need to reset themselves, being ready for a possible event

[1]Could actually at this time still be in the SIS and not even ejected into the FRS yes...

C, happening outside the reach of B, but hopefully within the now avoided read-out dead-time.

5.3 Combatting dead-time

The dead-time for (not) accepting a new trigger is a global combination (logical or) of all dead-time produced by individual components. There are only a few things to do to reduce this:

- Minimise component dead-time. Can some processing in a computer perhaps be done after releasing the dead-time? Then it is reduced to a second order effect, in that only if the next event comes quickly enough, the processor will not be able to handle it directly, and then can not release the dead-time after the second event early.
- Parallelise the dead-time. If one component by design anyhow has a certain dead-time at a specific point in time (e.g. the conversion time in a module), then it is for "free" to make some other component also perform any dead-time producing activity in this time frame.
- Have the dead-time happen when no triggers are expected. Then, the caused dead-time is also for free, since no events are lost due to it. This would be to perform any background processing needed during off-spill (LAND, e.g. pedestal determination), or when very few ions are expected due to their half-life (REX-ISOLDE, shortly before a proton on target pulse). Note however, that if the DAQ is always ordered to perform some activity at these specific intervals, data from these particular time periods will not be available at all, and the measurement get biased. The biasing would be limited to the suppression of background triggers (which cannot be eliminated anyhow, but are expected to be uniformly distributed). They will be rejected in a certain (small) time window, and can be corrected for.

5.4 Dead-time "quality"

From the above discussion, it is clear that although dead-time is the particular stretches of time (usually after each accepted event) when the DAQ is dead[2], such that it is conveniently measured in μs, one can of course also

[2]Or actually working - confusingly, the DAQ is never as active as when the system is having dead-time.

give a measure of the dead-time as the fraction of the time no triggers can be accepted. However, a count of triggers lost due to dead-time will not necessarily be equal for different triggers, particularly if certain (large stretches) of dead-time are introduced when the trigger rate is at a minimum, causing unproportionally few triggers to be lost. Live-time is just 1-DT.

5.5 Multi-event Read-Out

The hard dead-time mainly has two components: conversion dead-time and read-out dead-time. The first one is given by the design of the used modules, and can only be reduced if the modules are replaced by better ones. The latter consist mainly of the data transfer (over CAMAC, VME or some other data bus) to the processor and any other processing done. Usually the processing part is very small, and large data transfers within the processor can be scheduled such that the dead-time is released before these operations are performed. The effects of the read-out time can possibly be reduced by using multi-event read-out, i.e. deferring the read-out of the modules until the data of several events have accumulated in the DAMs, see Figure 5.2. First an important sanity check should be considered though:

5.5.1 Synchronisation

The time at which the read-out program as such is executing, i.e. after it got the knowledge of an occurred trigger, and before it has willingly released the dead-time, is special since we then know that dead-time is active[3], and no triggers can occur[4]. This means that all data collected (yet read-out or not) belongs to triggers before this instance in time, and all later events (that have not happened yet) are, well, later.

These synchronising points in time may also be used to do certain changes to the operation of the DAQ, the effect of which must be classifiable to happen exactly between events, like (automatically) adjusting pedestal cuts.

The synchronising points are however most important for knowing that data from different events are not mixed together in different modules. If one module would get a spurious trigger before another, real, one, in most cases the second will silently be ignored, causing event mixing. When running in multi-event mode (collecting several events before all of them are read out), the data from different events must be merged with the help of local

[3]This is e.g. by design done by using the dead-time output of the trigger module in MBS

[4]Minus any cabling errors in the trigger logic, of course.

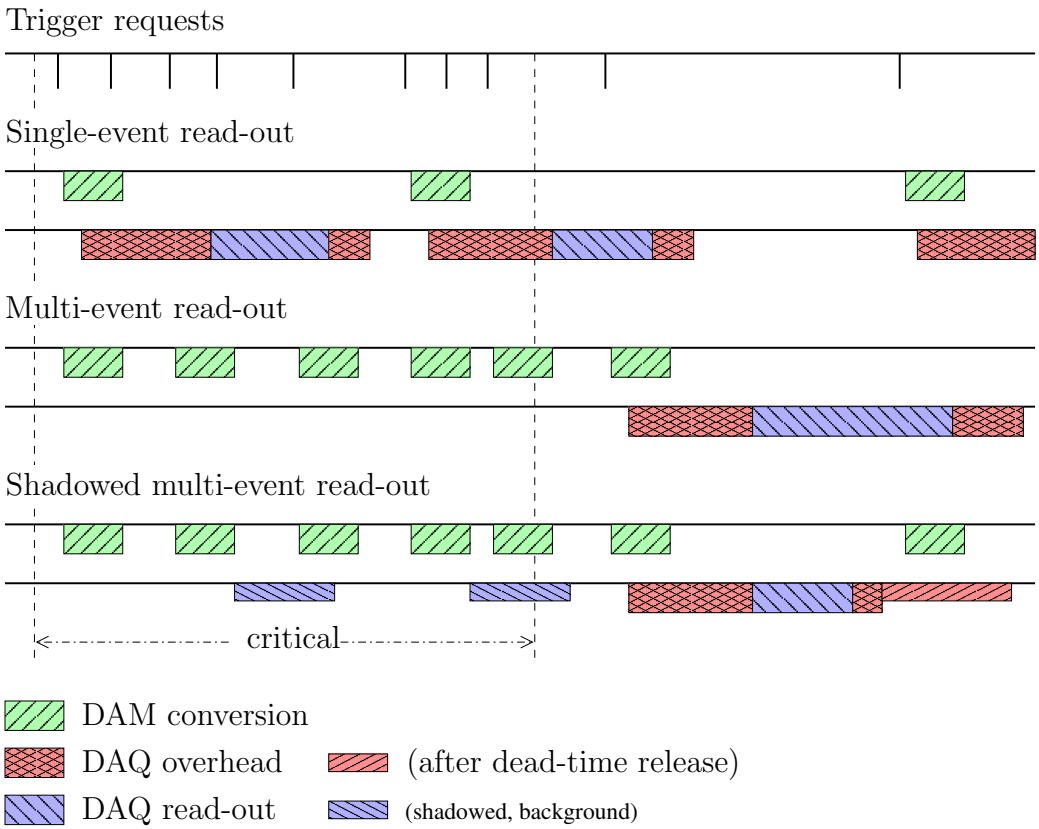

Figure 5.2: DAM conversion dead-time, DAQ overhead and read-out dead-time all contribute to the dead-time. With multi-event read-out it is possible to reduce and move the read-out time away from the most critical periods (with many triggers).

events counters in each module. If these counters have not increased by the same amount, then some spurious or lost triggers have occurred, causing unhandlable unpredictable mixing of event data. In practise, this will happen very rarely (if not, the trigger cabling/logic is not clean, and must be fixed), but must nonetheless be rigorously and continuously checked by the running DAQ.

5.5.2 Speeding the Read-out Up

One way to speed the read-out is to read in total less data. (Without sacrificing vital information). The main contributions here would be to activate zero-suppression (i.e. ignore data below a certain threshold) in any capable modules. A second would be to pack the data more densely. The packing is only effective against read-out dead-time when it results in less cycles on the data bus[5].

In cases where each DAM has a fair amount of data to deliver, another way of increasing the read-out throughput is to when available use more advanced transfer modes on the data bus, like BMA (block move) transfers. This does a bulk transfer of data in one go on the VME data bus instead of the basic single-shot data word read-out directly performed by the processor. The BMA is controlled by a special chip on the processor board which is first programmed by the processor, and then performs the transfer autonomously. The chip-programming involves some overhead, but the transfer time for each data word is drastically reduced. The overhead is further marginalised by running in multi-event mode, letting the data for several events accumulate in the modules before they are read out.

5.5.3 Shadowing the Read-out

When the conversion modules are capable of storing multiple events internally, they are usually also able to allow read out of previous data while digitising new events, letting the internal event buffer act like a FIFO. When this is the case, the DAQ program may start to read data out as soon as the first event has been converted, and continue to read the modules out until the next synchronising event appears, at which point only the then remaining data need to be read, instead of all. This way a large fraction of the read-out dead-time is completely shadowed by both the unavoidable conversion dead-time, and live-time of the trigger logic waiting for interesting enough triggers.

Optimally, this would be handled such that the modules send an interrupt (look-at-me, or rather, please-empty-me) when their internal buffers are almost full), the read-out processor will then queue them for read-out of the data into an internal intermediate buffer in the processor's memory (usually much cheaper and larger than the memory available in a DAM). Hopefully the processor (DAQ program) will actually start to read events out before

[5]The data bus (e.g. CAMAC, VME) is usually the time limiting factor of the read-out system, possibly nut rarely in competition with network bandwidth. Processor speed is usually sufficient.

the modules buffer is completely full, which would block the trigger from continuing selecting events. The internal buffers would then be merged with the data remaining in the modules at the next (forced or voluntary) synchronising event. In such a way, a modules with a 32 entries deep multi-event buffer, may act like it can handle hundreds of events, before a synchronising event becomes necessary.

Point being that a synchronising event (causing/forcing a complete read-out) is time consuming. It on average it needs to read half the used size of the multi-event buffers in the *modules*. But a few hundred events may like this have been acquired at zero read-out dead-time cost.

Chapter 6

Spin-off: ISOLDE DAQ

During this work, the opportunity arose to look at, go through, use, and improve one of the data acquisition systems at ISOLDE, CERN. Also there the complexity is increasing, from before using only a handful of channels, to today sometimes several hundred. Fortunately, in each experiment only a handful of detectors are in use, and the channel count is mainly caused by increased granularity of the read-out, e.g. more and smaller segments of silicon diodes.

There are large synergy effects that can be stimulated when handling and running several different DAQs[1]. The system studied at ISOLDE is housed in a rolling rack and consist of a NIM crate for a simple trigger logic and a VME crate with a processor and trigger module adapted to run the MBS system, a scaler, TDCs and ADCs as needed by the experiments using it.

As such, the DAQ is much simpler than e.g. the LAND DAQ, but one must also adapt the operation of the DAQ to the specific environment in which it runs. This mostly comes as the two scenarios considered (decay experiments, and reaction experiments at the REX post-accelerator), both have a much smaller and easily manageable average data rate, but an extremely high instantaneous rate.

6.1 Preliminaries: Dead-time

Dead-time is the time during which a certain part or parts of the experiment cannot handle a new event due to being busy handling a previous event. The evil cousin of dead-time is pile-up, since dead-time would just inhibit signals

[1]Which also happened. Actually, even quite some ideas that were later used also for the LAND DAQ were developed and refined during a few quiet weeks working at ISOLDE.

or events, while pile-up will generate whatever values usually larger than the true ones, making the information unreliable.

Dead-time can be caused at all levels of the electronics chain:

- Detector dead-time. Most detectors (if not all) work by having energy stored in a volume (usually in the form of an electrical field) on the limit to breakdown. (The depleted region in a semiconductor Si (particle) or Ge (gamma) detector, the field on the dynodes in a photo-multiplier, the field between the anodes and cathodes in a wire chamber). When a particle passes through it, the energy is released and as a result an electrical signal is produced. This is necessary as the detectors need to have very fast response times (on the order of some ten ps in the case of PM tubes).

  They are so to say always ready to signal the passing of particle. Just after producing the signal however, the field has been weakened such that another passing particle will not cause a signal, since it takes some time for the bias voltage supply of the detectors to rebuild the stored energy. Even though most detectors are operated in a proportional regime (such that the output signal is not saturated, but rather proportional to (primarily) the charge of the passing particle), enough of the stored field is used to make the signal of the second particle weaker than it should be.

- Electronics dead-time. Electronics can be dead, or in an ill-defined state when two signals must be handled too close in time. An illustrative case is a shaping amplifier. Such an amplifier is usually producing a signal with a well defined amplitude from a noisy input. The output signal is often on the order of a few μs long. If another signals has arrived just before, or comes a little later, they will be summed, such that the wanted signal rides on a tail, making the output useless and misleading.

  Pile-up occurs when the particles come so often that, the recovery does not complete and the signals get distorted due to that. Either in the detectors or the electronics.

- DAQ dead-time. After receiving a trigger, all the converter modules (TDCs, QDCs, scalers) must digitise the information. And if they are not equipped with any multi-event buffers, they also need to be read out. Until this is done, no further triggers may be accepted.

  This is usually the time-wise largest dead-time, but only occurs for events which had accepted triggers, while the detector and electronics

dead-times apply to all particles that pass the setup, either they are of interest or not.

I hope to have conveyed that dead-time as such is not bad, it is inevitable. Zero dead-time just means that no events are collected. It just need to be under control, and of course kept to a minimum.

6.1.1 Reducing dead-time

DAQ dead-time is usually composed of two distinct sources. Module (converter) dead-time, which usually cannot be affected, except by possibly disabling unused channels. The read-out dead-time, which is caused by the communication between the modules and processor used and the processor latency to respond to a trigger interrupt (when running in single event mode). The read-out time can be reduced by only reading appropriate channels out. The overhead of executing a few house-keeping instructions in the processor is usually negligible compared to the saved communication cycles.

A completely different regime of read-out dead-time can be entered by using modules which have internal buffers for multiple events. With these, read-out may sometimes be done in parallel to conversion, essentially obliterating the read-out associated dead-time, or in special cases done at times when no new triggers are expected, clearing the modules to be ready at the next shot.

6.1.2 Handling pile-up

As opposed to dead-time, which mostly just causes lost events on a random basis (i.e. also not affecting cross section measurements - however, bear Appendix E in mind), leading to a reduced event collection rate, pile-up destroys data for (affected) events by mixing signals associated with different incoming ions. One obvious method is to reduce that overall rate at the experiment. Another way is to be able to detect the affected events either before collecting them (thereby also eliminating their contribution to read-out dead-time and data storage) or during off-line analysis.

Pile-up is usually mostly a problem for detectors tracking the incoming particles, i.e. before the target, since these see all ions. If the pile-up affects a slower detector, and also an (almost) unaffected faster detector is available, the faster detector can be used to identify the troubling candidates (by simply seeing more that one ion during a specified time interval) and record this.

On-line removal: reject the trigger if another ion also passed. (This is easily done when the disturbing ion comes previously to the wanted event.

Off-line removal: reject the event based on collected data. This can be sufficient to remove events where the disturbing ion came after the event. The alternative is to implement a fast clear scheme in the DAQ.

If the rejection detector is location upstream the target, this operation is also safe for cross section measurements, since rejecting an ion based on an upstream detector is equivalent of the ion never appearing at the setup.

6.2 Two scenarios considered

6.2.1 Decay experiments

The interesting point here is that most of the events may come within a few ms after the protons hit the production target (every 1.2 s). During this time, dead-time should be kept to an absolute minimum. The rest of the time, the DAQ may very well by fairly slow (dead).

6.2.2 REX experiments

Are similar as above, except that the events are coming within a 50-few 100 μs long window each 20 ms (EBIS pulse). During this window, dead-time is lethal to the statistics collected. Except in this window, dead-time is not too dangerous.

6.3 Requirements

The requirements for the above experiments are: a number of ADC and TDC channels to be read out. The experiment is responsible for delivering one master trigger, from which a start/gate is produced and an event created. For measuring long times (ms) (time since protons on target etc), a 1 MHz (easily calibrated with processor on-board clock) is used to also measure the time from a) proton impact and b) (where applicable) relation to the EBIS pulse. The scaler also records some monitoring information, e.g. raw triggers and accepted triggers, number of proton and EBIS pulses. That's all.

6.3.1 Single-event read-out

Up till now, the rolling VME-based DAQs, featuring one scaler and a few ADCs and TDCs, were being read out in single event mode. Each master trigger fires the read-out via an input to the VME trigger module. The

invocation of the read-out function has an overhead of ≈ 35 μs. Additionally comes the read-out itself, at about 1 μs per word transferred on the VME backplane, which also includes book-keeping. The read-out of each event easily reaches 100 μs. For REX experiments this means that only one event per EBIS pulse can be recorded. For decay experiments, it limits the instantaneous rate to 10 kHz.

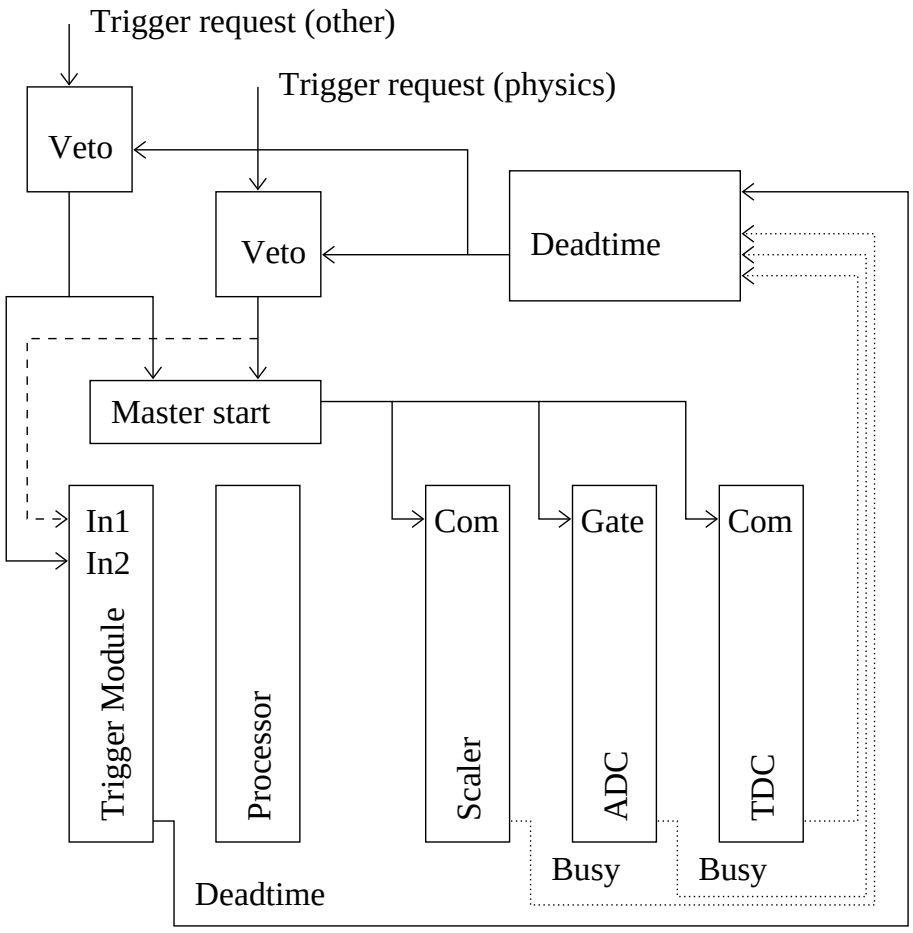

Figure 6.1: Trigger logic for single-event read-out. Dead-time can be handled solely by the trigger module, as it will not be released before the modules have been read out (and therefore are finished converting). For multi-event read-out, the dashed connection is removed, and the dotted connections added, assuring dead-time during conversion.

6.3.2 Multi-event read-out

The solution to the problem above is to avoid invoking the trigger module every event as in Figure 6.1. The scaler, ADCs and TDCs (but not the pattern unit, which has been thrown out) are all multi-event capable. Each trigger (or rather, start/gate) is sent to all modules, and they provide a busy output (or'ed to the global dead-time) until the event has been converted. Then another event can be recorded. Depending on the gate length, the dead-time may be as short as 8.5 μs (with a 1 μs gate), giving a possible instantaneous rate of 120 kHz[2].

The scaler can internally store $\approx$ 1000 full events, or more if fewer channels are used, while the ADC/TDCs only store 32 events before they must be read out. The modules notify the processor via interrupts when they start to get full, and can actually be read out while they continue to convert. If this is started before the modules are full, one might collect more events before a complete read-out is necessary, avoiding dead-time. Effectively, the buffers are deepened by using memory in the processor.

At regular intervals, preferably some time before the proton impact (or EBIS pulses), the trigger module is triggered by special events for protons (or EBIS), at which time global dead-time becomes active from the trigger module, and all events up till that point are completely read out and written to file, together with the firing event. Also, all modules internal event counters are compared so that they all saw the same number of triggers.

Should the processor-assisted deep buffers become full, a read-out is forced to empty the buffers, so that acquisition can continue without waiting for the special triggers. The buffers sizes should of course be adjusted to avoid this happening too often.

6.4 Additions

Trigger box. To be able to also record signals that have a very high rate, without completely block the acquisition, a trigger box might be of use to be able to down-scale the trigger rate from these detectors. However, this of course requires the detector to be faster than the busy time of multi-event-collection for it to be of large use. In any case, the output from this may still be a master trigger to the standard DAQ.

[2]This has been observed by the author; also as an average rate, but at that time only with very few channels active, due to single-shot VME read-out.

6.4.1 Network scaler view

To aid in running and setting the experiment up, the raw scaler data is on every special trigger for protons (every 1.2 s), sent out on the network for processing by an user client program, among other things calculating trigger rates. The transport is UDP, to avoid the DAQ program failing just because the user program went away.

Chapter 7

Outlook

7.1 Ordnung muß sein

Where the soup sticks to the spoon, is when one realizes that the extra layer of raw detector data between the experiment and the physical quantities in a certain sense actually dictates (or at least mandates) a certain, rather disciplined, way of working. The raw data as such looks essentially like random numbers[1] and only through a very laborious procedure can these random-look-alikes be transformed into the wanted quantities (momenta, particle-id). The upside is that most of this laborious work can (and due to the amounts must) be done by computer programs. The upside to this, is that these programs, just as the detectors can also be made in such a way that the same programs can be used for many experiments.

The downside is that unless written to be experiment independent, the programs will usually also not behave such, i.e. one will when trying to apply it to the next experiment then need to expend on the order of the same amount of effort as originally went into writing it to adapt it to be useful again. And if this effort is not used to make the program 'clean' (experiment-agnostic), the same effort will be spent again the next time. And this for only one of many parts of all needed to analyse an entire experiment.

To make this explicit: there is no such thing as a free lunch with computers! Either one plans ahead and make the computer programs work for oneself, which effectively means to spend a large fraction of the work up front. Or one tries to 'cheat' and avoid spending the effort preemptively, in which case, at the end a very much larger effort will in total have been spent, just on coping with the insufficient software. Effort which could have been spent on further other development instead, which most likely also would be

[1] Very poor random numbers, if treated as such, but still...

much more interesting. The same applies to hardware: build a detector or read-out system with limited debugging capabilities, and you'll find yourself spending your time trying to debug it.

7.2 Towards NuSTAR/FAIR and R^{3}B

The analysis of the ALADiN-LAND experiments has in the last years (almost unknowingly) been making the most use out of the fact that the setup has not evolved very quickly, and at least in such a way that the data rates has been fairly constant, while at the same time, computing capabilities improvements has almost kept up with Moore's law.

The NuSTAR experiments at the FAIR facility[2] are all sitting (either at, or) behind the Super-FRS in its capacity of looking for a needle in a haystack - in form of exotic nuclei. This means that they all (e.g. R^{3}B just as LAND) are sitting behind a long chain of equipment, meaning that beam-time is very expensive. And all data that can possibly be collected must then be harvested. Also, detectors have a tendency to become more and more segmented and granular (reducing pile-up), also that leading to higher data rates. It then is even more necessary to use the middle layer of large amounts of raw detector data and the computer as a comb to cook the data down.

[2]What a name? unfair? or just only fairly well?

Glossary

ADC	Amplitude-to-digital converter. Measures the maximum amplitude of detector signals during a gate. Commonly used for semiconductor detectors, e.g. silicon diodes. The normal meaning of *analog*-to-digital converter is only used briefly in Section 3.3.
ALADiN	A large acceptance dipole magnet.
BMA	Block move of data on a data bus. Faster than single-shot access.
CAMAC	Computer automated measurement and control. A data bus used frequently in nuclear physics experiments. Being replaced by the VME bus.
CERN	The acronym for it's planning committee, Conseil Européen pour la Recherche Nucléaire, has stuck as the name for the world's largest particle physics laboratory, located outside Geneva, Switzerland.
CFD	Constant fraction discriminator. Converts an analog pulse into a logical signal, in such a way that the timing of the logical signal is independent of the analog pulse height, i.e. at a constant fraction of the pulse height.

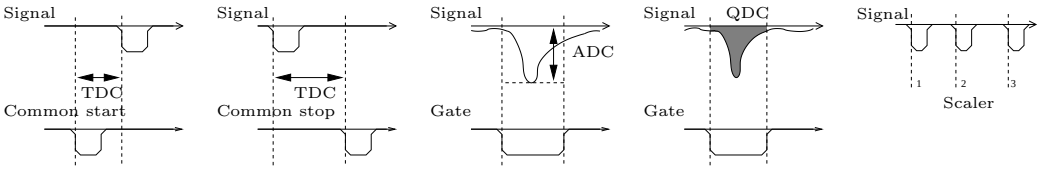

Figure 7.1: Basic DAM modules. From left to right, two TDCs operating in common start and common stop mode, measuring time interval between logical signals. An ADC measuring the highest pulse amplitude (negative signal) during a gate. A QDC integrating the signal during a gate. Finally a scaler counting the number of logical signals.

Channel	The word has two ambiguous usages. a) One detector or DAM channel = a read-out channel. b) The digital channels of one DAM channel (usually 4096), corresponding to the bins of a histogram of that read-out channel. Throughout this thesis, only the first meaning is used.
CVS	Concurrent versions system.
DAM	Digital acquisition module. See Figure 7.1 for examples.
DAQ	Data acquisition. The scoping of this word varies wildly - between only the program controlling the data read-out, and the entire read-out electronics, hardware as well as software.
Dead-time	Time during which another event cannot be accepted, due to one or more parts of the DAQ being busy, usually processing a previous event.
ELISe	ELectron-Ion Scattering in a Storage Ring (eA collider).
Event	The occurrence of each ion passing through a setup (possibly reacting), together with the data recorded from the detectors, is an event.
FASTBUS	A databus standard. Thought to be successor of CAMAC, but never really took off.
FIFO	First-in first-out. A queueing principle.
FRS	Fragment separator, magnetic spectrometer, located behind the SIS.
Gate	Signal during which the inputs of a DAM module are open. Usually derived from the common start/stop signal.
GFI	Große Fiberdetektor. Plastic scintillating fibre detector used for position measurement at the LAND setup.
GSI	Gesellschaft für Schwerionenforschung.
HV	High voltage.
ISOL	Isotope Separator On-Line. Method for producing exotic isotopes by bombarding a heavy target, thereby fragmenting it and then extracting the isotopes by diffusion into an ion source. The low-energy beams of radioactive nuclei can be mass separated by a magnetic spectrometer.
ISOLDE	ISOL facility located at CERN, *on* the Swiss-French border, with the PS-booster as bombarding proton source for the exotic isotope production.
LAND	Large Area Neutron Detector.

MWPC Multi-wire proportional chamber. Detector where a gas is ionised by passing charged particles. The charges are drawn and amplified via repeated collisions by an electric field toward a grid of wires serving both as signal collectors and high voltage electrodes.

NIM Nuclear Instrument Module. A mechanical and electrical standard for the construction of experimental electronics.

Pile-up Effect when a detector or other electronic component receives stimuli (hits or signals) so often that the output no longer is proportional to each input, but becomes a superposition (linear or non-linear).

PM tube Photo-multiplier tube. Detects photons (usually produced in a scintillator or crystals) by converting them into electrons and amplifies the electrical output signal by multiplying the electrons.

QDC Charge-to-digital converter. Integrates and measures the current of detector signals during a gate. Commonly used for signals from PM tubes.

REX-ISOLDE Radioactive Beam EXperiment at ISOLDE. Post-accelerates exotic isotopes after production and separation.

Scaler Counts the number of pulses. Used to see how often a detector channel or other signal fires.

Scintillator Organic or inorganic compound which has a high probability of emitting photons after being excited by a passing ion. The chemically nasty organic liquid scintillators are often embedded in a transparent plastic, making them easy to handle.

SIS Schwerionensynchrotron. Accelerator at GSI.

Slow control Computer assisted adjustment and recording of the parameters controlling an experimental setup. Replacement for potentiometers and screw-drivers.

TDC Time-to-digital converter. Measures the time between a common start or stop signal and the individual detector signals.

TFW Time of flight wall, a large segmented plastic scintillator detector. Part of the LAND setup.

TOF Time-of-flight. Used to determine the velocity of ions.

Trigger logic Part of the experimental electronics that based on fast coincidences from the detectors decide when an event has happened, and if it should be recorded. It is also assuring the dead-time blocking.

VME VERSA module eurocard, a databus, commonly used in industry for computing and control applications.

Acknowledgements

This work would not have been possible without the help, support and encouragement from numerous people.

Haik Simon, everything! Letting me share your office and sharing your knowledge. Help with everyday stuff. All the late evening discussions. This would have been impossible without your patience.

Björn Jonson, introducing me to the subject, arranging for my participation in several experiments at interesting places abroad. Not hesitating as the work turned in an unexpected direction.

Thomas Nilsson, pushing the work forward. Believing in the developments, being a first daring user of the code.

Hans Emling and Tom Aumann, letting me play around with the equipment, to learn how to use it.

My fellow PhD students, accepting my shrewed humour and coping with the numerous changes done to the analysis software.

Anton Lindahl, testing the incoming tracker, and bug-fixing it.

Stefanos Paschalis, showing that the framework can be used and modified, hammering the incoming tracker into use.

Yuliya Aksyutina, taking on the herculean task of calibrating and synchronising the S245 experiment. And not being content with a half-done result - attacking all channels sticking out.

Audrey Chatillon, thoroughly inspecting the time-of-flight wall code, finding and fixing many nasty bugs. Implementing the phase2 calibration routine for the same. Not forgetting our discussions of minus signs and other lethal details.

Kripamay Mahata, doing something about the long standing GFI reconstruction problem that has plagued generations of PhD students.

Acknowledgements

Tudi Le Bleis, embracing the cable documentation, spending long days of equipment testing.

Adam Klimkiewicz, giving very inspiring talks of LAND results.

Luis Fraile, being very helpful and supportive, making my visits to ISOLDE very enjoyable.

Nikolaus Kurz, inventor and creator of a flexible and powerful DAQ: the MBS. When the LAND group sees a little further, it is because it peeks from the shoulders of a giant!

Leonid Chulkov, always inspiring, and furnishing an exciting introduction.

The fantastic group in Göteborg: Christian, Elisabeth, Göran, Kate, Kristian, Lars, Mikael, Mikhail.

And many more: Chiara, Christine, Gerhard, Hans-Henrik, Heinrich, Henrik, Karsten, Konstanze, Oleg, Olof, Pontus, Sven...

My Parents, giving infinite support, and providing a fixed point.

The biggest thanks - a hug of giant proportions - to my brother Henrik, for never ever letting me down, always believing in me. Not to mention an excellent hotel service on numerous visits to Göteborg.

Håkan Johansson, Halmstad, December 2006

Bibliography

[1] L.V. Chulkov et al, *Three-body correlations in electromagnetic dissociation of Borromean nuclei: The ^6He case*, Nuclear Physics A759 (2005) 23-42.

[2] R. Palit et al, *Exclusive measurement of breakup reactions with the one-neutron halo nucleus ^{11}Be*, Phys. Rev. C 68, 034318 (2003).

[3] C.A. Bertulani and G. Baur, *Relativistic Coulomb collisions and the virtual radiation spectrum*, Nuclear Physics A442 (1985) 739-752.

[4] R.J. Glauber, *Deutron stripping processes at high energies*, Phys. Rev. Vol.99 No.5 (1955) 1515-1516.

[5] http://webcast.rice.edu/speeches/19620912kennedy.html (2006-12-16).

[6] F.M. Marques et al., *Detection of neutron clusters*, Phys. Rev. C 65, 044006 (2002).

[7] H.O.U. Fynbo et al, *New information on ^{12}C states from the decays of ^{12}N and ^{12}B*, Nuclear Physics A718 (2003) 541c-543c.

[8] H.O.U. Fynbo et al, *News on ^{12}C from β-decay studies*, Nuclear Physics A738 (2004) 59-65.

[9] C.Aa. Diget et al, *Properties of the ^{12}C 10 MeV state determined through β-decay*, Nuclear Physics A760 (2005) 3-18.

[10] B. Jonson, *Light dripline nuclei*, Phys. Rep. 389 (2004) 1-59.

[11] S. Müller et al., *$^{187}Re(\gamma,n)$ cross section close to and above the neutron threshold*, Phys. Rev. C 73, 025804 (2006).

[12] William R. Leo, *Techniques for Nuclear and Particle Physics Experiments*, Springer, Berlin; New York, 1994.

[13] T. Aumann, *Reactions with fast radioactive beams of neutron-rich nuclei*, Eur. Phys. J. A 26, (2005) 441-478.

[14] H.G. Essel and N. Kurz, *The general purpose data acquisition system MBS*, IEEE TNS Vol.47 No.2 (2000) 337-339.

[15] Th. Blaich et al, *A large area detector for high-energy neutrons*, Nucl. Instr. and Meth. A 314 (1992) 136-154.

[16] Y. Aksyutina, *Experimental studies using an energetic beam of* ^{8}He, ^{11}Li *and* ^{14}Be, Diploma Thesis, Karazin Kharkiv National University; GSI Darmstadt, 2006.

Appendix A

Time and Energy similarity

The handling of the energy values (amplitudes) in the analysis is immensely simplified by invocation of the logarithm[1]. This is based on two observations:

Firstly, the time values have a common scale, seconds, while the energy deposits have the feature of never being negative.

The digital representation of a time for one channel relative to the common start/stop needs both a slope and an offset to get a physical meaning when comparing it to another channel.

A.1 Reconstruction formulas, $ln(e) \sim t$

Secondly, an illustration of the above also comes when looking at the formulas used for reconstruction the time and energy deposited in a paddle from the times and amplitudes measured by the two PM tubes at the short ends (c.f. Section 1.3.2 and Figure 1.3 therein).

The time is calculated as the arithmetic mean

$$T_{mean} = \frac{1}{2}\left(t_1 + t_2\right), \tag{A.1}$$

since the time recorded in one PM tube is $t_i = T + v \cdot d_i$, where T is the time of the hit, v the velocity of light in the paddle and d_i the distance of the hit to the PM tube. The position dependence disappears when using two PM tubes, such that the light always need to in total travel a distance L, the length of the paddle.

The energy is calculated as the geometric mean

$$E_{mean} = \sqrt{e_1 \cdot e_2}, \tag{A.2}$$

[1]This usage is very similar to the way medieval merchants replaced complicated multiplications by additions of logarithms.

since the amplitude recorded in one PM tube is $e_i = Ee^{-\frac{d_i}{\lambda}}$, where E is the energy deposited in the paddle and λ the attenuation length of light in the paddle[2]. Using logarithms, this becomes

$$\ln E_{mean} = \ln \sqrt{e_1 \cdot e_2} = \frac{1}{2} (\ln e_1 + \ln e_2) . \tag{A.3}$$

Letting tilde mark a logarithmic value ($\tilde{e} = \ln e$), the resemblance to Equation (A.1) is clear

$$\tilde{E}_{mean} = \frac{1}{2} (\tilde{e}_1 + \tilde{e}_2) . \tag{A.4}$$

For the position reconstruction the logarithms are naturally needed when using the amplitude measurements,

$$P = \lambda \tilde{E}_{diff} = \frac{\lambda}{2} (\tilde{e}_1 - \tilde{e}_2) , \tag{A.5}$$

and consequently directly analogous to

$$P = vT_{diff} = \frac{v}{2} (t_1 - t_2) . \tag{A.6}$$

For purists (the author included) it is of course ill-defined to take the logarithm of a value with an unit, like an energy loss. It is implied that all logarithms are taken of the values in question divided by our unit of choice (1 MeV), see Appendix B.2. As long as the same basis is used throughout all computations, it never need to enter them explicitly. Also the base of the logarithms is arbitrary - chosen to be the natural (e), to minimise the computing overhead.

A.2 Determining calibration parameters

The reconstruction from raw (digitised) values to calibrated times, requires two parameters per channel, a slope k and an offset m, as shown by

$$t_{sync} = t_{tcal} + m = k \cdot t_{raw} + m. \tag{A.7}$$

Re-adjusting k necessarily means to also recalculate m, while a modification of m does not affect k. One could of course also think of using the formula

$$t_{sync} = k \cdot (t_{raw} + m') , \tag{A.8}$$

[2]The use of e and E as variable names, and the transcendental number $e = 2.718\ldots$ in the same formula, may lead to some confusion. The use of something unambiguous, like a for amplitude, would however loose the direct connection to the energy loss.

but this makes little sense, since usually k can be determined more easily and before m. Actually, since all times are "floating" relative to each other, their individual offsets m_i must be determined together, and this can only be done when the slopes k_i have been determined such that the $t_{tcal,i}$ values can be compared.

For the energies, the case is seemingly reversed. In the transformation from raw to calibrated values, the two needed parameters - the gain g and the location of the digitised value for zero amplitude p - are applied in the opposite order

$$e_{sync} = e_{adj} \cdot g = (e_{raw} - p) \cdot g. \tag{A.9}$$

But in this case, p can usually be determined directly (already by the DAQ, see Section 3.3) and in units of the raw values, while the determination of the gain requires either information about the passing ions (with the help of other detectors), or at least cooperation of the other channels in the same detector for an internal (relative) gain-matching. The use of logarithms

$$\begin{aligned} \ln e_{sync} &= \ln(e_{raw} - p) + \ln g, \\ \tilde{e}_{sync} &= \ln(e_{raw} - p) + \tilde{g}. \end{aligned} \tag{A.10}$$

make it possible to determine the set of parameters g_i using the same methods as for determining the time offsets m_i, since they both are additive, and each tractable with over-determined linear equation systems, as in Section 4.3.1.

Appendix B

Units and reference points

On the subject of coordinate systems, reference points, their definitions, and units.

In order to simplify the writing and maintenance of the analysis software, it is very helpful to make a few definitions (rules) of how the different values are stored[1], i.e. their units and reference points. Within each routine this may be arbitrary and then not of such gravity, but the information from each function will also be used by some other routine. As an example, all the data from all detectors' **hit** levels is combined by the tracker to produce a **track** level of data, see Section 4.5. It then simplifies the program layout a lot if a common set of principles is used.

What this boils down to, is to create an as smooth interface as possible between on one hand the input data structure (source level) of an algorithm (some certain reconstruction routine between two data levels) and the output data structure (destination level), and on the other hand the two algorithms producing and consuming the data in any data level. The simplest interface is of course when no extra transformation is needed. The reconstruction routines should each preferably work in (whatever is) natural units for it. Most often this can be achieved and at the same time have any surrounding routines also work in natural units without the need for data conversions in-between. When not possible, some compromise must be made. The most "compromised" data items are the energies (or amplitudes), since for some operations, they are best handled as logarithms, while for other, the linear value is needed. The trade-off made for such definitions is usually chosen such that the number of operations is minimised, such that unnecessary $\ln x$, $\exp x$, $\sqrt{x}$, x^2 are avoided.

[1] We are not talking about the actual storing of the floating point variables in bits and bytes on the computer, although this is important for data interchange. This has largely been solved nowadays by the IEEE-754 standard.

For units, it does not matter if a time is measured in ns or s, or a distance in cm or m. For a general program, the immediate suggestion would be to use the SI units! As our analysis program however deals with a specific setup, with somewhat odd scales, it makes sense to rather prefer units such that the quantities stored in the variables (and also their errors) have values around unity. And then rather somewhat larger, than smaller, i.e. to have the orders of magnitude chosen such that the values are within or around $0.1 - 100$. This is important for displaying the data.

For the reference points, they are selected such that not one variable or detector part arbitrarily becomes more special or important than another one, unless it actually is intrinsically special. They should also be defined so as to reduce the total number of calibration parameters. Accordingly, the centre of a detector is most often taken as it's origin.

Since there often may be several good choices for the principles to be decided on, the most satisfying way to choose is to find the "killer argument" for some choice - an argument such that it cannot be overthrown. A few examples are given in the following sections. Most often one does not find or realise the killer argument until one has written a large portion of the program, usually using varying and other definitions, such that some rewriting is necessary in order to conform to the preferred principles. The sooner the time is spent on adapting the already written code, the more work in total is saved, since the maintenance work is greatly reduced!

B.1 Times in ns, global common 0

Times are measured in ns. The typical length for a TOF measurement in the setup is 10 m, at about half the speed of light, giving values around 60 ns. Since the times will be used together at the track level, they also need to have a common (global) reference, i.e. be measured with a common clock.

This common reference is however still arbitrary. It happens to be the master start, but it could as well be randomised. In the LAND setup, the master start timing is usually derived from a scintillation detector close to, upstream the target, but this is not a true mean time of that detector. It is even less a good measurement of the time-on-target, particularly with a mixed beam due to velocity dispersion. By having all detectors' time be represented with a common reference (itself arbitrary), any time difference between two hits directly measures the TOF between them.

It should also be noted that we do not measure the absolute time of passing through a specific detector relative to the master start signal (at some unique location in the electronics chain), but rather a time with a

constant offset to that absolute time. This is since there are unknown (or at least uninteresting) lengths of cables, signal propagation delays in modules and the response time of the detector itself, offsetting the measured time. Important is that these delays are all equal in all events (their jitter of course becomes part of the time resolution).

So the principle becomes to at each data level have all the times using a common offset. And as soon as a common offset has been determined at a later (higher) level, it can also be applied at the lower level, since it there actually is arbitrary. This removes the need to apply a time offset between those levels later.

All detectors use the same time reference. Before the detectors can be synchronised together, each must be internally calibrated. The calibration goes in stages, putting bigger and bigger blocks of equipment on a common reference.

For a paddle, the time measured (by a PM tube) depends on the position where the ion hits the detector due to the light propagation time. This ambiguity is removed by defining the reference time for hits in the centre of the paddle. Other events can of course be used for the calibration, they just need to be corrected for their hit locations.

B.2 Energies in MeV

The amplitudes (energies) do not have the same global reference requirement as the times, since they represent energy losses in individual detectors. They should still be measured in the same units though, such that the tracker need not know too many details of the detectors[2].

Even though the energy measurement from each detector most often is a geometric average from several read-out channels (to remove the position dependence), the gains of the individual read-out channels must be determined. The gains should also be independent of the co-operating signal, e.g. other end of the paddle. The definition is therefore for the gains that the amplitude should be expressed as the energy loss of a particle passing through the centre of the detector. When the hit is not centred, there will be an additional position dependent correction corresponding the the attenuation (for scintillators) or charge-division (for semiconductors), of the signal.

[2]Actually, for the energies, the tracker need more information about the detectors than it need to know for times or positions, since the Bethe-Block calculation to determine charges also must have a velocity (β) correction, cannot be handled by the detectors themselves and must be postponed to the tracker.

B.3 Positions in cm, centre is origin

The tracker uses the hit level data from all detectors. To keep the tracking routines as detector-unaware as possible, it is decided that the hit locations are presented with common units (cm), with the origin at the centre of the detector. The tracking routines know where the detectors are located.

The natural units for describing lengths in detectors are most often determined by some internal segmentation: wire, strip or paddle spacing. These units are convenient to use for the calibration routines, and also the routines reconstructing the dhit and hit level data. The conversion to the cm system[3] is done at the end of the hit level reconstruction.

[3] As in "centimetre".

Appendix C

Cable documentation example

A simplified dissection of the anatomy of the cable documentation is given in Listing C.1. The entries are made in a C-inspired free-text format[1], and looks like a set of functions, one per module.

```
MODULE(MODULE_NAME)
{
  MODEL_AS(BASE);

  LABEL("function");

  LEMO_INPUT(connector);
  LEMO_OUTPUT(connector);
  ECL_INPUT(connector);
}
```

```
MODULE_NAME(location)
{
  SERIAL("serialnumber");
  LABEL("name");

 connector: other_loc/other_connector;
 connector: other_loc/other_connector;
 connector: other_loc(type)/conn;
 connector: "lbl"->"lbl",other_loc/conn;
}
```

Listing C.1: Basic layout of the cable documentation. Each module type that is used must first be declared (left). Each instance (actual piece of hardware) is described individually, along with all the cables connected to it (right).

Each type of module that is used must first be defined with the `MODULE` keyword, and an identifier (`MODULE_NAME`) as argument. The type declarations for commonly used modules are kept in files in a special directory tree, such that they can be used when documenting different experiments. Within the module body is then (in arbitrary order) optionally given a list of module types to inherit characteristics from (`MODEL_AS`), a `LABEL` with a descriptive

[1] One (sufficient) reason being that this way, most C-aware editors (e.g. the excellent emacs) can help indent and colour-code the documentation files.

name of the module. The names of all cable connectors are declared, with both their type (`LEMO_INPUT`, `ECL_INPUT`, etc.) and the `connector` names. Each signal is treated individually, i.e. an 8-fold ECL connector is declared as 8 connectors. When a module has many connectors of the same kind and function, e.g. `in1`, `in2`, `in3`..., they may however be abbreviated using a short-hand notation, `in1_8`. The module type declarations are not shown further in this example.

Each instance of a module is documented much like a function, with the type identifier (`MODULE_NAME`) and the module's location. The location is specified with a rack-crate-slot(-unit) `rcsu`-label, on the form `r5c6s7` or `r6c2s9u2` which uniquely specifies the location of a module. Modules that consist of several identical sections are declared as units, such that the type declarations need only be made once. Within the body of the declaration a `SERIAL` number or name can be declared, as well as a `LABEL` describing the signal produced. The `LABEL` is used to name the physical signals from detectors and is used when the origins of a signal entering a DAM are traced.

Each cable connected as input or output to a module is entered with the name of the `connector`, a colon, and the `other_location` (module) and `other_connector` describing where it goes. The short-hand notation for several connections may also be used here, provided the same number of connectors are specified at both ends of the cable(s). The location is given in the `rcsu` notation. The `type` of the other module may optionally be given, as well as any labels (tape markings) on the cable.

In this manner, all cables are documented twice, once per end of the cable, making cross-checks possible.

C.1 Electronics chain for LAND PM tubes

To give a flavour of what a cable documentation looks like, excerpts from the LAND setup are shown below. They correspond to the hardware depicted in Figure C.1.

```
// Declare two of the LAND PM tubes.

DETECTOR_LAND(rNP1c01s1) // Plane 1, paddle 1, pm 1
{
  LABEL("N01_01_1");
 signal: rNP1c21/in1;
 hv: rNP1c31/out1;
}
```

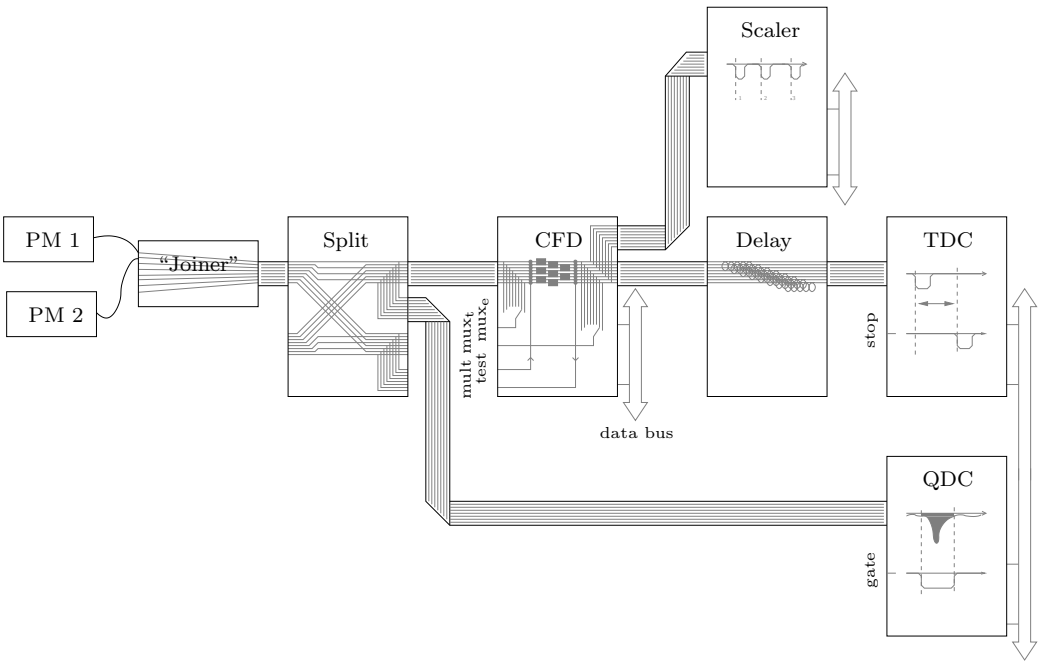

Figure C.1: An electronics chain for two LAND PM tubes, from detector to digitisers. See the listing captions for further explanations.

```
DETECTOR_LAND(rNP1c02s1) // Plane 1, paddle 2, pm 1
{
  LABEL("N01_02_1");
 signal: rNP1c21/in3;
 hv: rNP1c31/out2;
}
```

We started with the declarations of two LAND PM tubes. Also detectors are also declared as modules, to fit the scheme used for all other equipment. (The LAND detector has 398 more PM tubes. The declarations are quite similar from plane to plane, and for the full detector, the 9 rear planes are actually generated from the file describing the first plane.) Each PM tube (base) has one connection for HV supply, and one signal output.

Each place where cables are connected (and signals permutations may happen), such as vacuum flanges, patch panels, are documented. The following "joiner" is a patch panel at the side of the LAND detector where the single signal cables from each PM tube are combined into 8-fold cables going to the electronics racks, about 10 m away.

```
// A connector at the LAND mounting frame combining single cables into
// 8-fold cables (the special 'dis-order' of the channels is to prevent
// cross-talk)

JOINER_8(rNP1c21)
{
 in1: rNP1c01s1/signal;
 in2: rNP1c09s1/signal;
 in3: rNP1c02s1/signal;
 in4: rNP1c10s1/signal;
 in5: rNP1c03s1/signal;
 in6: rNP1c11s1/signal;
 in7: rNP1c04s1/signal;
 in8: rNP1c12s1/signal;

 out1_8: r13c1s1/in1_8;
}
```

The signals are (passively) split into two, one for trigger and time measurement, going to the constant fraction discriminator, the other being delayed (to give time for a possible trigger decision) and then passed into the QDC for charge integration. The split box internally also permutes the signals, such that odd inputs end up in the first 8 outputs, and even inputs in the last 8 outputs.

```
// Split the signal into two, one direct for timing to a CFD, the other
// delayed for integration (amplitude measurement) going to a QDC

SPLIT_BOX(r13c1s1)
{
 in1_8:   "N101" <- , rNP1c21/out1_8;
 in9_16: "N102" <- , rNP1c22/out1_8;

 e1_16:   "N11" -> , r13c2s18/in0_15;

 t1_8:    "N11" -> , r12c2s1/in1_8;
 t9_16:   "N12" -> , r12c2s2/in1_8;
}
```

The delayed signals from several split cards are routed directly to a QDC for charge integration.

```
// A LeCroy 1885 Fastbus QDC

LECROY1885F(r13c2s18)
{
  SERIAL("L_Q46993");

 in0_15:  "N11" <- , r13c1s1/e1_16;
 in16_31: "N12" <- , r13c1s2/e1_16;
 in32_47: "N13" <- , r13c1s3/e1_16;
 in48_63: "N62" <- , r13c3s1/e1_16;
 in64_79: "N71" <- , r13c3s2/e1_16;
 in80_95: "N72" <- , r13c3s3/e1_16;
}
```

The un-delayed analog signals from the split card are discriminated in a CFD. The logical output signals are sent both to a scaler for monitoring and a logic delay module before the time is measured with a TDC. A multiplicity signal for the trigger logic is also produced from the output signals. With programmable multiplexers, it is possible to view both the incoming and outgoing signals from each channel, one at a time. A test input is also connected, and is used to feed in time calibration signals.

```
CF8103(r12c2s1)
{
  SERIAL("LCF6343");

 in1_8: "N11 CFTN1" <- , r13c1s1(SPLIT)/t1_8;
 th1_8: "1/1"       -> , r11c1s1(SCALER)/in0_7;
 tb1_8: "CR2 SL1"   -> , .s15(DELAY)/in1_8;

 m:         .c11s3/in1;
 test:      .s23/out1;
 mux_tb:    .s22/in1a;
 mux_e:     .s22/in5a;
 mux_mon:  .s22/in9a;
}
```

The count rate of individual channels are monitored with scalers.

```
// A 48 channel scaler, for monitoring

SC4800(r11c1s1)
{
  SERIAL("SC6794");

 in0_7:   "1/1" <- , r12c2s1/th1_8;
 in8_15:  "1/2" <- , r12c2s2/th1_8;
 in16_23: "1/3" <- , r12c2s3/th1_8;
 in24_31: "1/4" <- , r12c2s4/th1_8;
 in32_39: "1/5" <- , r12c2s5/th1_8;
 in40_47: "1/6" <- , r12c2s6/th1_8;
}
```

The logical time signals must also be delayed, to make it possible for the trigger logic to reach a decision and supply an "arm" signal to the TDC before any signals of the event arrives.

```
// Delay of timing signal (before going to TDC)

DL1610(r12c2s15)
{
 in1_8:  <- "CR2 SL1" , .s1/tb1_8;
 in9_16: <- "CR2 SL2" , .s2/tb1_8;

 outa1_16:            r11c2s7(TDC)/in0_15;
}
```

The timing of the signals are measured relative to the master start, here coming as a common stop. However, these TDC modules require an "arm" signal before they will accept any input, making the delay requirements as stringent as for common-start operation. Both the "arm" and the stop signals are supplied via the back-plane of the fastbus crate, and are not explicitly documented.

```
// A LeCroy 1875 Fastbus TDC

LECROY1875(r11c2s7)
{
  SERIAL("L_T48834");

 in0_15:  r12c2s15/outa1_16;
```

```
 in16_31: r12c3s15/outa1_16;
 in32_47: r12c4s15/outa1_16;
  ;

  // The common stop and "arm" signals come via the back-plane
}
```

C.2 Parsing, Checking and Output Generation

The processing of the cable documentation is done in several steps:

1. The input file is passed through the `C` preprocessor, with all it's features. This removes all `C`-style comments. File inclusion via the normal `#include` directives, e.g. a "standard" library of module declarations, is also done. File and line number information, necessary for user-friendly error messages are generated. Parts of the documentation can also be made conditional with the `#define` and `#if`/`#ifdef` statements.

   All this comes at no development or maintenance cost - it's part of the system compiler!

2. The output of the previous step is then piped[2] into the actual program responsible for analysing the cable documentation, `ytcon`[3]. The first part of the program, which parses the input stream, is created with the help of a lexer and parser generator from a specification of the syntax of the cable documentation, see Appendix D. Embedded within the syntax specification is the program code necessary to create the data structure elements representing the input. Hence, with very little programming effort, the information is now transformed into a "native" format for the analysing program, see Figure D.1.

3. While the previous step has checked the syntactic correctness of the input, the consistency of the documentation is now assured while setting up the cross-pointers in the data structures representing the cable connections, dashed lines in Figure D.1.

[2] A pipe, |, is a standard operating system construct for passing the output of of process as the input to another.

[3] Short for Y & T connector, the only kind of connectors the program cannot handle - each cable has exactly two ends.

4. The linked data structures can now be "walked" in different ways to deduce various information about the setup, of importance to the DAQ, slow control and analysis. The specific detector channel read out by every DAM is determined by following the cables backwards from every DAM input to the detector signal outputs. This traversal is done backwards since it results in fewer possible ways to continue at each module (all must be tried). The question answered at each stage is: "Which inputs of a module can affect the output that was reached from the previously visited module?" A forward search would have had to answer the question: "Which outputs can be affected by an input?" This would at the locations where the raw trigger signals are generated also have led the searching into the trigger logic, literally exploding the search tree!

   The data paths between each computer controllable module and the corresponding processor or controller are also identified.

5. Read-out code, setup tables and unpack lists are generated for use by the various programs needed to perform an experiment. These are tailor made to suit the formats required by each program - generating information is much easier than reading it!

Unpack tables generated for the two signals described in the previous Section is shown below. The `ELECLOC` entries associate each serial number with a type of module and it's location (crate and slot). Each signal entry (`SIG_BEAM`) give the name of a signal, and the associated CFD module, HV supply, QDC, and TDC channel. Each module is specified by a serial number and channel index, except the HV, which is given by crate, slot, index[4]. The CFD and HV supply information is used by the slow control, while the QDC and TDC mappings are required by the analysis.

```
ELECLOC(LCF6343, CFD3, 2, 1, 1, OFF);
ELECLOC(L_Q46993, QDC_F, 1, 18, 0, FASTBUS);
ELECLOC(L_T48834, TDC_F, 2, 7, 0, FASTBUS);

SIG_BEAM(N01_01_1, LCF6343, 0, 1,0,0, L_Q46993, 0, L_T48834, 0);
SIG_BEAM(N01_02_1, LCF6343, 4, 1,0,1, L_Q46993, 4, L_T48834, 4);
```

[4]This is a legacy of the old slow control software.

Appendix D

Performing Text File I/O

One of the major recurring problems with almost all software associated with DAQ, slow-control and analysis of experiments (not only nuclear physics) is to get information in and out of the programs. Not making this general problem any easier is the diversity of all the tasks that need to be performed. Most tasks as such are fairly small, but they are all in one way or another interconnected. And the result from each task is used as input for other tasks.

Most of the information (calibration parameters) is generated by programs, but may for testing purposes be hand-crafted. It is also useful to be able to tweak some program-generated parameters to see their effect.

A substantial part of the (setup) information must however directly or indirectly be provided by the user. This is control parameters for the DAQ: what modules to read, in what mode to read them... Also similar information for the slow control: what equipment is connected where, the mapping between instances of hardware and detector channels, limiting and default values for the parameters that can be controlled... Finally the information needed by the analysis tool to unpack and map the data from the DAQ into physical detector channels. Either this information is provided directly and individually to the programs needing it, or it is provided from a common source (e.g. a cable documentation, see Section 3.2.2 and Appendix C), it must be easy to inspect and modify.

The obstacle to overcome is that the representations of data that are good for long term storage and user interaction, are not suitable for direct use (bit-for-bit, byte-for-byte) by a program operating on the data. For text files this is obvious, with e.g. 3.14159 being internally represented as `0x40490fd0`. Likewise, a rather loose text format is quite different from the rigid cross-pointed data structures used inside a computer program to represent the information - compare Listing C.1 and Figure D.1.

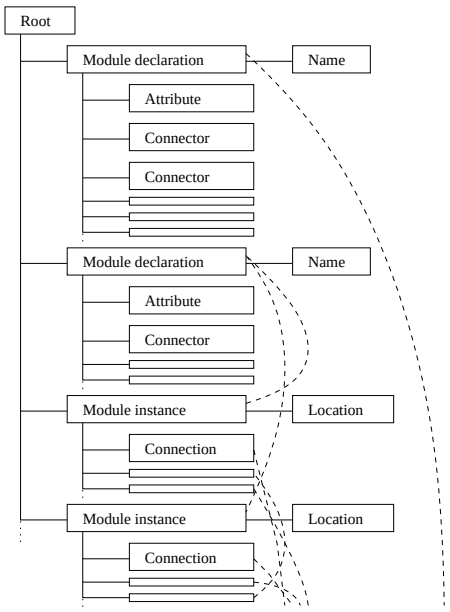

Figure D.1: A parse tree. This represents the contents of a cable documentation. Listing C.1 describes the textual representation used outside the computer program. The full lines are pointers made while parsing, while the dashed lines are created after parsing is complete, when the full information is available.

Binary storage formats do not fare much better. Pointers can not be stored verbatim, since their values change between invocations of the programs. And the more rigid format has drawbacks when new data members are introduced in that precise version information must be handled at a fine granularity to cope with changes done as the software develops.

With a parser generator, the creation of the code that will read an input file, syntax check it, and create the instances of the dynamic data structures representing the input is immensely simplified and also formalised.

D.1 Parser Generators: `lex` and `yacc`, `flex` and `bison`

The conversion of the input data, can be thought of as a kind of Fourier Transform. Here, instead of producing the representation of a wave-function in momentum space from the description in coordinate space, the lexical analyser and the parser manages the conversion of the information from a

readable input format into program-friendly data structures. The equivalent of the inverse transform is to, for each item in the data structures, write their contents. The inverse transform is much easier to perform, almost trivial, and not discussed further.

Although the "lexer" (lexical analyser, shown first) and the parser (shown thereafter) work in lock-step, i.e. the parser requesting one token at a time from the lexer, the interpretation of the input can be viewed as a two-stage operation. Each of the two stages is described by a specification of it's input. From these description (source) files, the appropriate code (e.g. in `C`) is generated by two programs, customarily called `lex` and `yacc`[1].

The lexical analyser combine the individual characters in the input to tokens, as seen in the example below. The tokens are recognised by *regular expressions* (patterns describing the combination of characters that can make up a particular token). For each token found, a small snippet of user-supplied code is executed, converting the characters into a suitable representation (e.g. an integer, an object, or just the copied string itself). The code fragment is made part of the lexical analyser by the generator (`lex`). The representation, along with an identifier specifying the type of item found is passed on to the parser. For keywords, it may be enough to specify the keyword identifier as such as a token, as that carries all the information.

```
// Parts of the lexical analyser for cable documentation files.

/* Recognise a number. */
[0-9]+     {
             yylval.iValue = atoi(yytext);
             return INTEGER;
           }
/* A rack-crate-slot-unit specifier. */
r([0-9]+|[_A-Z][_A-Z0-9]*)c[0-9]+(s[0-9]+(u[0-9]+)?)? {
             yylval.rcsuValue = create_rcsu(yytext);
             return RCSU;
           }
/* A rack-crate-slot-unit specifier, with range. */
r([0-9]+|[_A-Z][_A-Z0-9]*)c[0-9]+(s[0-9]+(u[0-9]+)?)?_[0-9]+ {
             yylval.rcsuRangeValue = create_rcsu_range(yytext);
             return RCSU_RANGE;
           }
/* Keywords. */
"CRATE"    { return CRATE; }
"MODULE"   { return MODULE; }
"UNIT"     { return UNIT; }
```

[1] `flex` and `bison` are the `GNU` versions of these fantastic programs.

```
"SETTING"  { return SETTING; }
"LEMO_INPUT" { return LEMO_INPUT; }
"LEMO_OUTPUT" { return LEMO_OUTPUT; }
"BROKEN"   { return BROKEN; }
"LABEL"    { return LABEL; }
"SERIAL"   { return SERIAL; }
"MODEL_AS" { return MODEL_AS; }
"->"       { return CABLE_TO; }
"<-"       { return CABLE_FROM; }

/* An identifier, i.e. a name. */
[_a-zA-Z][_a-zA-Z0-9]* {
             yylval.strValue = find_str_identifiers(yytext);
             return IDENTIFIER;
           }
/* Simple tokens remain simple, handled by the parser directly
   as part of the syntax. */
[-+*/;(){},:\|] { return *yytext; }
/* A simple quoted string. */
\"[^\"\n]*\" { /* Cannot handle \" inside strings. */
             yylval.strValue =
               find_str_strings(yytext+1,strlen(yytext)-2);
             return STRING;
           }
/* Spaces, tabs and newlines are eaten. */
[ \t]+     { } /* ignore whitespace */
[\n]+      { } /* ignore whitespace */
/* Anything else is an error. */
.          { char str[64];
             sprintf (str,"Unknown character: '%s'.",yytext);
             yyerror(str);
           }
```

The syntactic parser consumes the tokens provided by the lexer. After each token has been received, it attempts to reduce it along with a number of it's predecessors into one token, according to the rules specified. After each successful reduction, it is again attempted to reduce the newly constructed element along with the previous ones. When an entire input is syntactically correct, it will finally reduce to only one node, containing all the information.

The rules defining how tokens can be combined are seen as the last part of the example below. Each time a rule is used, a fragment of code is executed. The fragment is embedded into the parser by the parser generator (`yacc`). The fragment code has access to all tokens that are to be combined via the `$n` variables, and the output is placed into the variable `$$`. A rule would usually execute code which creates a new object representing the resultant

token, and give it the contained objects as arguments. This way, the full tree of data structures that represents the input is created.

The variables holding the (partially) merged tokens are all an instance of an `union`[2], whose layout is declared in the beginning of the parser specification. For complicated types, it is convenient to just store a pointer to an object in the `union` .

Between the union declaration and the reduction rules the mapping between both tokens (as received from the lexer), and nodes (as produced by the rules), and the objects within the `union` is given.

```
// Parts of the grammar for a cabling documentation file.

/* A union data structure which hold the state of each
   token recieved from the lexer, and also of the
   reduced tokens produced by the parser below. This
   is the data structure that each variable below with
   the name $n, and $$ has. */
%union {
  // What we get from the lexer:
  int       iValue;                 /* integer value */
  rcsu      rcsuValue;              /* rack,crate,slot,unit */
  rcsu_range rcsuRangeValue;        /* rack,crate,slot,unit range */
  const char *strValue;             /* string */

  // We generate internally:
  cabling_node       *nPtr;         /* node pointer */
  cabling_node_list  *nodes;        /* list of nodes */

  hardware_def       *hw_def;       /* hardware definition */
  hardware_attr      *hw_attr;
  hardware_attr_list *hw_attr_list;

  hw_module          *module;       /* hardware instance */
  hw_module_part_list *md_part_list;
  hw_module_part     *md_part;
  hw_connection      *md_connection;

  md_ident_fl        *md_idfl;

  std::vector<const char*> *string_list;
};
```

[2] An `union` is a data structure which can hold several different kinds of items, but only one at any point in time.

```
/* Things we get from the lexer */
%token <iValue>  INTEGER
%token <rcsuValue> RCSU
%token <rcsuValue> PARTIAL_RCSU
%token <rcsuRangeValue> RCSU_RANGE
%token <strValue> STRING
%token <strValue> IDENTIFIER
%token CRATE
%token MODULE
%token UNIT
%token LEMO_INPUT
%token LEMO_OUTPUT
%token BROKEN
%token LABEL
%token SERIAL

/* The statements (nodes) */
%type <nPtr> stmt
%type <nodes> stmt_list

/* Compounds */
%type <hw_def> hardware_definition
%type <hw_attr_list> hardware_attr_list
%type <hw_attr> hardware_attr
%type <hw_attr> connector_input
%type <hw_attr> connector_output
%type <hw_def> hardware_decl
%type <hw_def> hardware_decl_crate
%type <hw_def> hardware_decl_module
%type <hw_def> hardware_decl_unit

%type <module>      module_definition
%type <module>      module_decl
%type <md_part_list> module_stmt_list
%type <md_part>     module_stmt
%type <md_part>     module_setting
%type <md_part>     module_connection
%type <md_connection> md_cnct_other
%type <md_idfl>     md_ident
%type <rcsuRangeValue> part_rcsu_range

/* We want verbose error reporting
 */
%token YYERROR_VERBOSE

%%
```

```
/* This specifies the entire input */
program:
          stmt_list                { append_nodes(&all_cable_nodes,$1); }
        | /* NULL */
        ;
/* The input is a list of statements. */
stmt_list:
          stmt                     { $$ = append_node(NULL,$1); }
        | stmt_list stmt           { $$ = append_node($1,$2); }
        ;
/* Each statement is either a module declaration or specification. */
stmt:
          ';'                      { $$ = NULL; }
        | hardware_definition      { append_hardware($1); $$ = NULL; }
        | module_definition        { $$ = $1; }
        ;
/* A module declaration. */
hardware_definition:
          hardware_decl '{' hardware_attr_list '}'
            { $1->apply($3); $$ = $1; }
        ;
/* The header of a module declaration. */
hardware_decl:
          hardware_decl_crate      { $$ = $1; }
        | hardware_decl_module     { $$ = $1; }
        | hardware_decl_unit       { $$ = $1; }
        ;
hardware_decl_crate:
        CRATE '(' IDENTIFIER ')'
          { $$ = new_hardware_def(FILE_LINE,HW_CRATE,$3); }
        ;
hardware_decl_module:
        MODULE '(' IDENTIFIER ')'
          { $$ = new_hardware_def(FILE_LINE,HW_MODULE,$3); }
        ;
hardware_decl_unit:
        UNIT '(' IDENTIFIER ')'
          { $$ = new_hardware_def(FILE_LINE,HW_UNIT,$3); }
        ;
/* A module declaration body can contain several attributes. */
hardware_attr_list:
          hardware_attr                  { $$ = append_node(NULL,$1); }
        | hardware_attr_list hardware_attr { $$ = append_node($1,$2); }
        ;
```

```
/* An attribute of a module. */
hardware_attr:
          connector_input                { $$ = $1; }
        | connector_output               { $$ = $1; }
        | LABEL '(' STRING ')' ';'  { $$ = new ha_label($3); }
        | NAME '(' STRING ')' ';'   { $$ = new ha_name($3); }
        | MODEL_AS '(' IDENTIFIER ')' ';'
          { hardware_def *parent = find_hardware_def($3);
            $$ = new ha_model_as(parent);
          }
        ;
/* A connector specification. */
connector_input:
          LEMO_INPUT '(' IDENTIFIER ')' ';'
            { $$ = new lemo_input($3,0); }
        ;
connector_output:
          LEMO_OUTPUT '(' IDENTIFIER ')' ';'
            { $$ = new lemo_output($3,0); }
        ;
/* A definition of one piece of hardware. */
module_definition:
        module_decl '{' module_stmt_list '}'
          { $1->apply($3); $$ = $1; }
        ;
/* The definition header (with the location). */
module_decl:
        IDENTIFIER '(' RCSU ')'
          {
            hardware_def *model = find_hardware_def($1);
            hw_module *module = new hw_module(model,$3);
            model->_instances.push_back(module);
            module->_loc = FILE_LINE;
            $$ = module;
          }
        ;
/* A definition may have several items. */
module_stmt_list:
          module_stmt                { $$ = append_node(NULL,$1); }
        | module_stmt_list module_stmt { $$ = append_node($1,$2); }
        ;
```

```
/* A definition item. */
module_stmt:
          ';'                          { $$ = NULL; }
        | LABEL '(' STRING ')' ';' { $$ = new hw_label($3); }
        | SERIAL '(' STRING ')' ';' { $$ = new hw_serial($3); }
        | module_connection ';'   { $$ = $1; }
        | IDENTIFIER ':' ';'
            { $$ = NULL; } /* Unconnected cable (ignored). */
        | BROKEN ';'
            { $$ = new hw_broken(); } /* Module is broken. */
        ;
/* A connector. */
module_connection:
          md_ident md_cnct_other { $2->_loc = $1->_loc;
                                   $2->set_cnctr($1->_id);
                                   $$ = $2; delete $1;
                                 }
        | md_ident BROKEN /* Connector is broken. */
          { hw_module_part *part = new hw_conn_broken($1->_id);
            part->_loc = $1->_loc; delete $1; $$ = part;
          }
        ;
/* The name of a connector. */
md_ident:
        IDENTIFIER ':' { $$ = new md_ident_fl($1,FILE_LINE); }
        ;
/* The location of the module a cable goes to,
   and the connectors name on that module. */
md_cnct_other:
        part_rcsu_range '/' IDENTIFIER
          { $$ = new hw_connection($1,$3); }
        ;
/* The location of another module. Some short-hand notations
   are also accepted, and expanded as needed. */
part_rcsu_range:
          RCSU                 { $$ = to_rcsu_range($1); }
        | PARTIAL_RCSU         { $$ = to_rcsu_range($1); }
        | RCSU_RANGE           { $$ = $1; }
        | PARTIAL_RCSU_RANGE { $$ = $1; }
        ;
```

Provided no syntactic errors were found, the data is now ready for further use within the program.

Appendix E

Spill structure and dead-time interference

The peak (or burst[1]) spill structure of the SIS, as opposed to the wanted uniform distribution, effectively leads to a micro-spill structure within the spills as in Figure E.1.

Due to that, secondary isotopes with different A/Z and $B\rho$ and will have different times-of-flight from the production target at the FRS to the experimental target, some ions will arrive earlier within the micro-spill than others. When the dead-time length is comparable to, or longer than the micro-spill length, they will have a larger chance to be accepted by the trigger. For the effect to be noticeable, the time-of-flight dispersion also has to be large compared to the eject period of the micro-spills. Down-scaling of the triggers will lessen the effect, as they are effectively randomised, and the time at the beginning of the micro-spill where the trigger-logic waits for a trigger

[1]Ketchup effect.

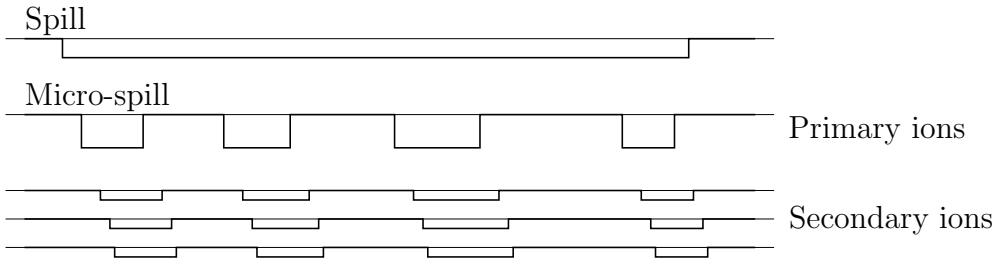

Figure E.1: The micro-spill structure can theoretically be shifted for different isotopes due to their velocity dispersion.

becomes longer[2].

In this manner, the μ-spill may bias which incoming ions are registered, but not how they react. So it will not affect angular correlations etc. It will affect the counting of raw and accepted trigger rates and their relative down-scale between reaction channels, such that cross sections can be affected. The SIS μ-spill is long enough such that the ion dispersion has no effect. Actually, if it should have, then the length would have to be so short, and the ions come so close, that the DAQ would inherently no longer be able to distinguish the events. A real problem is the intervals of the μ-spills, which generally are much longer than the dead-time, causing an inefficient use of the equipment, resembling the situation at REX-ISOLDE.

ISOLDE extraction

The dead-time can together with the spill structure bias the data collection more easily at ISOLDE.

At the low energies available, it is not feasible to detect the incoming ions. Therefore, the possibility of having a coincidence requirement with "good beam", i.e. incoming ion, in the trigger is not available. Hence, it is necessary to perform background subtraction of events collected when no beam is expected. The instantaneous dead-time can vary a lot due to the uneven even rate, see Figure E.2. If the dead-time percentage becomes much larger during the exotic isotope burst than when only background events occur, the relative amount of background events collected when the DAQ is saturated will be smaller, and the background subtraction need to be corrected for the varying efficiency of the data recording. The effective dead-time can be determined by sampling the number of wanted and accepted triggers as a function of time-within-burst, e.g. using a scaler.

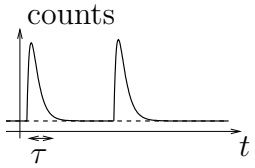

Figure E.2: The trigger rate is the sum of a constant background and events with exotic isotopes occurring after ISOL production as a function of the extraction curve, with a time constant τ mostly depending on the isotope life-time.

[2]Only the minimum bias trigger need to be down-scaled, as the reaction triggers already are reduced by nature (with a thin enough target).

Appendix F

So, what's the problem? → Recommendations!

Maintenance, stability - not just during an experiment, but through the lifetime of an experimental setup.

These suggestions were written in response to the conduction of a small experiment at ISOLDE. They apply generally, also to much larger experiments. In neither case is the use of singularis below meant to imply that the the experiments should be made dependant on single persons. But work must not be diluted to a degree where synergies both within and between experiments cannot be used.

- Someone has to be responsible[1]! Having it floating around just means that much double work and (unintentional) destruction will happen. *A DAQ is* one *interconnected system, and its constituents must be handled consistently.*

- This someone has to have the time for it. Allocated, appreciated, and not spare time!

- Responsible also means to be in control of the situation. The someone also says what goes and what doesn't (or rather: is supported or not). This is the only way he can help. Requirement: users specify what is wanted and needed well in advance (this is on the order of many weeks or months, not days and definitely not hours). Being responsible should not equal night shifts!

[1] "Someone's got to take responsibility if the job's going to get done! You think that's easy?" Capt. Keith Mallory, The Guns of Navarone, Columbia Pictures Corporation, 1961.

So, what's the problem? → Recommendations!

- A DAQ is just as defined by the software as the hardware. One way to formalise software maintenance is to keep track of it using a code revision control system, like CVS. The idea is to provide a well defined location where the current and previous versions of software are located, and not just some directory on some random machine somewhere.
- Along with the previous: Backups! Files without automatic backup procedures should be regarded as non-existing. This also make it formal where production code can be maintained.
- Analysis. It is also efficient if the DAQ-person also take part in the first stages of analysis, e.g. unpacking and ntuple-making, calculations of "static" variables like time from proton, dead-time etc. This since these tasks are recurring for almost every experiment. These are *not* last minute night-shift operations. Rather: users should in advance provide a (possible) very simple simulation so that unpacking and analysis code can be tested beforehand. This also helps setting the experiment up.

All this spells:

- Preparations!

The end of "THE DAQ ALWAYS RUNS"

`land02` will return