Fakultät für Physik und Astronomie

Ruprecht-Karls-Universität Heidelberg

Diplomarbeit
im Studiengang Physik

vorgelegt von
Martin Gerlowski
aus Stuttgart

Januar 1999

Die Slow Control des HERA-B Inner Trackers und Auswertung von Testmessungen mit kompletter Auslesekette

Die Diplomarbeit wurde von Martin Gerlowski ausgeführt am
Physikalischen Institut
unter der Betreuung von Herrn Prof. Franz Eisele

Das Experiment HERA-B wird momentan am DESY aufgebaut und soll die $\mathcal{CP}$-Verletzung im System der neutralen B-Mesonen messen. Teilchen- und Ereignisraten stellen höchste Anforderungen an die Detektorkomponenten.
Das Innere Spurkammersystem (Inner Tracker), bestehend aus Gas-Mikrostreifen-Kammern (GEM-MSGCs), wird seit 1994 von den Universitäten Heidelberg, Siegen und Zürich entwickelt. Der volle Aufbau des Inner Trackers umfaßt 184 MSGCs mit insgesamt über 140.000 Auslesekanälen.
Für den Betrieb eines solch komplexen Detektors ist eine stetige Überwachung von Betriebsparametern notwendig. Diese Aufgabe wird von der sogenannten Slow Control übernommen.
Im Rahmen der vorliegenden Arbeit wurden die fundamentalen Prozesse der Slow Control für den Inner Tracker entwickelt und getestet. Des weiteren wurde eine kollaborationsweit einsetzbare und leicht an die unterschiedlichen Bedürfnisse anpaßbare Graphische Benutzeroberfläche für die Slow Control entwickelt.
Außerdem werden Analyseergebnisse eines Strahl-Tests am Paul-Scherrer-Institut in Villigen/CH präsentiert. Aus den Daten wurden Pulshöhen und Effizienzen in Abhängigkeit von Kammerparametern bestimmt, um die Betriebsspannungen zu optimieren.

The Slow Control of the HERA-B Inner Tracker and analysis of MSGC-data with a full readout chain.

The experiment HERA-B is currently being assembled and commissioned at DESY and will measure the $\mathcal{CP}$ violation in the system of neutral B mesons. It is a challenge to design a detector for the high particle and event rates expected.
The Inner Tracker, consisting of Gas-Microstrip-Chambers (GEM-MSGCs), has been developed at the universities of Heidelberg, Siegen and Zürich since 1994. The whole detector will consist of 184 MSGCs and over 140.000 readout channels.
The operation of this complex detector needs a regular control of its operation parameters. This task is accomplished by the so-called Slow Control.
For the thesis presented here, the most important Slow Control programs for the Inner Tracker have been developed and tested. Furthermore a Graphical User Interface has been written which can be used by the whole collaboration after little modification for special needs.
Furthermore, results from an analysis of data taken at the Paul-Scherrer-Institut in Villigen/CH are shown. The main focus was set on the pulse height spectra and the efficiency as a function of chamber parameters to obtain the optimum settings for chamber voltages.

Inhaltsverzeichnis

Kapitel 1

Einführung

1.1 Aufbau der Arbeit

Die vorliegende Arbeit beschreibt im ersten Kapitel die physikalischen Grundlagen und den Aufbau des Experiments HERA-B, wobei der Schwerpunkt auf der Erklärung des Inneren Spurkammersystems liegt.

Das zweite Kapitel behandelt das Konzept der HERA-B Slow Control, die viele Betriebsparameter des Detektors überwacht. Die im Rahmen dieser Arbeit enstandenen Programme und Konzepte für die Slow Control des Inner Trackers werden erklärt. Eine graphische Benutzer-Schnittstelle, die nach geringen Modifikation für alle Slow Control Programme von HERA-B benutzt werden kann, wird vorgestellt.

Die Analyse von Daten, die während eines Strahltests am Paul-Scherrer-Institut in Villigen/CH genommen wurden, wird beschrieben und Ergebnisse präsentiert. Hierbei wurde besonderer Wert auf die Überprüfung der für den Betrieb wichtigen Kenngrößen wie Pulshöhenspektra, Effizienzen und Pulsbreiten gelegt.

1.2 Die $\mathcal{CP}$-Verletzung

Ein bedeutendes Prinzip in der Physik sind Symmetrien und damit verknüpfte Erhaltungsgrößen.

Für die Elementarteilchenphysik besonders wichtig sind die drei diskreten Symmetrieoperationen $\mathcal{C}$, $\mathcal{P}$ und $\mathcal{T}$. $\mathcal{C}$ ersetzt ein Teilchen durch sein Antiteilchen (Charge Conjugation), $\mathcal{P}$ bewirkt eine Raumspiegelung am Ursprung (Parität) und $\mathcal{T}$ die Zeitumkehr. Die 3 Transformationen sind über das $\mathcal{CPT}$-Theorem verknüpft, das aussagt, daß es keine lokale, kausale und lorentzinvariante Feldtheorie gibt, die die Symmetrie $\mathcal{CPT}$ verletzt.

Seit 1964 weiß man aus Experimenten [Chr64], daß die $\mathcal{CP}$-Symmetrie im K^0-$\bar{K}^0$-System verletzt ist. Die Ursache der $\mathcal{CP}$-Verletzung ist nach wie vor unklar, jedoch gibt das Standardmodell eine Möglichkeit, die $\mathcal{CP}$-Verletztung zu beschreiben. Aus der $\mathcal{CP}$-Verletzung folgt bei Gültigkeit des $\mathcal{CPT}$-Theorems auch $\mathcal{T}$-Verletzung.

1.2.1 CKM-Matrix und $\mathcal{CP}$-Verletzung im Standard Modell

Im Standardmodell wird die Mischung der Massen-Eigenzustände und Eigenzustände der schwachen Wechselwirkung über $W^{\pm}$-Bosonen durch eine unitäre 3×3 Matrix, der Cabbibo-Kobayashi-

Maskawa (CKM) Matrix [Kob73] beschrieben:

$$V = \begin{pmatrix} V_{ud} & V_{us} & V_{ub} \\ V_{cd} & V_{cs} & V_{cb} \\ V_{td} & V_{ts} & V_{tb} \end{pmatrix}. \tag{1.1}$$

Durch die Unitaritätsbedingung wird die Zahl der unabhängigen Parameter der Matrix auf vier reduziert, wobei drei Parameter Mischungswinkel und ein Parameter eine komplexe Phase darstellen.

Für die experimentell beobachteten kleinen Winkel hat sich die Wolfenstein-Parametrisierung [Wol83] mit den reellen Parametern λ, A, ρ und η als Nährerungsform durchgesetzt:

$$V = \begin{pmatrix} 1-\frac{\lambda^2}{2} & \lambda & \lambda^3 A(\rho - i\eta) \\ -\lambda & 1-\frac{\lambda^2}{2} & \lambda^2 A \\ \lambda^3 A(1-\rho-i\eta) & -\lambda^2 A & 1 \end{pmatrix} \tag{1.2}$$

mit $\lambda = \sin\Theta_{Cabbibo}$.

Aus der Unitaritätsbedingung folgt unter anderem

$$V_{ud}V_{ub}^* + V_{cd}V_{cb}^* + V_{td}V_{tb}^* = 0. \tag{1.3}$$

Mit den $V_{ud} \approx 1$, $V_{tb} = 1$ und $V_{cd} = -\lambda$ erhält man die Gleichung

$$V_{ub}^* + V_{td} = \lambda V_{cb}^*, \tag{1.4}$$

die man graphisch als Dreieck darstellen kann. Dieses sogenannte Unitaritätsdreieck liegt in der komplexen Ebene (ρ, η). Nur wenn die Fläche des Dreiecks ungleich Null ist, also $\eta \neq 0$, ist $\mathcal{CP}$-Verletzung im Standardmodell möglich.

1.2.2 Messung der $\mathcal{CP}$-Verletzung

Bis 1957 nahm man an, daß die mikroskopischen Naturgesetze unter den Symmetrieoperatioenen $\mathcal{C}, \mathcal{P}$ und $\mathcal{T}$ invariant sind. 1957 konnte die Paritätsverletzung bei der schwachen Wechselwirkung nachgewiesen werden [Wu57]. Dagegen sind die Eigenzustände der schwachen Wechselwirkung in sehr guter Näherung Eigenzustände zu $\mathcal{CP}$.

Im Jahr 1964 wurde im System der neutralen Kaonen eine kleine Verletzung der $\mathcal{CP}$-Symmetrie nachgewiesen. Man machte die Beobachtung, daß die langlebigen K_L^0-Mesonen sowohl in drei Pionen als auch in zwei Pionen zerfallen können. Der $\mathcal{CP}$-Eigenwert des drei Pionen Endzustandes ist -1, der des Zwei-Pionen-Zerfalls allerdings $CP = +1$. Dies kann mit einer Mischung von $\mathcal{CP}$-Zuständen erkärt werden, wobei der Effekt ist sehr klein ist. Die Zerfallswahrscheinlichkeit für einen Zwei-Pionen-Zerfall ist nur $2 \cdot 10^{-3}$, der Drei-Pionen-Zerfall hingegen 0.3.

Auch im System der neutralen B-Mesonen, B^0 und $\bar{B}^0$, erwartet man analog zu den Kaonen Mixing und $\mathcal{CP}$-Verletzung, allerdings sollten die Effekte erheblich besser beobachtbar sein.

Das Experiment HERA-B hat das Hauptziel, die $\mathcal{CP}$-Verletzung im B-System zu vermessen. Dafür soll die Asymmetrie

$$A = \frac{N(B^0 \to f) - N(\bar{B}^0 \to f)}{N(B^0 \to f) + N(\bar{B}^0 \to f)} \tag{1.5}$$

gemessen werden, die vergleicht, wie häufig ein B^0 bzw ein $\bar{B}^0$ in den gleichen $\mathcal{CP}$ Eigenzustand f zerfällt. Bei Erhaltung von $\mathcal{CP}$ zerfallen B^0 und $\bar{B}^0$ mit gleicher Häufigkeit in den Endzustand f.

HERA-B konzentriert sich auf den sogenannten "Goldenen" Zerfall in den $\mathcal{CP}$-Eigenzustand $f = J/\Psi K_s$ (Abb. 1.1). Dieser Zerfall eignet sich besonders für den Nachweis der $\mathcal{CP}$-Verletzung, da bei dem Zerfall nur eine hadronische Amplitude auftritt, die aus der Asymmetrie A herausfällt. Die beobachtete Asymmetrie A ist direkt proportional zu dem Sinus des Winkels β im Unitaritätsdreieck. Es gilt:

$$A \propto \sin 2\beta. \tag{1.6}$$

Für die Messung der Asymmetrie, also das Abzählen der entstandenen B^0 bzw. $\bar{B}^0$, muß bekannt sein, ob ursprünglich ein B^0 oder $\bar{B}^0$ produziert worden ist. Da bei der Erzeugung der b-Quarks durch die starke Wechselwirkung immer ein $b\bar{b}$-Paar entsteht, ist es möglich durch die Analyse des zweiten Zerfalls herauszufinden, ob im ersten Zerfall ein B^0 mit einem $\bar{b}$ oder $\bar{B}^0$ mit einem b entstanden ist. Bei HERA-B werden hierzu die semileptonischen Kanäle benutzt, bei denen die Ladung des Leptons zwischen B^0 und $\bar{B}^0$ unterscheidet.

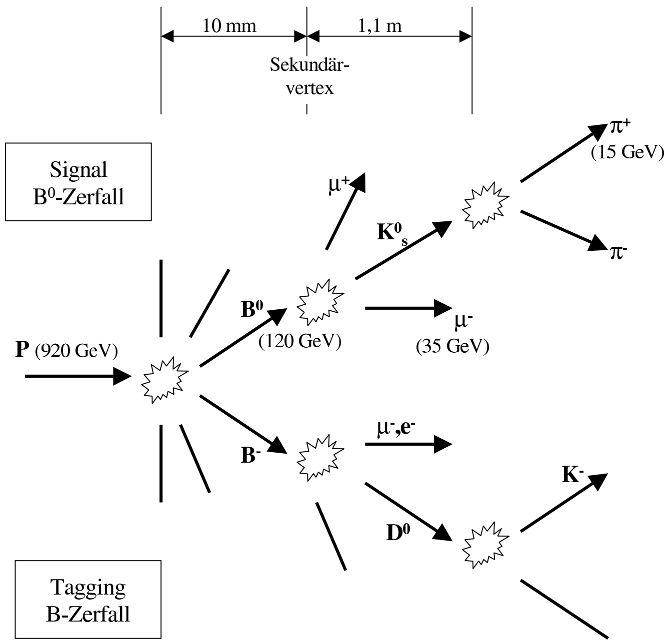


Abbildung 1.1: *Ein "Goldener" B^0 Zerfall $B^0 \to J/\Psi K_s \to \mu\mu\pi\pi$*

1.3 Der Speicherring HERA

HERA[1] ist ein Speicherring am Deutschen Elektronensynchroton (DESY) in Hamburg, in dem Protonen auf eine Energie von 920 GeV und Elektronen auf 27.5 GeV in gegensätzlicher Richtung beschleunigt werden. HERA ist bislang der einzige Elektron-Hadron-Collider. Der Umfang des Speicherringes beträgt 6,3 km. Bisher werden drei der vier "Interaction Zones" von den Exerimenten "H1", "ZEUS" und "HERMES" benutzt.

Als viertes Experiment befindet sich "HERA-B" momentan in der Halle West im Aufbau.

[1]Hadron-Elektron Ring Anlage

σ_{inel}	13 mb
$\sigma_{b\bar{b}}$	20 nb
$\sigma_{b\bar{b}}/\sigma_{inel}$	1.5 $\cdot 10^{-6}$
Verzweigungsverhältnis $B^0 \rightarrow J/\Psi K_s$	4.2$\cdot 10^{-6}$
Nachweiswahrscheinlichkeit eines $B^0 \rightarrow J/\Psi K_s$ Zerfalls aus einer pN-Kollision	6.3 $\cdot 10^{-12}$

Tabelle 1.1: *Für HERA-B interessante Wirkungsquerschnitte und Verzweigungsverhältnisse [Hau98]. $\sigma_{b\bar{b}}$ ist nicht genau bekannt.*

1.4 Das Experiment HERA-B

1.4.1 Anforderungen

Die Messung der $\mathcal{CP}$-Verletzung im B-System stellt in verschiedener Hinsicht eine große Herausforderung für den Detektorbau dar.

Die Wahrscheinlichkeit von 10^{-12} (s. Tabelle 1.1), einen "Goldenen" Zerfall bei einer Wechselwirkung nachzuweisen, erfordert eine hohe Wechselwirkungsrate, um die gewünschte Anzahl von B-Zerfällen rekonstruieren zu können. Der daraus resultierende Teilchenfluß von bis zu $10^7 cm^{-2}s^{-1}$ stellt ein Problem für alle Detektorkomponenten dar.

Der HERA-Speicherring arbeitet mit einer *bunch-crossing* Frequenz von 10 MHz. Um eine höhere Wechselwirkungsrate zu erzielen, muß bei einem *bunch-crossing* mehr als ein Ereignis auftreten. Die Variation der Wechselwirkungsrate geschieht bei HERA-B durch die Änderung des Abstands zwischen Targetdrähten (s. unten) und Strahl. Es wird eine konstante Wechselwirkungsrate von 40 MHz angestrebt.

Um die Flugbahn der entstandenen Teilchen rekonstruieren zu können, ist ein Spurkammersystem notwendig, das sowohl zeitlich als auch räumlich eine sehr gute Auflösung besitzt. Diesen Spuren sollen dann Elektronen, Myonen, Protonen, Kaonen oder Pionen zugeordnet werden können. Neben diesen sich aus der Physik ergebenden Anforderungen sollte der Betrieb von HERA-B möglich sein, ohne daß der Protonen-Strahl des Speicherrings HERA stark verändert wird und die anderen drei Experimente an dem Speicherring gestört werden.

1.4.2 Aufbau des Detektors

Der folgende Aufbau (s. Abb. 1.2) wird den obigen Anforderungen gerecht [HER94] [HER95]:

- Das Target besteht aus acht Drähten, die bis auf einen Abstand von wenigen Millimetern an den Protonenstrahl herangefahren werden können. Auf die Drähte treffen nur die Protonen aus dem Randbereich des Strahls (Halo), die für die anderen Experimente am Speicherring nicht nutzbar sind. Die übrigen Experimente sollen außer durch einen leicht erhöhten Untergrund nicht vom HERA-B Betrieb gestört werden.

- Für die genaue Bestimmung des Wechselwirkungspunktes und der Sekundärvertices wurde ein Siliziumstreifendetektor (Silizium Vertex Detektor, SVD) aufgebaut. Die x-y-Auflösung beträgt $\approx 25 \mu m$. Die z-Auflösung beträgt etwa 500 μm. Dies ist wesentlich kleiner als die durchschnittliche Zerfallslänge der B-Mesonen, die 9 mm beträgt. Der SVD besteht aus 28 Siliziumpanels, die senkrecht zur Strahlachse stehen und bei Betrieb nahe an die Strahlachse gefahren werden können.

- Die Spurrekonstruktion der Sekundärteilchen übernehmen das innere Spurkammersystem (ITR) in einem Bereich von 20 cm um die Strahlachse und das äußere Spurkammersystem

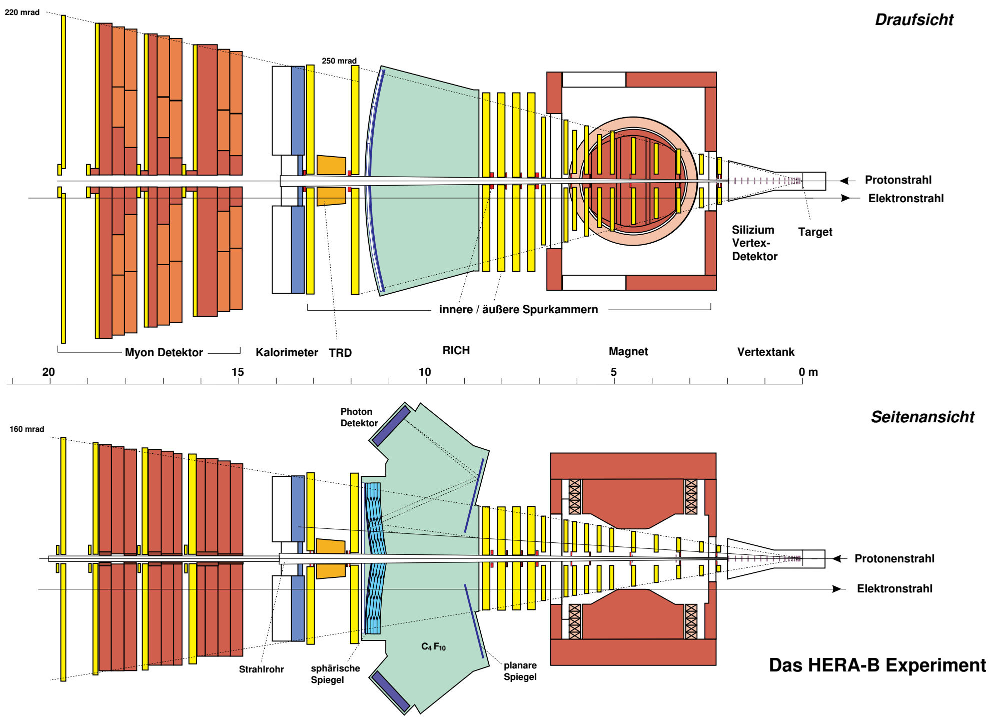


Abbildung 1.2: *Schematischer Aufbau des HERA-B Detektors*

(Outer Tracker, OTR) im Winkelbereich bis zu 220 mrad von der Strahlachse. Gemeinsam können sie Teilchendurchgänge mit einer Genauigkeit von 80 μm in x-y-Richtung auflösen. Der ITR wird im folgenden Kapitel näher beschrieben. Der OTR besteht aus Driftkammern mit sechseckigem Querschnitt, sogenannten Honeycomb-Driftchambers. Die Zellengröße variiert mit dem Abstand vom Strahlrohr von $5mm$ innen bis $10mm$ außen.

- Der Teilchenimpuls wird durch ein magnetisches Spektrometer gemessen. Es setzt sich aus einem Dipolmagneten und dem Spurkammersystem zusammen. Das Magnetfeld lenkt geladene Teilchen in der horizontalen Ebene ab.

- Kaonen und Pionen werden von einem ringabbildenden Čerenkov-Licht Zähler (RICH, Ring Imaging Čerenkov Counter) unterschieden. Er besitzt einen großen Radiatortank, der mit $\approx$100m^3 C_4F_{10} gefüllt ist.

- Ein elektromagnetisches Kalorimeter (ECAL) detektiert Elektronen, Positronen und Photonen. Im inneren Bereich besteht das Kalorimeter aus einer Wolfram-Nickel-Eisen Legierung und im äußeren Bereich aus Blei. Der Nachweis der Teilchen erfolgt mit szintillierenden Fibern.

- Der Übergangsstrahlungsdetektor (Transition Radiation Detektor, TRD) vereinfacht die Unterscheidung von Elektronen und Hadronen in Strahlrohrnähe. In dem TRD wechseln sich Schichten mit Übergangsstrahlung erzeugendem Material und Straw-Tube-Lagen ab. Er wird unmittelbar vor dem Kalorimeter aufgebaut.

- Das *High-p_t*-System im Magneten unterstützt den Trigger bei der Suche nach $B^0 \to \pi^+\pi^-$ Zerfällen. Mit seiner Hilfe können Teilchen mit hohem Transversalimpuls identifiziert werden. Er besteht aus Gaspixel- und Straw-Tube-Kammern.

- Die entstehenden Myonen-Paare werden mit einem Myonen-Detektor nachgewiesen. Dieser besteht aus einem dreiteiligen, 850 t schweren Absorber aus Eisen und Stahlbeton, sowie Driftröhren und Gas-Pixel-Kammern zum Nachweis der Teilchen.

1.4.3 Aufbau des Inner Trackers

Die Kammern

Der Inner Tracker wird gemeinsam von den Universitäten Heidelberg, Siegen und Zürich entwickelt. Er muß eine Fläche von 11.5 m^2 abdecken, mit einer Auflösung von $\Delta x \leq 100\mu m$ in horizontaler Richtung und $\Delta y \leq 1mm$ in vertikaler Richtung. Entlang des Strahlrohres sollen noch $\Delta z \leq 3mm$ aufgelöst werden können. Daneben muß der Detektor noch bei Teilchenraten von $10^4/mm^2s$ voll funktionsfähig sein.

Die Gas-Mikrostreifenkammer (Micro Strip Gas Chamber, MSGC) erfüllt diese Voraussetzungen. Sie ist eine Weiterentwicklung der Vieldrahtproportionalkammer. Im Gegensatz zur Drahtkammer werden allerdings auf Glas aufgebrachte Metallstrukturen als Anoden benutzt. Diese Metallstrukturen verursachen keine Probleme durch elektrostatische Anziehung wie bei Drähten. Neben den Anoden sind feldformende Kathoden auf dem Glas aufgebracht. Es können Strukturen mit einer Breite von von 10 μm für Anoden und 170 μm bei Kathoden auf einer Länge von 25 cm hergestellt und betrieben werden. Der Abstand zwischen zwei Anoden (Pitch) ist 300 μm, sowie 60 μm zwischen Anoden und Kathoden (Gap) (s. Abb. 1.3).

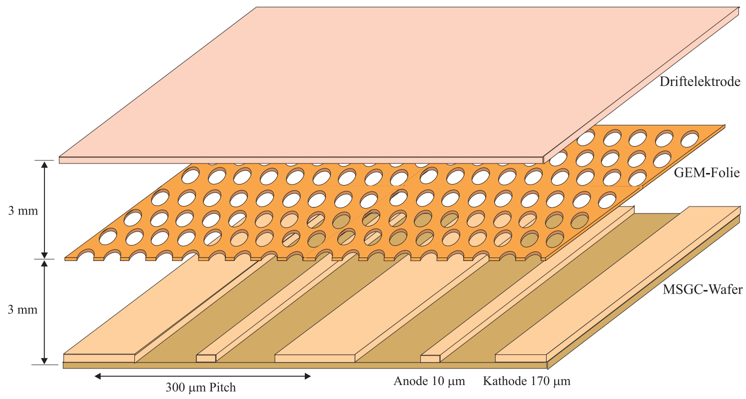


Abbildung 1.3: *Schematischer Aufbau einer GEM-MSGC*

Die Auflösung in der x-Richtung kann somit leicht erreicht werden. Die y-Richtung kann durch die Anordnung mehrerer Kammernlagen aufgelöst werden, die gegeneinander um 5° verdreht sind.

Neben der 25×25 cm^2 großen Glasplatte mit Kathoden und Anoden (dem Wafer) besteht die Kammer aus einem Rahmen, einer Gasverstärkungsfolie (Gas Electron Multiplier, GEM) und einen Deckel als Driftelektrode (s. Abb. 1.3). Auf einem Wafer sind 758 Anoden bzw. Kathoden aufgebracht.

Anfangs wurden Versuche gemacht, allein die Gasverstärkung zwischen Anode und Kathode zu nutzen. Es zeigte sich allerdings, daß es bei den benötigten hohen Spannungen im Hadronenstrahl zu Überschlägen zwischen Anode und Kathode kommt [Vis96]. Diese Überschläge führen zu massiven Schädigungen der Wafer-Struktur.

Nach dem Einbau von GEM-Folien zwischen den Wafer und den Deckel konnte die Kathodenspannung reduziert werden, da in der GEM-Folie eine Vorverstärkung stattfindet. Ein Betrieb von GEM-MSGCs bei HERA-B ist nun möglich.

Der mechanische Aufbau

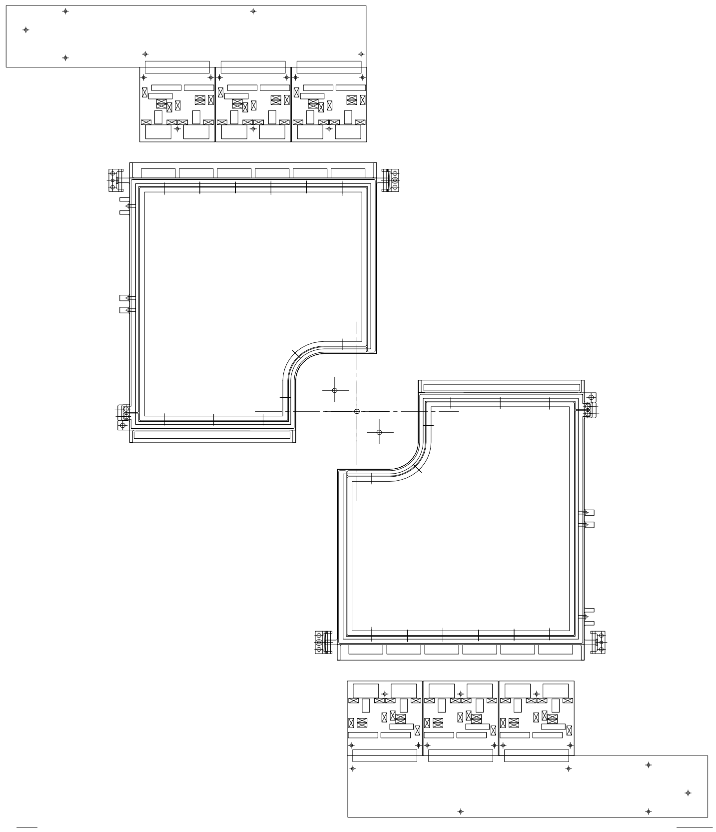


Abbildung 1.4: *Kammer mit PCBs und Signalverteiler Boards*

Jeweils vier Kammern werden um das Strahlrohr angeordnet und bedecken so ein Rechteck um das Strahlrohr. Sie überlappen sich gegenseitig, um tote Bereiche auszuschließen.

In der Abbildung 1.4 sind zwei von vier Kammern einer Ebene mit der Kammerelektronik und den Halterungen sichtbar. Die Kammern werden an drei Punkten fixiert, wobei es sich um einen Drehpunkt, einen Anschlag und einen Druckpunkt handelt. Unterhalb der rechten Kammer sind die drei Helix-PCBs und das damit verbundene Signalverteilerboard gezeichnet.

Auf einem PCB (Printed Circuit Board) befinden sich zwei Helixe, deren Anschlüsse auf dem PCB nach außen geführt werden.

Die Montage erfolgt auf CFK-Platten, in die Halterungen geklebt werden. CFK ist ein Kohlenstoff Verbundwerkstoff, der eine kleine Strahlungslänge aufweist und trotz seines geringen Gewichtes eine hohe mechanische Stabilität besitzt. Die Strahlungslänge gibt an, wieviele Wechselwirkungen auftreten. Da man möglichst ein unverfälschtes Signal messen möchte, versucht man die Strahlungslänge zu minimieren. Beim ITR beträgt sie pro Ebene $\approx 7 \cdot 10^{-3}$. Ein weiterer Vorteil ist, daß sich bei der Bestrahlung von CFK nur Isotope mit geringer Lebensdauer bilden. Somit ist ein Zugang zu dem Detektor schon nach kurzer Abklingzeit möglich.

Die Auslese

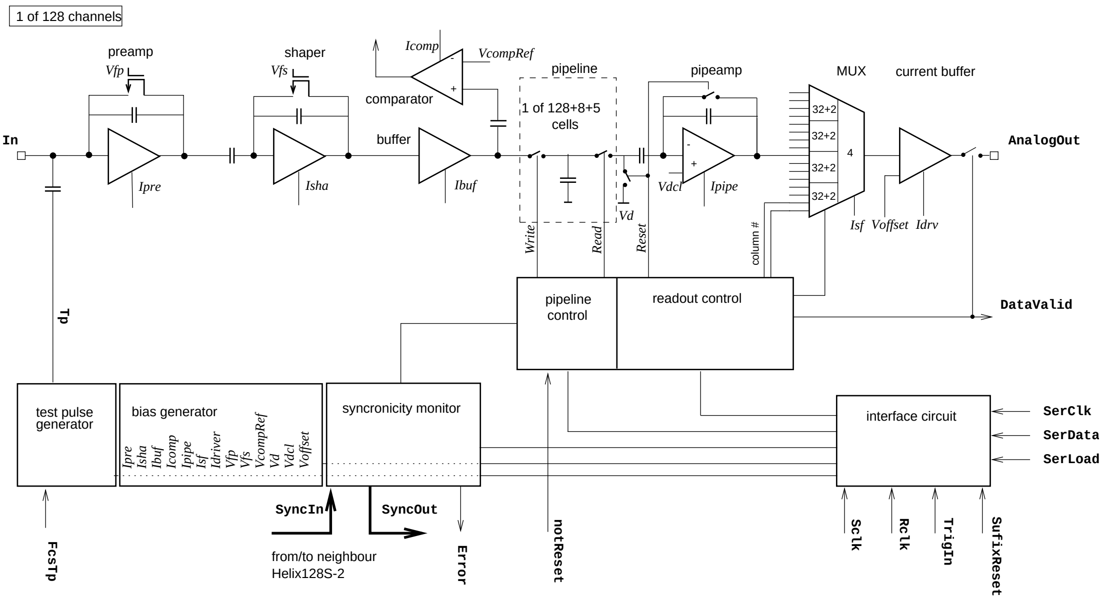


Abbildung 1.5: *Blockschaltbild des Helix-Auslesechips. Es ist nur einer der 128 Auslese-Kanäle sichtbar. (aus [Fal98])*

Die Auslese der Anoden ist bei einer bunch crossing Frequenz von 10 MHz eine weitere Herausforderung. Da erst nach $\approx 10\mu s$ bekannt ist, ob das gemessene Ereignis physikalisch interessant ist, muß die Ausleselektronik auf das Signal des Triggers warten, also die Meßwerte speichern.

Für diese Anforderungen wurde vom Asic-Labor Heidelberg ein anwendungsspezifischer integrierter Schaltkreis (ASIC), der "Helix 128", entwickelt, der 128 Auslesekanäle auf einem Chip integriert. Das Blockschaltbild eines der 128 Kanäle ist in Abbildung 1.5 gezeigt.

Jeder der 128 Kanäle besitzt einen ladungsempfindlichen Vorverstärker (preamp) und einen Shaper (shaper), der dafür sorgt, daß das Signal nach 100 ns wieder auf Null abgefallen ist. Der Helix besitzt einen 141 Ereignisse großen Zwischenspeicher (Pipeline), der aus jeweils einem Kondensator besteht. Auf dem Helix befinden sich 128×141 Zwischenspeicherzellen.

Jedes Analog-Signal wird nach dem Shaper mit einem Schwellwert verglichen (comparator). Die daraus resultierenden Signale von vier Kanälen werden miteinander verodert und sind eine Eingangsinformation für den First-Level-Trigger (FLT).

Vor dem Stromteiber am Ende (current buffer) werden die 128 Kanäle mit einem Multiplexer (MUX) auf einen Ausgangskanal umgesetzt.

Da jeder Chip 128 Eingangskanäle hat, werden 6 Helixe für eine Kammer benötigt, wobei sich jeweils 2 Chips auf einer Platine befinden und gemeinsam ausgelesen werden. Der Abstand der Helixeingänge beträgt 41,4 μm, der Abstand der Anoden jedoch 300 μm. Die Verdichtung der Kanäle erfolgt auf einem Folienkabel aus Kapton und einem Keramik Fan-In, der zusammen mit dem Helix auf eine Platine geklebt wird. In das Fan-In sind Widerstände integriert, die die Eingangsstufe vor Entladungen in der Kammer schützen.

Kapitel 2

Die Slow Control für den Inner Tracker

In Rahmen dieser Arbeit wurden Hardware-Komponenten der Slow Control für den Inner Tracker getestet und Programme für deren Steuerung und Überwachung entwickelt. Daneben wurde an der konzeptionellen Entwicklung eines Slow Control Frameworks für alle HERA-B Subdetektoren mitgearbeitet. Der Schwerpunkt lag dabei auf der Bereitstellung einer graphischen Benutzeroberfläche, die von allen Subdetektoren nach geringen Modifikationen genutzt werden kann.

Im ersten Abschnitt dieses Kapitels wird das allgemeine Konzept der HERA-B Slow Control vorgestellt. Der zweite Teil beschäftigt sich mit der Realisierung der Slow Control beim Inner Tracker, und am Ende des Kapitels werden eigene Programme der Inner Tracker Slow Control erklärt.

Um einen Detektor der Größe und Komplexität wie HERA-B zu betreiben, müssen Parameter wie z.B. Spannungen, Ströme oder Temperaturen einzelner Komponenten automatisch überwacht und gesteuert werden. Diese Funktion erfüllt die Slow Control. Da der Detektor bei normalem Betrieb nicht von den jeweiligen Subdetektor-Experten gesteuert wird, müssen die Slow Control Programme einfach zu bedienen sein und Schäden durch Fehler des Benutzers möglichst ausschließen.

Die von der Slow Control überwachten Parameter wie Spannungen und Temperaturen werden typischerweise wenige Male pro Minute bis einmal pro Sekunde ausgelesen.

Im Gegensatz zu der von dem Fast Control System (FCS) überwachten DAQ (Data Aquisition) ergeben sich nicht Datenraten von der Größenordnung Megabytes/sec, sondern nur Bytes/sec.

Alle Parameter werden in einer Datenbank protokolliert. Bei einer späteren Analyse der Detektordaten sind daher die Betriebsparameter zugänglich und dokumentieren den Detektorzustand.

2.1 Beschreibung des HERA-B Konzepts

Die Slow Control hat die Aufgabe, bei allen Subdetektoren von HERA-B Parameter aus der jeweiligen Hardware zu lesen oder sie dort zu setzen. Die Daten sollen bei einer Änderung des Parameters nach einem einheitlichen Konzept in der Datenbank gespeichert werden. Ferner müssen alle Komponenten des Detektors synchronisierbar sein, um eine zentrale Steuerung zu ermöglichen.

Diese gemeinsamen Anforderungen der Subdetektoren legen es nahe, diese Funktionen in einer Bibliothek zur Verfügung zu stellen. Das reduziert zum einen die Arbeit für die einzelnen Gruppen und verbessert zum anderen durch die einheitliche Realisierung des Konzepts die Wartbarkeit der Software. Auch für den späteren Benutzer der Slow Control Programme während des Schichtbetriebs erleichtert ein einheitliches Konzept und Erscheinungsbild die Einarbeitung in die Detektorsteuerung und erhöht die Benutzerfreundlichkeit.

Da die Basisfunktionalität durch die Bibliothek verfügbar ist, muß nur die Implementierung des konkreten Hardwarezugriffs durch die jeweilige Subdetektor-Gruppe geschehen. Dieses auf die Hardware zugreifende Programm wird bei der HERA-B Slow Control "Parameter Control" genannt.

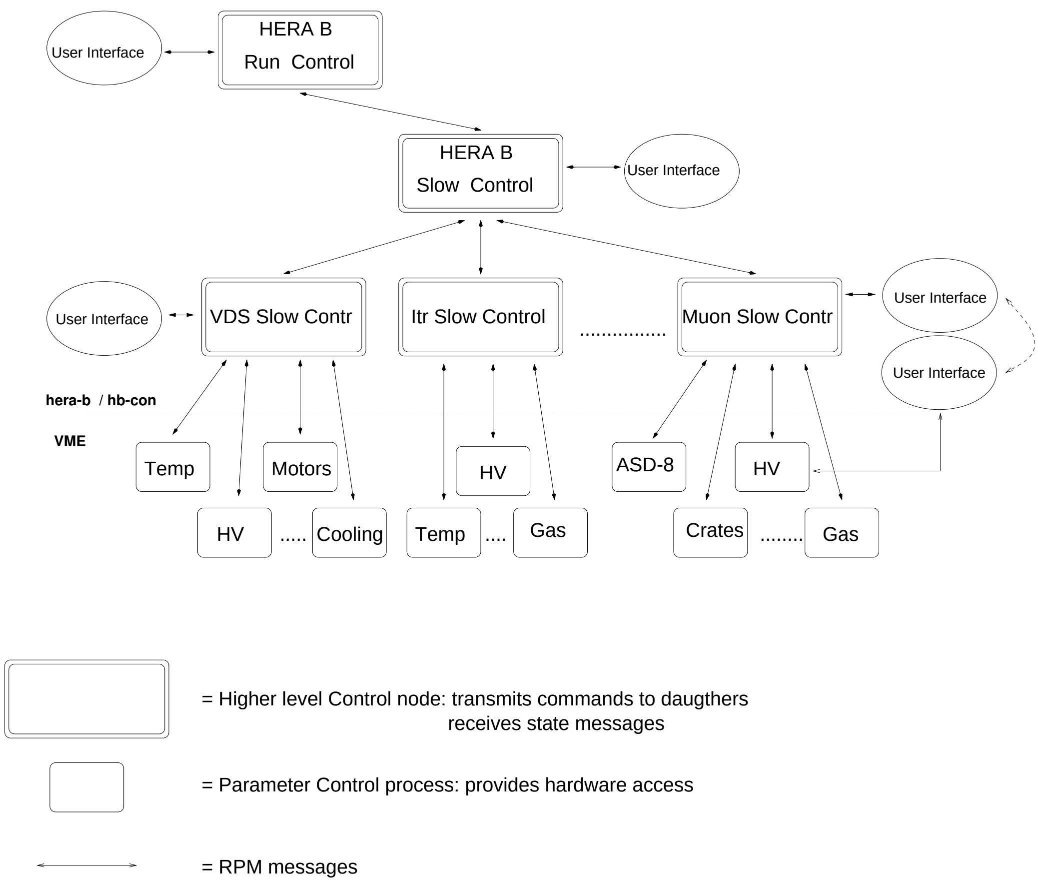


Abbildung 2.1: *Schematischer Aufbau der HERA-B Slow Control*

Auch bei der Hardware wird eine Vereinheitlichung angestrebt. Zum einen kann bei der Sammelbestellung einer größeren Anzahl von Geräten über Preisnachlässe verhandelt werden, zum anderen ist das Expertenwissen über ein Gerät in mehreren Arbeitsgruppen vorhanden. Außerdem müssen Bibliotheken für den Hardware-Zugriff nur einmal entwickelt werden. Gemeinsam eingekauft wurden z.B. die VME-Crates und Hochspannungsversorgung für den Inner und Outer Tracker, RICH und ECAL.

Die Slow Control ist als Baumstruktur aufgebaut (s. Bild 2.1). Der oberste Knoten ist die HERA-B Run Control, die den gesamten Detektor steuert. Von diesem gehen die Befehle an alle darunterliegenden Knoten aus. Die unterste Ebene, die Parameter Control, hat Zugriff auf die Hardware und setzt den Befehl der Run Control konkret für die entsprechende Hardware um. Nach erfolgreicher Ausführung des Befehls meldet die Parameter Control dies über die Baumstruktur an die Run Control. Mit jedem Knoten kann ein User Interface (UI) verbunden werden, das die Steuerung des jeweiligen Knotens und der darunter liegenden Äste übernimmt.

Jeder Knoten dieses Baumes wird durch eine "Finite State Machine" (s. z.B. [Str98]) repräsentiert. Das Konzept basiert darauf, daß sich jedes Programm nur in einem bestimmten Zustand/State (z.B. `INITIALIZE, CONFIGURE, ERROR...`) oder in einem definierten Übergang zwischen zwei Zuständen befinden kann.

2.1.1 Parameter Control

Für jede Komponente (Hochspannung, Strommesser...) existiert ein einzelner Parameter Control Prozeß, welcher eigenständig in die Datenbank schreibt und die Hardware überwacht. Diese Prozesse laufen auf CPUs, die einen direkten Zugriff auf den VME-Bus besitzen, sogenannten VME-CPUs. Die VME-CPUs arbeiten mit dem Echtzeit-Betriebssystem LynxOS [Lyn]. Die zu den anderen Knoten im Baum gehörigen Prozesse laufen auf PCs mit dem Betriebssystem LINUX [Lin].

Eine Parameter Control kann viele physikalische Kanäle überwachen und konfigurieren. Jeder dieser Kanäle hat einen Namen, eine Status und einen Satz von Variablen. Alle Kanäle haben einen einheitlichen und eindeutigen Namen nach dem Schema `detector/system/gruppe/kanal`, also z.B. `itr/hv/sl1/3` [Uwe97b].

Der Status spiegelt den Zustand eines physikalischen Kanals (z.B. Driftspannung der Kammer 4) wider, der State hingegen den Zustand des gesamten Knotens (z.B. Hochspannung Inner Tracker). Jeder Kanal befindet sich immer in einem der folgenden Status: `OK, Warning, Alarm, Error` oder `Unknown`. Eine Übersicht über die States und Status findet man in Tabelle 2.1.

Darüber hinaus besitzt jeder Kanal einen Satz von Monitor- und Konfigurationsparametern (z.B. Strom, Spannung, Temperatur ...). Konfigurationsparameter werden für die Konfiguration der Hardware bzw. Software verwendet und in der Datenbank gespeichert. Sie werden beim Programmstart und einer interaktiven Änderung in der Hardware gesetzt. Jede interaktive Änderung der Parameter durch den Benutzer speichert die Parameter Control in der Datenbank.

Monitorparameter hingegen werden nicht gesetzt, sondern in regelmäßigem Abstand überwacht. Ein typisches Beispiel für einen Konfigurationsparameter ist die Hochfahrgeschwindigkeit der Hochspannung; die aktuelle Spannung ist aber ein Monitorparameter.

Die Datenstruktur für die Monitorparameter besteht aus folgenden Variablen:

- `actual` (tatsächlicher Wert),
- `delta` (Unterschied, bei dem der neue Wert in der Datenbank gespeichert und an das GUI weitergeleitet wird),
- `warningLo/warningHi` (Grenzen, bei denen der Status des Kanals auf `Warning` gesetzt wird),
- `alarmLo/alarmHi` (Grenzen, bei denen der Status des Kanals auf `Alarm` gesetzt wird).

Die Konfigurationsparameterstruktur besteht aus:

- `actual` (tatsächlicher Wert),
- `nominal` (nomineller Wert),
- `dbase` (Wert in der Konfigurationsdatenbank).

In der Hardware wird der nominelle Wert gesetzt, jedoch in der Datenbank nur der `dbase` Wert gespeichert. Der nominelle Wert muß in dem Program gesetzt werden und kann eine Funktion von äußeren Paramtern z.B. der momentanen Temperatur (`temp * dbase = nominal`) oder des States sein (`INITIALIZE: 0.5*dbase = nominal`, `READY: 1*dbase = nominal`).

Wenn sich ein Parameter außerhalb der Warning oder Alarm Grenzen befindet, ändert sich der Status des Kanals zu `Warning` bzw. `Alarm`. Der State ist eine Funktion der Status des Knotens. So kann z.B. im State `READY` noch ein Kanal im Status `Warning` akzeptiert werden, zwei Kanäle im Status `Warning` jedoch nicht mehr. Diese Folgerung der States aus der Summe der Status muß für jede Parameter Control von dem jeweiligen Entwickler definiert werden.

In definierten Notfällen kann die Parameter Control die Geräte abschalten und nach vorgegebenen Algorithmen versuchen, das Problem automatisch zu beheben. Diese Error Recovery muß für jedes Programm definiert werden. Zeigen die Parameter eine Störung der Detektorkomponente an, so meldet die Parameter Control dies über die Baumstruktur an die Run Control und über ein evt. verbundenes User Interface an den Benutzer.

2.1.2 Die Baumstruktur

Die unterste und zentrale Ebene der Slow Control Baumstruktur stellen die Parameter Controls dar.

Untereinander tauschen die Prozesse in dem Slow Control Baum (s. Abbildung 2.1) nur in vertikaler Richtung Informationen auf einem abstrakten Niveau aus. Die vertikal benachbarten Knoten werden nur über den momentanen State und den Status einzelner Kanäle informiert. Detailliertere Informationen für den jeweiligen Prozeß sind nur über ein User Interface zugänglich.

Der oberste Knoten wird durch die HERA-B Run Control repräsentiert. Von dort gehen die Befehle zur Änderung der States aus. Jeder Knoten des Baumes empfängt von oben den Befehl zur Änderung des States und leitet diesen an die darunter liegenden Knoten weiter. Der Knoten merkt sich den von oben kommenden Befehl, ändert aber seinen eigenen State nicht. Erst bei der Bestätigung von unten, daß der neue State erreicht wurde, ändert auch dieser Knoten seinen State.

Die Parameter Control ändert als unterstes Glied des Baums zuerst ihren State und sendet eine Nachricht mit der Bestätigung des Wechsels nach oben in den Baum. Erst dann können die darüber angeordneten Knoten auch ihren State ändern und somit am Ende die Run Control feststellen, daß sich der gesamte Baum im neuen State befindet. So wird gewährleistet, daß ein Knoten stets die Information über alle darunter liegenden Knoten repräsentiert.

Eine Sonderrolle spielt der `ERROR` State. Dies ist der einzige State, in den die Parameter Control eigenständig wechseln kann. Außerdem wird nur dieser State ohne Anfrage nach oben gemeldet. Es ist sofort einsichtig, daß sich ein Knoten im `ERROR` State befinden kann, ohne daß alle darunter liegenden Knoten auch in diesem State sind.

Mit jedem Knoten kann man ein Benutzer-Interface (User Interface, UI) verbinden, das die direkte Steuerung des Knotens ermöglicht. Ohne UI ergibt sich die Steuerung der States durch die hierarchische Struktur im Baum.

Das User Interface wird mit verschiedenen Benutzerrechten ausgestattet: man kann sich sowohl als Master verbinden und die Parameter und States ändern oder als Client und dann nur Parameter überwachen. Für den normalen Schichtbetrieb sind die User Interfaces nur mit Client-Rechten ausgestattet.

State	Status
BRAINDEAD	OK
INITIALIZE	WARNING
CONFIGURE	ALARM
PREPARE	ERROR
READY	UNKNOWN
SAFE	
ERROR	
STOP	
UNKNOWN	

Tabelle 2.1: *Übersicht über die definierten States und Status*

2.2 Beschreibung der HERA-B Slow Control Software Pakete

Für die Speicherung der Daten, die Übertragung von Informationen zwischen den Programmen und die Realisierung der State Machine wurde auf schon früher bei HERA-B entwickelte Programmpakete zurück gegriffen. Diese Pakete werden im folgenden erläutert.

2.2.1 Finite State Machine

Das von Vladimir Rybnikov entwickelte Programmpaket "SMC" [Ryb97] wird bei der gesamten HERA-B DAQ und Slow Control für die Realisierung der Finite State Machine benutzt. Das Paket liegt als C-Bibliothek vor und kann einfach in die jeweiligen Programme eingebunden werden.

Mit der Finite State Machine (FSM) wird gewährleistet, daß die Knoten in dem Slow Control Baum synchronisiert sind, sich also z.B. gleichzeitig im `INITIALIZE` oder `READY` State befinden.

Für die Slow Control ist eine Menge von zulässigen States definiert worden: `BRAINDEAD, INITIALIZE, CONFIGURE, PREPARE, SAFE, READY, STOP` und `ERROR`. Bei jeder Parameter Control kann eine passende Auswahl von States getroffen werden. Für einen Temperatursensor ist der State `SAFE` wahrscheinlich nicht sinnvoll, für ein Hochspannungssystem hingegen sehr wichtig. Zwischen den States sind nur gewisse, in einer Konfigurationsdatei definierte, Übergänge erlaubt (s. Bild 2.2). Die Syntax der Konfigurationsdatei ist in [Ryb97] beschrieben.

Die mit den States verbundenen Aktionen wurden folgendermaßen definiert [Uwe97a]:

- `INITIALIZE:` liest Monitor- und Konfigurationsparameter aus der Datenbank, initialisiert die Software und den VME-Bus-Zugriff bzw. eine andere Schnittstelle für den Hardware Zugriff.
- `CONFIGURE:` lädt Konfigurationsparameter in die Hardware
- `SAFE:` Detektor befindet sich in einem sicheren Zustand
- `PREPARE:` bereitet den Detektor für den Run vor
- `READY:` Detektor ist bereit für die Datennahme
- `ERROR:` die Datennahme/der Detektor ist ernsthaft von den aufgetretenen Problemen betroffen
- `STOP:` beendet das Programm, das den Knoten repräsentiert

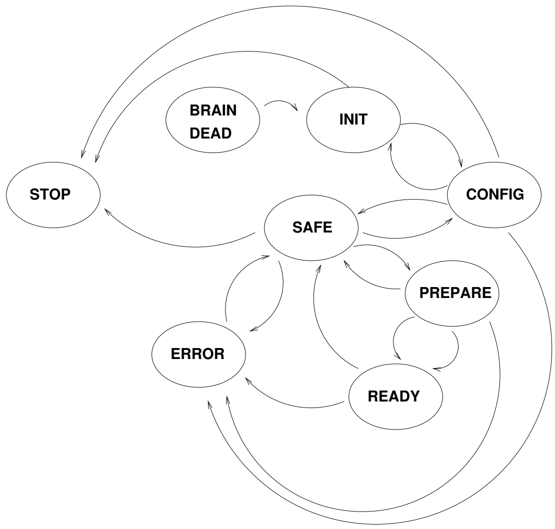


Abbildung 2.2: *Beispiel für State Übergänge der HERA-B Slow Control*

Layer	Title	Purpose	Example
7	Application layer	Services to Users	File transfer, eMail etc.
6	Presentation Layer	Data Representation	ASCII text. Formating the data for the above Layer
5	Session layer	Control of Session(s)	Printing, fils sharing/saving
4	Transport layer	Reliable Packet Delivery	TCP, UDP, SPX, ATP, and ADSP
3	Network layer	Logical addressing, routing	IP, IPX, AppleTalk,
2	Data link layer	Physical addressing, topology (bus or ring). line access method	Ethernet, Token Ring, PPP, HDLC
1	Physical layer	Electrical, optical	10baseT, Fiber Optic, Wireless

Tabelle 2.2: *Das "Open Systems Interconnection" (OSI) Reference Model unterteilt den Transfer einer Nachricht auf verschiedene Schichten (Layer) (nach [Tan96]).*

2.2.2 Message System

Für die Kommunikation zwischen den einzelnen Prozessen auf unterschiedlichen Computern wurde "RPM"(Real Powerful Messaging), ein High Level Nachrichtenprotokoll entwickelt [Zwe98]. Die Anforderung war, ein einheitliches Protokoll zu entwickeln, das auf verschiedenen Plattformen wie Unix und WindowsNT genauso verwendbar ist wie auf betriebssytemlosen Prozessoren, z.B. dem SHARC [Res97], einem digitalem Signalprozessor, aus dem der Second-Level-Buffer aufgebaut wird.

RPM kann mit verschiedenen Netzwerk-Protokollen realisiert werden. Wird IP[1] als Layer 3 (OSI Referenz Modell) benutzt, baut RPM auf UDP[2] auf (s. Tabelle 2.2). Eine Besonderheit von UDP ist, daß Nachrichten, die auf mehrere Pakete verteilt verschickt werden müssen, nicht auf ihre Vollständigkeit überprüft werden. Dies erhöht die Geschwindigkeit der Übertragung, kann aber auch den Verlust von Daten bedeuten. Mittlerweile[3] wird für die Slow Control eine

[1] Internet Protocol: Das Standard Internet Protokoll

[2] User Datagram Protocol: ein Standard Netzwerk Protokoll der TCP/IP-Protokoll Familie

[3] Dezember 1998

RPM-Version benutzt, die solche Verluste ausschließt.

Für die Lokalisation der Prozesse ist ein zentraler Name-Server [Nam] notwendig, welcher auf jedem beliebigen Rechner gestartet werden kann. Jeder Prozeß meldet sich während des Programmstarts bei diesem Nameserver. Die Zuordnung einer Nachricht zu einem gewissen Kanal erfolgt über den Kanalnamen. Es herrscht also eine eindeutige Zuordnung von Kanalnamen zu Programmen.

Da die gesamte Kommunikation für den Benutzer nicht sichtbar ist, soll hier nicht näher auf RPM eingegangen werden.

2.2.3 Die Datenbank

Die "Table Based Data Base" [Kih97b] ist ein von Th. Kihm entwickelter Aufsatz auf die GNU-Database. Diese Datenbank wird in der gesamten Kollaboration eingesetzt.

Die Slow Control verwendet die Datenbank als Speicher für die Konfigurations- und Monitorparameter.

Da die Datenbank auf Tabellen basiert, sind Tabellen (Tables) und Felder (Fields) die Objekte, mit denen in dieser Datenbank gearbeitet wird. Die Datenbank enthält keine relationalen Funktionalitäten, mit denen sich Bezüge zwischen Daten definieren lassen. Mittlerweile gibt es aber Erweiterungen, die auch relationales Datenmanagment ermöglichen.

Die Einträge in der Datenbank enthalten zwei Ordnungsgrößen: den Major- und den Minor-Tick. Mit welchen realen Größen die Ticks gefüllt werden, ist nicht festgelegt.

Bei der Slow Control werden als Major-Tick die Sekunden seit dem 1.1.1970, 0:00 verwendet und als Minor-Tick die μ-Sekunden seit der letzten vollen Sekunde.

Der Datenbankzugriff kann auf zwei Arten erfolgen:

- Die Anwendung öffnet direkt das Datenbankdatei. Durch den direkten Zugriff sind die Lese- und Schreibvorgänge sehr schnell. Auch bei großen Dateien (mehrere Mega-Byte) stellt die Geschwindigkeit kein Problem dar, jedoch kann nur eine Anwendung auf diese Datenbankdatei zugreifen.

- Die andere Möglichkeit ist der Zugriff auf die Datenbank über den Datenbankserver `sdbserver`. Hierbei hat nur der Datenbankserver direkten Zugriff auf das Datenbankfile, alle anderen Anwendung lesen die Daten mittels Datenbankserver über das Netzwerk. Der Datenbankserver kontrolliert die Lese- und Schreibzugriffe und verteilt die Informationen an die Anwendungen. Da der Datenbankserver bei großen Dateien (einige Megabyte) zu langsam ist, werden die Datenbankfiles auf mehrere Dateien verteilt und von einem Hilfsprogramm verwaltet.

Für die Datenbankeinträge der Slow Control wird für jeden Kanal eine Tabelle angelegt[4]. In einer Tabelle ist je eine Spalte für jeden Monitorparameter vorgesehen. Die Tabelleneinträge werden Felder (Fields) genannt. Den Feldern wird ein Datentyp zugewiesen, z.B. ASCII, ganze Zahlen (Integer) oder Fließkomma-Zahlen (Floats). Für die Monitor- und Konfigurationsparameter sind nur ganze Zahlen vorgesehen. Eine typische Tabelle der Slow Control Datenbank wird in der Abbildung 2.3 gezeigt. In der ersten Spalte steht der momentane Status als Text, es folgen Spalten für den Strom und die Anzahl der Warnungen, jeweils als ganze Zahl (Integer), formatiert. Jede dieser Tabellen repräsentiert einen Meßpunkt.

Mit dem Hilfsprogramm `dbedit`[Lef96] kann die Datenbank editiert werden. Mit diesem Programm kann man sich einzelne Tabellen aus der Datenbank anschauen und verändern. Abbildung 2.3 wurde mit `dbedit` erstellt. Das Hilfsprogramm `hbdis` zeichnet den zeitlichen Verlauf

[4] Stand: Oktober 1998

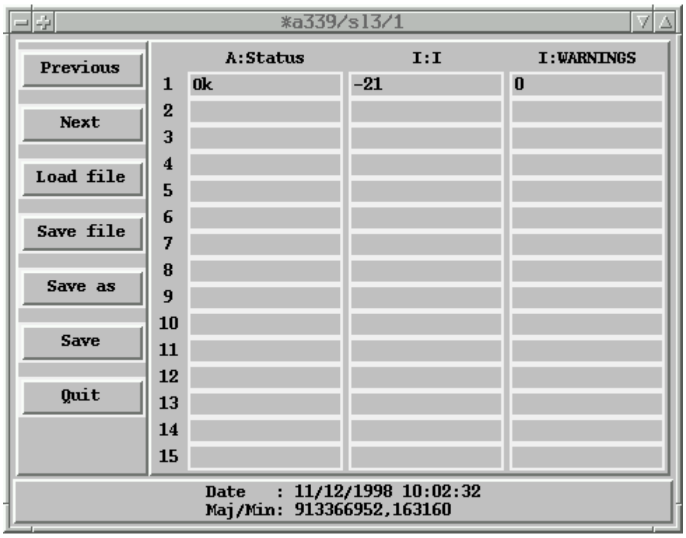


Abbildung 2.3: *Eine Tabelle des Inner Tracker HV Systems in der Slow Control Datenbank. Die erste Spalte beschreibt den momentanen Status, die zweite den momentanen Strom und die dritte die Anzahl der Warnungen, die bisher aufgetreten sind. Das* `A` *vor dem Status bedeutet, daß in diesem Feld ASCII Zeichen, also Text, eingeben werden dürfen, im den beiden anderen Tabellen dürfen nur Daten des Typs* `Integer`*, also ganze Zahlen, eingetragen werden. Am unteren Rand erkennt man das Datum und die Uhrzeit der Tabellenerstellung sowie den Major- und Minor-Tick. Diese Tabelle wurde von dem Programm* `dbedit` *dargestellt, mit dem man Datenbankeinträge lesen und editieren kann.*

der Datenbankeinträge. `hbdis` kann als Online Plotter verwendet werden, zeigt aber auch den Verlauf vorangegangener Runs.

2.2.4 SPC Slow Control Framework

Die Slow Control Bibliothek "SPC" stellt ein Framework zur Erstellung der jeweiligen Slow Control Subdetektoranwendung zur Verfügung. Nur der Zugriff auf die Hardware und die speziellen Anforderungen eines Subdetektors müssen implementiert werden.

Das gesamte Programmpaket wird von "Events" gesteuert, d.h. daß eine definierte Anzahl von Ereignissen auftreten kann und diese Ereignisse definierte Reaktionen des Programms zur Folge haben. Ein mögliches Event ist `READ_CONFIG`, welches das Lesen der Konfigurationsparameter aus der Hardware zur Folge hat, oder `UPDATE_MONITOR`, woraufhin die Monitorparameter aktualisiert werden. Events können aus der Änderung von States folgen oder auch vom Programm selbst erzeugt werden.

Das Graphische User Interface

Für das Graphical User Interface (GUI) wird Tcl/Tk [Ous95], eine Skript-Programmiersprache verwendet. Mit ihr kann in kurzer Zeit eine graphische Benutzeroberfläche erstellt werden, da Elemente wie Knöpfe, Regler und Rahmen Objekte der Programmiersprache sind. Durch ein C-Interface ist die Anbindung an C-Programme möglich. Außerdem ist Tcl/Tk für eine Vielzahl von Plattformen kostenlos verfügbar.

Jedes Tcl/Tk-Skript wird erst bei der Ausführung interpretiert. Der dafür notwendige Interpreter kann selbst erstellt werden, wobei andere Bibliotheken hinzugefügt werden können. Der

Interpreter für das HERA-B Graphical Interface enthält zusätzlich die Bibliothek des RPM-Nachrichtenprotokolls, die der Finite-State Machine und die Anbindung an die Datenbank. Dem Programmierer der Subdetektor-Gruppe stehen in dem Tcl/Tk Skript Funktionen für den Zugriff auf die Parameter Control zur Verfügung. Diese Funktionen werden wie gewöhnliche Tcl/Tk-Funktionen aufgerufen.

Hierbei unterscheidet man zwischen zwei Arten von Funktionen:

- Request Funktionen: Sie senden eine Nachricht an die Parameter Control, die folgende Aktionen der Parameter Control auslösen kann:
  - Ändern von Monitor- oder Konfigurationsparametern
  - Wechsel von States
  - Senden von Monitor- oder Konfigurationsparametern, States oder Status von der Parameter Control an das GUI.
- Receive Funktionen: Diesen Funktionen werden als Parameter bestimmte Informationen der Parameter Control übergeben. In diesen Funktionen können die Daten weiterverarbeitet werden. Eine Receive Funktion kann als Antwort auf eine Request Funktion aufgerufen werden.

Request Funktionen werden von dem Tcl/Tk Skript aufgerufen, z.B. nachdem der Benutzer einen Knopf gedrückt hat. Receive Funktionen werden automatisch von dem Tcl/Tk Interpreter aufgerufen, d.h. man muß sich bei der Programmierung nicht darum kümmern, daß diese Funktionen aufgerufen werden.

Ein Beispiel: Um den Status des Kanals `itr/hv/sl1/1` zu erfahren, ruft man im Tcl/Tk Skript zuerst die Request Funktion `spc_RqChaStatus` mit den Argumenten `"Display␣Status"␣ itr/hv/sl1/1` auf. Dieser Aufruf kann z.B. an das Drücken eines Knopfes im GUI gebunden werden. In der Parameter Control wird daraufhin das entsprechende Event ausgelöst. Als Antwort auf diesen Request wird im Tcl/Tk Skript die Funktion `spc_NewChaStatus` aufgerufen. Als Argumente werden der Kanalname und der Status übergeben.

Request Funktionen können sich auch auf mehr als nur einen Kanal beziehen. Wenn bei dem obigen Beispiel als Kanal nicht `itr/hv/sl/1`, sondern `itr/hv` benutzt worden wäre, hätte man die Informationen über den Status aller Kanäle der Hochspannungsversorgung des ITR erfahren. Als Antwort wäre dann die Funktion `spc_NewChaStatus` nicht nur einmal, sondern mehrmals aufgerufen worden. Jedesmal hätte das Argument der Funktion den Status eines weiteren Kanals beschrieben.

Die Parameter Control

Das SPC Framework enthält ein Gerüst für eine Parameter Control, worin alle allgemein benötigten Funktionen (wie z.B. lese die Parameter aus der Hardware, ändere den Status) deklariert sind. Für den Subdetektor müssen die Funktionen für die jeweilige Hardware und deren Aufgabe gefüllt werden. Daneben ist eine allgemeine Datenstruktur, auch für die Monitor- und Konfigurationsparameter, definiert.

Die User-Funktionen werden bei bestimmten Events aufgerufen, z.B. beim Programmstart `spc_user_start_up`. Andere Funktionen werden bei jedem State-Wechsel aufgerufen, z.B. `spc_user_check_config`. Wie die Konfigurationsparameter aus der Hardware ausgelesen werden, muß jeweils neu definiert werden. Welche Funktionen bei welchen Events aufgerufen werden, ist in [Zhe98b] beschrieben.

In regelmässigen Abständen wird die Funktion `spc_main_loop` aufgerufen[5]. In den gleichen Abständen werden auch in der Funktion `spc_user_read_monitor` die aktuellen Monitorparameter aus der Hardware gelesen und in die globale Datenstruktur `spc` geschrieben. Diese Datenstruktur enthält als wichtige Strukturen `spc_Channel` mit Kanal-Namen, Status, Monitor- und Konfigurationsparameter `spc_Monitor` und `spc_Config`. Die Monitor- bzw. Konfigurationsparameter-Struktur wurde bereits in Abschnitt 2.1.1 beschrieben [Zhe98a].

2.3 Komponenten der ITR Slow Control

2.3.1 Konzept der ITR Hochspannung

Für den Betrieb einer GEM-MSGC werden vier Hochspannungen benötigt: U_{Drift}, $U_{Gem-Oben}$, $U_{Gem-Unten}$ und $U_{Kathode}$. Die Hochspannungen werden in der Elektronikhütte erzeugt und über eine Entfernung von bis zu 50 m zu den MSGC Paketen geführt. Um die Anzahl der Kabel und Hochspannungsnetzgeräte zu reduzieren, wird jede der Drift- bzw. Kathodenspannungen unter der Detektorplattform auf acht parallel geschaltete Ausgänge verteilt. Aus U_{Drift} wird für jede Kammer separat mit einem Spannungsteiler $U_{Gem-Oben}$ und $U_{Gem-Unten}$ erzeugt (s. Abb. 2.5). Der Spannungsverteiler für U_{Drift} und $U_{Kathode}$ fungiert gleichzeitig als Strommesser. So kommt man mit 2 langen Kabeln für acht Kammern aus, anstatt 32 ohne Verteilung.

Für Probleme in einer Kammer sind außergewöhnliche Ströme ein Indiz. Um bei auftretenden Problemen während des Einschaltens schnell reagieren zu können, wird die Spannung in mehreren Schritten erhöht und nach dem Abklingen der Lade- und Induktionsströme der Kathoden- und Driftstrom mit Grenzwerten verglichen. Sind die Ströme innerhalb dieser Grenzen, wird die Spannung weiter erhöht, andernfalls sofort die Hochspannung heruntergefahren.

Während des Normalbetriebs werden die Ströme mit $\approx$1 Hz kontrolliert.

2.3.2 Hochspannungserzeugung

Als Hochspannungsnetzgerät wird das Gerät *SY527* der Firma *CAEN* mit verschiedenen Einschüben eingesetzt.

Diese Geräte der Firma *CAEN* werden auch von den HERA-B Subdetektoren Outer-Tracker, RICH und ECAL benutzt, wodurch die Wartbarkeit und Transparenz auch bei der Bedienung von Experten anderer Subdetektoren erleichtert wird.

Die Ansteuerung erfolgt entweder über die serielle Schnittstelle RS232 oder das *FAST CAEN-NET*. Das *FAST CAEN-NET* kann über eine VME-Karte angesprochen werden.

2.3.3 Strommessung

Der Zustand einer MSGC wird gut durch die Ströme über die Driftanode und die Kathode beschrieben. Der Strom steigt bei Gasverstärkung an und erhöht sich bei Kurzschlüssen in der Kammer extrem. Daher ist eine ständige Kontrolle der Ströme sehr wichtig.

Die Überwachung von I_{Drift} und $I_{Kathode}$ geschieht mit einem 2 x 8 Kanal Strommesser, der Strommeßbox A339[vWa99a]. Die Strommeßbox wurde in der Elektronikwerkstatt des Physikalischen Instituts der Universität Heidelberg entwickelt. Eine dieser Boxen verteilt eine Drift- und eine Kathodenspannung auf jeweils acht Ausgangskanäle und mißt dabei die Ströme.

Der Strommesser verfügt über programmierbare Stromschwellen, bei deren Überschreitung ein Alarm ausgelöst wird. Bei einem Alarm wird durch ein Relais die Verbindung zu dem Hochspannungsgerät unterbrochen und an einem Kontrollausgang die Spannung von 5 V im Normal-

[5] Stand Oktober 1998. Seit Dezember 1998 gibt es eine neue Version des SPC-Frameworks, das die Funktion `spc_main_loop` nicht mehr aufruft.

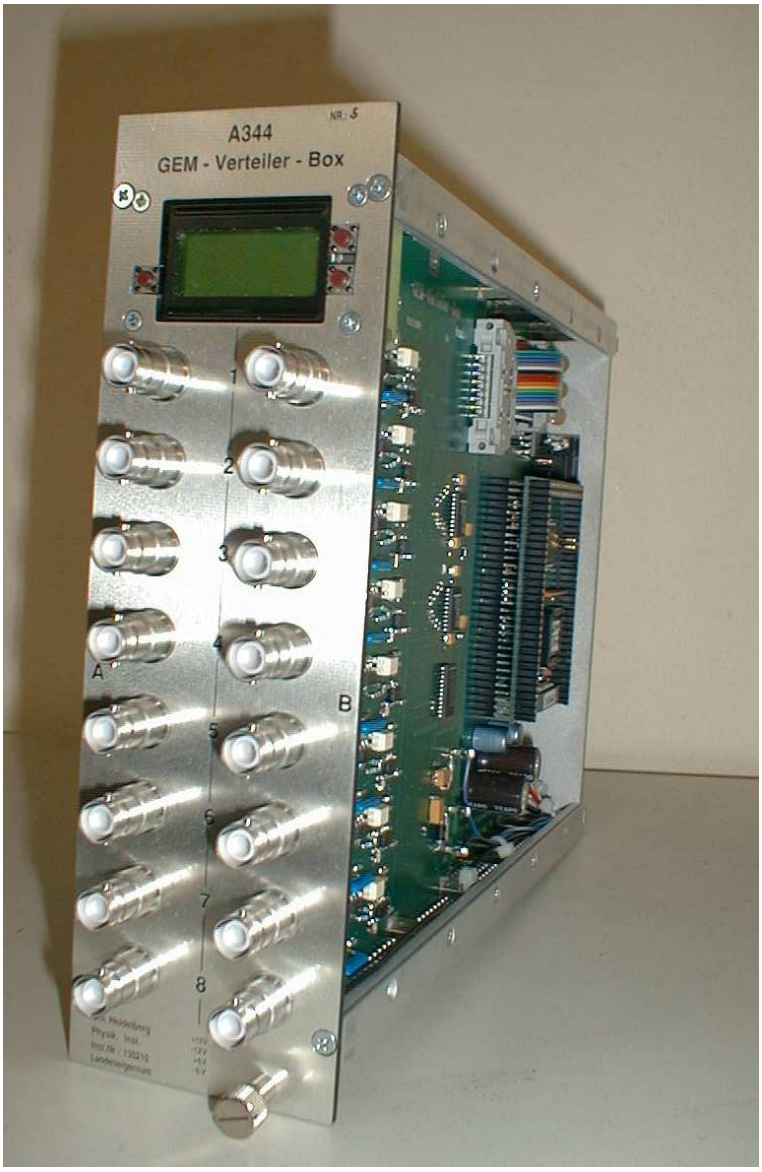


Abbildung 2.4: *Photo einer GEM-Verteiler Box*

zustand auf 0 V im Alarmzustand gesetzt. Die Ladung, die sich noch auf den Kathoden befindet, fließt über die Beschichtung der Glasplatte zur Anode ab. Die Ladung auf der Driftelektrode bzw. auf der GEM-Folie wird über die GEM-Verteilerbox abgeleitet (s. Abb. 2.5).

Der Strom wird aus dem Spannungsabfall über einem bekannten Widerstand berechnet. Um den Strommessbereich zu verändern, wechselt man die Meßwiderstände. Der jeweilige Widerstand muß dem in der Strommessbox laufenden Programm mitgeteilt werden. Mit einem typischen Meßwiderstand von 10 $k\Omega$ kann man Ströme von einige μA messen. Ansteuerung und Programmierung der Strommessers erfolgt entweder über die serielle Schnittstelle RS232 oder den CAN-Feldbus.

Beide Eingangsspannungen werden an je einen Ausgang durchgeschleift, wobei die Ströme bei diesen Spannungen nicht gemessen werden. Im Alarmfall sind diese beiden Spannungen auch von der Abschaltung durch das Relais betroffen. Die Ausgangsspannung von U_{Drift} wird als Eingangsspannung für die GEM Verteiler Box verwendet.

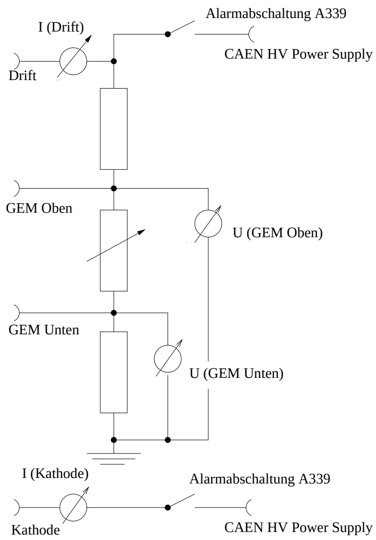


Abbildung 2.5: *Erzeugung der Kammer-Spannungen: Die Ströme der Drift- und Kathodenspannung werden, wie auch die Spannungen an den Ober- und Unterseite der GEM-Folien, protokolliert. Bei einem Alarm der Strommeßbox werden die Drift- und Kathodenspannungen von der Hochspannungsversorgung getrennt.*

2.3.4 GEM Verteiler Boxen

Die GEM-Verteiler Box A344 [vWa99b] erfüllt zwei Funktionen: zum einen wird aus einer Driftspannung die GEM-Spannung für 8 Kammern erzeugt, zum anderen kann die GEM-Differenzspannung für jede Kammer einzeln geregelt werden. Die Regelung dient sowohl der Kompensierung von Ratenschwankungen, als auch zur Variation der Verstärkung jeder Kammer.

Die GEM-Verteilerbox funktioniert nach dem Prinzip eines Spannungsteilers (s. Bild 2.5). Der regelbare Widerstand wird durch einen Mikrocontroler angesteuert, der die Spannungen vor und nach dem regelbaren Widerstand mißt und danach versucht durch Veränderung des Widerstandes die vorgegebene GEM-Spannung zu erreichen.

Bei einem Spannungsteiler hängen die erzeugten Spannungen von den Strömen ab, jedoch die Ströme wiederum von der Teilchenrate in der Kammer. Daher kann eine Nachregelung des Widerstandes bei Variation der Teilchenrate notwendig sein.

Die Regelung erfolgt durch einen DAC (Digital Analog Converter), wobei eine Änderung um ein Bit einem Spannungssprung von einigen Volt entspricht und dadurch Ladeströme verursacht. Diese Ladeströme sollten minimiert werden. Deswegen wird um den momentanen Wert ein Fenster definiert, in dem eine Spannungsänderung toleriert und nicht nachgeregelt wird. Dieses Fenster kann für jeden Kanal programmiert werden.

Ein weiteres Problem entsteht bei einer Entladung zwischen der GEM-Ober- und GEM-Unterseite: die GEM-Differenzspannung sinkt bei einer Entladung auf 0 Volt. Da die GEM-Folien über 2 MΩ Widerstände geschützt sind, ist die Ladezeit der GEM-Folien im Vergleich zu der Regelzeit der GEM-Verteilerbox groß. Darum versucht die GEM-Verteilerbox durch Erhöhung des regelbaren Widerstandes die programmierte Differenzspannung zu erzeugen. Sobald sich die GEM-Folie allerdings wieder aufgeladen hat, wird der Widerstand nicht schnell genug verkleinert, so daß eine zu hohe Spannung anliegt. Da die Gasverstärkung exponentiell mit der GEM-Differenzspannung ansteigt, ist dieser Zustand sehr gefährlich für die gesamte Kammer und kann zu Überschlägen und Beschädigungen in der Kammer führen.

Darum wird der DAC, der den regelbaren Widerstand ansteuert, nach oben begrenzt. So können Veränderungen der Spannungen ausgeglichen werden, die Spannungsdifferenz ist aber limitiert.

Die Auslese und Programmierung der GEM-Verteilerbox erfolgt entweder über die serielle Schnittstelle RS232 oder den CAN-Feldbus.

2.4 Erstellte Programme

2.4.1 Strommessung

Es existieren drei Programme zur Ansteuerung der Strommeßbox, die für verschiedene Anforderungen entwickelt wurden:

- Ein "Stand-alone" Programm, das die Strommeßbox konfiguriert und regelmäßig ausliest. Die Daten werden in einer Datenbank gespeichert, es existiert jedoch kein Benutzerinterface. Für die Darstellung der Daten müssen diese mit dem Datenbank-Hilfsprogram `hbdis`[Lef96] wieder aus der Datenbank gelesen werden. Die Auslese der Box erfolgt über die serielle Schnittstelle RS232, die durch die VME-RS232-Adapter Karte DL620 mit einer VME-CPU verbunden ist. Das Programm wurde bei der Überwachung der Ströme während des PSI-Tests im Juli 1998 genutzt. Das Programm wurde im Oktober durch das im folgenden Absatz beschriebene Programm ersetzt.

- Das Programm `spc_a339` wurde als Parameter Control konzipiert, kann also in die HERA-B Slow Control integriert und zentral von einer Run Control gesteuert werden. Es können die Ströme überwacht und neben anderen Parametern die Stromgrenzen für einen Alarm festgelegt werden. Für das Programm wurde ein Graphisches User Interface entwickelt, mit dem die Box (auch online) konfiguriert und der Stromverlauf gezeichnet werden kann. Die Steuerung des Programms erfolgt mit der Finite State Machine. Mit diesem Programm wurden bei dem PSI-Test im Oktober 1998 die Ströme aufgezeichnet. Das GUI für dieses Programm wird in Abschnitt 2.4.4 vorgestellt.

- Mit Hilfe der graphischen Programmiersprache LabVIEW[Lab93] entstand ein Programm für Computer der Firma Apple. Hierbei wird keine VME-CPU oder VME-RS232-Adapter-Karte benötigt, vielmehr wird die serielle Schnittstelle des Computers für die Ansteuerung des Strommessers genutzt. Die Oberfläche des Programms ist in Abbildung 2.6 gezeigt. Die Daten werden auf der graphischen Oberfläche numerisch und graphisch dargestellt und in einer Textdatei gespeichert. Dieses Programm wird noch für Strommessungen gebraucht, bei denen keine VME-CPUs benutzt werden können.

Die beiden erstgenannten Programme basieren auf einer im Rahmen dieser Arbeit entwickelten Bibliothek zur Ansteuerung der Strommeßbox über die serielle Schnittstelle mit der VME-RS232-Adapter-Karte. In dieser Bibliothek wird die Synchronisation der Hardware mit Interrupts erreicht. Die ankommenden Informationen werden dann in einem Ringbuffer gespeichert. So wird eine asynchrone Nachrichtenverarbeitung ermöglicht: während die eine Nachricht noch aus dem Ringbuffer gelesen und verarbeitet wird, kann schon die nächste Nachricht eintreffen und im Ringbuffer gespeichert werden.

Bei dem zweiten Programm kann über das Netzwerk[6] auf die momentanten Stromwerte von bestimmten Kanälen zugegriffen werden. Das Programm enthält einen Subprozess, einen sogenannten Thread, der die Verbindung mit dem anfragenden Programm herstellt. Er empfängt einen Kanalnamen und sucht aus der Datenstruktur die momentanen Ströme des gesuchten

[6] Für die Netzwerkkommunikation wurde eine TCP/IP Bibliothek von Th. Kihm verwendet.

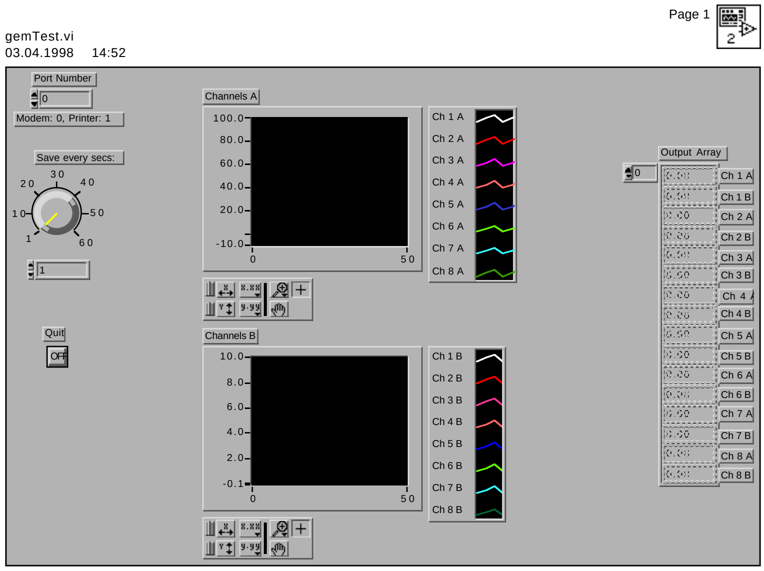


Abbildung 2.6: *Oberfläche des LabVIEW Programs zur Ansteuerung der Strommeßbox. Mit dem Schalter links oben kann man den Ausgang (Modem oder Printer) einstellen, an dem die Strommeßbox angeschlossen ist. Der Drehschalter links im Bild steuert die Meßfrequenz und mit dem Knopf darunter kann das Programm gestoppt werden. Die beiden in der Bildmitte angeordneten Displays zeigen die Ströme der Kanäle des Moduls A (oben) bzw. des Moduls B (unten) an. Diese Ströme erscheinen auch numerisch in der Anzeigenleiste am rechten Bildrand.*

Kanals ohne daß das Hauptprogramm im Ablauf beeinflußt wird. Dieser Stromwert wird über das Netzwerk an das anfragende Programm übermittelt.

2.4.2 GEM-Verteiler

Die GEM-Verteilerbox A344 kann mit drei verschiedenen Programmen gesteuert werden:

- Das erste Programm war eine "Stand alone"-Version, das bei Programmstart die GEM-Differenzspannung einstellt, die in einer Konfigurationsdatei gespeichert ist. Danach werden in regelmässigen Abständen die Potentiale an der Ober- und Unterseite der GEM-Folie und die GEM-Differenzspannungen auf dem Bildschirm angezeigt. Es existiert kein User Interface und die Meßwerte werden nicht gespeichert.

- Eine weiterentwickelte Version des Programms schreibt die momentane GEM-Differenzspannung und die Potentiale an der Ober- und Unterseite der GEM-Folie in eine Datenbank. Diese Programmversion fand Anwendung bei dem PSI-Test im Juli 1998.

- Als drittes Programm wurde die Parameter Control und das GUI für die GEM-Verteilerbox entwickelt. Das Programm `spc_a344` wird mit der Finit State Machine gesteuert und kann in das Slow Control System von HERA-B integriert werden. Mit dem Programm werden die Eingangsspannung der GEM-Verteilerbox und die Potentiale an der Unter- und Oberseite der GEM-Folie überwacht. Es kann die GEM-Differenzspannung, das Spannungfenster (in dem Spannungsänderungen ohne Nachregelung toleriert werden) und der maximal DAC-Wert (s. Abschnitt 2.3.4) eingestellt werden. Das Programm wurde erstmals bei dem PSI-Test im Oktober 1998 genutzt. Das GUI hierfür wird im folgenden Abschnitt erklärt.

2.4.3 Die Hochspannungssteuerung

Für die Ansteuerung des CAEN Hochspannungsgeräts über das Fast-CAEN-Net beim Inner Tracker wurde ein vorhandenes Programm des Outer Trackers verändert. Das Programm war das erste Beispiel einer Parameter Control.

Die Anforderung des Inner Trackers ist, die Spannungen in mehreren Schritten hochzufahren. Nach jedem Schritt wird eine bestimmte Zeit gewartet um Ladeströme abklingen zu lassen. Nur wenn dann die von der Strommeßbox gelesenen Ströme innerhalb eines definierten Toleranzbereichs liegen, wird weiter hochgefahren. Die momentanen Ströme werden über das Netzwerk aus dem Programm `spc_a339`, das die Strommeßboxen steuert, ausgelesen (s. Abschnitt 2.4.1). Die Spannungen der einzelnen Kanäle bei jedem Schritt und der Strombereich, in dem die Spannung noch erhöht werden soll, sowie die Wartezeit nach jedem Schritt ist in einer Konfigurationsdatei gespeichert.

Es besteht außerdem die Möglichkeit, daß der Benutzer nach jedem Spannungsschritt entscheidet, ob die Spannung weiter erhöht oder herunter gefahren werden soll. Hierfür muß lediglich die Umgebungsvariable `ASK_USER` auf `YES` gesetzt sein. Eine Abfrage der Stromwerte bei dem Strommeßprogramm erfolgt dann nicht.

2.4.4 Das graphische User Interface

In diesem Abschnitt werden die GUIs der Strommeßboxen, der GEM-Verteilerboxen und der Hochspannungsversorgung vorgestellt. Die drei GUIs werden hier zusammen beschrieben, da sie alle eine Modifikation eines im Rahmen der Arbeit entwickelten Beispiel-GUIs für die HERA-B Kollaboration war, das einen Umfang von $\approx$ 2000 Zeilen hat. Dieses GUI stellt für die Subdetektoren OTR, RICH und VDS die Grundlage ihrer eigenen Programme dar und zeigt den Umgang mit Tcl/Tk, die Anbindung an Datenbanken und die Einbindung in das SPC-Framework.

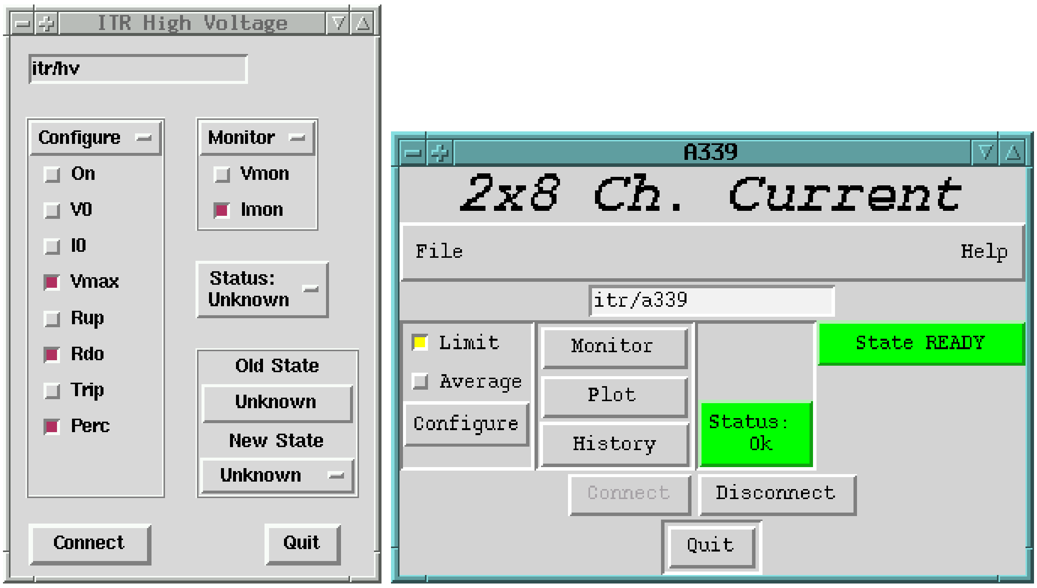


Abbildung 2.7: *Zwei Fenster der GUIs für die Hochspannungregelung (links) und die Strommeßbox A339 rechts. Es gibt jeweils Elemente für die Konfigurationsparameter, die Monitorparameter, den Status der Kanäle und den State des Programms.*

In Abbildung 2.7 sind die Fenster der Hochspannungssteuerung und der Strommeßbox zu sehen, die bei Start des GUIs erscheinen. In beiden Fenstern sieht man Elemente für die Konfigurationsparameter, die Monitorparameter, den Status der Kanäle und den State der Parameter Control. Konfigurationsparameter können mit der Maus ausgewählt werden, was durch ein farbiges Quadrat vor dem Parameternamen angezeigt wird. Mit einem Pulldown Menu, das sich unter dem Knopf "Configure" verbirgt, können diese Parameter gesetzt bzw. angezeigt werden.

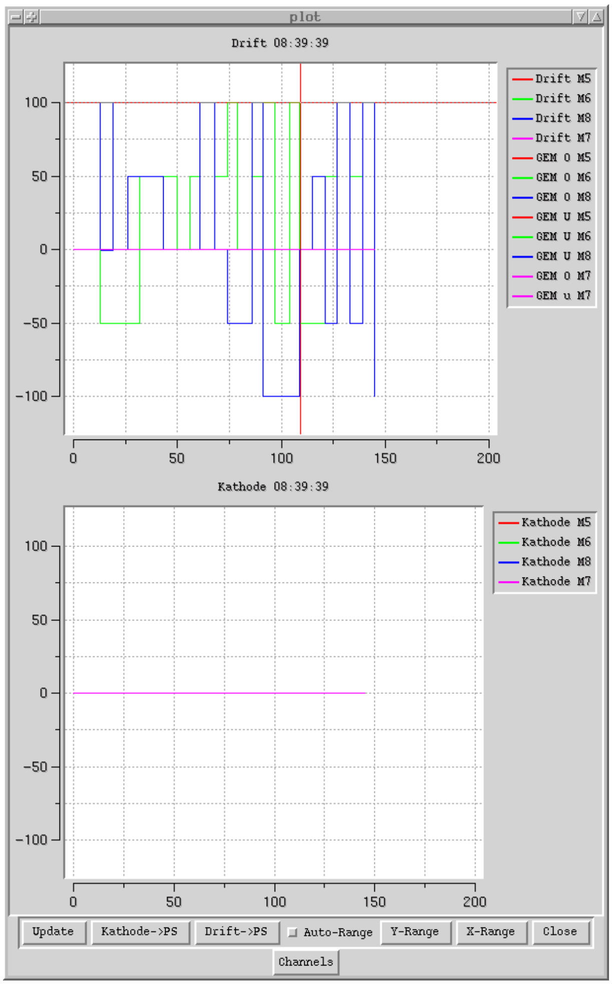


Abbildung 2.8: *Fenster, in dem der Stromverlauf online dargestellt wird. In dem oberen Plot sind die Ströme der Driftspannungen und der GEM-Folie gezeichnet, in dem unteren die Kathodenströme.*

Da bei der Strommessung nur die Ströme von Interesse sind, wurde dort auf eine Auswahl der Monitorparamter verzichtet. Nach dem Drücken des Knopfes "Plot" öffnet sich ein weiteres Fenster, in dem der aktuelle Stromverlauf dargestellt wird, das in Abbildung 2.8 gezeigt wird. Einzelne Kanäle können ausgeblendet und der Stromverlauf als Postscript-Datei gespeichert werden. Mit der Maus kann man einen bestimmten Punkt einer Stromkurve auswählen und erhält den Kanalnamen, der zu dieser Kurve gehört, den Stromwert und die Zeit.

Durch das Drücken des Knopfs "History" wird ein Fenster geöffnet, in dem man die Kanäle angeben kann, von denen man den Verlauf in einem wählbaren Zeitraum sehen will. Danach werden die Ströme für diese Kanäle aus der Datenbank gelesen und graphisch in einem Fenster ähnlich Abbildung 2.8 dargestellt.

2.5 Hardwired Slow Control

Die Sicherheit beim Betrieb des Detektors darf nicht alleine auf der Überwachung von Parametern durch Computer und Software basieren. Auch wenn Programme bzw. Computer den Detektor nicht mehr steuern können, muß ein kritischer Zustand vermieden oder, wenn dies nicht möglich ist, zumindest gemeldet werden. Zustände wie partielle Stromausfälle müssen ebenso berücksichtigt werden, wie die Reaktion auf einen zentralen Alarm in der Halle.

Daher existiert eine zweite Überwachungsebene, die zum größten Teil auf elektro-mechanischen Bauteilen basiert (s. Abbildung 2.9).

Verschiedene Szenarien wurden bei der Planung berücksichtigt:

- Wenn sich das Gassystem des Inner Trackers in einem kritischen Zustand befindet, wird die Hochspannung durch das CAEN Hochspannungsgerät schnellstmöglich heruntergefahren. Im Kontrollraum wird dies akustisch und optisch signalisiert.
- Der Strom fällt in einem Crate aus, welches die Strommeßboxen und die GEM-Verteilerbox versorgt. In diesem Fall wird das Relais in der Strommessbox geöffnet und somit die Verbindung mit der Spannungsquelle unterbrochen. Die Kammer entlädt sich über die GEM-Verteilerbox.
- Eine Strommeßbox mißt einen zu hohen Strom und geht in den Alarmzustand. Daraufhin öffnet das in der Strommeßbox laufende Programm das Relais in dem Strommesser, im Kontrollraum leuchtet eine Lampe auf und ein Signal ertönt.
- Bei einem externen Alarm (Gasalarm in der Halle, Feueralarm...) wird so schnell wie möglich die Hochspannung durch das Hochspannungsgerät ausgeschaltet.

2.6 Ausblick

Bisher existieren für die Slow Control des Inner Trackers Programme für die Hochspannungssteuerung, die Strommessung und die Regelung der GEM-Spannungen. Diese Programme können in die Baumstruktur der gesamten HERA-B Slow Control integriert werden. Bis Oktober 1998 existierten allerdings noch keine der Parameter Control übergeordneten Knoten im Slow Control Baum. Mittlerweile[7] wurden diese Programme entwickelt und müssen nun noch mit den existierenden Parameter Controls getestet werden.

Die Ansteuerung der Strommeßboxen und der GEM-Verteilerbox soll im Jahr 1999 auf den CAN-Bus umgestellt werden, da der Betrieb über die serielle Schnittstelle RS232 nur bei einer geringen Anzahl von Boxen möglich ist.

Weitere Änderungen im Programm sind notwendig, weil im November 1998 eine neue Version der SPC-Bibliothek veröffentlicht wurde. Neben der Behebung von Fehlern wurde das Datenbankformat geändert. Da im GUI des Strommessers die Möglichkeit besteht, Ströme aus der Datenbank zu lesen und darzustellen, muß dieser Programmteil dem neuen Format angepaßt werden.

Das Konzept einer Slow Control Arbeitsgruppe für die gesamte Kollaboration und die Bereitstellung eines Frameworks mit Beispielen hat sich bewährt. Ein Beispiel für die gelungene Zusammenarbeit ist die Hochspannungssteuerung: Zu Beginn wurde von der ECAL-Gruppe eine Bibliothek für die Hardware entwickelt und diese von dem OTR in das SPC-Framework integriert. Das Beispiel-GUI des ITR wurde vom RICH verbessert und modifiziert, bevor es für den ITR erweitert wurde.

[7] Stand Dezember 1998

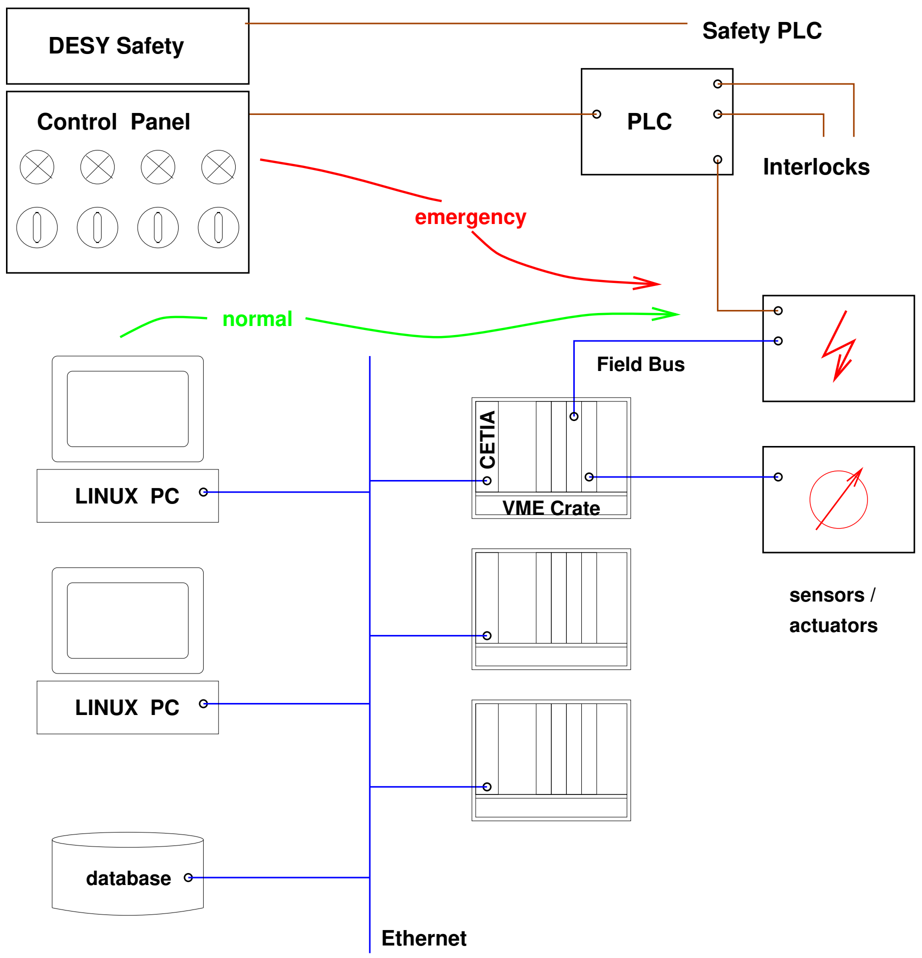


Abbildung 2.9: *Schema der HERA-B Slow Control: Im unteren Bildteil sieht man den auf Rechnern aufgebaute Slow Control Teil für eine Hochspannungsversorgung. Auf einer VME-CPU läuft eine Parameter Control, die die Hochspannung überwacht und die Meßwerte in eine Datenbank schreibt. Im oberen Bildteil ist der rechnerunabhängige Teil der Slow Control zu sehen. Die PLC (Programmable Logic Controller) wird bei dem Inner Tracker durch Relais-Schaltungen ersetzt. Eine Kontrolleiste im Kontrollraum signalisiert den Zustand der Slow Control und besitzt auch einen NOT-AUS-Schalter für eine schnellstmögliche Abschaltung im Notfall.*

Kapitel 3

Analyse von Kammerdaten

Im Rahmen dieser Arbeit wurden Daten, die mit der vollständigen Readout-Kette genommen waren, unter verschiedenen Aspekten analysiert.

In dem folgenden Kapitel wird im ersten Teil beschrieben, in welcher Form die Kammer-Daten für die Analyse vorliegen und mit welchen Algorithmen die Aufbereitung erfolgt.

Im zweiten Teil werden die Ergebnisse einer Auswertung von Daten vorgestellt, die bei einem Strahltest am Paul-Scherrer-Institut Villigen (CH) im Oktober 1998 genommen wurden. Das Ziel dieses Test war die Funktion der gesamten Readout Kette zu untersuchen und die Betriebsparameter der Kammern zu überprüfen.

Eine Auswahl der Runs mit den jeweiligen Bedingungen findet man in Tabelle 3.1 und eine vollständige Übersicht im Anhang D.

3.1 Beschreibung des Aufbaus am PSI

3.1.1 Mechanischer Aufbau

Die Kammern wurden auf CFK-Strukturen montiert, die die gleichen Halterungen für die Kammern und die Elektronik besitzen wie später für HERA-B vorgesehen. Dabei kann die Kammer mit einer Genauigkeit von 10 μm auf die CFK-Platte montiert werden. Um die Kammern gegeneinander zu verschieben und die Anordnung zu variieren, können die CFK-Platten auf Schienen parallel zueinander bewegt werden.

Für jede Kammer wurde ein eigener Gaskreislauf betrieben, damit für jede Kammer der Gasfluß kontrolliert werden konnte. Als Zählgas wurde ein Argon-DME (Dimethylether) Gemisch im Verhältnis 50:50 verwendet.

Daten wurden mit den drei Kammern M5, M6 und M8 genommen. Die Kammern unterscheiden sich in folgenden Punkten:

Run No.	Spannung	Strahl	Position [mm]	Kollimator
272	$U_{Kathode} = 465V$, $U_{Gem} = 390V$	π^-	$x = 5.6, y = -10.6$	100^4
273	$U_{Kathode} = 485V$, $U_{Gem} = 390V$	π^-	$x = 5.6, y = -10.6$	100^4
274	$U_{Kathode} = 485V$, $U_{Gem} = 410V$	π^-	$x = 5.6, y = -10.6$	100^4

Tabelle 3.1: *Eine Auswahl der am PSI im Oktober genommenen Daten. Die Position bezeichnet die Anzeige des Fahrtisches für das Szintillatorkreuz. Der Exponent bei der Kollimatorstellung bedeutet, daß die vier Kollimatoren auf den jeweiligen Wert gestellt waren. Das Schema für die Dateinamen der Rohdaten ist:* `readout.testrun.[Run No.].data`

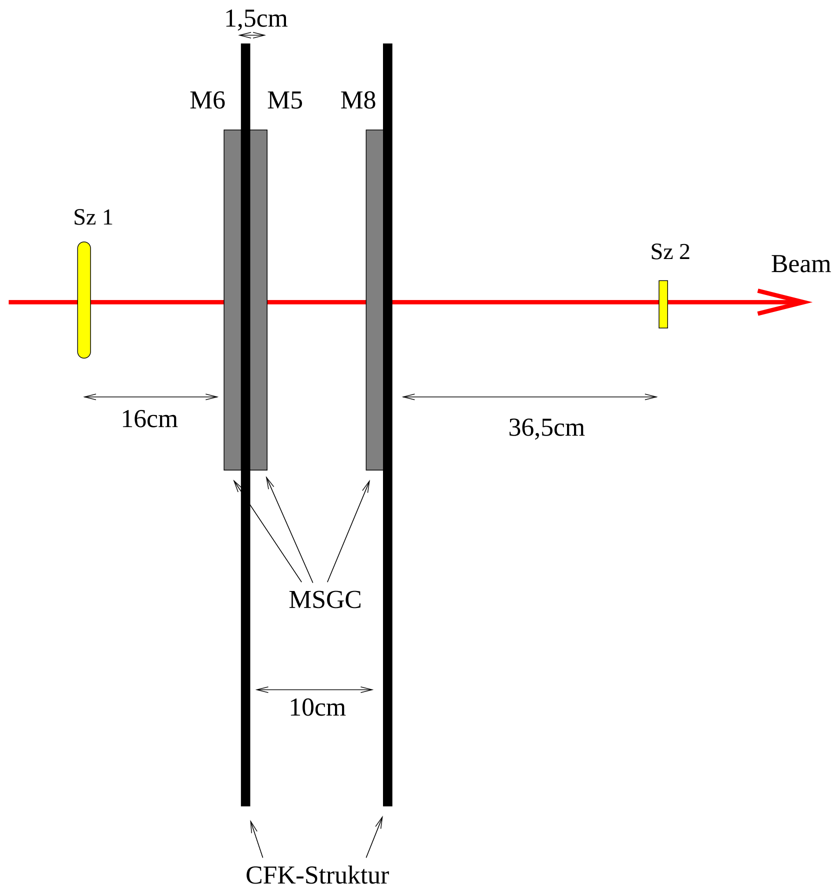


Abbildung 3.1: *Der Aufbau am PSI im Oktober 1998. Sz2 ist ein fahrbares Szintillatorkreuz aus zwei* $5 \times 5mm^2$ *großen Szintillatoren und Sz1 ein Fingerszintillator, der parallel zu den Anodenstreifen der Kammer montiert war. Die CFK-Strukturen können gegeneinander parallel verschoben werden.*

- M8 wurde mit einem 5-fach segmentierten GEM gebaut. Die Segmentierung sollte bei Überschlägen zwischen der GEM-Folie und der Drift-Elektrode die freiwerdende Ladung reduzieren und so Schäden vermeiden. Jedoch traten bei den angelegten Spannungen in keiner Kammer Überschläge auf, so daß der Vorteil der Segmentierung nicht überprüft werden konnte. Die GEMs der Kammern M5 und M6 sind identisch mit den GEMs, die für den späteren Einbau bei HERA-B produziert wurden.

- Die Kammer M5 besitzt für die Kathoden 1 MΩ Vorwiderstände, die Kammern M6 und M8 hingegen 10 MΩ Vorwiderstände. Da die Kathoden und Vorwiderstände "in Reihe" geschaltet sind, hängt die an der Kathode anliegende Spannung von dem Vorwiderstand und dem Strom durch den Vorwiderstand ab. Durch den Vorwiderstand fließen ein bestrahlungsunabhängiger Blindstrom und ein durch die Gasverstärkung in der Kammer hervorgerufener Strom.

Die Kammern M5 und M6 wurden als Doppellage eingesetzt, wobei an M6 eine Triggerkarte angeschlossen wurde und hiermit erstmals das Zusammenspiel der Auslese mit Kammer und der ITR-Triggerelektronik studiert werden konnte.

Mit den in Kapitel 3 vorgestellten Slow Control Programmen wurden die Ströme der Kathode, der Driftanode und der GEM-Folie protokolliert.

Da die Kammer M8 mit einer leicht veränderten Auslesekette betrieben wurde, hätte die Analyse weitere Korrekturschritte erfordert. Deshalb werden die Daten der Kammer M8 in dieser Arbeit nicht berücksichtigt.

3.1.2 Auslese

Die Auslese von Kammersignalen erfolgt mit dem Auslesechip "Helix 2.2" [Fal98], der 128 Eingangskanäle besitzt. Für jeden Kanal erzeugt der Helix ein Analogsignal, das zu der auf der Anode deponierten Ladung proportionales ist. Die Analogsignale jeweils zweier Helixe, die auf einem PCB im Daisy-Chain-Mode betrieben werden, werden seriell über eine Glasfaserstrecke zum FED gesendet [Hau98], wo sie von einem 8 Bit Analog Digital Converter (ADC) digitalisiert wird. Außerdem werden ein Header vor das digitalisierte Signal und ein Trailer nach dem Signal der 128 Kanäle eingefügt. Der Header enthält Informationen des Fast Control Systems und Informationen über die Herkunft der Daten (Subdetektor, FED). Der Trailer enthält nochmals Fast Control System Informationen. Diese Informationen wurden allerdings nicht ausgewertet. Das FED-Board befindet sich in einem VME-Crate und leitet die Daten in einen Signalprozessor, den SHARC, weiter. Mit einer VME-CPU können aus dem SHARC im sogenannten "Spy Modus" Daten ausgelesen werden. Es besteht so die Möglichkeit, die Daten direkt auf Festplatte zu schreiben. Da alle Daten bei dem Test am PSI von der VME-CPU zu einem Server mit Festplatte über ein Netzwerk gelangen, ist die Geschwindigkeit auf mehrere Hz limitiert. Bei dem späteren Betrieb werden die Daten von dem SHARC über sogenannte "SHARC Links" der zentralen DAQ zugeführt.

Als Trigger wurden drei in Koinzidenz geschaltete Szintillatoren eingesetzt: zwei gekreuzte $5 \times 5mm^2$ große Szintillatoren hinter den Kammern und ein Szintillator vor den Kammern (jeweils in Strahlrichtung) liefern bei einem Teilchendurchgang drei koinzidente Signale (s.Abb. 3.1).

3.1.3 Analyse der Daten

Jeder Eventdatensatz setzt sich aus Datensätzen für 2 Helixe auf einem Printed Circuit Board (PCB) zusammen. Ein Beispieldatensatz eines PCBs ist im Bild 3.2 gezeigt. Man sieht von den Helixen 0 und 1 jeweils 128 Helix-Kanäle, wobei der Analogwert zu dem Triggerzeitpunkt gezeigt ist. Außerdem sieht man in der Abbildung die vom Helix versendete Information, aus welcher Spalte (Column) das Ereignis gelesen wurde. Da alle Helixe synchron laufen, muß man aus jedem Helix bei einem Trigger die selben Spalte auslesen und somit die gleiche Columnnumber sehen.

Bei jedem Helix-Datensatz muß zwischen vier verschiedenen Offsets unterschieden werden, dem ADC-Offset, der Baseline-Schwankung, der Pedestal-Variation und der Speicherzellen-Variation:

- Um negative Signal mit dem Helix nachweisen zu können, muß die Grundlinie (Baseline) der Signale zu positiven Werten verschoben sein. Dieser Offset ist in Abbildung 3.2 $\approx$ 100 ADC Counts und kann bei den optischen Empfängern und am FED-Board verändert werden.

- Die Variation der Baseline bei verschiedenen Ereignissen und Helixen ist nicht gewünscht. Die Änderung des gemeinsamen Offsets aller Kanäle eines Helix wird "Baseline-Schwankung" genannt und durch externen Pickup und eine Schwingen der Helix-internen Referenzstromquelle verursacht. Der externe Pickup kann durch eine gute Erdung der Elektronik am Detektor und die Schwingung der Stromquelle durch Kondensatoren vermindert werden.

- Da die Verstärkerkette auf dem Helix bei jeden Kanal produktionsbedingt leicht verschieden ist, gibt es einen von Kanal zu Kanal verschiedenen Offset, der jedoch bei jedem Ereignis gleich bleibt.

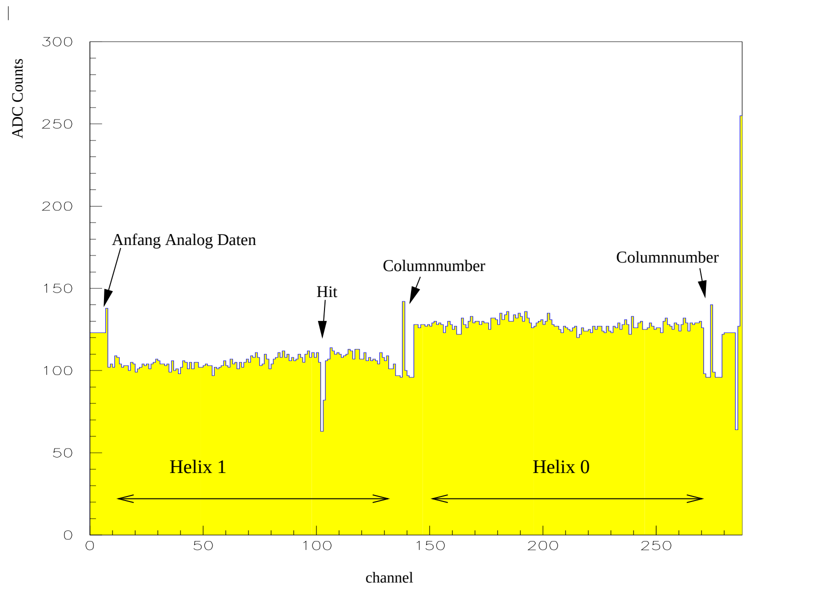


Abbildung 3.2: *Der Rohdatensatz enthält von jedem Helix (Helix 0, Helix 1) jeweils 128 Kanäle. Außerdem befinden sich noch Kontrollinformationen wie die Columnnumber im Datensatz. Der erste Kanal des linken Helix hat wegen mit einer speziellen Auslese einen viel größeren Wert als die restlichen Kanäle des Helix und wurde somit nicht für die Analyse verwendet.*

- Die dritte Variation des Offsets rührt von der produktionsbedingten Verschiedenheit der 128×141 Zwischenspeicherzellen her. Dieser Effekt ist allerdings klein im Vergleich zu den beiden ersten.

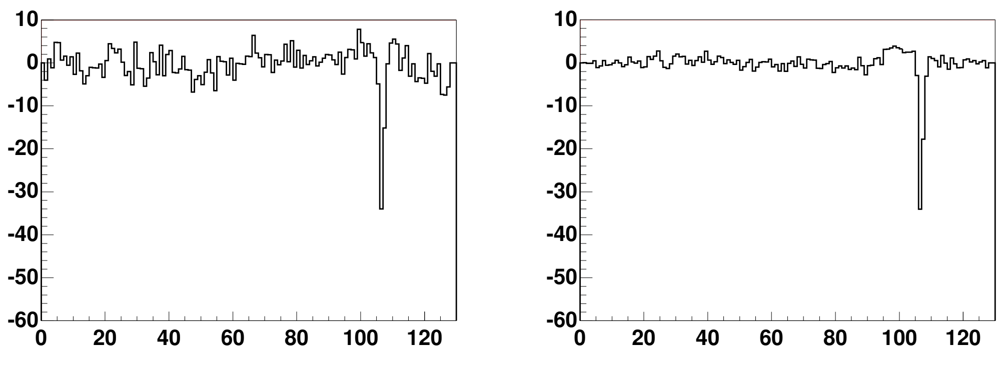


Abbildung 3.3: *Ein Event nach zwei Analyseschritten. Im ersten Analyseschritt wurde ein gemeinsamer Offset aller Kanäle abgezogen, im zweiten Schritt (Bild rechts) ein für jeden Kanal individueller Offset subtrahiert.*

Die Verarbeitung der Rohdaten erfolgt in mehreren Schritten:

- Zuerst muß der Anfang der digitalen Daten gefunden werden, da der Anfang der Daten

(s. Abb. 3.2) nicht immer an der gleichen Stelle ist. Der Anfang wird durch den ersten Sprung von einem Kanal zum nächsten definiert. In der Abbildung 3.2 sieht man den ersten Kanal des linken Helix deutlich erhöht im Vergleich zu den folgenden Kanälen, da an dem ersten Kanal eine spezielle Auslese angeschlossen war. Deshalb wird dieser Kanal bei der folgenden Analyse nicht berücksichtigt.

- Danach wird die Information gesucht, in welcher Pipelinespalte das Ereignis stand. Mit der Information, der Columnnumber, könnte man die Zwischenspeicherzelle feststellen und die individuelle Variation korrigieren. Bei der vorliegenden Analyse wurde diese Korrektur allerdings noch nicht durchgeführt, da diese Korrektur nur sehr klein ist.

- Die Daten für ein PCB, also zwei Helixe, werden in zwei Datensätze aufgeteilt, für jeden Helix einen Datensatz. Die Columnnumber, die Information, aus welcher Pipelinespalte das Event gelesen wurde, wird aus den Analogdaten entfernt und in der Datenstruktur gesondert gespeichert.

- Da die Baseline-Schwankung für alle Kanäle von Event zu Event sehr stark variiert, wird ein Polynom zweiten Grades in der Kanalnummer an die Baseline angepaßt und danach dieses Polynom subtrahiert ($\approx$ 100 ADC-Counts). In Abb. 3.3 links sind die Daten nach dieser Korrektur gezeigt. Diese Korrektur variiert von Event zu Event um ± 20 ADC-Counts..

- Der Offset von Kanal zu Kanal, die Pedestal-Variation, variiert in einem Bereich von ≈ 10 ADC-Counts. Darum wird das Pedestal jedes Kanals über alle Ereignisse gemittelt und in einem zweiten Schritt der Mittelwert von dem jeweiligen Kanal abgezogen. Die Korrekturwerte sind für alle Ereignisse gleich, da sie nur von der elektronischen Beschaffenheit der Kanäle auf dem Helix abhängen. Abbildung 3.3 rechts zeigt den linken Datensatz nach diesem Korrekturschritt. Die Variation zwischen den Kanälen hat sich auf ≈ 4 ADC-Counts reduziert, was eine deutlich sensitivere Pulssuche erlaubt.

- Letztendlich wird nach den Signalen von Teilchendurchgängen gesucht. Nach den Korrekturen erwartet man in den meisten Helix-Kanälen einen ADC-Wert von ungefähr Null, nur bei Teilchendurchgängen sollte der Wert deutlich kleiner Null sein. Nun betrachtet man das korrigierte Signal jedes Helix-Kanals und definiert einen Schwellwert. Ist das korrigierte Signal eines Kanals über diesem Schwellwert (z.B. 5 ADC-Counts), wird in den folgenden nach weiteren Kanälen über dem Schwellwert gesucht und alle gefundenen Kanäle zu einem Puls zusammengefaßt, für den folgende Parameter bestimmt werden:

  - Maximale Pulshöhe
  - Integrierte Pulshöhe
  - Pulsbreite; Zahl der benachbarten Kanäle über Schwellwert
  - Ort der maximalen Pulshöhe
  - Schwerpunkt des Pulses

- Da ein geschlossener Stromkreislauf zwischen der Auslese-Elektronik und der Kammer besteht, fließt ein dem Signal entgegengesetzter Strom durch die benachbarten Kanäle. Die rückfließenden Ströme haben das entgegengesetzte Vorzeichen wie der Strom des Signals und erzeugen damit bei großen Pulsen einen Offset, der deutlich in Abbildung 3.4 zu sehen ist. Die Signalhöhe wird deshalb nicht von der Nulllinie gerechnet, sondern es wird aus den benachbarten Kanälen ein mittlerer Offset berechnet, mit dem die Signalhöhe korrigiert wird.

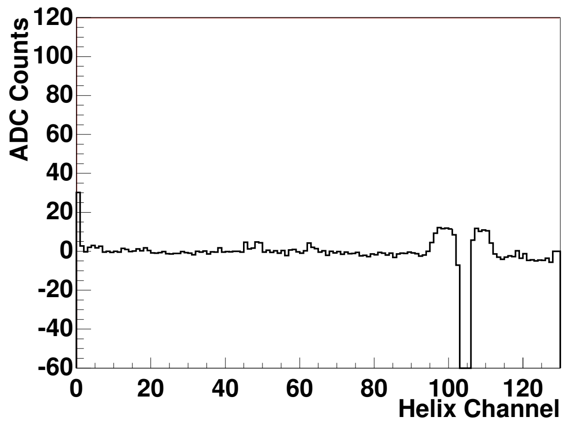


Abbildung 3.4: *Ein Event, bei dem die Veränderung der Baseline durch den Rückfluß bei großen Pulsen sehr deutlich ist. Der ADC befindet sich bei diesem Event im Bereich des Kanals 105 in der Sättigung.*

Die Verarbeitung der Rohdaten zu Datensätzen für jeden Helix mit Informationen über Kammersignale geschieht mit dem Analysepakt "ROOT" [ROOT], dem in C++ [Str92] geschriebenen Nachfolger von "Paw" [PAW]. Sämtliche Analyseskripte benutzen die C++ Syntax, weswegen keine neue Makrosprache gelernt werden muß.

Die im folgenden beschriebenen Analysen erfolgte mit Programmen, die gemeinsam mit Th. Walter von der Universität Zürich entwickelt wurden [Wal98]. Eine Beschreibung der wichtigen Routinen befindet sich im Anhang C.

3.2 Pulshöhenanalyse

3.2.1 Variation der Kammerspannungen

Für drei verschiedene Gasverstärkungen wurde jeweils die integrierte Pulshöhe und die maximale Pulshöhe ausgewertet. Ziel war es, die im Hinblick auf die Auslese optimalen Betriebsparameter für die Kammern festzulegen.

In den Histogrammen 3.5 werden nur Hits auf Anoden berücksichtigt, die von den drei Szintillatoren überdeckt sind. Da auf die Koinzidenz der drei Szintillatoren getriggert wurde und der Teilchenfluß klein war, stammen diese Hits von Teilchendurchgängen. Wäre der Teilchenfluß sehr hoch, würden mit einer großen Wahrscheinlichkeit mehrere Teilchen zu einem bestimmten Zeitpunkt die Kammer durchqueren. Man würde also in den Daten nicht nur einen Puls, sondern mehrere Pulse sehen und es ergäbe sich das Problem herauszufinden, welches Teilchen das Signal im Szintillator ausgelöst hat. Dies ist von Bedeutung, da eine Aussage über Pulshöhen nur bei optimaler Einstellung des Zeitpunktes möglich ist, zu dem der Wert des Helix-Vorverstärkers in den Zwischenspeicher übernommen wird. Ein nicht triggernder Puls durchquert mit einer geringen Wahrscheinlichkeit genau zum optimalen Zeitpunkt die Kammer. Deshalb möchte man einen Teilchenfluß, bei dem mit großer Wahrscheinlichkeit nur ein Signal eines Teilchens sichtbar ist.

Der Beam bestand bei diesen Runs zu ≈ 90 % [Gra98] aus π^- mit einer Energie von 350 MeV. Pionen sind bei dieser Energie minimal ionisierend, was die Zusammensetzung des Teilchenflusses durch die Kammern bei HERA-B gut widerspiegelt. Bei jedem Run wurden jeweils 8000 Events aufgezeichnet.

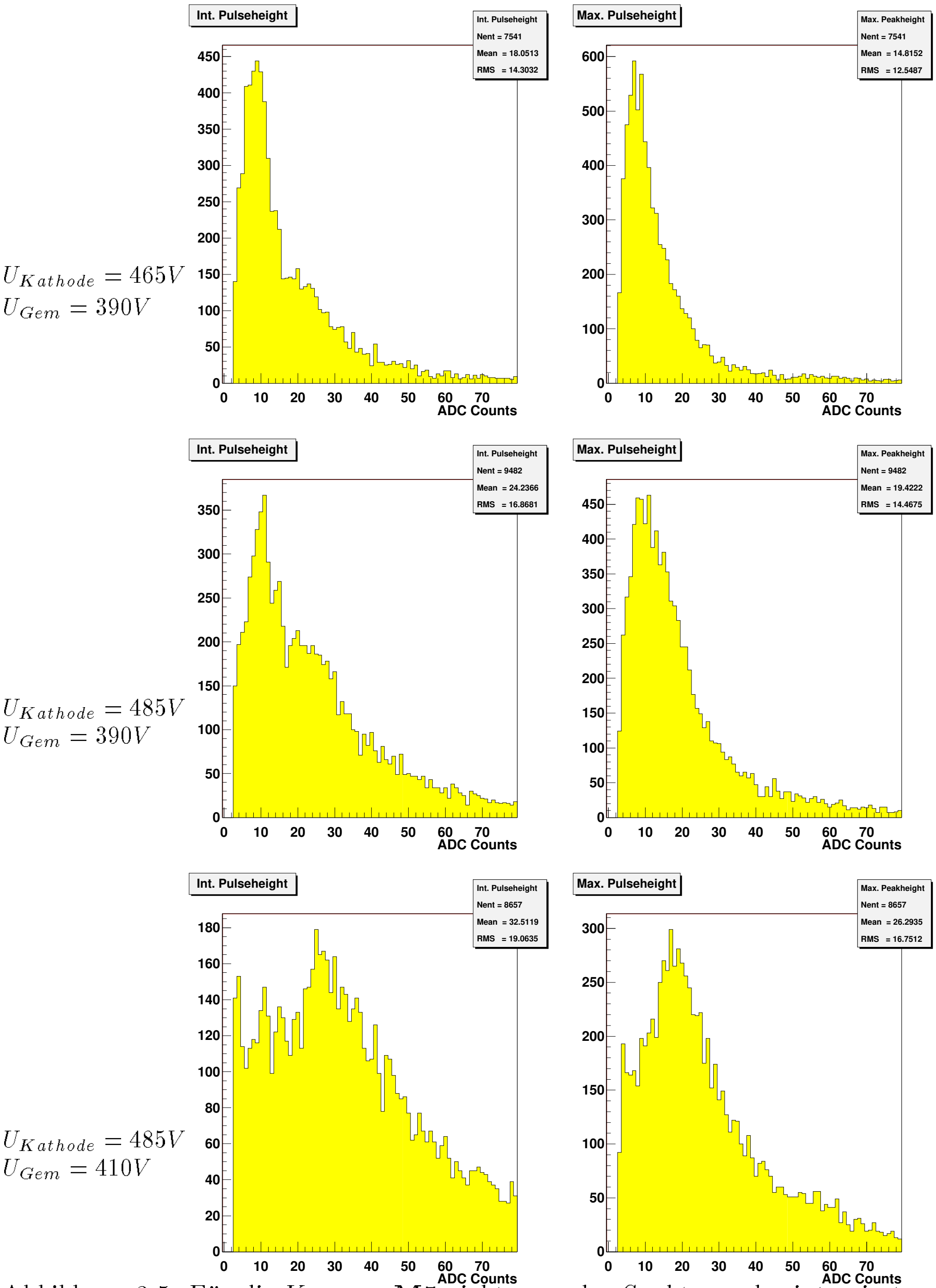


Abbildung 3.5: *Für die Kammer* **M5** *sieht man das Spektrum der integrierten (links) und maximalen (rechts) Pulshöhe bei verschiedenen Spannungen.*

Aus den Pulshöhenhistogrammen wird deutlich, daß sich allein bei $U_{Kathode}$ = 485 V und U_{Gem} = 410 V das Maximum der Landau-Verteilung deutlich vom Rauschen abhebt. Die vor dem PSI Test definierte "Standardkonfiguration" mit $U_{Kathode}$ =465 V und U_{Gem} =390 V zeigt eine deutlich zu geringe Gasverstärkung.

Selbst bei der höchsten Spannung wäre eine höhere Gasverstärkung wünschenswert.

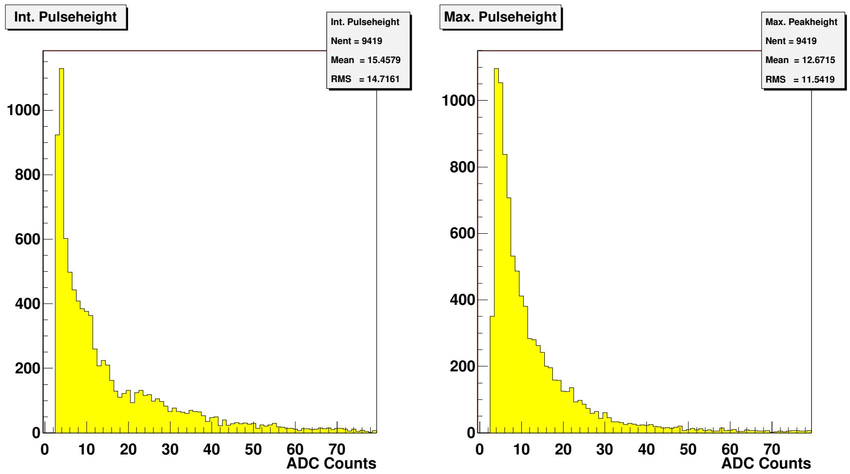


Abbildung 3.6: *Kammer* **M6** *zeigt eine geringere Gasverstärkung als die baugleiche Kammer M5. Der Signal-Peak verschwindet bei der Kammer M6 im Rauschen. Die Spannungen sind* $U_{Kathode}$ *=485 V,* U_{Gem} *=410 V. Man vergleiche diese Histogramme mit der untersten Zeile in Abbildung 3.5*

3.2.2 Vergleich der Kammern M5 und M6

Obwohl die Kammern M5 und M6 bis auf die Kathoden-Vorwiderstände baugleich sind, wurde eine starke Variation der Gasverstärkung beobachtet (s. Bild 3.6). Wenn man die Kathodenströme der beiden Kammern vergleicht und berücksichtigt, daß die Bestrahlung, und damit auch die durch Gasverstärkung ausgelösten Ströme, nur in einem kleinen Bereich der Kammer auftrat, erhält man eine Spannungsunterschied an den Kathoden der beiden Kammern. Dieser Spannungsunterschied kann evt. den Unterschied in der Gasverstärkung erklären [Eis98], ist aber auch noch Gegenstand weiterer Untersuchungen.

Um welchen Faktor genau die Gasversträrkung variiert, kann aus den Bildern nicht abgelesen werden, da bei der Kammer M6 das Maximum der Verteilungen im Rauschen verschwindet.

3.2.3 Bestimmung der Gasverstärkung

Da die Clustersize in einem weiten Bereich nicht von der Verstärkerkette bestimmt wird, kann davon ausgegangen werden, daß die integrierte Pulshöhe ein gutes Maß für die Gasverstärkung ist und damit nicht stark vom Übersprechen der Verstärkerkanäle abhängt.

Die Pulsgröße hängt von der Primärionisation, der Verstärkerkette und dem ADC-Wandler ab. Es gilt:

$$< \text{int. Puls}_{ADCCounts} >=< \text{Primärion.} > \cdot \frac{mV}{\text{Elektron auf der Anode}} \cdot \frac{\text{ADC-Counts}}{mV} \cdot \text{Gasverst.} \tag{3.1}$$

Die Primärionisation für ein MIP (Minimal Ionizing Particle) wird mit $30e$ angenommen. Mit der Verstärkerkette wurde gemessen, daß 38 mV einer Ladung von $24.000e$ entsprechen. Da der ADC-Wandler 1 Volt mit 8 Bit auflöst ergibt sich $\frac{1ADC}{3.9mV}$. Die mittlere integrierte Pulshöhe kann nicht aus Abbildung 3.5 abgelesen werden, da dort nur über den gezeigten Bereich gemittelt wird. Sinnvoller ist es, den Bereich bis zu 300 ADC-Counts zu erweitern, womit sich eine mittlere integrierte Pulshöhe von 47 ergibt. Daraus folgt für die Gasverstärkung:

$$\text{Gasverstärkung} = \frac{47\text{ADC-Counts}}{30e\text{Primärionisation}} \cdot \frac{24.000e}{38mV} \cdot 3.9\frac{mV}{\text{ADC-count}} \approx \mathbf{3900} \qquad (3.2)$$

Eine Gasverstärkung von 3900 sagt aus, daß ein Teilchendurchgang in der Kammer M5 bei einer Kathodenspannung von $485V$, eine Driftspannung von $3200V$ und einer GEM-Differenzspannung von $410V$ auf der Anode 3900 nachweisbare Elektronen erzeugt. Man nennt diese Gasverstärkung "visible gain", da nur die nachweisbare Verstärkung berücksichtigt wird.

3.3 Wiremap

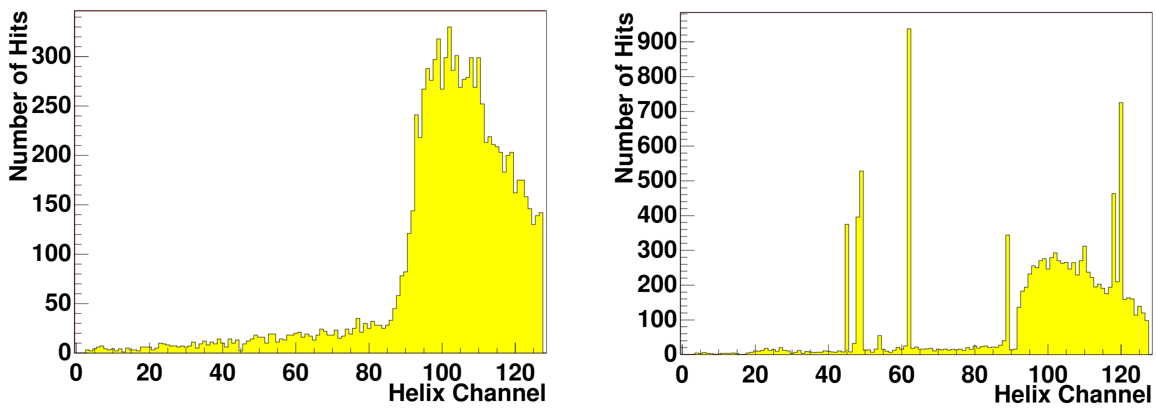


Abbildung 3.7: *Wiremaps der Kammern M5 (links) und M6 (rechts). Man sieht rechts im Bild die von dem Szintillator überdeckten Fläche und links davon einen Bereich, wo nur Zufallsereignisse aufgezeichnet wurden. Die Wiremaps wurden aus Daten des Runs 274 mit dem Schwellwert 3 erstellt.*

Unter einer "Wiremap" versteht man ein Histogramm, in dem die Anzahl der detektierten Teilchendurchgänge für jeden Helix-Kanal dargestellt ist.

In Abbildung 3.7 wird die Wiremap des Runs 274 für die Kammern M5 und M6 gezeigt. Eingetragen wurde der Helix-Kanal mit der maximalen Pulshöhe. Als Trigger wurde ein Szintillator-Kreuz, bestehend aus zwei $5 \times 5mm^2$ großen Szintillatoren verwendet, deren Koinzidenz einen Teilchendurchgang anzeigte. Da die Teilchenrate gering war, erwartet man nur in dem vom Szintillator überdeckten Bereich Hits. Außerhalb des überdeckten Bereichs sollte man nur zufällige Ereignisse und "heiße" Kanäle beobachten. Als "heiß" bezeichnet man Kanäle, die deutlich häufiger über dem Schwellwert sind als aus der Umgebung des Kanals zu erwarten wäre.

Bei beiden Kammern läßt sich der vom Szintillator überdeckte Bereich gut am rechten Bildrand erkennen. Bei Kammer M6 sind zusätzlich eine Reihe von "heißen" Kanälen sichtbar.

Die Breite des Bereichs mit vielen Einträgen sollte der Ausdehnung des Szintillators entsprechen. Da der Szintillator $5mm$ breit war, erwartet man, daß sich der Szintillator auch auf dieser Breite in der Wiremap abzeichnet. Tatsächlich ist der Bereich mit vielen Einträgen statt der erwarteten 17 Kanäle jedoch $\frac{5mm}{300\mu m\text{Pitch}}$=30 Kanäle breit.

3.4 Column Number

Die Column Number gibt an, in welcher Spalte der Helix-Pipeline das ausgelesene Ereignis zwischengespeichert wurde. Da keine der 141 Spalten ausgezeichnet ist, erwartet man eine Gleich-

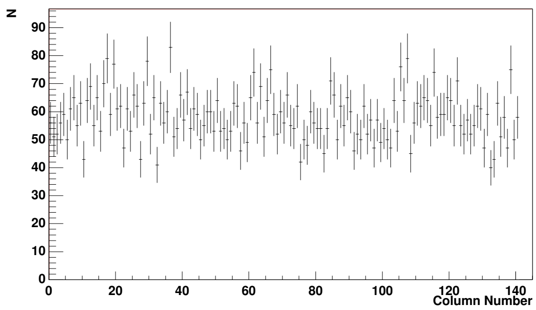


Abbildung 3.8: *Verteilung der Column Numbers bei Run 274*

verteilung der Spalten, die in der Abbildung 3.8 auch sichtbar ist.

Innerhalb der Fehler kommen alle Spalten gleich häufig vor. Der erwartete Wert liegt bei den 8000 analysierten Ereignissen bei $\frac{8000}{141} \approx 57$.

3.5 Clustersize

In diesem Abschnitt wird der Begriff Clustersize in zwei Bedeutungen verwendet. Die erste und gebräuchlichere Bedeutung gibt an, wieviele Kanäle bei einem Puls über dem Schwellwert sind. Diese eher technische Clustersize-Definition wird im folgenden "Clustersize über Schwellwert" genannt und wird für die Berechnung der Occupancy (durchschnittliche Anzahl der Kanäle über dem Schwellwert bei einem Event) benötigt.

Das Verhältnis von integrierter Pulshöhe zur maximalen Pulshöhe ist die zweite Definition von Clustersize, die hier untersucht wird. Sie wird im folgenden "relative Clustersize" genannt. Sie ist unter anderem eine Kennzahl dafür, wieviel Ladung nicht in dem höchsten Kanal deponiert wurde. Da nur auf die maximale Pulshöhe diskriminiert wird, ist diese Ladung bei dem angewendeten Pulssuchalgorithmus nicht sichtbar.

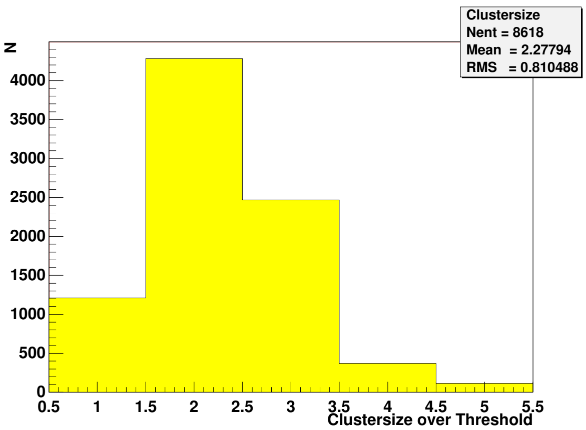


Abbildung 3.9: *Mit Clustersize ist hier gemeint, wieviele Kanäle über dem Schwellwert von 4 ADC-Counts liegen.*

In Abbildung 3.9 ist sichtbar, daß im Durchschnitt 2.3 Kanäle über dem Schwellwert 4 ADC-Counts liegen. Diese durchschnittliche Clustersize ist stark von dem gewählten Schwellwert abhängig. Bei einem Schwellwert von 10 ADC-Counts beträgt sie nur noch 1.7 Kanäle.

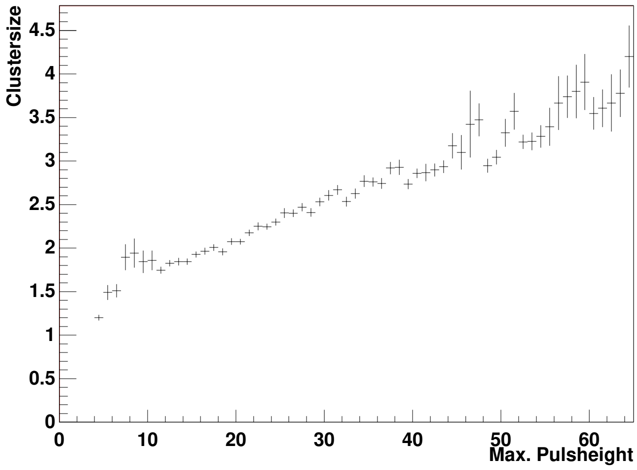


Abbildung 3.10: *Die Clustersize über dem Schwellwert ist in Abhängigkeit von der maximalen Puslhöhe aufgetragen. Wie erwartet sieht man einen linearen Anstieg mit der Pulshöhe.*

Welche "Clustersize über Schwellwert" bei Pulsen mit einer bestimmten maximalen Pulshöhe am wahrscheinlichsten ist, kann in Abbildung 3.10 abgelesen werden. Da die Form der Pulse in erster Näherung unabhängig von ihrer Höhe ist, wächst ein immer größerer Teil des Randbereichs über die Schwelle. Damit läßt sich der Anstieg der Clustersize über Schwellwert mit der maximalen Pulshöhe in Abbildung 3.10 erklären.

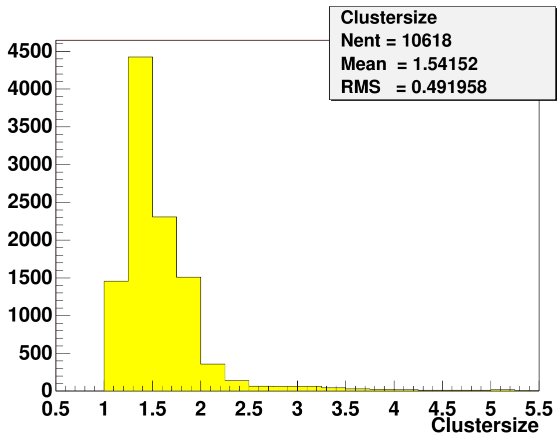


Abbildung 3.11: *In der obigen Abbildung wird die relative Clustersize der Kammer M5 bei einem Schwellwert von 4 ADC Counts gezeigt. Mit Clustersize ist hier das Verhältnis von integrierter Pulshöhe zu maximaler Pulshöhe gemeint.*

In Abbildung 3.11 ist die relative Clustersize für die Kammer M5 bei einem Schwellwert von 4 ADC Counts gezeigt. Die durchschnittliche Clustergröße ist 1.5 Kanäle. Besonders interessant ist jedoch, ob sich die relative Clustersize mit der maximalen Pulshöhe ändert. Wie in Abbildung 3.12 zu sehen ist, hängt die relative Clustersize in dem Bereich zwischen 10 - 40 ADC-Counts nicht von der Pulshöhe ab. Bei einem Übersprechen von Verstärkerkanälen würde diese Kurve in allen Bereichen ansteigen. Der Anstieg im rechten Bildteil kann durch die Beschränkung des ADCs erklärt werden. Große Pulse können oft nicht in ihrer gesamten Höhe aufgelöst werden und somit erscheint der Puls kleiner und die Clustersize steigt.

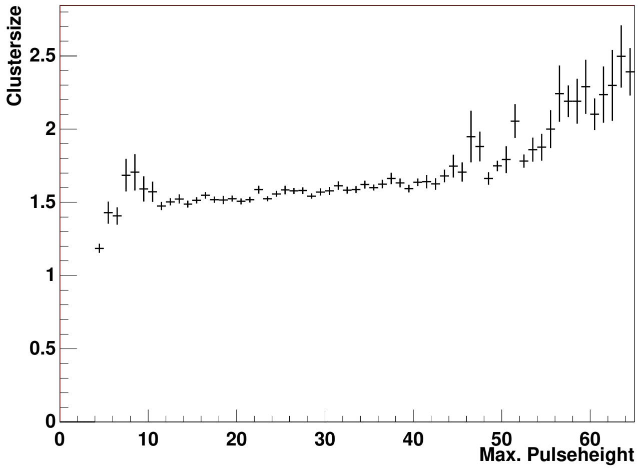


Abbildung 3.12: *In der Abbildung ist der Mittelwert der Clustersize (Int. Pulse/ Max. Höhe) gegen die maximale Pulshöhe aufgetragen.*

3.6 Verschiebung der Kammern

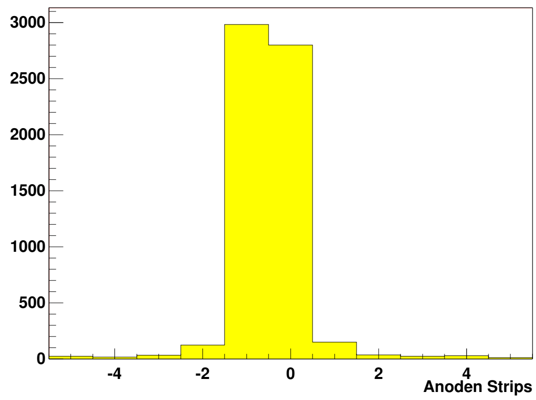


Abbildung 3.13: *Für jeden Hit in Kammer M5 wird der Abstand zum nächsten Hit in der Kammer M6 aufgetragen. Bei Null enthält das Histogramm 3260, bei minus eins 3377 Einträge. Daraus ergibt sich eine Verschiebung von 0.51 Streifen.*

Für die Bestimmung von Effizienzen möchte man eine Kammer als Trigger verwenden und eine Vorhersage über den angesprochenen Kanal in der anderen Kammer machen. Damit kann die Effizienz für jeden einzelnen Kanal bestimmt werden.

Die Vorhersage ist allerdings nur mit bekannter Verschiebung der beiden Kammern gegeneinander möglich. Der Abstand der Hits der Kammern M5 und M6 in Anoden-Streifen ist in

Abbildung 3.13 aufgetragen. Für diese Analyse wurde der Run 274 ausgewählt.

Mit folgendem Algorithmus wurde dieses Histogramm erstellt:

- Es wird in Kammer M5 ein Hit innerhalb des vom Szintillator überdeckten Bereichs gesucht.
- Findet man in M6 im selben Kanal auch einen Hit, wird in das Histogramm eine 0 gefüllt.
- Wenn nicht, wird in den benachbarten Kanälen nach Hits gesucht. Sind zwei Hits gleich weit entfernt, werden die Abstände von beiden Hits in das Histogramm gefüllt, was allerdings nur selten auftritt. Es ist deshalb möglich, bei einem Ereignis zwei Einträge in das Histogramm zu produzieren.

Die Abweichung des Peaks von Null spiegelt die Verschiebung der beiden Kammern gegeneinander wider. Über die Form läßt sich auf den Anteil der zufälligen Pulse schließen.

Die Kammern sind ungefähr ≈ 0.51 Streifen gegeneinander verschoben, was 150 μm entspricht.

Wenn die Hits der zweiten Kammer vollkommen unkorreliert (die Wahrscheinlichkeit einen Hit in einem Kanal zu finden ist überall gleich hoch) mit Hits in der ersten Kammer auftreten würden, sähe man in der Abbildung eine Gaußverteilung. Da nur wenige Pulse außerhalb der Verschiebung 0 und 1 sind, kann davon ausgegangen werden, daß fast alle Pulse in dem Histogram von einem Teilchen stammen, das auch einen Trigger ausgelöst hat.

3.7 Effizienz

Die Effizienz wird auf zwei Arten bestimmt: Zuerst bestimmt man die Effizienz mit einem Szintillator-Trigger und dann im zweiten Abschnitt die Kanal-Effizienz, wobei eine andere Kammer als Trigger dient.

Alle Effizienzbestimmungen wurden mit den Daten des Runs 274 durchgeführt.

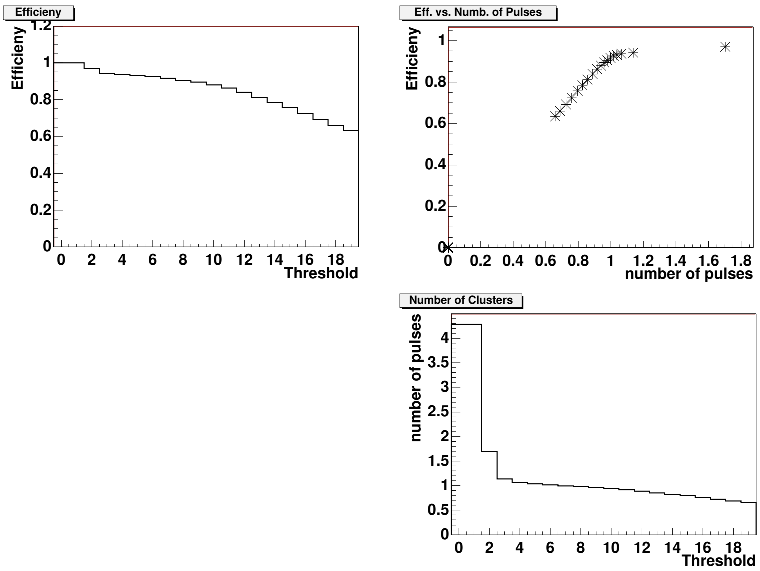
3.7.1 Szintillator-Trigger

Für jede der beiden Kammern M5 und M6 wurden folgende drei Graphiken erstellt, wobei immer nur der vom Szintillator überdeckte Bereich ausgewertet wurde:

- Anzahl von Pulsen in einem Event in Abhängigkeit vom Schwellwert. Bei optimaler Einstellung des Schwellwertes erwartet man einen Puls pro Event.
- Effizienz gegen den Schwellwert.
- Anzahl der Pulse gegen die Effizienz.

Wenn man keine zufälligen Teilchendurchgänge beoachten würde und eine Effizienz von 100 % hätte, erwartete man genau einen Puls pro Event. Wenn man den Schwellwert zu niedrig setzt, detektiert man deutlich mehr als einen Puls pro Event, also auch Rauschen. Bei zu hohem Schwellwert sinkt die Effizienz.

Aus den Abbildungen 3.7.1 und 3.14 kann man ablesen, daß für die Kammer M5 ein Schwellwert von 4 ADC Counts sinnvoll ist, da dann nur 1 Cluster pro Event gefunden wird. Bei diesem Schwellwert ist die Effizienz ≈ 95 %. Der Schwellwert der Kammer M6 sollte auch bei 4 ADC Counts liegen, allerdings sinkt bei Kammer M6 die Effizienz auf 85 %.



Effizienz der Kammer M5, Run 274

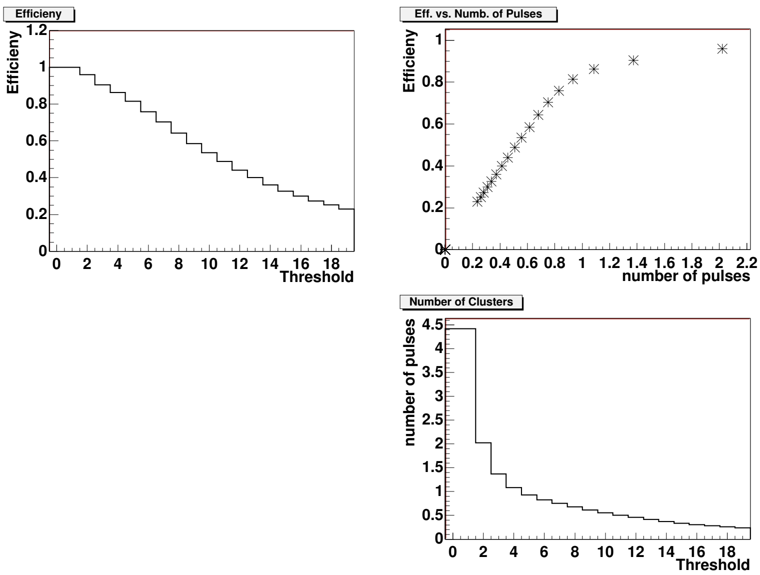


Abbildung 3.14: *Effizienz der Kammer M6, Run 274. In dem Plot links wurde die Effizienz gegen den Schwellwert aufgetragen. Der Plot rechts unten zeigt, wieviele Hits man bei einem Event findet, wenn man einen bestimmten Schwellwert setzt. Wie hoch die Effizienz ist, wenn man eine gewisse Anzahl von Hits bei einem Event hat, ist in der Graphik rechts oben zu sehen.*

3.7.2 Kammer Trigger

Wenn man die Verschiebung der beiden Kammern M5 und M6 gegeneinander kennt, läßt sich bei einem Puls in Kammer M5 vorhersagen, wo auch in Kammer M6 ein Teilchen detektiert wird und umgekehrt. Da diese Vorhersage auf zwei Kanäle genau ist, kann nicht für jeden Kanal einzeln die Effizienz bestimmt werden, da die Kanal-Effizienz mit der Ortsauflösung verschmiert ist. Jedoch kann eine Aussage über die Homogenität der Kanäle gemacht werden.

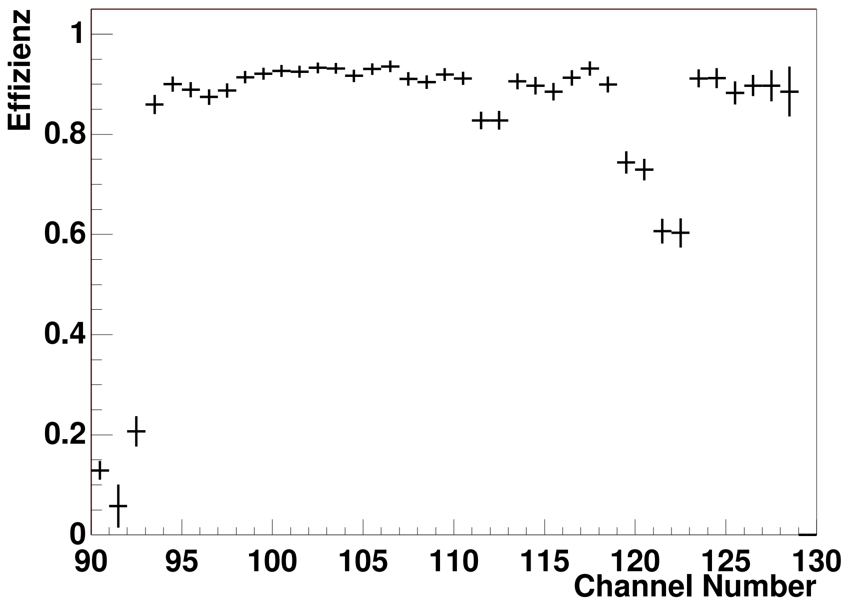


Abbildung 3.15: *Kanal Effizienz der Kammer M5, Run 274,Schwelle = -4*

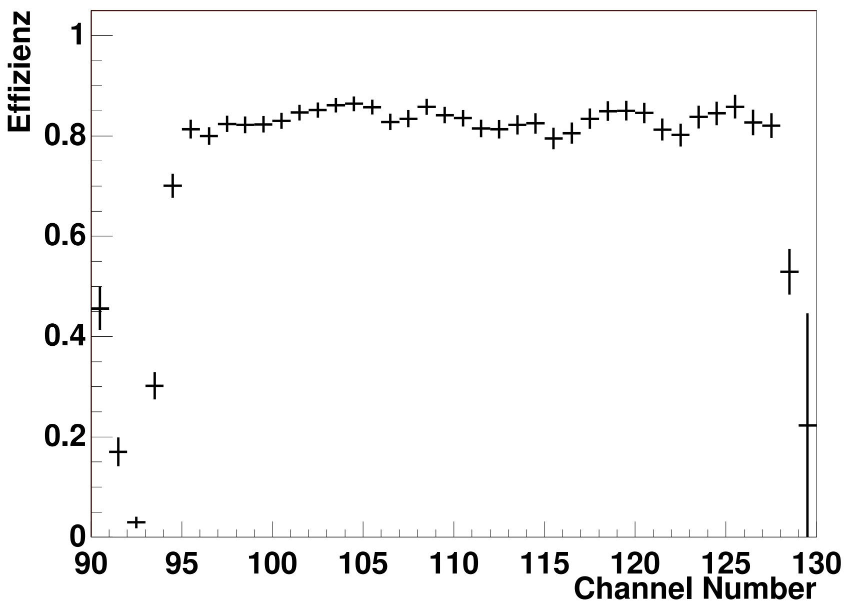


Abbildung 3.16: *Kanal Effizienz der Kammer M6, Run 274, Schwelle = -4*

Die Effizienzen in Bild 3.15 und 3.16 wurden mit Hilfe von zwei Histogrammen erstellt. In dem ersten Histogramm, `hitsSoll`, werden die vorausgesagten Pulse gezählt. Will man also die Kanal-Effizienz von Kammer M5 bestimmen, sucht man in Kammer M6 nach Pulsen. Findet man dort einen Hit, macht man eine Voraussage über die beiden Kanäle in M5 und füllt das Histogramm `hitsSoll` mit beiden Kanälen. Wird in M5 in einem der beiden vorhergesagten Kanäle ein Puls gefunden, werden **beide** Kanäle in `hitsIst` gefüllt.

Am Ende werden die Einträge von `hitsIst` durch `hitsSoll` geteilt und ergeben die leicht verschmierte Effizienz für jeden Kanal.

Die Fehlerbalken der Graphik wurden bei N erwarteten Pulsen und einer daraus resultierenden Effizienz von ϵ wie folgt berechnet [Gra97]:

$$\sigma_\epsilon = \sqrt{\frac{\epsilon(1-\epsilon)}{N}} \tag{3.3}$$

Die Homogenität ist in beiden Kammern groß. Die Abweichungen von Kanal zu Kanal liegen im Bereich der Fehler. Die Einbrüche bei Kammer M5 lassen sich nach einem Blick auf die Wiremap von Kammer M6 erkären. Bei den Einbrüchen gibt es in Kammer M6 heiße Kanäle, weshalb in Kammer M5 vergeblich nach Teilchendurchgängen gesucht wird.

3.8 Zusammenfassung

Erstmals wurde eine größere Datenmenge (mehrere MBytes) mit der kompletten Ausleskette, wie sie auch für den Betrieb in HERA-B vorgesehen ist, analysiert. In der vorliegenden Arbeit wurden Daten aus Runs mit einem π^- Strahl ausgewertet. Pionen mit einer Energie von 350 MeV sind minimal ionisierend, wie auch der größte Teil der bei HERA-B entstehenden Teilchen.

Es wurde gezeigt, daß die vor dem PSI-Test definierten Betriebsspannungen zu niedrig waren. Erst bei deutlicher Erhöhung der GEM-Differenzspannung von 390 V auf 410 V und der Kathodenspannung von 465 V auf 485 V zeichnet sich ein Maximum vom Rauschen ab. Der Einfluß der geänderten Kathoden-Vorwiderstände auf die Gasverstärkung wurde vor dem PSI-Test bei der Festlegung der Betriebsspannungen nicht berücksichtigt.

Die Effizienz wurde bei der erhöhten Spannung berechnet. Es wurde gezeigt, daß die Effizienz bei der Kammer M5 mit 95 % noch durch Erhöhung der Spannung vergrößert werden sollte. Durch die Bestimmung der Kanaleffizienz konnte nachgewiesen werden, daß die Kammer in dem vom Szintillator überdeckten Bereich homogene Verstärkung zeigt.

Die erreichten Effizienzen sind für beide Kammern deutlich tiefer als 99 %, die bei optimaleren Betriebsbedingungen mit einer anderen Readout-Kette erreicht wurden[Hil98].

Es wurde gezeigt, daß mit einer MSGC und der vollständigen, für den Einsatz bei HERA-B entwickelten Readout-Kette Daten in einem Hadronenstrahl genommen und analysiert werden können.

Anhang A

Graphisches User Interface

Das Graphisches User Interface (GUI) ermöglicht die intuitive Steuerung der Slow Control durch die Maus und zeigt graphisch den momentanen Zustand des Detektors an.

Für das Graphische User Interface wurde die Skript-Programmiersprache Tcl/Tk [Ous95] in der Version 7.6 gewählt. Diese stellt Elemente wie Knöpfe, Skalen usw. zur Verfügung und enthält ein C-Interface.

Die graphische Darstellung von zeitlichen Verläufen wird mit der Tcl/Tk Erweiterung BLT [BLT] realisiert. Diese enthält Elemente wie Graphen und Vektoren, für die vielfältige Konfigurationsmöglichkeiten bestehen.

Jedes GUI besteht aus zwei Teilen: einem Interpreter, der für jedes Skript gleich ist, und dem Tcl/Tk Skript. Der hierfür von A. Zhelezov erstellte Interpreter enthält folgende Funktionalitäten:

- Tcl/Tk Interpreter
- die Tcl/Tk Erweiterung BLT für die Online Plots
- Anbindung an das Nachrichtenprotokoll RPM
- Kommunikationsmöglichkeit mit der Parameter Contol
- Funktionalität der State Machine
- Zugriffsmöglichkeiten auf die Datenbank

Jedes GUI kann sich mit einem Parameter Control Programm als *Master* oder als *Client* verbinden. *Client* GUIs können Parameter nur überwachen. Sie können keine Werte oder States der Parameter Control ändern, dieses Recht haben nur die *Master* GUIs. Während des normalen Schichtbetriebs des Detektors sollte das GUI nur im *Client* Modus gestartet werden können. Bisher ist keine Password-Abfrage für *Master* Verbindungen implementiert. Durch eine Änderung der Skripts kann sich so auch eine unbefugte Person als *Master* mit der Parameter Control verbinden. Eine Password-Abfrage ist für eine spätere Version der SPC-Frameworks vorgesehen.

Die Kommunikation mit der Parameter Control erfolgt mittels vordefinierter Funktionen, die im Tcl/Tk Skript aufgerufen werden. Bei jedem Aufruf der Tcl/Tk Funktion `spc_loop` werden automatisch die eingetroffenen Daten bearbeitet und an die unten erläuterten Funktionen übergegeben.

Die Bibliothek, die diese Funktionen zur Verfügung stellt, stammt von A. Zhelezov [Zhe98a] und U. Uwer.

Die Daten der Parameter Control werden in folgenden Funktionen übergeben:

- `spc_NewState` wird bei einer Änderaung des States einer Parameter Control aufgerufen und übergibt den neuen und den alten State als Argument.

- `spc_NewSysInfo` übergibt den Debug Level, den System Name (z.B. `itr/hv`), die Anzahl der Kanäle und die Anzahl der Konfigurations- und Monitorparameter. Diese Funktion wird beim Start des GUIs und nach einem Request (s. 2.2.4) des GUIs aufgerufen.

- `spc_NewChaStatus` wird aufgerufen, wenn sich bei einem oder mehreren Kanälen der Paramter Control der Status geändert hat.

- `spc_NewChaInfo` übergibt einen Kanalnamen und die dazu gehörigen VME Addresse, die VME Basisaddresse und die Hardware ID;

- `spc_NewChaConfig` wird aufgerufen, wenn sich die Konfiguration eines oder mehrerer Kanäle geändert hat. Der Aufruf erfolgt normalerweise in Folge eines Requests (s. 2.2.4). Die Daten werden in dem Array `spc_confData` gespeichert; Elemente sind: Kanal-Name, Kanal-Status, Typ des Parameters, nominaler Wert, aktueller Wert und Wert aus der Datenbank;

- `spc_NewChaMonitor` wird aufgerufen, wenn sich mindestens ein Monitorparameter eines oder mehrerer Kanäle geändert hat oder eine Anfrage des GUI an die Paramter Control erfolgt ist.

Variable	Beispiel	Bedeutung
`serverName`	`spc_a339`	Name des Parameter Control Prozesses
`spc(StateTable)`	`8`	Definiert, welche Konfigurationsdateien für die State Machnine gelesen werden. Wenn `spc(StateTable)` 8 ist, werden die Dateien 8.stt und 8def.stt geladen.
`spc(path)`	`itr/hv`	Definiert den gemeinsamen Teil aller logischen Kanälen.
Liste[a] `configInfo`	`Max. Voltage, Ramp Up`	Erzeugt eine Namensliste mit Konfigurationsparametern. Die Anzahl und Bedeutung muß mit den Parametern in der Konfigurations `config.db` der Paramter Control übereinstimmen.
Liste `monitorInfo`	`Voltage, Current`	Erzeugt eine Namesliste mit Monitorparametern. Die Anzahl und Bedeutung muß mit den Parametern in der Konfigurations `monitor.db` der Paramter Control übereinstimmen.

Bedeutung einiger globaler Variablen der Tcl/Tk Skripte

[a] Eine Liste ist eine Tcl/Tk Datenstruktur

Zu beachten ist, daß Fehler im Tcl/Tk Skript dieser Funktionen nicht zu einer Fehlermeldung führen, sondern abgefangen werden. Für den Entwickler erscheint es so, als ob die Funktion nur bis zu der Fehlerquelle abgearbeitet würde.

Außerdem möchte man Aktionen des Benutzers, wie z.B. das Drücken von Knöpfen des GUIs, mit Anfragen an die Parameter Control verbinden. Hierfür können folgende Request-Funktionen benutzt werden:

- `spc_RqConnect` verbindet das GUI mit der Parameter Control. Als Argument muß der Name des Parameter Control Prozesses und der Modus (*Master* oder *Client*) des GUIs angegeben werden.

- `spc_RqSysAction` kann die Hardware in einen definierten Ausgangszustand bringen (Reset), eine Anfrage über den Zustand des Systems abschicken oder eine benutzerdefinierte Aktion auslösen. Das Reset löst keine Antwort der Parameter Control aus. Der Zustand des Systems wird der Funktion `spc_NewSysInfo` übergegeben und die Antwort auf die benutzerdefinierte Anfrage erfolgt durch `spc_NewUser`.

- `spc_RqChaStatus` liest bei dem Argument `Display Status` den Status des Kanals aus der **internen Datenstruktur** der Parameter Control, wohingegen bei `Read Status` erst noch die aktuellen Daten aus der Hardware gelesen, daraus der Status berechnet und dann der Status an die Funktion `spc_NewChaStatus` übergeben wird.

- `spc_RqChaConfig "Display"` liest die Konfigurationsdaten in der Parameter Control aus der **internen Datenstruktur** und ruft dann im GUI die Funktion `spc_NewChaConfig` mit den Konfigurationsdaten eines Kanals als Argument auf.

- `spc_RqChaConfig "Read"` liest die Konfigurationsdaten in der Parameter Control aus der **Hardware** und ruft dann im GUI die Funktion `spc_NewChaConfig` mit den aktuellen Konfigurationsparametern als Argument auf.

- `spc_RqChaConfig "Set"` setzt die Konfigurationsparameter in der Datenstruktur der Parameter Control und ändert diese auch bei der Hardware. Danach wird die Funktion `spc_NewChaConfig` im GUI mit den neuen Konfigurationsparametern aufgerufen.

- `spc_RqChaMonitor "Display"` liest die Monitordaten in der Parameter Control aus der **internen Datenstruktur** und ruft dann im GUI die Funktion `spc_NewChaMonitor` mit den aktuellen Monitor Parametern als Argument auf.

- `spc_RqChaMonitor "Read"` liest die Monitordaten durch die Parameter Control aus der **Hardware** und ruft dann im GUI die Funktion `spc_NewChaMonitor` auf.

- `spc_RqChaMonitor "Set"` setzt die Monitorparameter sowohl in der Datenstruktur der Parameter Control, als auch bei der Hardware. Anschließend wird als Antwort die Funktion `spc_NewChaMonitor` mit den gesetzten Parametern als Argument aufgerufen.

Die `spc_NewCha*` Funktionen können sowohl als Reaktion auf Anfrage des GUIs, als auch von der Parameter Control selbstständig aufgerufen werden. Die eindeutig Zuordnung von Anfrage und Antwort der Parameter Control ist nicht möglich!

Konfigurationsdateien

Für jedes GUI existiert ein Konfigurationsfile, das eine Konfiguration der Plot-Routinen ermöglicht. Sie heißen: `a339Names.tcl` und `a344Names.tcl`

In dem Array `spc_channel2Name` erfolgt die Zuordnung von den logischen Kanalnamen zu benutzerdefinierten Namen(z.B.`spc_channel2Name(itr/hv/sl1/1) ``Drift MS14 ``). Außerdem kann jedem Kanal mit dem Array `spcColor` eine Farbe für die Plotfunktionen zugeordnet werden. Die Anzahl und Namen der Plots werden in der Liste `spc_Plots` festgelegt. Anschließend kann man jedem Plot eine Anzahl von Kanälen zuordnen.

Die Variable `spc(max_vector_length)` legt die maximale Länge des internen Datenarrays fest, das BLT benutzt.

Die Bedeutung einiger globaler Variablen im Tcl/Tk Skript wird in Tabelle A erklärt.

Anhang B

Hardware der Inner Tracker Slow Control

In diesem Kapitel werden Betriebshinweise für Komponenten der Inner Tracker Slow Control gegeben und die Programme zur Ansteuerung detaillierter beschrieben.

B.1 A339

Die 2×8 Kanal Strommeßbox A339 wurde in der Elektronikwerkstatt des Physikalischen Instituts speziell für den Einsatz des Inner Trackers bei HERA-B gebaut.

Auf der Vorderseite befinden sich drei Knöpfe neben dem LCD-Display, einer links und zwei rechts. Mit dem linken Knopf kann man zwischen verschiedenen Anzeigemodi (momentaner Strom, Alarmlimits, Anzahl der Warnungen, maximaler gemessener Wert) wechseln, die beiden rechten Knöpfe schalten zwischen den 8 Kanälen um. Im Display werden immer die Werte für beide beide Kanäle angezeigt.

Ein Reset des Strommessers löst man aus, indem man gleichzeitige alle drei Knöpfe drückt und den Knopf rechts oben als erstes los läßt. Den Alarm kann man durch gleichzeitiges Drücken der beiden unteren Knöpfe ausschalten.

Auf der Rückseite befinden sich je 9-polige Sub-D Stecker und eine Buchse, der Stecker für den CAN-Bus und die Buchse für die serielle Schnittstelle RS232. Mit Jumpern unter der rechten seitlichen Abdeckung können für die RS232 Schnittstelle die Receive und Transmit Leitung vertauscht werden. Mit einer seriellen Konsole kann die Strommeßbox direkt angesprochen werden. Auf das Kommando **?␣** hin werden alle möglich Befehle mit Erklärkung über die serielle Schnittstelle ausgegeben. Eine Auswahl der Befehle:

- **Relay A ON/OFF:** schaltet das Relais für Kanal A;
- **Relay B ON/OFF:** schaltet das Relais für Kanal B;
- **Channel Set/Get:** Setzt bzw. liest den den auf dem Display angezeigten Kanal an.
- **Display text at position:** schreibt Text auf das Display des A339 Moduls. Überschreibt die vorherige Anzeige;
- **Get Keys:** Gibt an, welche der drei Knöpfe gerade gedrückt sind.
- **Gain=Shunt..:** Ermöglicht eine Änderung des Widerstandwertes, der für die interne Umrechnung von DAC Werten in Ströme benutzt wird.

- **HV On/OFF:** Schaltet den Alarm und beide Relais an bzw. aus. Alarm wird durch ein Blinken den Anzeige signalisiert. Der Status einer Box wird auch über ein TTL-Signal angezeigt, das im Normalbetrieb HIGH ist.

- **Currents Channel n/All:** Gibt den Strom eines bzw. aller Kanäle mit Einheit aus. Bei der Umrechnung von ADC Werten in Ströme wird der programmierte Widerstand berücksichtigt.

- **Key Lock/Unlock:** Sperrt die Knöpfe an der Frontseite des Moduls

- **Mode Set/Get:** Schaltet zwischen den verschiedenen Anzeigemodi des Displays (Strom, bisher gemessener Strombereich, Limits, Warnings)

- **Raw Data:** Gibt die ADC Werte aus. Aus diesen wird mit den Widerstandswerten der Strom berechnet.

- **Range Channel:** Zeigt den maximalen und minimalen gemessen Strom der Submodule A und B an.

- **Average Count Set/Get:** Hier kann die Anzahl der Messwerte angegeben werden, über die gemittelt wird bevor mit den Alarm/Warning-Grenzen verglichen wird. Die Messfrequenz variiert jedoch nach der sonstigen Belastung des Einschubs durch externe Abfragen, liegt jedoch in der Größenordnung von 10 Hz.

- **Warnings Channel (A,B):** Zeigt die Anzahl der bisherigen Überschreitungen der Stromlimits für jedes Submodul an.

- **Monitor ON/OFF:** Wenn Monitor angeschaltet ist, springt bei einem Alarm die Anzeige auf den auslösenden Kanal.

- **Reset Range:** Setzt die maximalen bzw. minimalen Stromwerte wieder auf 0.

- **Reset Warnings:** Setzt den Zähler der aufgelaufenen Warnungen wieder auf 0.

B.2 A344

Das von der Elektronik-Werkstatt des Physikalischen Instituts entwickelte GEM-Verteiler Modul besteht im Prinzip aus acht einzeln programmierbaren Spannungsteilern. Diese Differenzspannungen werden aus einer Eingangsspannung hergestellt.

Die Spannungen werden an die GEM-Folie angelegt und sorgen so für die Gasvorverstärkung in der GEM-Folie.

Es gibt unter anderem folgende Befehle:

- **Set/Get voltage:** ermöglicht die Kalibration der Spannungsmessung in beiden Submodulen. Außerdem kann die Spannung an jedem Kanal der einzelnen Submodule abgefragt werden.

- **Channel Set/Get:** Der im Display angezeigte Kanal läßt sich so einstellen bzw. abfragen.Dies kann aber auch, wie beim A339, mit den Knöpfen links neben dem Display geschehen.

- **Get Input voltage:** Die Spannung vor dem Spannungsteiler wird für den jeweiligen Kanal bzw. mit der Kanalnummer 0 für alle Kanäle abgefragt.

- **List all ADCs, DACs**: listet alle DAC/ADC Werte und Spannungen auf. Alle Informationen über den momentanen Zustand der Box werden angezeigt.
- **Set/Get A-B voltages:** Erlaubt es, die zu erzeugende Differenzspannung anzugeben. Diese muß negativ sein! Das Gerät kann allerdings nur einen bestimmten Spannungsbereich einstellen. Ist es nicht möglich, den gewünschten Wert zu erreichen, erscheint ein "*" links oben im Display. Die tatsächlich eingestellte Spannung kann mit "v" abgefragt werden.

B.3 Beschreibung der Bibliothek A339 und A344

Für die Strommeßbox A339 und die GEM-Verteilerbox A344 wurde im Rahmen dieser Arbeit eine Bibliothek aufgebaut, die über die VME-RS232-Karte DL620 eine zuverlässige Kommunikation zwischen dem Programm auf der VME-CPU und den Boxen erlaubt.

Die Boxen (im folgenden ist mit Boxen die Strommeßbox und die GEM-Verteilerbox gemeint) antworten nur auf Anfragen über die serielle Schnittstelle, schicken also von sich aus keine Daten. Der übliche Informationsweg ist, daß die VME-CPU der DL620-Karte über den VME-Bus einen Befehl für die Box schickt. Die DL620-Karte schickt diesen Befehl über die Schnittstelle RS232 an die Box und speichert die Antwort der Box in einem internen Speicher zwischen. Die VME-CPU liest dann die Zeichen aus der DL620-Karte.

Eine Schwierigkeit besteht darin, daß es zwischen den Boxen und der DL620-Karte, kein Hardware-Handshake gibt und die DL620-Karte einen nur 8 Zeichen großen FIFO[1] besitzt. Wenn mehr als 8 Zeichen von der Box gesendet und nicht schnell genug aus der DL620-Karte von der CPU abgeholt werden, gehen die übrigen Zeichen verloren. Das Auslesen der Zeichen aus der DL620-Karte ist zeitkritisch!

Daher wird die Kommunikation zwischen der VME-CPU und der DL620-Karte mit Interrupts gesteuert. Ein Interrupt ist hier ein von der DL620-Karte ausgelöstes elektronisches Signal, wonach in dem Programm auf der VME-CPU in einer vom Betriebssystem garantierten Zeit eine Routine aufgerufen wird. Dies geschieht asynchron zu der "normalen" Abarbeitung des Programms. Ein Interrupt signalisiert der CPU, daß in dem Zwischenspeicher der DL620-Karte ein Zeichen für sie vorliegt. Daraufhin wird das Zeichen von der CPU aus der DL620-Karte gelesen und in einem Ringbuffer gespeichert. Da somit geleichzeitig auch den Ringbuffer zugegriffen werden kann, ist ein Schutzmechanismus notwendig, der den Zugriff regelt. Hierfür werden in dem Programm Semaphore[2] benutzt.

Bei relativ hoher Zugriffsfrequenz der beiden Programme `spc_a339` und `spc_a344` auf den VME-Bus gehen Zeichen im Zwischenspeicher der DL620-Karte verloren. Um zu gewährleisten, daß immer nur ein Programm auf den VME-Bus zugreift, wurde ein gemeinsamer Semaphor, ein sogenannter "shared semaphore" eingeführt. Aus einem Dateinamen und einer Zahl wird ein "Key" generiert, der den gemeinsamen Semaphor in beiden Programmen beschreibt. Dieser Semaphor wird in dem Programm `spc_a339` erzeugt. Ob ein gemeinsamer Semaphor gesetzt ist, kann mit dem Programm `ipcs`[3] (Inter Process Communication Status) gesehen werden. Da es in dem SPC-Framework noch keine Funktion gibt, die beim Beenden des Programms aufgerufen wird, muß das Löschen des Semaphors vor dem erneuten Programmstart von `spc_a339` mit `ipcrm -s 19`[4] durch den Benutzer geschehen.

Das Programm `spc_a339` öffnet einen TCP/IP-Port, über den Stromwerte abgefragt werden können. Mehrmals in der Sekunde wird überprüft, ob ein Programm versucht sich mit diesem

[1] First In First Out: ein Pipeline-Konzept

[2] [grch. Zeichenträger] der oder das, Mast mit schwenkbaren Armen zum Angeben von opt. Signalen, z.B. zur Nachrichtenübertragung von Schiff zu Schiff [Bro89]

[3] s. Onlinedokumentation mit `man ipcs`

[4] s. Dokumentation mit `man ipcrm`

Port zu verbinden. Das anfragende Programm sendet zuerst den Kanalnamen und die Anfrageart, z.B. `CURRENT itr/hv/sl1/1`. Aus dem Kanalnamen wird die Zeichenkette zwischen dem zweiten und dritten Schrägstrich, hier `sl1` extrahiert. In sämtliche Kanalnamen des Programms `spc_a339` wird nach dieser Zeichenkette gesucht und die Ströme dieser Kanäle in nA werden an das anfragende Programm zurück gegeben. Diese Zuordnung wurde gewählt, da zu einem Kanal der Hochspannungsversorgung mehrere Ströme gehören.

Beschreibung von Variablen

Für die VME-Addressen gibt es in der Struktur `spc` zwei Elemente, `vmemod` und `vmeaddr`. Die VME-Addresse in der Konfigurationsdatei steht in `vmemod`, `vmeaddr` gibt jedoch die Baseaddress an, auf die das Programm zugreifen soll. Die Verbindung der beiden erfolgt so:

```
for(i=0;i<nchannel;i++){
  chano=cha[i].no;
  vmead = spc.channel[chano].namead->vmemod;
  spc.channel[chano].namead->vmeaddr = vmea16d16init(vmead, 0xFF);
}
```

wobei `vmea16d16init` eine Funktion aus der VME-Bibliothek[Kih97a] von T. Kihm ist.

`volatile char buffer[BUFFER_SIZE]`	Ringbuffer zum Speichern der ankommenden Zeichen
`int readPtr, writePtr`	Read und Write Pointer des Ringbuffers
`csem_t sem_desc1`	Semaphor, der den Zugriff auf den Ringbuffer verwaltet
`char port[8]`	Port, auf dem ein anderes Programm Strominformationen aus der Strommeßbox lesen kann.

Defines

```
/* Registers of the VME-Modul DL620 */
#define RDWR          0x00    /* Read/Write Register */
#define IER           0x02    /* Interrupt Enable Register */
#define IIR           0x04    /* Interrupt Identification Register */
#define PR            0x06    /* Protocoll Register */
#define MCR           0x08    /* Modem Control Register */
#define LSR           0x0A    /* Line Status Register */
#define MSR           0x0C    /* Modem Status Register */
#define FCR           IIR     /* FIFO Controll Register */
#define DLL           RDWR    /* Divisor Latch Low */
#define DLM           IER     /* Divisor Latch High */

/* Registers of the VME-Basismodul DL600 */
#define ICR           0x82    /* Interrupt Control Register */
#define IP            0x04    /* Interr. logik enabled/IRQ4 */
#define IM            0x00    /* RORA Mode                  */
#define M             15      /* enable IR from Module 3   */
```

Beschreibung von Funktionen

Im folgenden werden einige wichtige Funktionen der Bibliotheken zur Ansteuerung der Strommeßboxen und der GEM-Verteilerboxen beschrieben. Da bei beiden Boxen die Programme sehr ähnlich sind, wird auf eine Unterscheidung verzichtet.

`void sendCommand(long addr, int cmd, char channel, char *data, char *str, int strLen, int *error)`	wird aufgerufen, um den Befehl `cmd` an die VME-Adresse `addr` zu senden. Zu einem Befehl gehört häufig noch ein anzusprechender Kanal `channel` und eine Zeichenkette `data`, in der z.B. die Stromgrenzen stehen. Aus der Box ausgelesene Daten werden bis zu einer Länge von `strLen` in `str` geschrieben.
`int getStrinIRQ1(long addr, char *out, int n):`	Die Funktion liest eine Zeichenkette aus dem Ringbuffer bis zu dem Zeichenkettenendzeichen `\0`. Die Zeichenkette wird in `out` gespeichert, wobei maximal `n` Zeichen gelesen werden. Die globale Varible `TIMEOUTTIME` gibt die Zeit in Sekunden an, die für für das Lesen der Zeichenkette benötigt werden dürfen. `addr` wird nicht mehr benötigt.
`char getChar(void)`	liest ein Zeichen aus dem Ringbuffer, wird von `getStrinIRQ1` mehrmals aufgerufen.
`void initRS232(long addr)`	initialisiert die RS232-Schnittstelle auf der DL620-Karte, setzt die Baud-Rate und ermöglich die Benutzung des FIFOs.
`short WriteRS232(long addr, char c)`	schreibt das Zeichen `c` in die Addresse `addr`, wobei solang in dem Register `LSR` geschaut wird, bis die DL620-Karte durch Setzen des ersten Bits vom Register `LSR` zeigt, daß sie schon bereit für das nächste Zeichen ist. Bei Überschreiten der `TIMEOUTTIME` wird die Routine mit einer Fehlermeldung abgebrochen.
`int [a339,a344]_start_up()`	wird zu Beginn der Programme aufgerufen und initialisiert die Semaphor. Außerdem wird eine Routine zum Abfangen von Signalen gestartet. Wenn der Benutzer mit `Ctrl-C` das Programm beendet, wird vorher noch die Routine `sig_handler` aufgerufen, die Semaphor löscht und Interrupts wieder frei gibt. Auch der Thread für die Abfrage von Stromwerten mit anderen Programmen wird hier (nur bei A339) gestartet.
`void interrupt_handler(void)`	wird bei jedem Interrupt aufgerufen und liest aus der DL620-Karte ein Zeichen und schreibt dies in den Ringbuffer.
`void CAEN_loop(void)`	wird von dem Thread zehn mal in der Sekunde aufgerufen und überprüft, ob ein anderes Programm versucht, eine Verbindung herzustellen.

B.4 CAEN

Die Hochspannung für den Inner und Outer Tracker, RICH und das Myonen System wird von einem *CAEN* Hochspannungsgerät erzeugt, welches aus einem Einschubrahmen mit Steuereinheit (*CAEN SY 527*) und 5 Einschüben besteht.

Bei dem Betrieb der Einschübe ist zu beachten, daß die Lemo Stecker auf der Frontplatte der Einschübe für den Betrieb auf Masse liegen müssen (man kann dies durch einen 50 Ω Abschlusswiderstand realisieren). Die Steuerung und Überwachung der Spannungen kann sowohl über eine RS232 Schnittstelle/Serielle Konsole als auch über ein *FAST CAEN NET* VME Interface Karte erfolgen.

Beide Steuerungen sind gleichberechtigt und schließen sich nicht aus!

Der VME Zugriff auf die *FAST CAEN NET* Karte wird von einer Bibliothek von einer zentral entwickelten Bibliothek ermöglicht [Atk98]. Für den Outer Tracker wurde ein Beispielprogramm für die Steuerung der Hochspannung mit einer Parameter Control bereit gestellt. Die Anpassung dieses Programms an die Bedürfnisse des Inner Trackers wird im folgenden Abschnitt beschrieben.

Das Programm `spc_itr_hv`

Die als Konfigurationsvariable definierte Hochspannung für einen Kanal soll in mehreren Schritten erreicht werden. Nach jedem Schritt soll eine bestimmte Zeit gewartet werden und aus dem Strommeßprogramm die Ströme ausgelesen werden. Nur wenn die Stromwerte innerhalb definierter Grenzen sind, wird die Spannung weiter erhöht.

Die Spannungschritte für alle Kanäle stehen in der Datei `rampUp.Dat`. Ein Ausschnitt aus dieser Datei sieht wie folgt aus:

```
#names of channels which should be ramped up
CHANNELNAMES
itr/hv/sl1/1
itr/hv/sl2/2
END
# name      voltage  low. Li. Wa. , up. Li. Wa, low Al. , up. Al, wait[secs]
RAMPUPDAT
itr/hv/sl1/1    100        -1            10        -1         10        20
itr/hv/sl1/1    200        -1            10        -1         10        20
itr/hv/sl1/1    300        -1            10        -1         10        20
itr/hv/sl1/1    400        -1            10        -1         10        20
itr/hv/sl2/2     0         -10            0         -1         1        20
itr/hv/sl2/2     0         -10            0         -1         1        20
itr/hv/sl2/2     20        -10           60         -1         1        20
itr/hv/sl2/2     40        -10           60         -1         4        20
END
```

Die Ströme in der gesamten Inner Tracker Software werden in nA angegeben!

Bei Programmstart wird die Konfigurationsdatei eingelesen und in einer dynamisch erzeugten Liste gespeichert.

Der State `PREPARE` ist für das schrittweise Hochfahren der Kammerspannungen vorgesehen. In diesem State wird die Routine `configureCAEN` aufgerufne. Für jeden Kanalnamen wird die erste Spannung (hier 100 V und 0 V) eingestellt und danach 20 Sekunden gewartet. Nun werden die beiden Kanalnamen `itr/hv/sl1/1` und `itr/hv/sl2/2` an das Programm `spc_a339` geschickt. Dies sucht wie oben beschrieben die Ströme zu den Kanalnamen, die `sl1` bzw `sl2` enthalten, heraus und das Programm `spc_a339` gibt die aktuellen Stromwerte zurück. Nun werden diese mit den Stromgrenzen verglichen. Wenn der Strom eines Kanals außerhalb den Grenzen für eine Warnung liegt, ändert sich der Status des Kanals zu `WARNING`, die Spannung wird jedoch weiter hochgefahren. Erst wenn auch die Alarm-Grenzen überschritten werden, fährt das Gerät die Hochspannung auf 0 V herunter.

Während des Hochfahrvorgangs ist das Programm nicht steuerbar. Benutzeraktionen werden erst nach dem Verlassen der Funktion `configureCAEN` abgearbeitet. Dies Problem wird nach dem setzen der Umgebungsvariable `ASK_USER` aus `YES` umgangen. Nach jedem Schritt kann der Benutzer den Vorgang abbrechen.

Es ist darauf zu achten, daß die Spannungen in dem Konfigurationsfile `config.db` mit den Endspannungen der Datei `rampUp.Dat` übereinstimmen!

Wenn beim kompilieren für den Präpozessor `NO_A339_CHECK` definiert ist, wird der Stromwert nicht bei dem Strommeßprogramm abgefragt!

Anhang C

Die ROOT Klassen

Die Daten wurden bei dem PSI-Test im Oktober 1998 in Text-Dateien geschrieben. Vor der Analyse werden von einem Konvertierungsprogramm die ASCII-Dateien in die Datenstruktur des Analysepaketes ROOT [ROOT], die ROOT-Trees, gefüllt. ROOT-Trees sind die ROOT-eigenen Datenstrukturen, in denen neben den Rohdaten auch Informationen über das jeweilige Event (Event Nummer, Run Nummer ...) und Daten aus den ersten Analyseschritten, wie z.B. die Columnnumber, gespeichert werden. Ein Tree enthält die Information eines gesamten Runs, ist aber in Strukturen, die den einzelnen Events entsprechen, untergliedert. Die ROOT-Trees entsprechen den Ntupels bei PAW [PAW].

C.1 Objektorientiertes Programmieren

Da die bei ROOT benutzte Programmiersprache C++ ist, wird kurz objektorientiertes Programmieren erläutert und wichtige Begriffe beschrieben.

Objektorientiertem Programmieren liegt ein Programmiermodell zu Grunde, das ein Programm als Sammlung diskreter **Objekte** betrachtet, das heißt als in sich abgeschlossene Sammlungen von Datenstrukturen und Routinen, die mit anderen Objekten kommunizieren.

Ein Objekt ist eine gekapselte Datenstruktur, die einerseits Datenkomponenten enthält (wie eine `struct`-Variable in C), andererseits aber auch alle zu ihrer Bearbeitung dienenden Funktionen. In der objektorientierten Theorie wird nicht von Funktionen, sondern von **Methoden** gesprochen. Eine **Klasse** ist ein Typ eines Objekts. Sie beschreibt Typ und Aufbau der Datenkomponenten und Methoden. Von den Datenkomponenten gibt es einen Satz pro Objekt, die Methoden sind an die Klassen gebunden. Der Zugriff auf die Daten eines Objekts unterliegt einer "Zugriffskontrolle", für die auch der Begriff **Kapselung** verwendet wird.

Neue Klassen können aus bereits definierten gebildet (**abgeleitet**) werden, indem Komponenten hinzugefügt oder modifiziert werden. Eine neue Klasse **"erbt"** die strukturellen Eigenschaften und Methoden einer anderen, der sogenannte Basisklasse.

Als **Polymorphismus** bezeichnet man die Eigenschaft einer Methode, in verschiedenen Zusammenhängen unterschiedlich zu wirken. Die Funktion einer Methode hängt von dem damit verbundenen Objekt ab. Die Methode "+" kann für die Klasse der ganzen Zahlen anderes wirken als bei der Klasse der komplexen Zahlen.

Für die Analyse der Daten fiel die Entscheidung für eine objektorientierte Programmiersprache, da durch die Definition der Klassen "Event" und "Helix" eine gute Strukturierung der Daten möglich ist. Daneben verbessert die Verwendung von Methoden die Übersichtlichkeit, und die Arbeit von mehreren Personen an einem Projekt wird durch die Definition von Methoden und Schnittstellen erleichert.

C.2 Die ROOT Klassen

Im Folgenden wird eine Auswahl der erstellten ROOT Klassen beschrieben.

C.2.1 TItrCluster

Die Klasse **TItrCluster** enthält alle Informationen über einen gefundenen Puls (Cluster). Zu Beginn der Analyse werden in den Daten Pulse gesucht und gespeichert. In den folgenden Analyseschritten braucht nur noch auf diese Klasse der Pulse zugegriffen werden wenn Histogramme aus Daten der Pulse erstellt werden sollen.

class **TItrCluster**: public TObject

public:

class TItrCluster	**TItrCluster**(void)
class TItrCluster	**TItrCluster**(Int_t pos, Int_t start, Int_t stop, TArrayF& data)
class TItrCluster	**TItrCluster**(Int_t ipos, Int_t istart, Int_t istop, Float_t iMaxHeight, Float_t iCOG, Float_t iIntegral)
Int_t	**GetMaxPos** (void)
Int_t	**GetStart** (void)
Int_t	**GetEnd** (void)
Int_t	**GetCSize** (void)
Float_t	**GetMaxHeight**(void)
Float_t	**GetCOG** (void)
Float_t	**GetIntegral** (void)

Data Members

protected:

Int_t	**fMaxPos**	*Position des Maximums*
Int_t	**fStart**	*Start des Clusters*
Int_t	**fEnd**	*Ende des Clusters*
Int_t	**fCSize**	*Cluster Größe*
Float_t	**fMaxHeight**	*Maximale Pulshöhe*
Float_t	**fCOG**	*Schwerpunkt des Clusters*
Float_t	**fIntegral**	*Integrierte Pulshöhe*

Klassenbeschreibung

TItrCluster(void)
Default Konstruktor

TItrCluster(Int_t pos, Int_t start, Int_t stop, TArrayF& data)
Diesem Konstruktor wird die Position der maximalen Pulshöhe, der Start- und Stopkanal und ein Array mit den Daten übergeben.

TItrCluster(Int_t ipos, Int_t istart, Int_t istop, Float_t iMaxHeight,
Float_t iCOG, Float_t iIntegral)

Dieser Konstruktor kann benutzt werden, wenn der Schwerpunkt und die integrierte Pulshöhe bereits berechnet worden sind.

C.2.2 TItrEvent

Die Klasse **TItrEvent** enthält alle Informationen, die ein Event charakterisieren. Diese Klasse wird bei der Verarbeitung der Daten als erstes gefüllt.

class **TItrEvent**: public TObject
public:

class TItrEvent	**TItrEvent**(void)
class TItrEvent	**TItrEvent**(istream & inIStream)
bool	**ChooseNColNo**(void)

Data Members

public:

RunNo_t	**fRunNo**	*Run Nummer*
EventNo_t	**fEventNo**	*Event Nummer*
ColNo_t	**fColNo**	*Spaltenndummer, in der das Ereignis im Helix gespeichert wurde.*
Bool_t	**fGood**	*Ist wahr, wenn bei allen Helix Daten eines Events die gleiche Spaltennummer gefunden wurde.*
TObjArray*	**fFEDDatas**	
TObjArray*	**fHelixDatas**	

C.2.3 TItrHelix

class **TItrHelix** : public TItrFED
public:

class TItrHelix	**TItrHelix**(void)
class TItrHelix	**TItrHelix**(SharcNo_t, LinkNo_t, HelixNo_t)
class TItrHelix	**TItrHelix**(const TItrFED& inFED, HelixNo_t inHelixNo)
class TItrHelix	**TItrHelix**(const TItrHelix& inHelix)
ULong_t	**HelixID**(SharcNo_t inSharcNo, LinkNo_t inLinkNo, HelixNo_t inHelix)
virtual ULong_t	**Hash**(void)

Data Members

public:

HelixNo_t	fHelixNo	*Helix Nummer: 0 oder 1.*

C.2.4 TItrHelixData

class **TItrHelixData** : public TItrHelix
public:

class TItrHelixData	**TItrHelixData**(void)
class TItrHelixData	**TItrHelixData**(TItrFEDData* inFEDData, HelixNo_t inHelixNo,
adsf	const TItrFEDCor* inFEDCor)
class TItrHelixData	**TItrHelixData**(TItrFEDData* inFEDData, HelixNo_t inHelixNo,
nsl	const TItrFEDCor* inFEDCor, const TItrHelixCor* inHelixCor)
void	**ReCalcCorrSignals**(Signal_t inThresold)
void	**SearchPulses** (int thresh)
inline Int_t	**GetNumClusters**(void)

Data Members

public:

TItrFEDData	***fFEDData**	
TArrayF	**fSignals**	*Signale vor der Korrektur*
TArrayF	**fCorrSignals**	*baseline korregierte Signale*
TObjArray	**fCluster**	*Array der gefundenen Cluster*
Int_t	**fNumClusters**	*Anzahl der Cluster*

void **SearchPulses** (int thresh)
Sucht in den Daten des jeweiligen Helix nach Pulsen deren Höhe mindestens `thresh` ist. Die gefundenen Pulse werden in dem Array `fCluster` gespeichert.

C.3 Erfahrungen mit ROOT

Die im Folgenden beschriebenen Erfahrungen wurden während der ungefähr sechswöchigen Analyse mit ROOT gesammelt. Die Wertungen stellen eine persönliche Meinung dar.

ROOT kann auf drei verschiedene Arten benutzt werden:

- Jeder Befehl wird einzeln in der Kommandozeile eingegeben. Da die Befehle der C++ Syntax gehorchen, ist dies sehr mühsam und für eine Analyse nicht geeignet. Änderungen im graphischen Erscheinungsbild (Anordnung der Plots, Farben, Liniendicke usw.) können so jedoch gut durchgeführt werden.

- ROOT kann auch Dateien, die viele Befehle enthalten, laden und interpretieren. Hierbei wurden guten Erfahrungen gemacht. Die Geschwindigkeit ist genügend hoch für nicht zu komplexe Analyseschritte.

- Die dritte Möglichkeit ist, ein Programm zu kompilieren, das aus den eigenen ROOT-Befehlen und der ROOT-Bibliothek besteht. Das so entstandene Programm ist die schnellste der drei Möglichkeiten. Nach jeder Änderung im Analyseverfahren muß das Programm allerdings neu kompiliert werden. Solche Programme wurden für die ersten beiden Analyseschritte verwendet. Hierbei traten jedoch nicht dokumentierte Inkompatibilitäten auf, so daß die Programm nur auf bestimmten Rechnern laufen konnten. Die Ursache hierfür wurde bisher nicht gefunden.

Die Benutzung der C++ Syntax hat die Erstellung von Analysemakros erleichtert. Allerdings war die Komplexität von ROOT für eine schnelle Analyse der Daten zu groß.

Anhang D

Verzeichnis der Dateien

230-241: Triggerdelay Scans 2000 events, low gain, x=-9.5, y=-25
243-250: Phasedelay Scans 2000 events, low gain, x=-9.5, y=-25
251-258: Phasedelay Scans 2000 events, low gain, x=-21.9, y=-25
288-298: Bxdelay Scans 2000 events, high gain, x=5, y=-20.3
299-304: Bxdelay Scans 2000 events, high gain, x=5, y=-20.3
305-311: Bxdelay Scans 2000 events, med gain, x=5, y=-20.3
312-318: Bxdelay Scans 2000 events, low gain, x=5, y=-20.3
319-325: Bxdelay Scans 2000 events, med* gain, x=5, y=-20.3 $U_{GEM} = 410$, $U_{Kathode} = 465$
336-348: Trigger studies Vcomp
Grounding festival
349-368: Trigger studies, delay scans

Run Bereich	(Sharc, Link)	Kammer	PCB	Notes
- 325	(5, 0)	M6	Mitte	
	(5, 1)	M5	Mitte	
	(5, 2)	M8	Mitte	*
326 - 335	(3, 0)	M8	Mitte	*
	(4, 0)	M5	Links	
	(4, 1)	M5	Mitte	
	(4, 2)	M5	Rechts	
	(5, 0)	M6	Links	
	(5, 1)	M6	Mitte	
	(5, 2)	M6	Rechts	
336 - 350	(5, 0)	M6	Mitte	Triggertreiber-Test-
	(5, 1)	M5	Mitte	runs mit Testpulsen
351 - 369	(5, 0)	M6	Mitte	Trigger-Tests mit
	(5, 1)	M5	Mitte	Teilchen
370 - 376	(5, 0)	M6	Mitte	
	(5, 1)	M5	Mitte	
	(5, 2)	M7	Mitte	Beschichtetes GEM, *
377 - 379	(5, 0)	M6	Mitte	
	(5, 1)	M5	Mitte	
	(5, 2)	M8	Mitte	*

Tabelle D.1: *Zuordnung der Sharc-Links zu den Kammern bei bestimmten Runs. Das Zeichen * bedeutet "High gain" Verteilerboard. Rechts und links beziehen sich auf die Sicht durch den Wafer auf den Deckel. Die Tabelle wurde von W. Gradl zusammengestellt.*

Size	Run	Strahl	Zweck	x	y	Gain	U_{GEM}	$U_{Kathode}$	Events	Trig	Coll
72801304	261	π^-		5	25	low	390	465	20000	89	100
7275679	262	π^-		15	-25	low	390	465	2000	89	100
21901602	265	π^-		5	-25	med	390	480	6028	89	600
134962503	266	π^-		5	-25	med	390	480	40k	89	600
145700616	267	π^-		5	-25	med	390	480	40k	89	600
73899728	268	π^-		5	-25	low	390	465		89	100
42299729	270	π^-		5	-25	low	390	465		189	100
32789180	271	π^-	gain	5	-25	low	390	465	9018	189	400
28172813	272	π^-	gain	5	-25	low	390	465	7751	189	100
33001804	273	π^-	gain	5	-25	med	390	485	9074	189	100
28172813	274	π^-	gain	5	-25	high	410	485	8000	189	100
79780971	275	π^+	gain	5.6	-10.4	high	410	485	21921	89	100
113494481	276	π^+		5	-10	high	410	485	31119	189	150
38517196	277	π^+	high rate	5	-20	low	390	465	10k	189	200
68963164	278	π^+	high rate	5	-20	low	390	465	10k	189	200
1739667	279	π^+	high rate	5	-20	low	390	465	lowvolt	189	200
36527078	280	π^+	high rate	5	-20	low	390	465	10032	189	200
51066779	281	π^+	high rate	5	-20	med	390	480	14024	189	200
61612754	284	π^+	high rate	5	-20	low	390	465	2000	189	600 Vfp156
42090518	326	π^+	Scan y	5	100	low	390	465	5000	89	70
42083976	327	π^+	Scan y	5	30	low	390	465	5000	89	70
42096647	328	π^+	Scan y	5	-30	low	390	465	5000	89	Coll. wrong
25641007	330	π^+	Scan y	5	-100	low	390	465	3000	89	70
25252117	331	π^+	Scan y	5	-30.5	low	390	465	3000	89	70
25249410	333	π^+	Rand	100	-80	low	390	465	3000	89	70
25262070	334	π^+		-148	-35	low*	390	475	3000	89	70
21307392	359	π^+	Trigger	11.5	-25.5	high	410	485		89	70
132522763	359a	π^+	Trigger	11.5	-25.5	high	410	485		89	70
4609018	360	π^+	Trigger	11.5	-25.5	high	410	485		89	70
7484854	361	π^+	Trigger	11.5	-25.5	high	410	485		89	70
5234746	362	π^+	Trigger	11.5	-25.5	high	410	485		89	70
2511607	363	π^+	Trigger	11.5	-25.5	high	410	485		89	70
5008385	364	π^+	Trigger	11.5	-25.5	high	410	485		89	70
4962795	365	π^+	Trigger	11.5	-25.5	high	410	485		89	70
5013953	366	π^+	Trigger	11.5	-25.5	high	410	485		89	70
4998495	367	π^+	Trigger	11.5	-25.5	high	410	485		89	70
5005689	368	π^+	Trigger	11.5	-25.5	high	410	485		89	70
599536	369										
18443879	370	pπ^+									
1772945	371	pπ^+	M7MSGC mode	15	0	MSGC	2.8	650	1400		70
7913933	372	pπ^+	M7MSGC mode	15	0	MSGC	2.8	650	6337		70
2859874	373	pπ^+	M7MSGC mode	30	0	MSGC	2.8	650	2292		70
635202	374	pπ^+	M7MSGC mode	30	0	MSGC	2.8	600	480		70
14064400	375	pπ^+	M7MSGC mode	30	0	MSGC	2.8	600	11k		70
3403278	376	pπ^+	M7MSGC mode	30	0	MSGC	2.8	650	2732		70
7327234	377	pπ^+	30Grad	0	0	high	410	485	200	89	70
18281482	378	pπ^+	30Grad	0	0	high	410	485	5067	89	70
21958952	379	pπ^+	30Grad	0	0	high	410	485	6005	89	70

Tabelle D.2: *Liste sämtlicher aufgezeichneter Runs, zusammengestellt von K. Müller*

Anhang E

Kurzanleitung der Slow Control Programme

Für den "Stand alone" Betrieb, d.h. den Betrieb ohne die DESY Infrastruktur, von Slow Control Programmen, die die SPC Bibliothek benutzen, müssen folgende Progamme gestartet werden:

- Der **Nameserver** für das RPM-Nachrichtenprotokoll mit dem Befehlt: `namserv -root`. Die anderen Knoten der Slow Control erfahren aus der Umgebungsvariable `HBNAMEROOT` den Rechnernamen, auf dem der Nameserver gestartet wurde. Die Umgebungsvariable wird mit `export HBNAMEROOT=myComputer.myHost.de` gesetzt und kann mit `echo $HBNAMEROOT` überprüft werden. In festen Zeitabständen gibt das Programm eine Meldung im Kommandozeilenfenster aus. Auch die An- und Abmeldung von einzelnen Prozessen wird so dokumentiert. Wenn diese Meldungen in der Datei `nameLog` stehen und das Programm im Hintergrund laufen soll, startet man es wie folgt: `nohup namserver -root> nameLog` .
- Der **Datenbankserver** ermöglicht das Lesen und Schreiben von Daten von jedem Rechner und verschiedenen Programmen über einen bestimmten Port. Soll das Programm eine Datenbank mit dem Namen `myDbase.db` anlegen und der Zugriff über den Port `myPort` erfolgen, so startet man den Datenbankserver mit `sdbserver [myPort] [myDbase.db] &` . Das Programm kann im Hintergrund laufen und erzeugt keine Meldungen in der Kommandozeile. Die geschriebenen Datenbankdateien können mit dem Programm `dbedit` leicht eingesehen werden.

Wenn Probleme mit den Slow Control Programmen auftreten, sollten folgende Punkte überprüft werden:

- Ist `HBNAMEROOT` richtig gesetzt und läuft der Nameserver auf dem richtigen Rechner? Auf einen solchen Fehler deuten RPM Fehlermeldungen beim Starten des Slow Control Programs wie (`Err:RPM init return error code 5`) hin.
- Wenn bei dem Programmstart ohne Fehlermeldung das Programm stehen bleibt, ist möglicherweise der Datenbankserver nicht gestartet oder das jeweilige Programm, das auf den Datenbankserver zugreifen will, nicht richtig konfiguriert.
- Wird das Programm gestartet und ist ein Zugriff mit dem User Interface möglich, werden jedoch keine Kanäle bearbeitet, stimmen evt. die Kanalnamen in der Datei für die Konfigurationsparameter mit denen in der Datei für die Monitorparameter nicht überein.
- Wenn die Strommeßbox/GEM-Verteilerbox über RS232 nicht angesprochen werden kann, ist unter Umständen entweder ein Reset der Box notwendig oder eine Vertauschung der

Receive- und Transmit-Leitung in den Boxen. Dies kann durch Steckbrücken im Inneren der Geräte geschehen, oder aber durch den Einsatz des richtigen Kabels.

Ich möchte allen danken, die zum Gelingen dieser Arbeit beigetragen haben:

- Prof. Eisele für die engagierte und menschliche Betreuung während der Diplomarbeit. Ich habe viel von ihm lernen können. Die Möglichkeit, das Arbeiten in einer Kollaboration kennenzulernen habe ich sehr genossen.
- Prof. Stachel für die Übernahme der Zweitkorrektur.
- Ueli Strauman für zwei PSI-Tests, seine Begeisterung und daß er immer wieder die richtigen Worte fand.
- Wolfgang für die optimale Betreuung, die vielen Erklärungen und den Spaß an der Arbeit. Herzlichen Dank auch für das wiederholte Korrigieren der Arbeit.
- Sebastian für seine motivierende Art und die Einführung in die Geheimnisse der Hardware-Programmierung.
- Katharina für ihre Besonnenheit und die vielen Tips und Ratschläge.
- Ulrich Uwer für die Möglichkeit in der Slow Control Gruppe am DESY mitzuarbeiten.
- Stefan, für seine Gabe auch dem trockensten Meeting einen gewissen Witz zu geben.
- Carsten für das gemeinsame Durchstehen des Studiums.
- Malte, Sonja, Marcus, Carola, Hans-Bernd, Martin F.-R., Arnd, Peter R., Thomi, Urs und Jörg für die gute Zusammenarbeit.
- Den Mitarbeitern der Werkstätten für ihre Geduld und Hilfe, besonders Rudi und Christian.
- Meinen Eltern, die mir das Studium ermöglicht haben und mich während dieser Zeit unterstützten.
- Susi, für das Zuhören und das eröffnen neuer Blickwinkel.
- Allen, mit denen ich gemeinsam arbeiten durfte und die ich hier nicht erwähnt habe. Es war für mich ein abwechslungsreiches und gutes Jahr!

Literaturverzeichnis

[Atk98] Atkins, Marty, Padilla, C., *Library for CAEN SY527 steering*, 1.1997, http://www-hera-b.desy.de/subgroup/slowctrl/docu/hvlib.ps

[Bro89] Brockhaus Lexikon, F.A. Brockhaus, Mannheim 1989

[BLT] http://www.tcltk.com/blt/index.html

[Chr64] Christensen, JH, Cronin W, Fitch VL, Turkey R, *PRL* 13, 183 (1964) und *PR* 140, B74 (1965)

[Eis98] Eisele, Franz, *persönliche Mitteilung*

[Gra97] Gradl, Wolfgang, *Analyse harter γp Ereignisse bei großem x_γ mit dem H1-Detektor*, Diplomarbeit am Physikalischen Institut Universität Heidelberg, 8.1997

[Gra98] Gradl, Wolfgang, *persönliche Mitteilung*

[Hau98] Hausmann, Sebastian, *Das Auslesesystem der Inneren Spurkammern bei HERA-B*, Dissertation Universität Heidelberg, 12.1998

[HER94] *HERA-B Proposal*, DESY-PRC 94/02, ftp://ftp.desy.de/pub/herab/proposal/prop.ps.gz

[HER95] *HERA-B Design Report*, DESY-PRC 95/01, ftp://ftp.desy.de/pub/herab/tdr/tdr.ps.gz

[Hil98] Hildebrandt, Malte, *persönliche Mitteilung*

[Fal98] Fallot-Burghardt, Wolfgang et al. , *Helix128-2.x User Manual V1.9*, Asic Labor Heidelberg, http://wwwasic.ihep.uni-heidelberg.de/~feuersta/projects/Helix/helix.ps.gz

[Kih97a] Kihm, Thomas, *VME*, 1997, http://ftp.desy.de/pub/herab/daq/VMEdriver.ps

[Kih97b] Kihm, Thomas, *Table Based Database*, http://ftp.desy.de/pub/herab/daq/DBdocu.ps

[Kob73] M. Kobayashi, T. Maskawa *Prog. Theor. Phys.* **49**, 652, 1973

[Ous95] Ousterhout, John K., *Tcl und Tk, Addison-Wesley,1995*

[ROOT] Brun, Rene, http://www-root.cern.ch

[Nam] *Hera-B Message System Name Service* , 5.1998 http://ftp.desy.de/pub/herab/daq/nam.ps

[Lab93] *LabVIEW for Macintosh User Manual*, National Instruments, 8.1993, Austin Texas USA

[Lef96] Leffers, Georg *Table Data Base Display and Editor* , ftp://ftp.desy.de/pub/herab/daq/projects/DBDISP.html, Oct 1996

[Lin] http://www.linux.org

[Lyn] Produktinformation LynxOS, *http://www.lynx.com*

[PAW] Brun, Rene *http://wwwinfo.cern.ch/asd/paw/index.html*

[Res97] Ressing, Dominik, Wurth, Rainer, *Implementation of the SLB using SHARCs*, HERA-B internal note, March 17th, 1997

[RRZ93] Regionales Rechenzentrum für Niedersachsen, *C++ für C-Programmierer*, Hannover, Oktober 1993

[Ryb97] Rybnikov, Vladimir,
The State Machine Control Paracke for the HERA-B Run Control System,
http://ftp.desy.de/pub/herab/daq/smc.ps

[Str92] Stroustrup, Bjarne, *Die C++ Programmiersprache, Addison-Wesley 1992*

[Str98] Strauman, Ueli, *Elektronik für Physiker, Skriptum zur Vorlesung WS 98/99,* http://doc.physi.uni-heidelberg.de/~strauman

[Tan96] Tanenbaum, Andrew S. , *Computer Networks*, Prentice Hall International Editions, Third Edition, 1996

[Uwe97a] Uwer, Ulrich, *Slow Control Status Report*, 12.1997,
http://www-hera-b.desy.de/subgroup/slowctrl/docu/talk-dec-97.ps

[Uwe97b] Uwer, Ulrich, *Towards a HERA-B Slow Control System*,
http://www-hera-b.desy.de/subgroup/slowctrl/docu/paractrl.ps

[Vis96] Visbeck, Sven, *Untersuchung an MSGCs für das Innere Spurkammersystem bei HERA-B*, Diplomarbeit am Physikalischen Institut der Universität Heidelberg, 9.1996,
http://www.physi.uni-heidelberg.de/groups/herab/text/DiplArb_SVisbeck.ps.gz

[vWa98] v.Walter, Peter, *persönliche Mitteilung*, Heidelberg 1998

[vWa99a] v.Walter, Peter, *2*8 Kanal HV Current Meter*, Dokumentation der Elektronikwerkstatt des Physikalischen Institus Heidelberg, 1.1999

[vWa99b] v.Walter, Peter, *GEM-Verteiler-Box*, Dokumentation der Elektronikwerkstatt des Physikalischen Institus Heidelberg, Januar 1999

[Wal98] Walter, Thomas, *persönliche Mitteilung*

[Wol83] Wolfenstein, L., Phys. Rev. Lett. **51**, 1945 (1983)

[Wu57] C.S. Wu et al., *Phys. Rev* **105**, 48, 1957

[Zhe98a] Zhelezov A., *SPC System User's Guide*,
http://www-hera-b.desy.de/subgroup/slowctrl/docu/spc_usage.ps

[Zhe98b] Zhelezov A., *SPC system programmers guide*, April 1998, http://www-hera-b.desy.de/subgroup/slowctrl/docu/spc_user.ps

[Zwe98] Zweizig, John, *Proposal for a HeraB Message Handler Interface*, 6.1998, http://ftp.desy.de/pub/herab/daq/rpm_doc.ps

Abbildungsverzeichnis

Erklärung:

Ich versichere, daß ich diese Arbeit selbständig verfaßt und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Heidelberg, den 19. Januar 1999

....

Martin Gerlowski