PETER RÜBHANN

Physik-Institut
der Universität Zürich

Februar 1990

Aufbau und Inbetriebnahme einer Apparatur zur Absoluteichung einer Driftkammer

Experimentalphysikalische Diplomarbeit
vorgelegt von
Thomas Kindler

ausgeführt unter der Leitung von
Prof. Dr. P.Truöl

Meinen Eltern gewidmet

Inhaltsverzeichnis

I. Motivation und Rahmenprojekt der Arbeit

In den folgenden Abschnitten werde ich kurz das internationale Grossprojekt vorstellen, in dessen übergeordneten Rahmen diese Diplomarbeit eingebettet ist.
Im Speziellen soll auf die gemachten Entwicklungen in der Zürcher Gruppe, in der ich das Vergnügen hatte, mitzuarbeiten, eingegangen werden.
Im letzten Teil dieses Kapitels möchte ich versuchen, die eigentliche Motivation für diese Arbeit zu schildern.

I.1 Der HERA-Speicherring

Auf dem Gelände des Deutschen Elektronen Synchrotrons (DESY) in Hamburg wird im Moment ein neuer Beschleuniger mit dem Namen HERA[1] gebaut und soll 1990 in Betrieb genommen werden.
In diesem Beschleuniger-System werden ein normalleitendes Elektron-Synchroton und ein supraleitendes Proton-Synchrotron miteinander kombiniert.
HERA benutzt in seinem Injektionsteil einige bereits vorhandene Komponenten (Abb I.1):

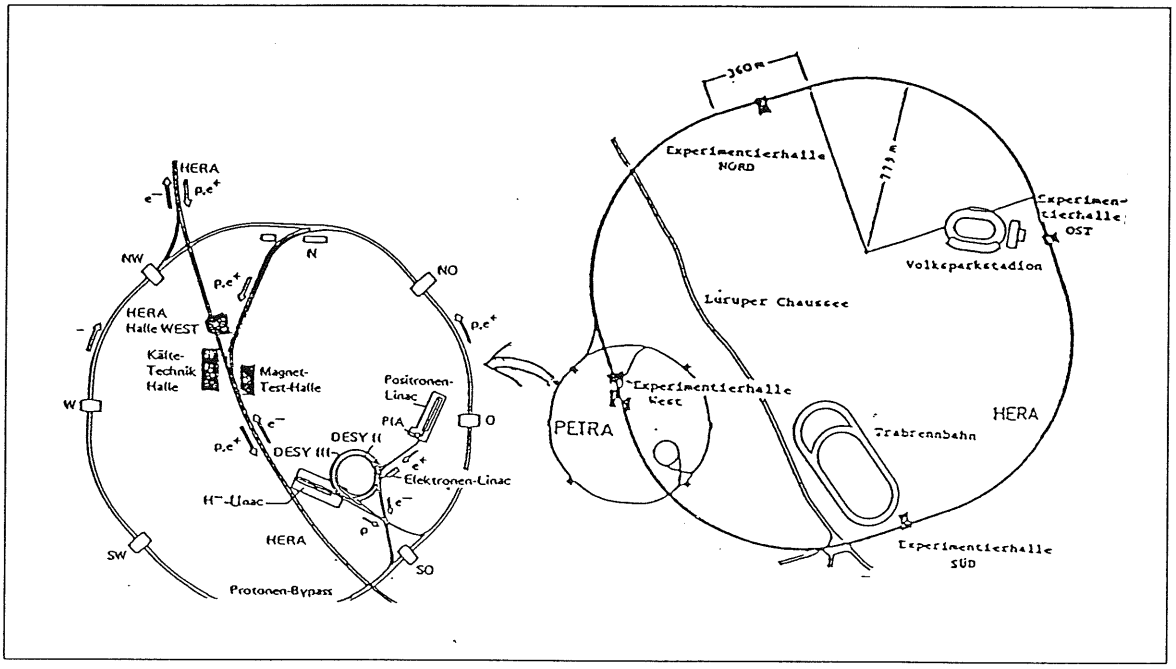


Abbildung I.1 Übersicht über DESY

[1] HERA steht für Hadron Elektron Ring Anlage

H-Ionen werden in einem neu gebauten Linearbeschleuniger auf 50 MEV vorbeschleunigt, bevor sie in einem neuen Protonsynchrotron (DESY III) auf 7,5 GeV und in einem bereits vorhandenen e^+e^- Speicherring (PETRA) auf 40 GeV beschleunigt werden. Mit dieser Energie werden sie in HERA eingeschossen.
Elektronen durchlaufen nach dem Linearbeschleuniger, der ihnen zwischen 50 und 400 MeV mitgibt, das neuerbaute Synchrotron DESY II, bringen 9 GeV beim Eingang zu Petra mit sich und werden mit 14 GeV in HERA eingeschossen. Die Füllzeit für jeden Strahl beträgt zwischen 14 und 20 Minuten.
In HERA stehen für Elektronen und Protonen schliesslich Energien von 30 GeV bzw. 820 GeV zur Verfügung. Das entspricht einer **Schwerpunktsenergie** von **314 GeV.**
In Tabelle I.1 sind die wichtigsten Parameter von vergleichbaren Beschleunigern zusammengefasst.

	SppS (CERN)	TEVATRON (Fermilab)	HERA (DESY)	LEP (CERN)	LHC (CERN)	SSC (USA)
Inbetrieb-nahme	1981	1987	1990?	1989	1995?	1995?
Beteiligte Teilchen	pp	pp	ep	ee	pp	pp
Max. Energie (TeV)	0.026	0.9-1.0	e: 0.03 p: 0.82	0.06	8	20
Länge des Ringes (km)	6.911	6.28	6.336	26.66	26.659	83.831

Tabelle I.1 (aus [25])

Um die an den Wechselwirkungspunkten entstehenden Sekundärteilchen nachweisen zu können, werden an zwei der Kollisionspunkte (insgesamt vier vorhanden) Detektoren mit dem Namen **Zeus** und **H1** entwickelt und aufgebaut.

I.2 Der Detektor H1

Die Abbildung I.2 zeigt einen Querschnitt durch den H1 Detektor.

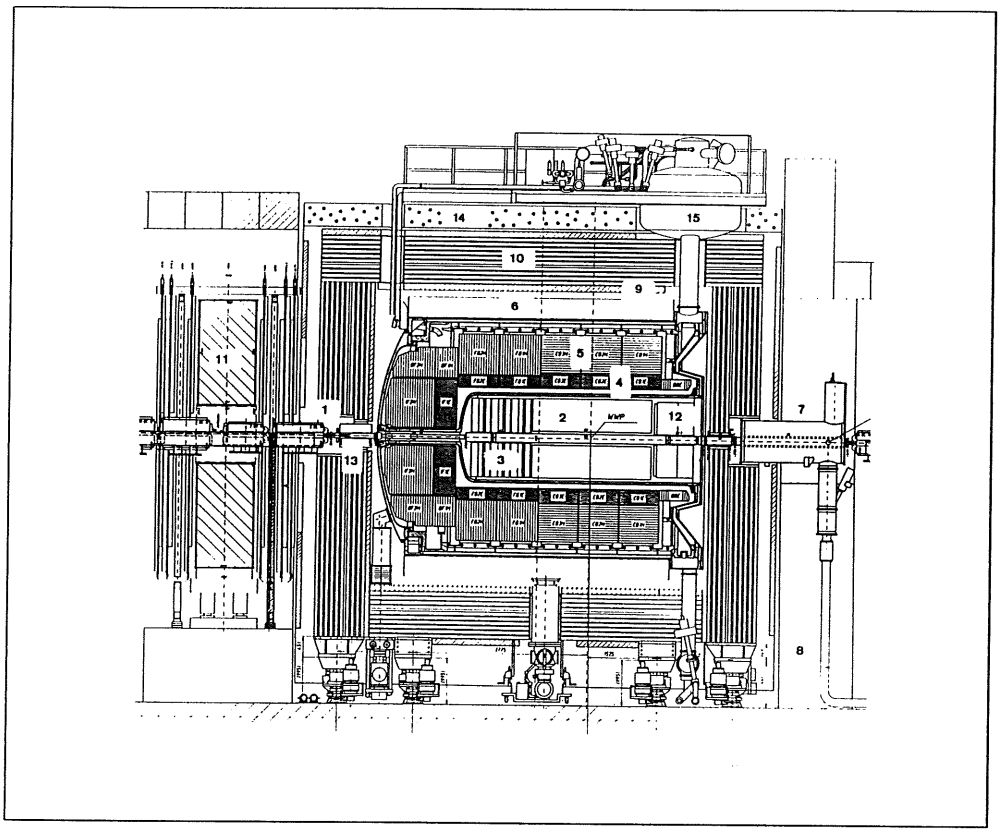


Abbildung I.2 Querschnitt durch H1 (aus [26])

1 Strahlrohr und Strahlmagnete. 2 Zentrale Spurkammern. 3 Vorwärtsspurkammern. 4 Elektromagnetisches Kalorimeter (Blei). 5 Hadronisches Kalorimeter (Edelstahl). 6 Supraleitende Spule (1.2 Tesla). 7 Kompensationsmagnet. 8 Transferleitung zur Flüssig-Helium Versorgung. 9 Myon-Kammern. 10 Instrumentiertes Eisen (Eisenplatten und "Streamer"-Röhren). 11 Myon-Toroid-Magnet. 12 Warmes, elektromagnetisches Kalorimeter. 13 Vorwärts-Kalorimeter. 14 Betonabschirmung. 15 Flüssig-Argon Kryostat.

Weil die Energie der Protonen wesentlich über der der Elektronen liegt, wird ein Grossteil der Reaktionsprodukte in Richtung der Protonen (Vorwärtsrichtung) emittiert. Dadurch ist die auffallend asymetrische Form des Detektors erklärt.
Ausgehend vom Wechselwirkungspunkt erfolgt der Nachweis von geladenen Teilchen durch den zentralen Spurdetektor **CTD**[2] und durch den Vorwärtsspurdetektor **FTD**[3].

[2] Central Tracking Detector
[3] Forward Tracking Detector

Der CTD besteht aus zwei konzentrisch angeordneten Jetkammern (CJC1[4] und CJC2), die durch je zwei **z-Kammern** und **MWPC's**[5] voneinander getrennt sind (Abbildung I.3).

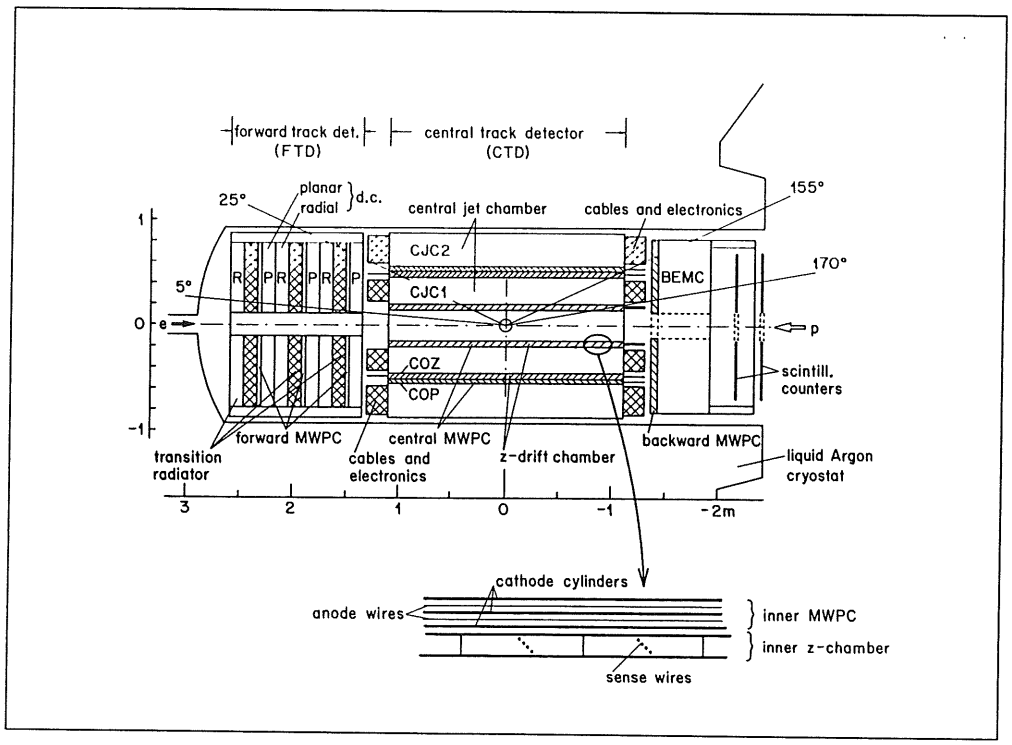


Abbildung I.3 Querschnitt durch den Central Track Detector (aus [26])

Die nächste Schicht wird durch verschiedene Kalorimeter gebildet. In Vorwärts-und Rückwärtsrichtung liegen die elektromagnetischen Kalorimeter für den Nachweis von Photonen und Elektronen; radial dazu befindet sich das hadronische Kalorimeter.
Die äussersten Detektorelemente werden durch das Vorwärtskalorimeter und die Myonkammern gebildet.

[4] Central Jet Chamber
[5] Multi Wire Proportional Chamber

I.3 Physik bei HERA

HERA ist bis anhin weltweit der einzige Elektron-Proton-Speicherring. Nach der heute gültigen Vorstellung ist das Proton aufgebaut aus drei Quarks (uud), die durch die starke Wechselwirkung untereinander gebunden sind. Ihre Bindungsenergie ist jedoch im Verhältnis zu den kinetischen Energien der Elektronen und Quarks im Schwerpunktsystem vernachlässigbar klein, so dass man es mit quasifreien Elektron-Quark-Stössen zu tun hat.
Mit 30 GeV-Elektronen und 820 GeV-Protonen sind Impulsüberträge bis zu $Q^2=s=(314\ \mathrm{GeV})^2= 10^5\ \mathrm{GeV}^2 = 4E_eE_p$ bei der Elektron-Proton-Streuung möglich. Bei diesen hohen Energien ist HERA der Welt mächtigster Erzeuger von Elektron-Quark-Stössen.
Die Kräfte zwischen Quarks und Leptonen können nach dem Standard-Modell auf den Austausch von Feldquanten der elektroschwachen Wechselwirkung (γ,Z^0,W^+,W^-) zurückgeführt werden. Grafisch kann man diesen Vorgang wie in Abbildung I.4 darstellen. Das einfallende Elektron emittiert ein Lepton l und tauscht einen Strom j mit einem der Quarks des einfallenden Protons aus. Dies führt zur Emission eines Quarks q', des sogenannten Stromquarks; die beiden übrigen werden Zuschauerquarks genannt.

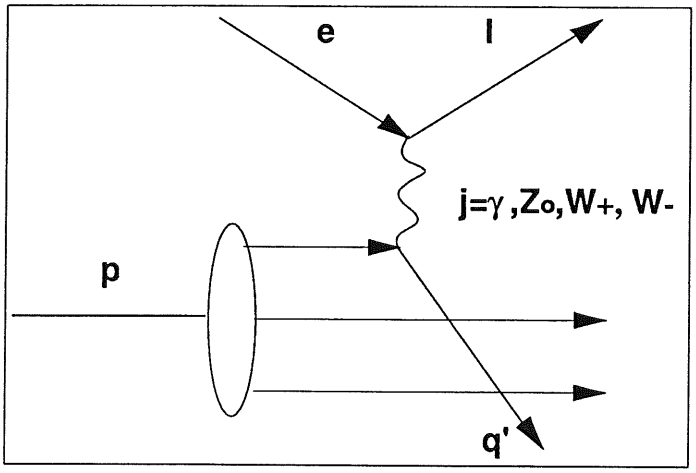


Abbildung I.4 Standardmodell für Kräfteaustausch

Je nachdem, ob der Strom neutral (j= γ, Z^0) oder geladen (j=W^+,W^-) ist, wird das Lepton im Endzustand zu einem Elektron oder einem Neutrino (Abbildung I.5). Für die Erzeugung von Teilchen ausserhalb des Standard-Modells können andersartige Ströme neue Quarks und Leptonen mit Massen bis zur kinematischen Limite von 314 GeV beitragen .

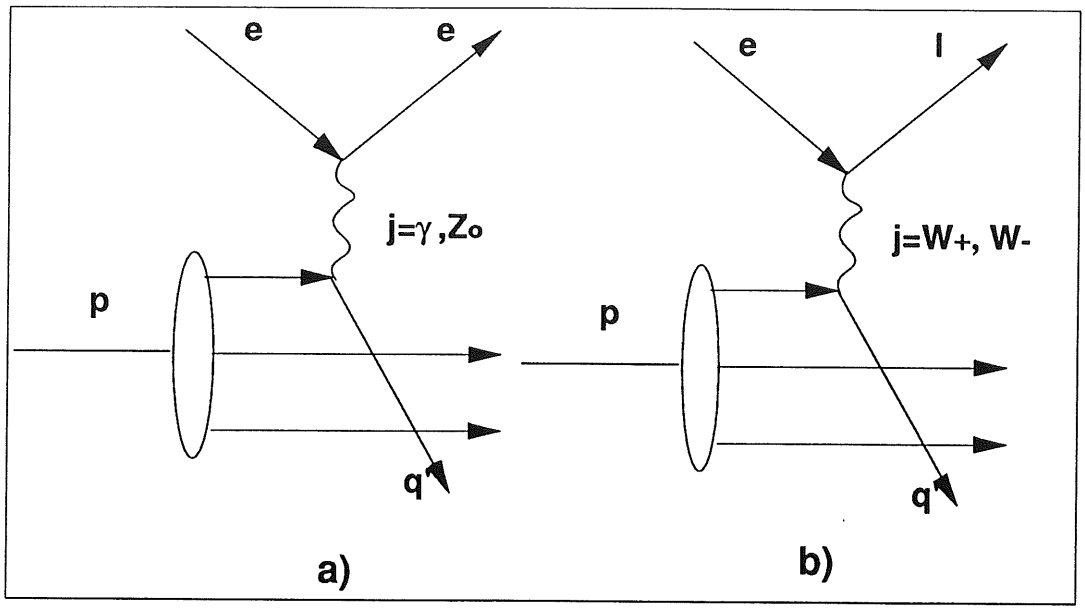


Abbildung I.5 Modell für geladenen und neutralen Strom

Neben der Elektron-Quark-Streuung spielt auch die Fusion des Stromes mit den die Quarks umgebenden Quanten des Farbfeldes, den Gluonen, eine grosse Rolle. Die Photon-Gluon-Fusion ergibt den Hauptbeitrag zur Erzeugung schwerer Quark-Antiquark-Paare. ($Q\bar{Q}$=$c\bar{c}$, $b\bar{b}$, $t\bar{t}$) (Abbildung I.6)

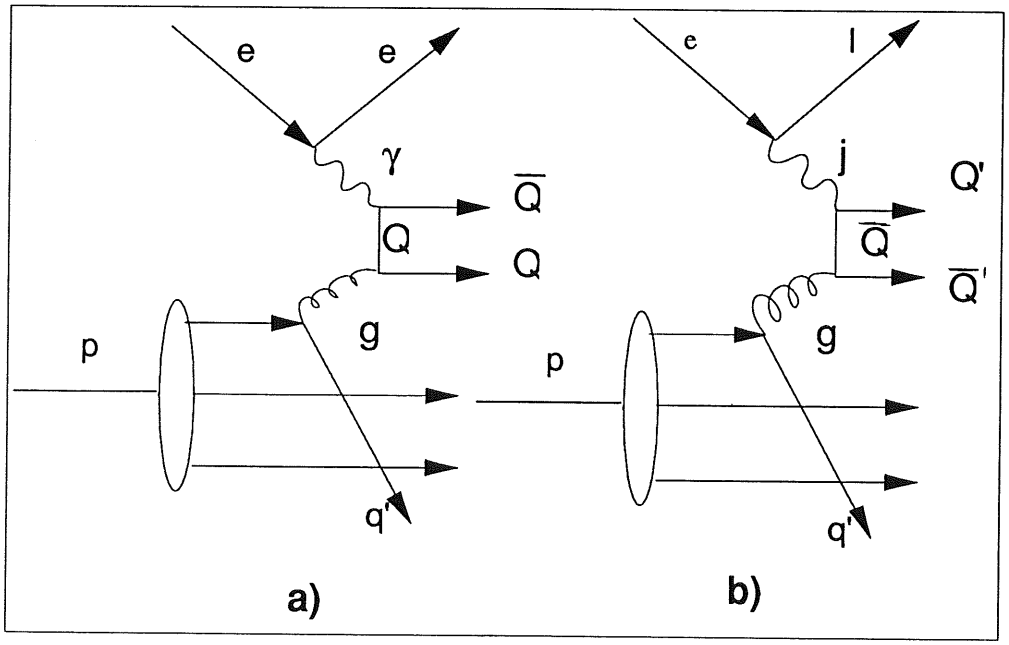


Abbildung I.6 Fusion des Stromes mit Gluonen

I.4 Der Zürcher Beitrag

Der Zürcher Gruppe obliegt es, neben anderen Aufgaben, die innere MWPC und die innere z-Kammer des CTD zu planen, zu bauen und zu betreuen
Ein Prototyp der z-Kammer ist gebaut und erprobt[6]. Die endgültige Kammer sollte bis Ende 1989 ebenfalls fertiggestellt sein.
Die Kammer soll eine möglichst präzise Messung der z-Koordinate der Teilchen- und Jetspuren liefern, die in einem Winkelbereich zwischen 20 und 170 Grad gestreut werden.

I.4.i Der Prototyp der z-Kammer

Der Prototyp besteht aus zwei Ringen mit je 16 Driftzellen. Der Innen- und Aussendurchmesser entspricht den in Abbildung I.7 angegebenen Grössen.

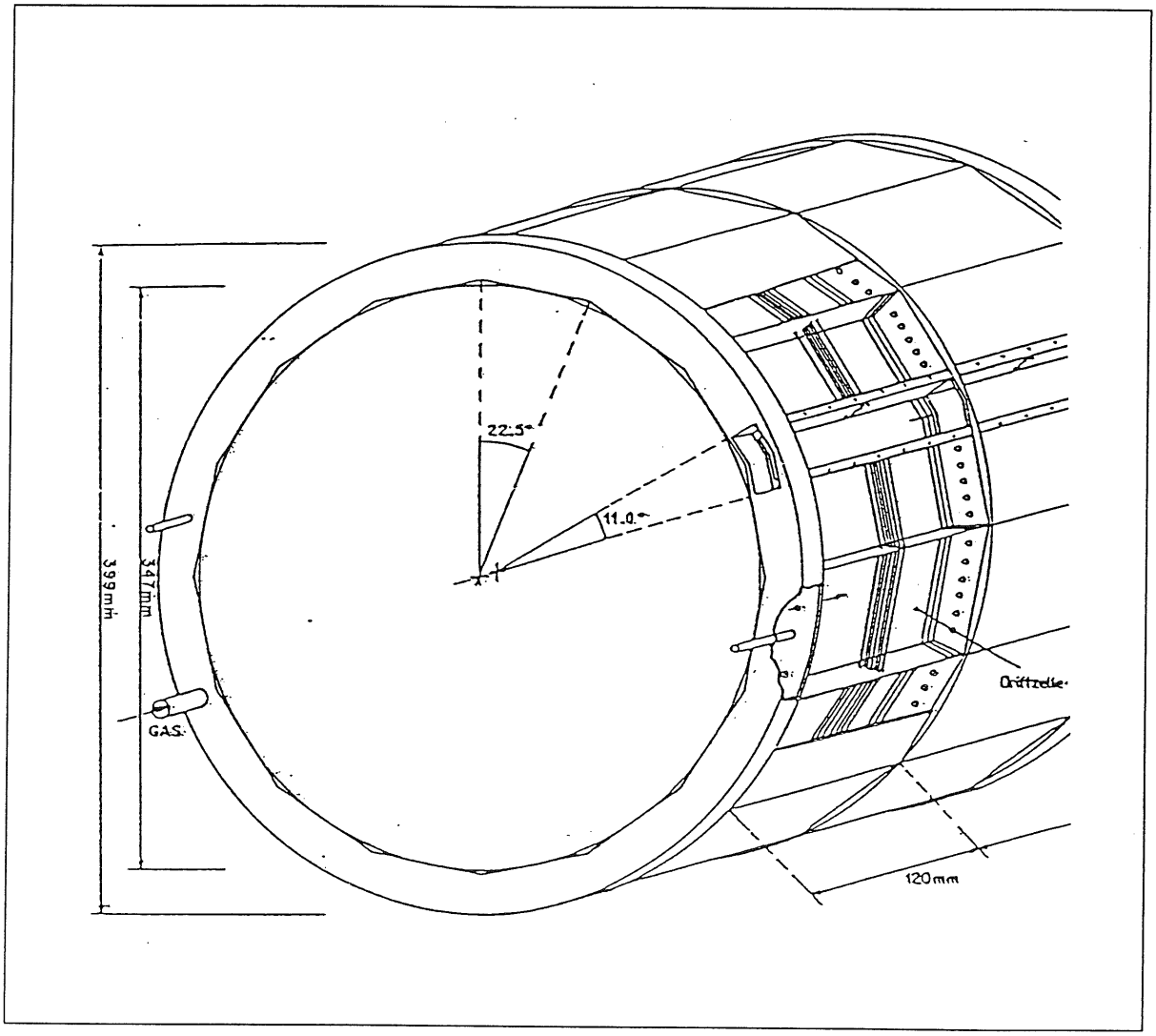


Abbildung I.7 Prototyp der z-Driftkammer (aus [6])

[6] Siehe Ref. [6]

Die gewählte Driftstrecke von 6 cm bestimmt die Ringbreite (12 cm). Um die Strecken der Elektronen bis zum Signaldraht einigermassen konstant zu halten, wurden die Signaldrähte nach intensiven Tests[7] in einer relativ zur Kammerachse geneigten Position angebracht. Das Kammergas wird in Achsenrichtung durch die Zellen geleitet. Die elektrischen Verbindungskabel werden im Kanal zwischen den Endstegen verlegt und mit der Kammer verbunden.

I.5 Motivation der Arbeit

In dieser Arbeit wird eine neue Methode der Absoluteichung bei Ortsmessungen mit Driftkammern erprobt.
Für eine solche Eichung muss durch eine äussere Anordnung die genaue Flugbahn der durch die Kammer fliegenden Teilchen vorgegeben sein. Bei unserer Methode messen wir diese Bahn mit zwei **Proportionalkammern** mit **1mm** Drahtabständen.
Die vorgegebenen Koordinaten (z) werden dann mit denjenigen verglichen, die aus der Zeitinformation der Driftkammerpulse stammen. Diese Information kann aus der mit schnellen Analog-zu-Digital-Wandlern (FADC) gespeicherten Pulsform von jedem Signaldraht der Driftkammern ermittelt werden. Hierzu verwenden wir ein von C.Meyer geschriebenes Programm[8].
In erster Linie ging es bei dieser Diplomarbeit dann darum, verschiedene bereits vorhandene Apparaturen und Geräte zusammen in Betrieb zu nehmen. Etwas salopp ausgedrückt bedeutet das, **MacII - MacVEE - MICRON - VME - FADCF1001 - z-Kammerprototyp - Analyseprogramm** zusammen zum Laufen und Funktionieren zu bringen. Hiermit ist gemeint, dass die FADC-Karte in einer VME-Umgebung residiert, die mit einem MacVEE-MICRON-"Interface" ausgelesen werden kann, so dass die Pulsform dann auf dem MacII-Personal-Computer vorverarbeitet und einem Datenspeicher zugeführt werden kann.
Der grösste Teil dieser Arbeit machte dabei die Entwicklung der Software für den Betrieb und die Auslese der neu entwickelten FADC-Karte aus.
Nach Bereitstellung der Werkzeuge, sollte dann die Messung überprüfen, ob das von uns verwendete Programm zur z-Bestimmung der Driftkammer tatsächlich eine lineare Beziehung zu den z-Werten der Proportional-Kammer ergibt.
Diese Beziehung liefert mit der Steigung der Geraden auch eine Messung der Driftgeschwindigkeit, die für eine genaue Ortsbestimmung der Teilchenspuren durch die Driftkammer notwendig ist.
Alle die in dieser Diplomarbeit ausgeführten Abläufe werden schliesslich für Test und Eichung der endgültigen, für H1 bestimmten Driftkammer verwendet werden.

---

[7] Siehe [7]
[8] Siehe [13]

II. Funktionsweise von Driftkammern

In diesem Kapitel möchte ich einen kurzen Überblick über die Funktionsweise von Driftkammern geben. Diese Art von Detektoren wird heute in nahezu allen Experimenten der Teilchenphysik verwendet und ist weit verbreitet. Sie dienen vor allem der Spurrekonstruktion und der Teilchenidentifikation.
Eine detaillierte Abhandlung der Funktionsweise von Driftkammern würde den Umfang dieser Arbeit sprengen; genauere Angaben können bei F.Sauli[9] und K.Kleinknecht[10] nachgelesen werden.

II.1 Allgemeiner Aufbau

Eine Driftkammer besteht im wesentlichen aus einem Gasvolumen und einer speziellen Anordnung von Signaldrähten (Anoden) und Potentialdrähten (Kathoden). Das Gasvolumen setzt sich zusammen aus Driftbereich und Gasverstärkungsbereich(Abbildung II.1).

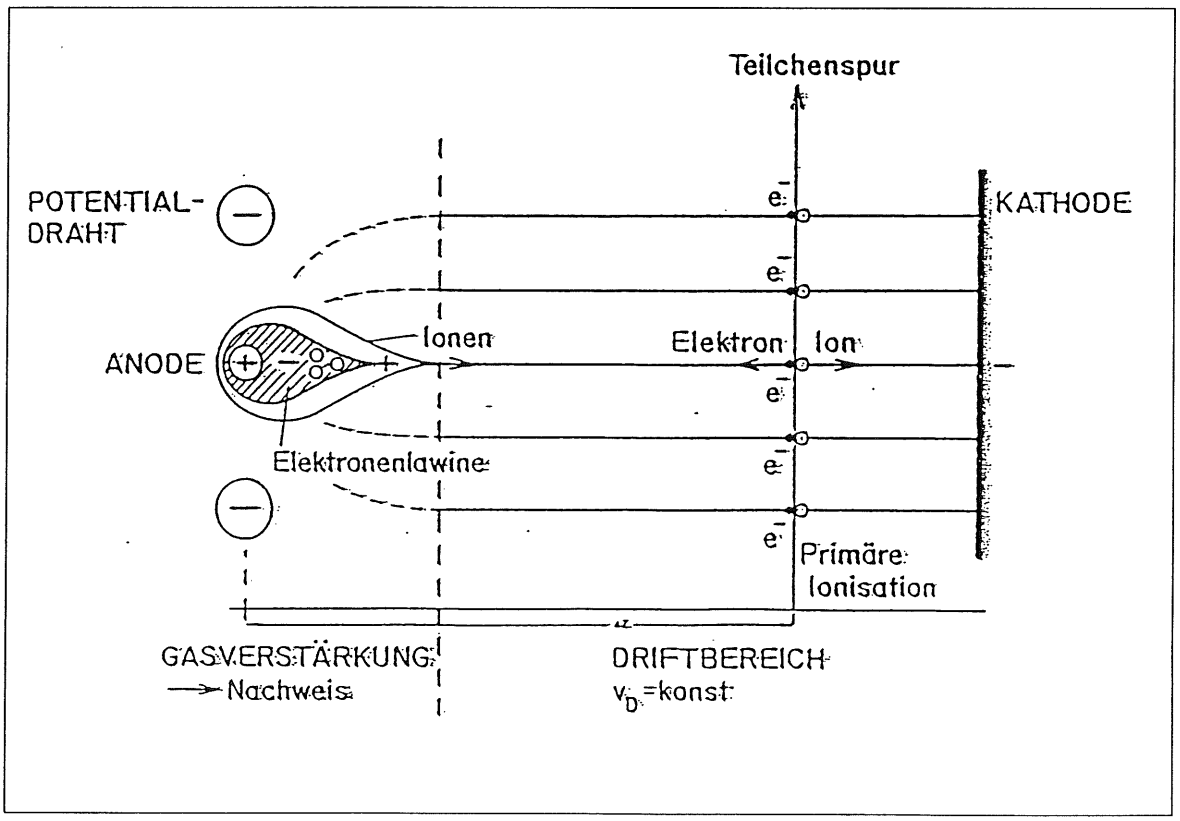


Abbildung II.1 Gasvolumen einer Driftkammer (aus [6])

[9] siehe [8]
[10] siehe [9]

Im Driftbereich versucht man, mit Hilfe einer bestimmten Anordnung von Kathodendrähten ein konstantes, homogenes Feld zu erzeugen (die Homogenität des Feldes wird im allgemeinen nicht allein durch die Kathodendrähte gewährleistet, so dass meistens noch sogenannte Feldformungsdrähte eingebaut werden müssen, deren Spannung so gewählt wird, dass das Feld bis auf wenige Prozent genau homogen wird).
Im Gasverstärkungsbereich, der ein sehr viel kleineres Gebiet um einen Anodendraht herum umfasst, steigt das Feld stark an ($E \sim \frac{1}{r}$).
Das ganze Kammervolumen wird nun mit einer geeigneten Gasmischung gefüllt, die meistens aus einem Edelgas und einer organischen Komponente besteht.
In dem von uns gebrauchten Prototypen sind pro Zelle vier Signaldrähte und drei Feldformungsdrähte eingebaut. Das Gasgemisch besteht im Moment zu 98% aus Argon und zu 2% aus Kohlendioxyd.
Durchquert nun ein geladenes Teilchen die Driftkammer, werden entlang der Flugbahn Gasatome ionisiert (inelastische Stösse). Unter dem Einfluss des konstanten Feldes im Driftbereich bewegen sich die Elektronen Richtung Anode und die positiven Ionen zur Kathode. Durch Stösse mit den Gas-Molekühlen stellt sich eine gleichförmige Driftgeschwindigkeit v_d ein, die von der Zusammensetzung des Gases, dem Druck und der Feldstärke abhängt.
Kommen die Elektronen in den Gasverstärkungsbereich der Anode, so nehmen sie in dem starken, elektrischen Feld mehr Energie auf, als sie in thermischen Stossprozessen abgeben können. Ihre Energie reicht zur Ionisation weiterer Atome des Gases. In Drahtnähe steigt dadurch die Zahl der Elektronen-Ionen-Paare in kurzer Zeit (einige nsec.) stark an, es bilden sich Elektronen-Lawinen. Die primäre Ladung wird dadurch um das 100-fache vergrössert. Der über die Anode abfliessende Elektronenstrom kann mit empfindlichen Verstärkern nachgewiesen werden.

II.2 Ionisation

Durchquert ein geladenes Teilchen ein Gas, so werden bei dieser elektromagnetischen Wechselwirkung[11] vier Prozesse auftreten: die Atome können ionisiert werden, das Teilchen kann Cherenkov-Licht emittieren, es kann durch Bremsstrahlung Energie verlieren oder es kann in inhomogenen Materialien Übergangsstrahlung verursachen.

[11] Andere Wechselwirkungen fallen bei Reaktionen mit Gas nicht ins Gewicht

Vernachlässigt man Cherenkov- und Übergangsstrahlung sowie die für schwere Teilchen ($m>m_e$) unwichtige Bremsstrahlung, so gilt für den Energieverlust pro Längeneinheit die sogenannte Bethe-Bloch-Gleichung (aus [8]):

$$\frac{dE}{dx} = K\frac{Z\rho}{A\beta^2}\left(\ln\frac{2mc^2\beta^2E_M}{I_o^2(1-\beta^2)} - 2\beta^2-\delta\right) \qquad K=\frac{2\pi Nz^2e^4}{mc^2}$$

Gas:
N=Avogadro Zahl m=Elektronenmasse
e=Elektronenladung
Z,A,ρ=Atom Zahl,Masse und Dichte
I_O=mittleres Ionisationspotential
δ=Dichte-Effekt-Parameter

Teilchen:
z,β=Ladung und Geschwindigkeit
$E_M \cong 2m_e\beta^2\gamma^2$=maximaler Energieübertrag beim Einzelstoss

Diese Näherung beschreibt den tatsächlichen Energieverlust recht genau (Abbildung II.2):

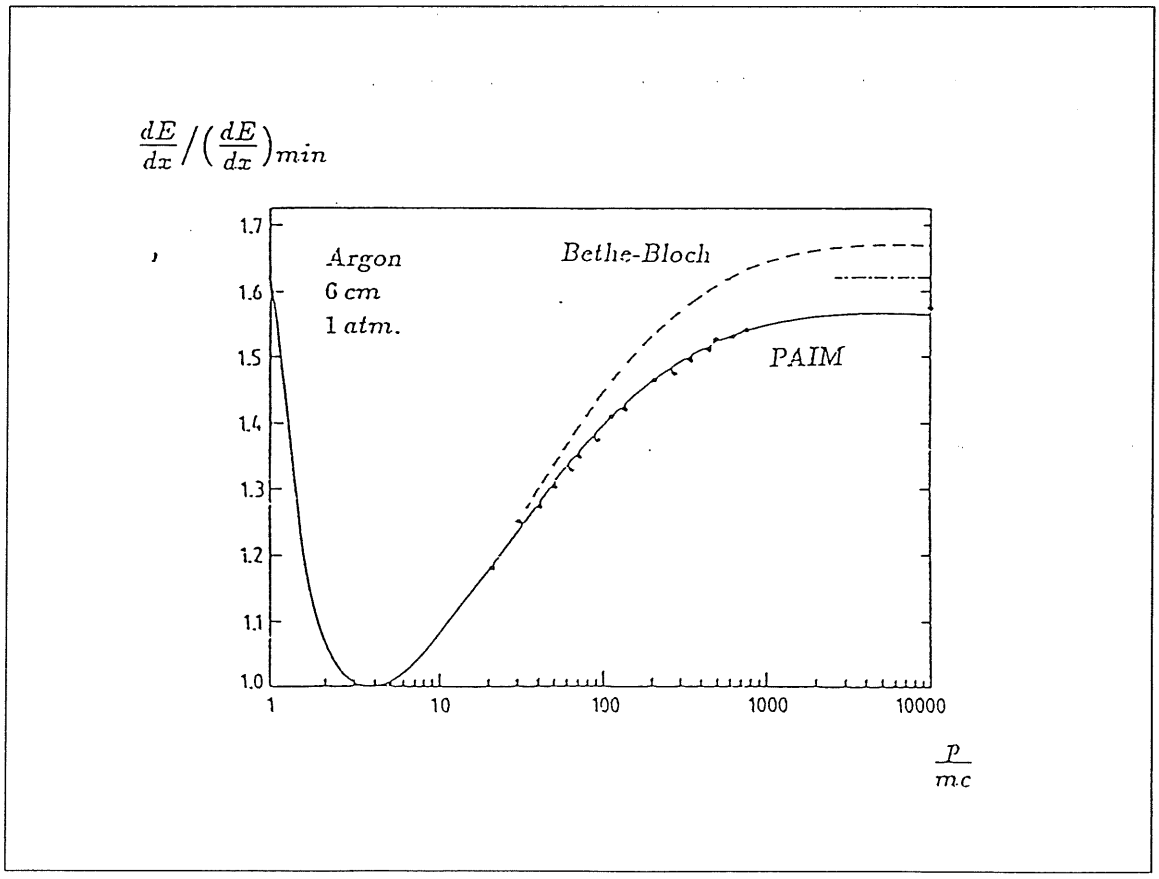


Abbildung II.2 Energieverlust durch Ionisation (aus [9])

Der Energieverlust ist unabhängig von der Masse des ionisierenden Teilchens.Er fällt im Bereich nicht-relativistischer Energien ungefähr mit $1/\beta^2$ ab, bevor er bei etwa $\beta=0.96$ ein Minimum erreicht. Zu sehr hohen Energien steigt er dann langsam wieder an, bevor er ein Plateau erreicht.
Mit dieser an das Gas abgegebenen Energie werden nun Elektron-Ionen-Paare von ihren Atomen gelöst (Primärionisation). Einzelne dieser Elektronen haben nun immer noch genügend grosse Energie (grösser als Ionisationsenergie des Gases), um weitere Elektronen-Ionen-Paare loszulösen (Sekundärionisation). Die totale Anzahl von Elektronen-Ionen-Paare kann man ausdrücken mit:

$$n_T = \frac{\Delta E}{W_i}$$

ΔE=verlorene Energie im Gas

W_i=Energie, um Elektron-Ionen-Paar zu erzeugen

Experimentelle Daten für obige Formel können in Tabelle II.1 nachgeschaut werden:

Gas	Z	A	δ (g/cm³)	E_{ex}	E_i (eV)	I_o (eV)	W_i	dE/dx (MeV/g cm⁻²)	dE/dx (keV/cm)	n_p (i.p./cm) a)	n_T (i.p./cm) a)
H_2	2	2	8.38×10^{-5}	10.8	15.9	15.4	37	4.03	0.34	5.2	9.2
He	2	4	1.66×10^{-4}	19.8	24.5	24.6	41	1.94	0.32	5.9	7.8
N_2	14	28	1.17×10^{-3}	8.1	16.7	15.5	35	1.68	1.96	(10)	56
O_2	16	32	1.33×10^{-3}	7.9	12.8	12.2	31	1.69	2.26	22	73
Ne	10	20.2	8.39×10^{-4}	16.6	21.5	21.6	36	1.68	1.41	12	39
Ar	18	39.9	1.66×10^{-3}	11.6	15.7	15.8	26	1.47	2.44	29.4	94
Kr	36	83.8	3.49×10^{-3}	10.0	13.9	14.0	24	1.32	4.60	(22)	192
Xe	54	131.3	5.49×10^{-3}	8.4	12.1	12.1	22	1.23	6.76	44	307
CO_2	22	44	1.86×10^{-3}	5.2	13.7	13.7	33	1.62	3.01	(34)	91
CH_4	10	16	6.70×10^{-4}		15.2	13.1	28	2.21	1.48	16	53
C_4H_{10}	34	58	2.42×10^{-3}		10.6	10.8	23	1.86	4.50	(46)	195

Tabelle II.1 (aus [9])

ρ=Dichte E_{ex}=minimale Energie für Anregung E_i=minimale Energie für Ionisation I_o=mittleres effektives Ionisationspotential pro Hüllenelektron n_T=Gesamtzahl von Ionenpaaren pro cm Wegstrecke n_p Zahl der primären Ionen pro cm Wegstrecke $\left(\frac{dE}{dx}\right)_o$ = minimaler Energieverlust

Betrachten wir zum Beispiel unser Ar - CO_2 Gemisch (98% - 2%), so erhalten wir folgende Werte:

$$\text{Totale Anzahl der Paare} = n_T = \frac{2440}{26} * 0.98 + \frac{3010}{33} * 0.02$$
$$= \mathbf{93.79\ Paare\ /\ cm}$$
$$\text{Anzahl der primären Paare} = n_p = 29.4 * 0.98 + 34 * 0.02$$
$$= \mathbf{29.49\ Paare\ /\ cm}$$

Man stellt also fest, dass die durchschnittliche Strecke zwischen zwei Primärionisationen 340μm lang ist und jedes primär ionisierte Elektron im Durchschnitt 3.1 Sekundär-Paare produziert.

II.3 Drift und Diffusion im elektrischen Feld

Die durch die Ionisation freiwerdenden Elektronen-Ionen-Paare befinden sich nach kurzer Zeit mit dem Gas und dem elektrischen Feld im Gleichgewicht. Sie werden wohl durch das Feld beschleunigt, verlieren dabei aber die gewonnene Energie ständig durch Stösse mit dem Gas. Es stellt sich darum eine konstante Driftgeschwindigkeit v_d mit Richtung parallel zu den Feldlinien ein (bei konstantem Feld). Kennt man diese Driftgeschwindigkeit und auch die Zeit zwischen der Ionisation und dem Eintreffen der Elektronen am Signaldraht, so kann man den Abstand zur Durchgangsspur des Teilchens berechnen:

$$z = \int_{t_0}^{t_0+t_d} v_d dt$$

II.3.i Driftgeschwindigkeit der Ionen

Die Driftgeschwindigkeit v^+ der Ionen ist bis zu hohen Feldern proportional zum reduzierten Feld E/p. Man führt darum die Beweglichkeit $\mu^+ = \frac{v^+}{E/p}$ ein. Werte für die Beweglichkeit sind in Tabelle II.2 angegeben:

Gas	Ionen	Beweglichkeit ($cm^2V^{-1}sec^{-1}$)
Ar	$(OCH_3)_2CH_2^+$	1.51
$IsoC_4H_{10}$	$(OCH_3)_2CH_2^+$	0.55
$(OCH_3)_2CH_2$		0.26
Ar	$IsoC_4H_{10}^+$	1.56
$IsoC_4H_{10}$	$IsoC_4H_{10}^+$	0.61
Ar	CH_4^+	1.87
CH_4	CH_4^+	2.26
Ar	CO_2^+	1.72
CO_2	CO_2^+	1.09

Tabelle II.2 aus [8]

Bei dem von uns verwendeten Argon-Gemisch erhalten wir also eine Driftgeschwindigkeit von ca. 0.017 mm/µs.

II.3.ii Driftgeschwindigkeit der Elektronen

Für die Driftgeschwindigkeit der Elektronen kann man als grobe Näherung untenstehende Formel angeben:

$$v^- \cong \frac{e}{2m_e} E \tau_e \qquad \tau_e \cong 1/p$$

τ_e bezeichnet die mittlere Zeit zwischen zwei Stössen. Aus Abbildung II.3 kann man den Verlauf von v^- in Abhängigkeit von E/p für unsere Gasmischungen herauslesen.

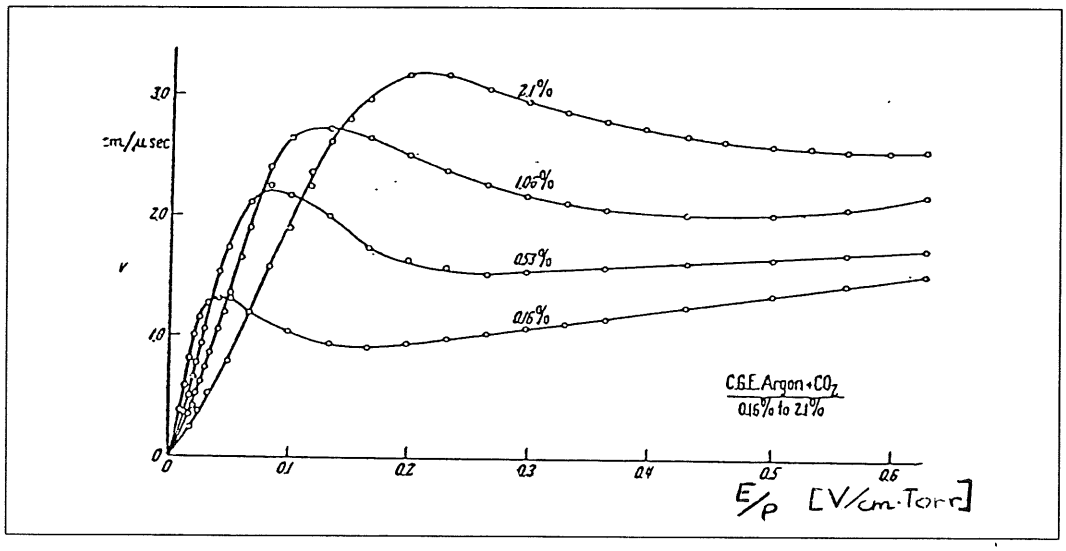


Abbildung II.3 Driftgeschwindigkeit als Funktion der Spannung (aus [8])

Da man die Kammer bei konstanter Driftgeschwindigkeit betreiben will, wählt man E so, dass v^- über einen möglichst breiten Bereich von E konstant bleibt.
In unserer Kammer beträgt die Driftgeschwindigkeit der Elektronen etwa 3cm/µsec.

II.3.iii Diffusion

Der Drift im elektrischen Feld ist die Diffusion durch thermische Bewegung überlagert. Bei Abwesenheit eines Feldes verändert sich eine lokalisierte Ladungsverteilung gemäss dem Gausschen Gesetzt wie folgt:

$$\frac{dN}{N} = \frac{1}{\sqrt{4\pi Dt}} e^{-\frac{x^2}{4Dt}} dx$$

dN/N=Anteil der Ladungen im Intervall [x,x+dx]
D = Diffusionskoeffizient

Das ergibt eine Gaussverteilung. Für die Standardabweichung der Verteilung nach einer Strecke $z = tv_d$ gilt:

$$\sigma_z = \sqrt{\frac{2Dz}{v_d}}$$

Dieses Auseinanderdiffundieren der Ladungsverteilung setzt dem Auflösungsvermögen einer Driftkammer eine natürliche Grenze (Abbildung II.4).

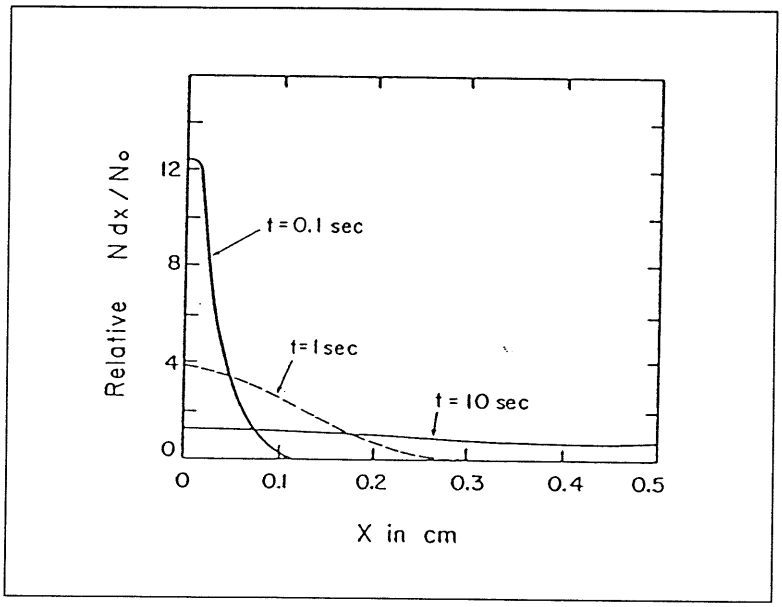


Abbildung II.4 Diffusion des Ladungs-Stromes (aus [8])

II.4 Gasverstärkung

In II.2 haben wir gesehen, dass ein durch den Detektor laufendes Teilchen in etwa 100 Elektronen-Ionen-Paare produziert. Sammelt man diese Elektronen an einer Anode; so misst man :

$$V = \frac{ne}{C} \cong 2\,\mu V \quad \text{falls } C = 10\ pF \text{ (typische Systemkapazität)}$$

Das ist natürlich viel zu klein, um gemessen zu werden. Man braucht also eine massive Vergrösserung der Ladung. Dafür sorgt die Feldstärke unmittelbar um die Anoden herum. E nimmt in dieser Region ungefähr mit 1/r zu. Die Elektronen nehmen zwischen zwei Stössen soviel Energie auf, dass sie immer mehr Elektronen-Ionen-Paare erzeugen können, es kommt so zu einer lawinenartigen Zunahme von Elektronen. Erst diese Ladungsvervielfachung führt zu einem an der Anode messbaren Signal.
Diese Ladungsvervielfachung ist abhängig von der gewählten Spannung. Man unterscheidet darum verschiedene Arbeitsbereiche (Abbildung II.5):

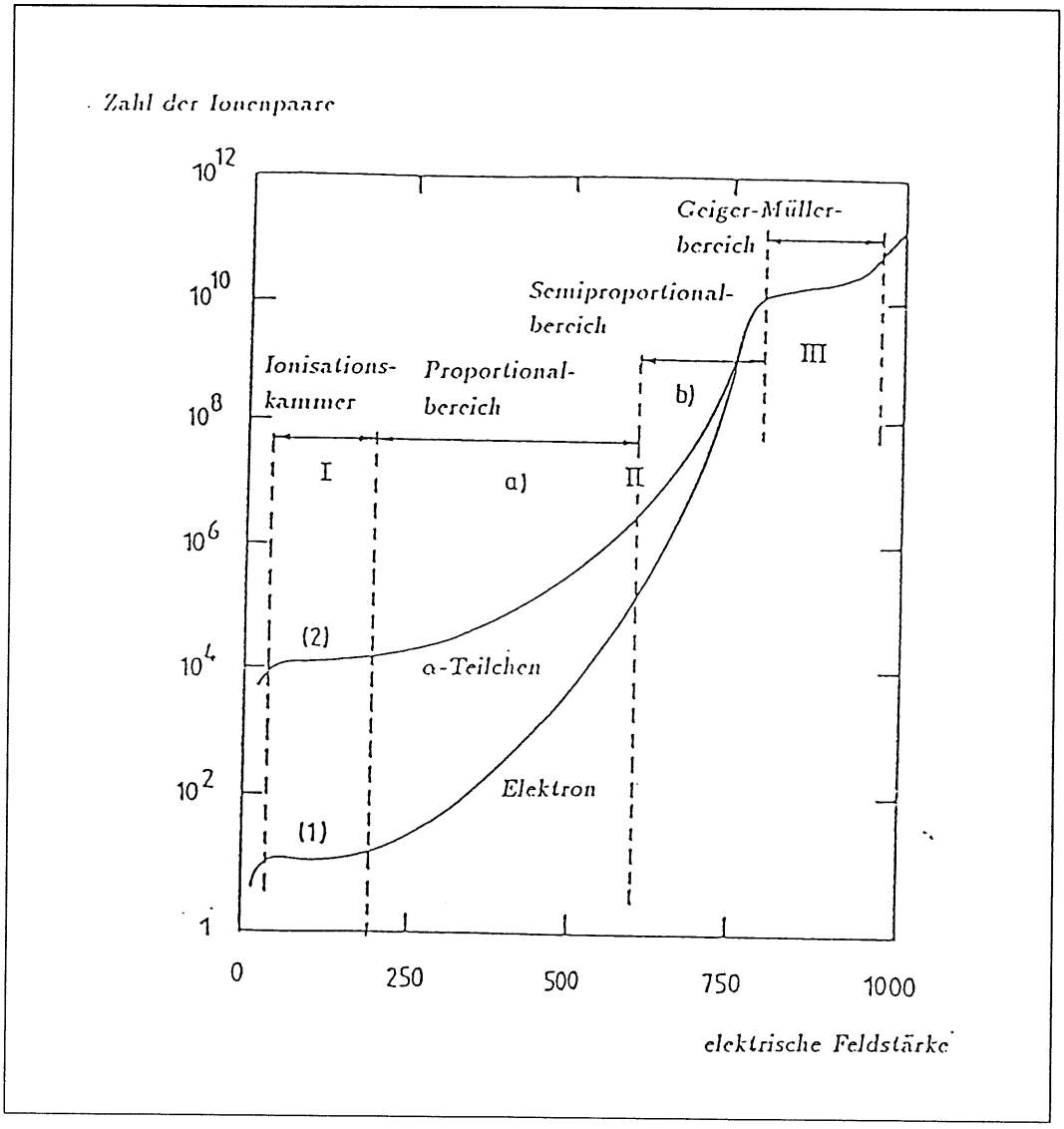


Abbildung II.5 Arbeitsbereiche einer Driftkammer (aus [8])

In der **Ionisationskammer** findet keine Gasverstärkung statt.
Im **Proportionalbereich** sollte eine Driftkammer betrieben werden. Hier steigt der Verstärkungsfaktor mit wachsender Spannung bis zu Werten von 10^6 und ist annähernd proportional zu der Zahl der primär erzeugten Elektronen. Man geht davon aus, dass jedes Elektron seine eigene Lawine erzeugt.
Wird die Spannung weiter erhöht, so tritt ein Sättigungseffekt ein; man gelangt in den **Semiproportionalbereich**. Die in einer Lawine entstehenden Ionen beginnen eine nachfolgende Lawine zu stören; der Verstärkungsfaktor ist nicht mehr proportional zu den primär erzeugten Elektronen.
Erhöht man die Spannung noch weiter, so wird die Gasverstärkung unabhängig von der Anzahl der Primärelektronen, man ist nun im **Geiger-Müller-Bereich**, der völlig unbrauchbar ist.

III. Versuchsanordnung

In diesem Kapitel werde ich die Versuchsanordnung sowohl für den Normalbetrieb der Driftkammer wie auch den Betrieb für die Absoluteichung beschreiben. In den ersten Abschnitten möchte ich über die wichtigsten Apparaturen einige Worte verlieren, bevor ich im letzten Abschnitt die elektronische Schaltung darstelle.

III.1 Übersicht

In Abbildung III.1 ist eine Übersicht der experimentellen Anordnung im Normal-Betrieb der Driftkammer dargestellt:

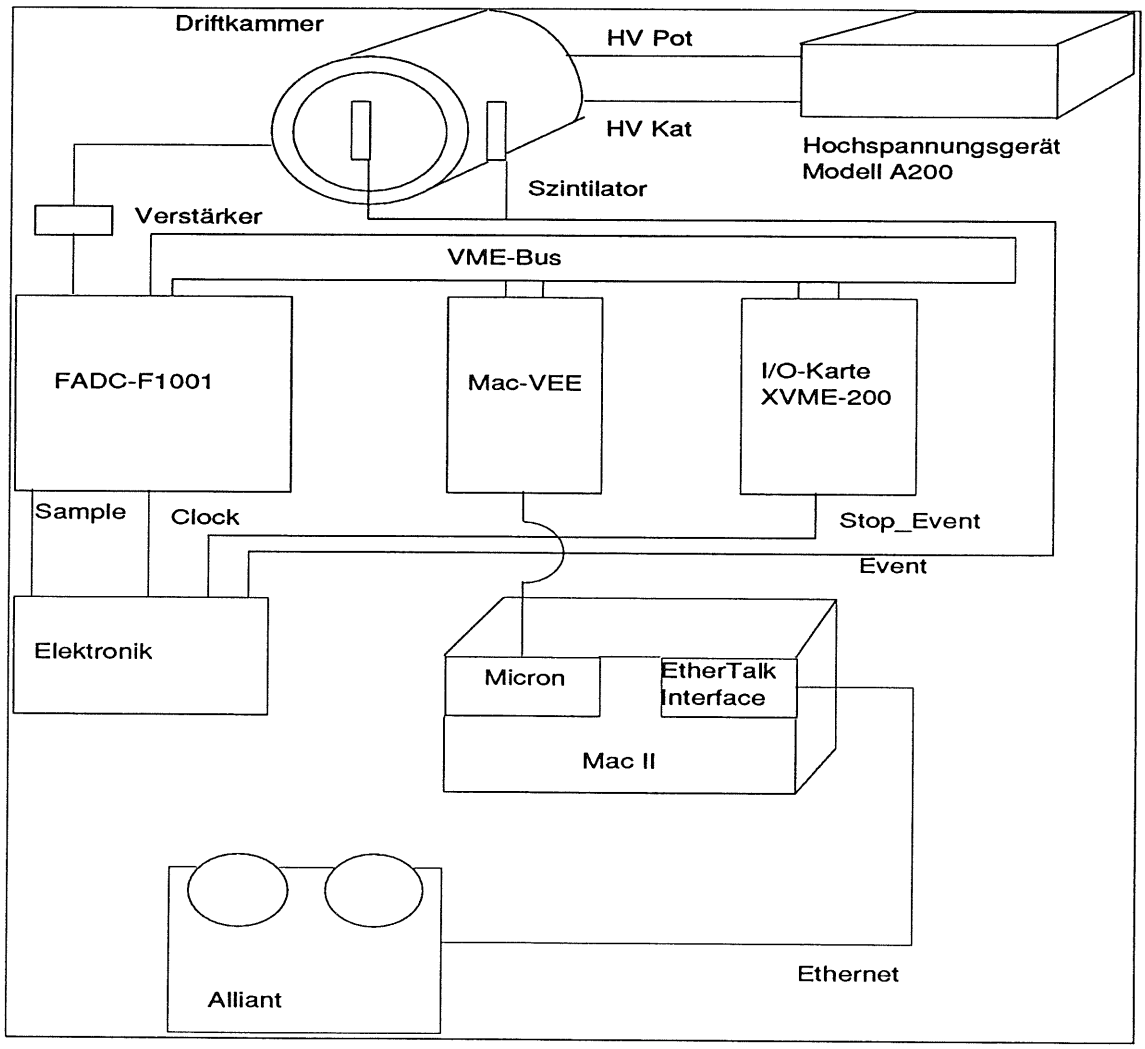


Abbildung III.1 Versuchsanordnung im Normalbetrieb

Der schnelle Analog-Digital-Konverter (100 MHz FADC) konvertiert dauernd (sample) die Signale der Driftkammer und speichert sie in einem Ringspei-

cher. Wenn die Szintillatoren den Durchgang eines Teilchens detektieren, wird von der Elektronik die laufende Konversion des FADC gestoppt und der Computer MacII über ein Signal (Stop-Event) informiert. Dieser liest die Daten des Ereignisses via VME-Bus (Micron Interface-MacVEE) aus dem FADC aus und gibt den normalen Betrieb wieder frei.
Die Daten werden nun mit Hilfe eines "Ethernet"-Anschlusses auf den Grossrechner[12] des Institutes transferiert. Dort werden sie durch ein speziell für diese Driftkammer entwickeltes Programm analysiert und ausgewertet.
Für den Betrieb während der Absoluteichung werden an den VME-Bus zusätzlich die Elektronik und Apparaturen für die Auslesung der 1mm-Proportionalammer (MWPC) angehängt[13] (siehe Abbildung III.2).

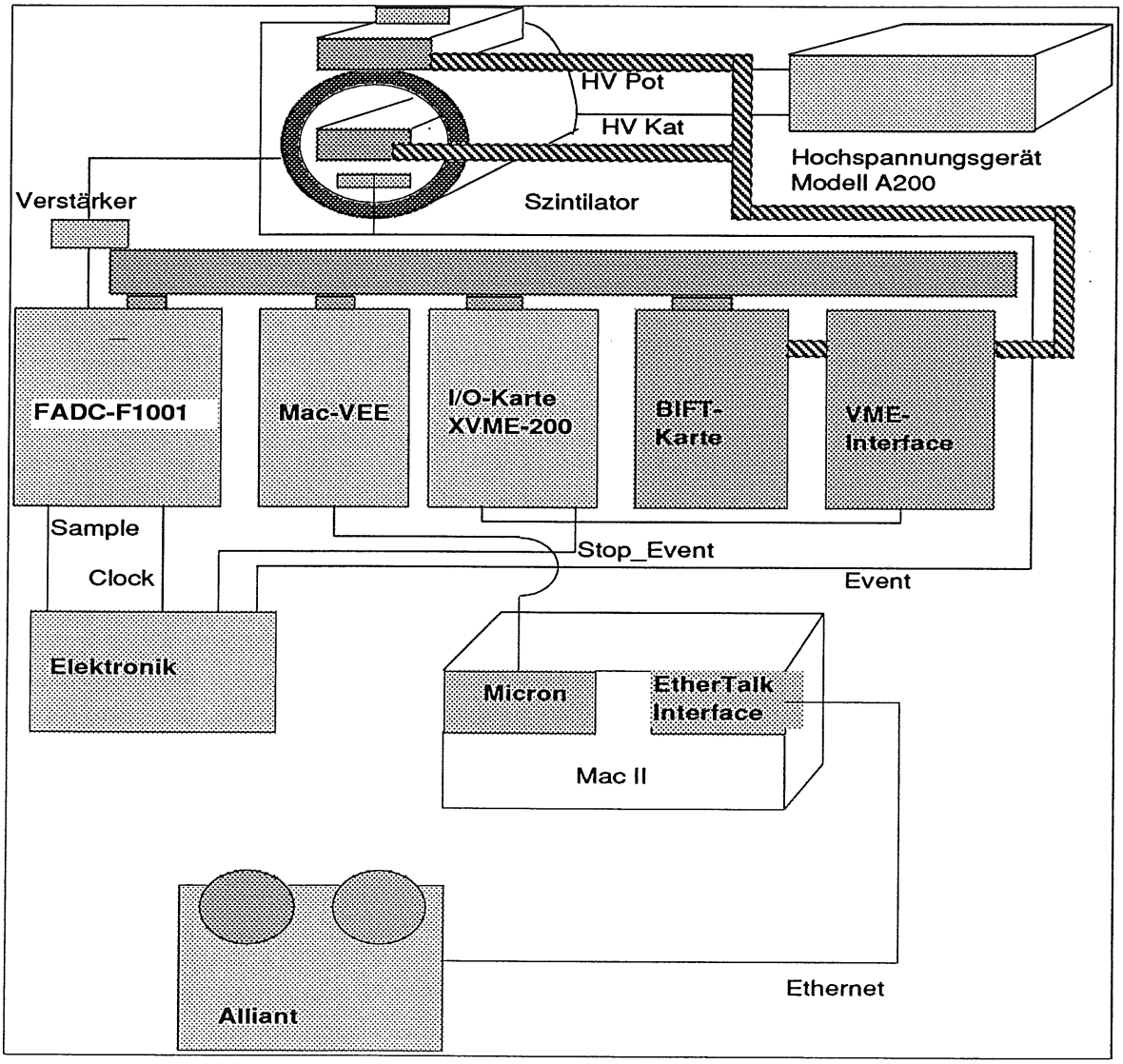


Abbildung III.2 Versuchsanordnung für Absoluteichung

Wird nun von den Szintillatoren ein Teilchen detektiert, so wird durch den MacII nicht nur der FADC ausgelesen, sondern auch die BIFT-Karte, welche

[12] Alliant
[13] Für detailliertere Angaben siehe [10]

die Informationen über die MWPC enthält. Den zeitlichen Ablauf dieser Vorgänge steuert die Elektronik, so dass zu einem Ereignis in der FADC-Karte das entsprechende Ereignis der MWPC ausgelesen wird. Nach einer Messreihe kann wiederum auf dem "Alliant" die Auswertung vorgenommen werden.

III.2 Wissenswertes zu MacII, MPW und C

Dieser Abschnitt dient dazu, Benützern meiner Apparatur einen Einstieg in die Handhabung und Programmiertechnik eines MacII zu bieten. In III.2.ii werde ich auch einige Informationen über die Programmiersprache C vermitteln.

III.2.i MacII

Der MacintoshII arbeitet mit einem 68020-Prozessor von Motorola. Dieser Prozessor macht den MacII hauptsächlich aus zwei Gründen wesentlich schneller als seine Vorgänger: Der 68020 arbeitet mit einer Taktfrequenz von 16 MHz und einer Busbreite zwischen Hauptspeicher und Prozessor von 32 Bits.
Das ist aber heute auf der Stufe PC schon fast normal und unterscheidet den Mac nicht wesentlich von IBM-kompatiblen Computern. Der Hauptunterschied liegt in einem völlig neu und andersartig konzipierten Betriebssystem (DOS =IBM / Multifinder = Macintosh), das für einen reinen Anwender grosse Vorteile bietet. Für "Dos" und die auf dieses Betriebssystem angewiesenen Anwendungsprogramme werden recht grosse Fachkenntnisse für deren sachgemässe Bedienung vorausgesetzt. Ganz anders das von "Maus", "Fenstern" und "Pull Down-Menus" beherrschte Macintoshprinzip, das konsequent bei sämtlichen Anwendungen eingehalten wird. Hat man es bei einer Anwendung erlernt (was sehr anschaulich und einfach ist), so kann man sämtliche weiteren Programme auf die genau gleiche Art bedienen und erhält brauchbare Ergebnisse innert kürzester Zeit, ohne auch nur eine Seite einer Bedienungsanleitung gelesen zu haben.
Genau dieses von Macintosh verlangte Prinzip der "Fenster" und "Pull Down-Menus" macht aber das eigene Entwickeln und Programmieren von Programmen für einen Einsteiger sehr nervenaufreibend. Man kann dieses Problem lösen, indem man nur innerhalb einer Umgebung, in die "Compiler", "Editor" und "Bibliothek" eingebettet sind, programmiert. Um das Programm laufen zu lassen, ist man dann aber immer auf diese Umgebung angewiesen, braucht viel Speicherplatz für Unnötiges und kann vor allem viele für den MacII spezifischen Eigenschaften beim Programmieren nicht ausnützen.
Aus diesem Grund lohnt es sich, sich näher mit dem MacII auseinanderzusetzen, um auch in den selbstgeschriebenen Programmen alle die bei offiziellen Anwendungen als selbverständlich geltenden Mac-Vorteile zur Verfügung zu haben.

a. Mac "User Interface Toolbox"

Unter Mac "User Interface Toolbox" versteht man verschiedene Funktionen und Prozeduren, die direkt aus dem "Read-Only-Memory" (ROM) des Macs dem Benutzer zur Verfügung gestellt werden und die für die Programmierung von Anwendungen unabdingbar sind. Sie sind in sogenannte Untergruppen , die mit dem Namen "Manager" bezeichnet werden, eingeteilt. Teile dieser "Manager" sind auch eine Stufe weiter unten im Betriebssystem untergebracht. Einen Überblick über den logischen Aufbau gibt Abbildung III.3:

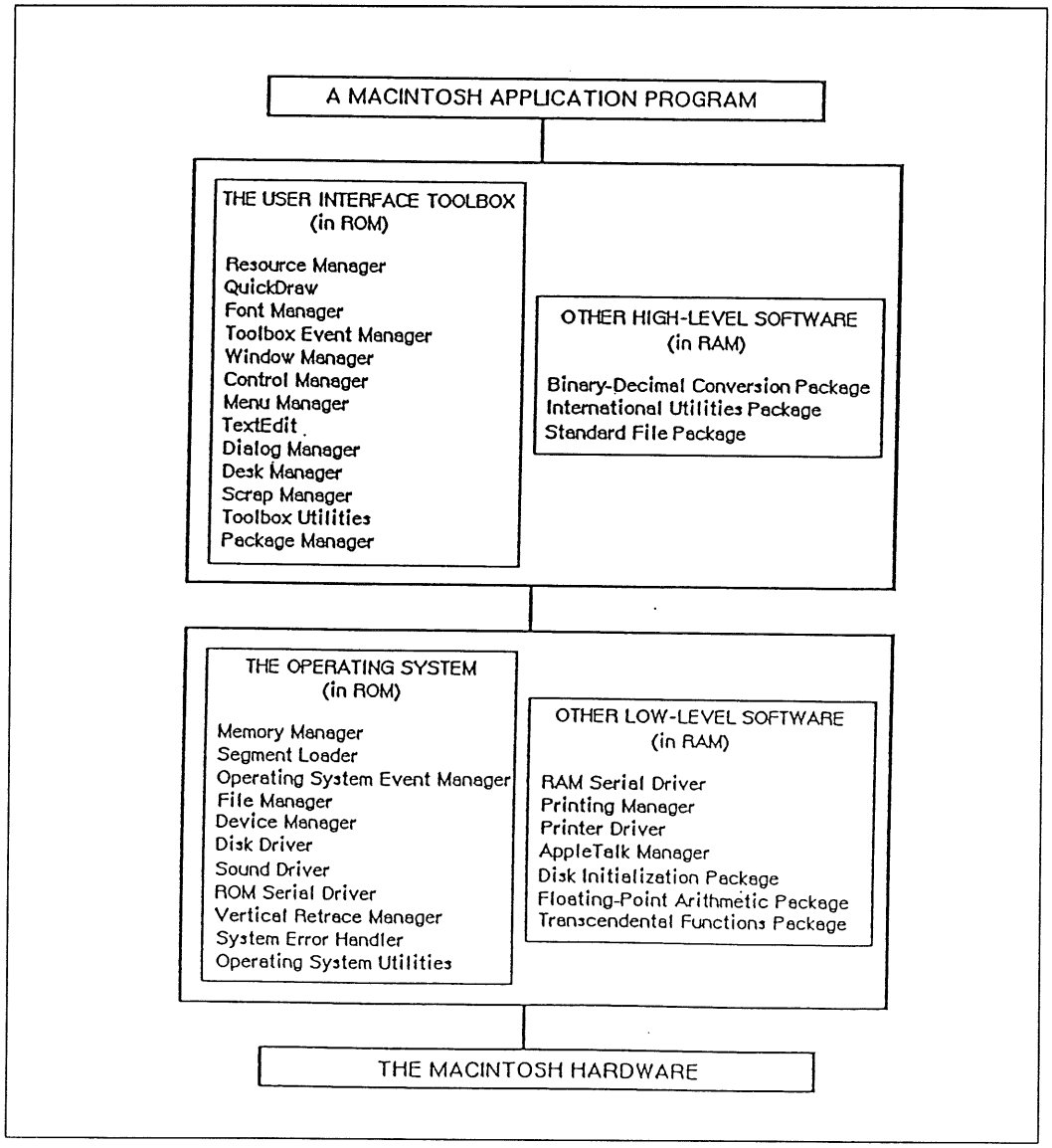


Abbildung III.3 Mac "User Interface Toolbox" (aus [17])

Den genauen Inhalt dieser Manager und die Funktionsweise der darin enthaltenen Funktionen wird ausführlich in "Inside Macintosh"[14] beschrieben.

[14] Siehe [14]

Um aber als Einsteiger einen Überblick zu erhalten, befasst man sich besser mit "Macintosh Revealed"[15].

b. Grundstruktur einer Anwendung

Jedes Anwendungsprogramm auf einem Mac, auch wenn es selber geschrieben ist, sollte die gleiche Grundstruktur verwenden. Diese werde ich hier kurz zu erläutern versuchen.
Einer der Eckpfeiler der Macintosh-Philosophie ist der Versuch, dem Anwender die Möglichkeit zu bieten, dem Computer jederzeit sagen zu können, welchen Befehl er jetzt gerade ausführen soll (und nicht umgekehrt). Der Benützer soll mit Tastatur oder Maus ein Ereignis (Event) generieren können, das dann sofort entsprechend seiner Funktion ausgeführt wird. Das Programm darf also nichts anderes machen, als in einer Schlaufe zu warten, bis ein Ereignis generiert wird um dann den entsprechenden Befehl auszuführen. Das sieht in etwa wie folgt aus:

```
programm Beispiel

BOOLEAN     Finished

MAIN()
   {
      Initialize()
      repeat
            MainLoop()
      until Finished
  }
```

Das Wesentliche geschieht natürlich im "MainLoop()". Hier werden alle verschiedenen Ereignisse, nicht nur die vom Anwender generierten, verarbeitet. Man unterscheidet folgende Ereignisse:

"Maus Event"	Der Anwender hat die Maus gedrückt oder losgelassen.
"Keyboard Event"	Der Anwender hat auf der Tastatur eine Taste gedrückt, sie losgelassen oder sie solange gedrückt, dass sie zu repetieren begann (auto-key-event).
"Disk-Inserted Event"	Eine Diskette wurde ins Laufwerk geschoben.
"Window Event"	Der Anwender hat ein Fenster aktiviert (activate event), es verschwinden lassen (deactivate event) oder ein Fenster verschoben, so dass verdeckte Teile von Fenstern sichtbar wurden (update event).
"Null Event"	Es gibt kein Ereignis auszuführen.

[15] Siehe [15]

In der Funktion "MainLoop()" wird hauptsächlich die Funktion "DoEvent()" ausgeführt:

```
DoEvent()
  {
      if GetNextEvent(EveryEvent,TheEvent) then
        {
            case  TheEvent.what of
                  MouseDown:          DoMouseDown()
                  KeyDown,AutoKey:    DoKeystroke()
                  UpdateEvt:          DoUpdate()
                  ActivateEvt:        DoActivate()
                  otherwise:          DoNothing()
            end case
        }
  }
```

Der Befehl "GetNextEvent" ist ein erstes Beispiel einer Funktion aus der "User Interface Toolbox". Er kontrolliert eine sogenannte "Event Queue" und retourniert in "TheEvent" einen Integer für das entsprechend anliegende Ereignis, das dann in der "case-Schlaufe" ausgeführt wird.
Ich möchte hier nur noch die "DoMousDown"-Funktion etwas weiter verfolgen, weil mit ihr die eigentlichen, vom Anwender geplanten Aktionen zur Ausführung gebracht werden. Zuerst wird mittels der "User Interface Toolbox" Routine "FindWindow()" nachgeschaut, wo genau die Maus gedrückt wurde.

```
DoMouseDown()
  {
      thePart=FindWindow(TheEvent.where,whichWindow)
      case thePart of
            InMenuBar:          DoMenuClick()
            InDrag:             DoDrag(whichWindow)
            InGrow:             DoGrow(whichWindow)
                .                       .
                .                       .
                .                             .
      end case
  }
```

Neben anderen Möglichkeiten, kann das "Mausdrücken" in einem sogenannten Menu vorgenommen worden sein. Mit der Funktion "MenuSelect()" erhält man das angeklickte Menu wie auch das darin ausgewählte "Item" (wird sichtbar, wenn man Menu anklickt).

```
DoMenuClick()
  {
      menuChoice=  MenuSelect(TheEvent.where)
      theMenu=HiWord(menuChoice)
      theItem= LoWord(menuChoice)
      case theMenu of
            AppleID:            DoAppleChoice(theItem)
            FileID              DoFileChoice(theItem)
            EditID:             DoEditChoice(theItem)
            OwnPrgID:           DoPrgChoice(theItem)
      end case
  }
```

Will man ein selber geschriebenes Programm (OwnPrgID) laufen lassen, macht man das mittels der Funktion "DoPrgChoice" und kehrt nach der Ausführung wieder in die "Main"-Schlaufe zurück.
Hält man sich an den soeben beschriebenen Programmaufbau, ist ein erster Schritt getan Kompatibilität mit der Macintosh-Philosophie gewahrt zu haben.
Am Schluss dieses Exkurses über Macintosh möchte ich noch den "Ressource"-Teil eines Files erwähnen, der bei Macintosh-Files immer vorhanden, normalerweise aber nicht zugänglich ist. Im "Ressource"-Teil einer Anwendung können Dinge wie das Aussehen von Fenstern und Menus, die man immer wieder von neuem verwendet, einmal definiert und dann mit speziellen "User-Interface-Funktionen" immer wieder aufgerufen und angewendet werden. Das spart enorm viel Zeit und ist sehr gängig unter Macintosh-Programmierern. Für das Definieren dieser "Ressourcen" braucht man spezielle Programme wie "ResEdit" [16].

III.2.ii MPW und C

MPW [17] ist der Name für einen komplexen Zusammenschluss von verschiedenen Unterprogrammen für die Entwicklung von Programmen für Macintosch-Computer. Neben den Programmiersprachen Assembler, Pascal und C (C++) sind darin folgende Elemente enthalten:

MPW Shell:	Programmier-Umgebung inkl. "Editor"
Projectmanagement system:	Hilfe für die Entwicklung von grossen Programmen, an denen mehrere Leute arbeiten
ResEdit:	"Resource Editor" (siehe III.2.i.b)
Linker:	Wird gebraucht, um Programmteile aneinander zu binden
Make:	Für die Verwaltung von File-Verbindungen
Dialog Interface:	Für die Kommandogebung (z.B. für Shell)
SADE:	Debugger

MPW ist eine Umgebung, in der sehr vieles machbar ist; dadurch entstehen aber beim Einarbeiten gewisse Probleme. Für Files, die mit der richtigen Buchstabenkombination (Extension) enden (für C: .c,für Pascal: .pas), kann mit dem Menu "Create Build Commands" selbständig ein "Make-File" generiert werden. Man kann dabei zwischen einer Anwendung (siehe III.1.b) oder "Tool" (läuft nur innerhalb MPW) wählen. Mit dem Befehl "Build" wird

[16] Siehe [16]
[17] Macintosh Programmer's Workshop

dann das entsprechende File kompiliert. Alles weitere entnehme man "Macintosh Programmer's Workshop"[18].
Die Programmiersprache C wurde von Dennis Ritchie in den Bell-Laboratorien entworfen und 1972 auf einer "PDP-11" implementiert, welche "UNIX" als Betriebssystem benützte. C hat sich aus diesen Wurzeln zu einer reifen Programmiersprache entwickelt und ist heute weit verbreitet. Ein Grund dafür ist sicherlich die enge Verflochtenheit mit "UNIX", in der C sehr oft die bevorzugte Programmiersprache ist. C verbindet die Vorteile von Assembler und Pascal miteinander, kann doch einerseits problemlos auf einzelne Bits oder Adressen zugegriffen werden und andererseits ist strukturiertes Programmieren einfach zu handhaben. Der grösste Vorteil dürfte aber in der grossen "Portierbarkeit" von C-Programmen liegen, ist es doch z.B. möglich, auf dem Mac Programme zu schreiben und sie auf einem IBM-PC laufen zu lassen. Ein weiterer wichtiger Aspekt ist die bescheidene Anzahl von Schlüsselwörtern, nämlich 32 (BASIC kennt 159), was eine grosse Vereinfachung für das Schreiben von neuen C-Compilern ist. C gilt als eine sehr vielseitige Sprache, da sehr viele Operatoren zur Verfügung stehen. Natürlich kennt C auch Nachteile; einer davon ist zugleich auch einer seiner grössten Pluspunkte: C ist nicht so streng wie andere Programmiersprachen. So darf der "Compiler" etwa die Reihenfolge der Auswertung innerhalb von Ausdrücken oder Parametern verändern. Auch werden die "Array"-Grenzen nicht automatisch überprüft. Zusammenfassend meine ich, dass C eine Programmiersprache ist, die dem erfahrenen Programmierer viele Möglichkeiten gibt, ein Programm effizient zu gestalten. [19]

[18] Siehe [17]
[19] Weitere Informationen [18],[19],[20]

III.3 VME-System

Busstrukturen sind die Verbindungsschienen zwischen den Bausteinen und -gruppen eines Mikro- oder Minicomputersystems. Da der gesamte Datenaustausch über derartige Bus-Systeme erfolgt, beeinflusst deren Struktur die Leistungsfähigkeit des Gesamtsystems in entscheidendem Masse. Für moderne Hochleistungsprozessoren braucht man darum technologische Spitzenprodukte als Bus-System, will man die Leistungsfähigkeit der Prozessoren voll ausschöpfen.
VME ist eines dieser neu entwickelten 16/32 Bit Systeme. Der VME-Bus zeichnet sich durch folgende Leistungsmerkmale aus[20] :

- Unterstützung von Mikroprozessor-Architektur bis zu 32-Bit Wortbreite
- Unterstützung von "Multiprozessor-Systemen"
- Datendurchsatz im praktischen Betrieb 34 MBytes/s (theoretisch 57 MBytes/s)
- Vollständig asynchrones, multiplexfreies Busprotokoll
- Prioritätsgesteuerte Busbelegung über vier Prioritätsebenen und zusätzlichem "Daisy-Chain" auf jeder Ebene
- Unterstützung von zentraler oder verteilter Interrupt-Verarbeitung in sieben Prioritätsebenen

Als 24-Bit Adress- und 16-Bit-Datenbus ist der VMEbus auf Einfach-Europa-Karte implementierbar, für den vollen 32-BitAdress- und Datenbereich wird eine Doppeleuropakarte benötigt (siehe Abbildung III.4).

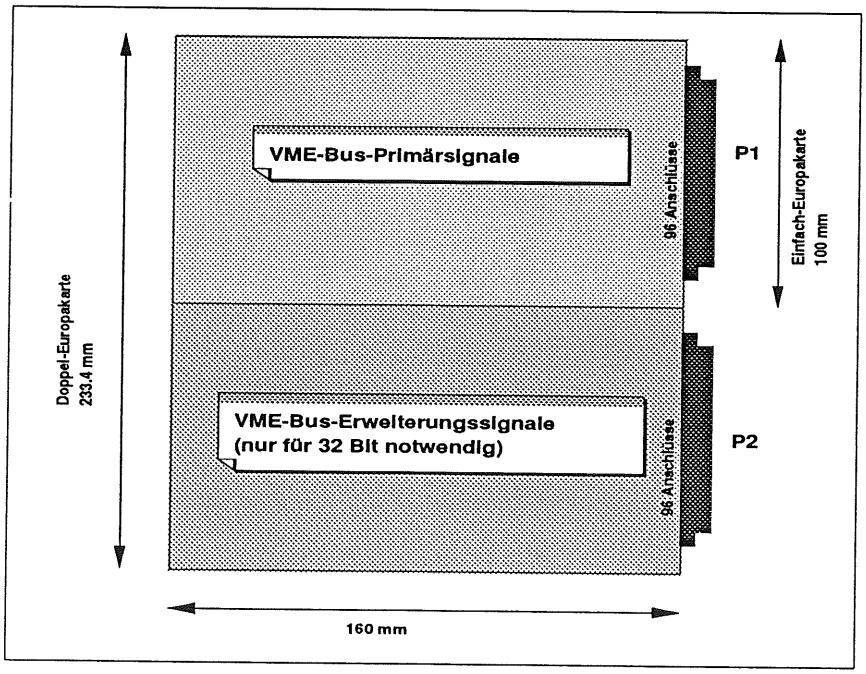


Abbildung III.4 Grössen der VME-Karten

[20] Aus [11]

Der VMEbus gliedert sich in vier unabhängige Teilsysteme:

- Der "Daten-Transfer-Bus" (DTB) enthält alle Daten- und die Adressleitungen sowie die zum "Datentransfer" notwendigen Steuerleitungen.
- Der "Arbitrations-Bus" liefert alle Signale, die zur Steuerung in einem "Multi-Master-System" notwendig sind.
- Der "Interrupt-Bus" dient zur Behandlung von Unterbrechungsanforderungen.
- Unter dem Begriff "Versorgungs- und Hilfsleitungen" werden alle restlichen Signale, die Stromversorgung und z.B. Fehlererkennung zusammengefasst (siehe Abbildung III.5).

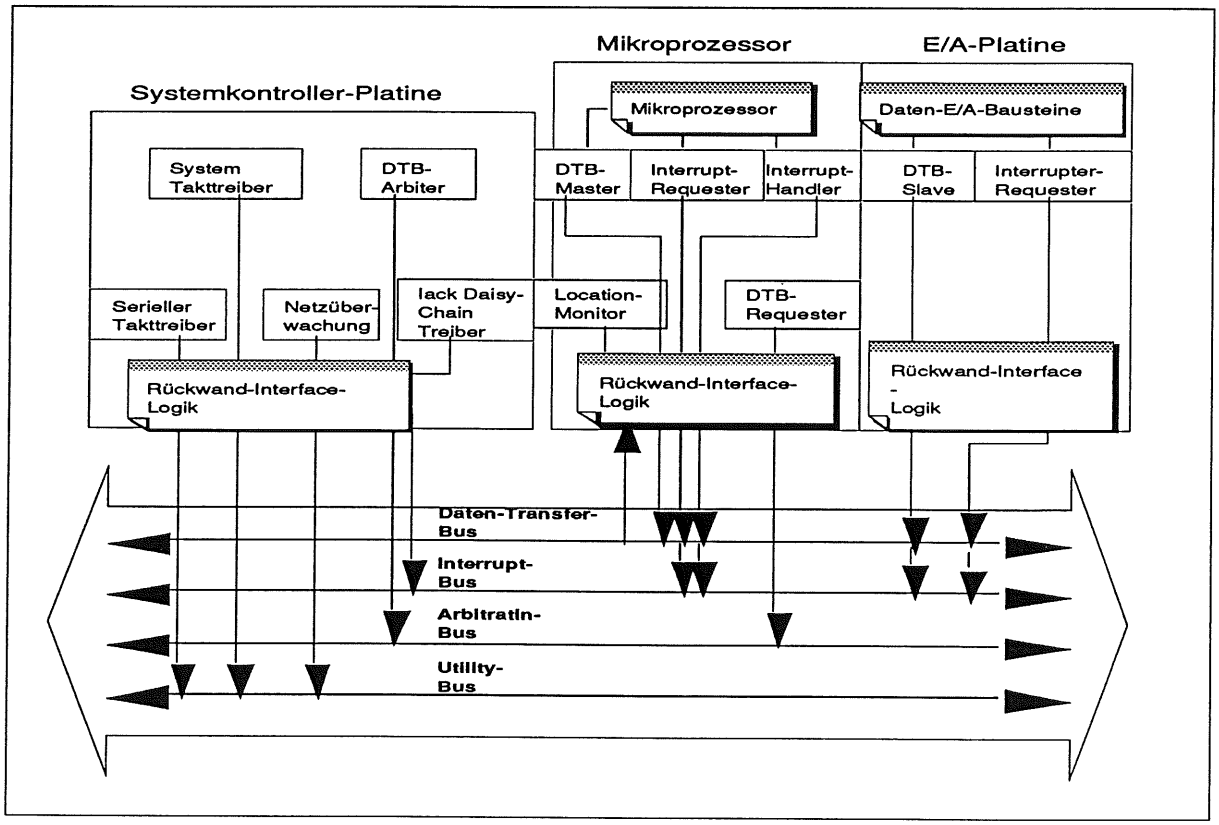


Abbildung III.5 VME-Bus

III.3.i "Daten-Transfer-Bus"

Der "Daten-Transfer-Bus" erlaubt eine asynchrone Datenübertragung mit einer maximalen Geschwindigkeit 8.5 Mio. 32-Bit-Transfers/s(= 34 MByte/s). Die Datenübertragung zwischen Sender (Master) und Empfänger (Slave) läuft folgendermassen ab (Abbildung III.6):

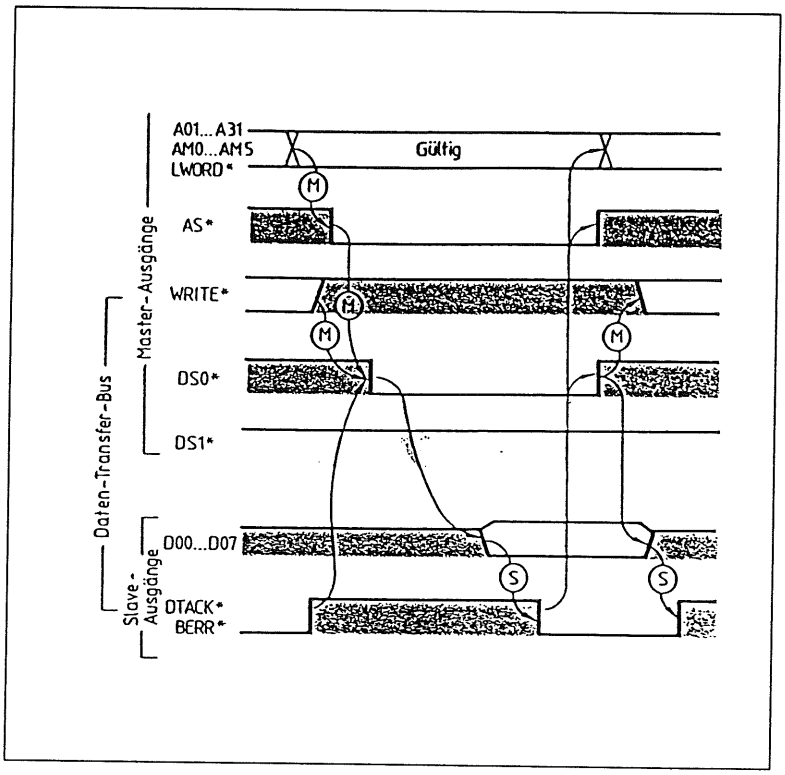


Abbildung III.6 Ablauf des Daten-Transfers

Das Anliegen einer gültigen Adresse auf dem Adressbus wird dem "Slave" durch Aktivierung des Signals "Address Strobe" (AS) mitgeteilt. Das Signal "Write" zeigt an, ob es sich um eine Schreib- oder Leseoperation handelt. Bei einem Lesezyklus wird dem "Slave" über die beiden "Data-Strobe"-Leitungen DS0 und DS1 mitgeteilt, dass der "Master" zum Einlesen der Daten vom "Slave" bereit ist. Daraufhin legt der adressierte "Slave" die Daten auf den Datenbus und signalisiert, dass gültige Daten anliegen über die Leitung "Data Acknowledge" (DTACK). Analog werden beim Schreibzyklus zunächst vom "Master" die Daten auf den Datenbus gelegt und danach DS0 und DS1 aktiviert. Wenn der "Slave" die Daten aufgenommen hat, signalisiert er dies durch Aktivierung von "DTACK". In beiden Fällen gibt der "Master" den Bus für den nächsten Zyklus erst frei, wenn er das Quittierungssignal "DTACK" vom "Slave" erhalten hat. Der "Slave" seinerseits gibt den Datenbus erst frei, wenn der "Master" die "Datenstrobes" DS0 und DS1 deaktiviert hat. Über dieses sogenannte "Handshaking" ist die problemlose Zusammenarbeit von Einheiten unterschiedlicher Geschwindigkeit möglich. Falls der "Slave" aufgrund einer fehlerhaften Signalkombination auf dem "DTB" nicht in der Lage ist, die Schreib/Lese-Anforderung ordnungsgemäss zu bearbeiten, so sendet er statt "DTACK" das Signal "Bus ERROR" (BERR)

aus. Der "Master" geht dann in die entsprechende Fehlerbehandlungsroutine.

III.3.ii "Arbitrationsbus"

In einem Multiprozessor-System können zwei oder mehrere "Master" auf den "DTB" zugreifen. Die Koordination wird vom "Arbiter" vorgenommen. Jeder "Master" kann über eine der vier "Bus-Request"-Leitungen BR0..BR3 den Bus anfordern. Jede Leitung entspricht einer bestimmten Priorität. Der "Arbiter" vergleicht die Busanforderung mit der Priorität des den Bus innehabenden "Masters"; ist letztere höher, so kann die Anforderung erst bedient haben, wenn der Bus freigegeben wird. Andernfalls aktiviert der Arbiter das Signal "BusClear" (BCLR), um den gerade aktiven "Master" zur Busfreigabe aufzufordern. Wenn dieser an einem passenden Punkt zur Unterbrechung seiner Aktivität angekommen ist, signalisiert er dies durch Deaktivierung von "Bus-Busy" (BBSY). Daraufhin schickt der "Arbiter" der höchsten anstehenden Priorität eine Busfreigabe (Bus-Grant BG0...BG3) und deaktiviert "BCLR".

III.3.iii Hilfs- und Versorgungssignale

Der VMEbus verfügt über spezielle unabhängige Signale für Synchronisation, Initialisierung, Systemtest und Fehlerdiagnose. Diese Hilfssignale stellen einen wesentlichen Beitrag zur Gesamtleistungsfähigkeit eines modernen Busystemes dar. Der VMEbus unterstützt folgende Funktionen:

- Systemtakt (SYSCLK)	Ist ein vom Prozessortakt unabhängiges 16-MHz-Taktsignal, das in keiner festen Phasenbeziehung zu anderen VMEbus-Signalen steht. "SYSCLK" kann beispielsweise für Zähler- und Synchronisationsfunktionen herangezogen werden.
- System-Reset (SYSRESET)	Über "SYSRESET" können alle an den VMEbus angeschlossenen "Module" in einen definierten Anfangszustand gebracht werden. "SYSRESET" kann manuell oder von einem "Power-Monitor-Modul" automatisch beim Einschalten der Stromversorgung aktiviert werden.
- System-Test-Leitung (SYSFAIL)	Über "SYSFAIL" kann ein Fehler im System gemeldet werden. Ein typischer Anwendungsfall sind Einschübe, die nach der Aktivierung von "SYSRESET" einen Selbsttest auf der Platine durchführen. Zeigt der Test einen Fehler, so wird "SYSFAIL" nicht deaktiviert und der "System-Controller" muss entsprechende Massnahmen ergreifen.

- (ACFAIL) Über diese Leitung wird dem System ein Spannungseinbruch gemeldet. Die VMEbus-Spezifikation schreibt eine Auslegung des Netzteils vor, die für den ordnungsgemässen Abbruch der laufenden Aktivitäten minimal 4ms zulässt.

III.3.iv Mac-Interface zu VME

Um mit dem Mac auf das VME-System zugreifen zu können, braucht es ein spezielles "Interface" sowohl auf der Seite des Mac's (Micron) wie auch auf der Seite des VME-Crates (MacVEE). Ausführlich sind sie beschrieben in "The MICRON User Manual" [21].

a. MICRON

Der MacII enthält sechs Erweiterungs-"Slots" mit 96 "Pin"-Steckern, die alle mit dem speziellen "NuBus" verbunden sind. In diese "Slots" können die verschiedensten Erweiterungs-Karten eingeschoben werden, unter anderem die Video-Karte für den Bildschirm und auch die Micron-Karte. "NuBus"-Karten konfigurieren sich selber, da sie in einem "ROM-Chip" alle nötigen Informationen für ihren Betrieb gespeichert haben und beim Einschalten des Mac's, diese ans Betriebssystem weitergeben. Das bedeutet, dass man die Micron-Karte an jeder beliebigen Stelle einschieben kann, ohne irgendwelche Adress-Schalter setzten zu müssen. Der "NuBus" ist eine asynchrone Busstruktur mit 32-Bit Adress- und Datenlinien. Man sollte sich von der Angabe "32-Bit Adresslinien" nicht täuschen lassen, im Normalzustand arbeitet der Mac im "24-Bit-Modus" und muss also unbedingt vor jedem Zugriff auf das Micron auf den "32-Bit-Modus" umgeschaltet werden. Dazu dient die "User Interface Toolbox"-Funktion "SwapMMUMode". Das sieht in C wie folgt aus:

```
#define  MMU32 {char mode; mode = true32b; SwapMMUMode(&mode);}
/* schaltet Mac II um von "24 Bit-Mode" auf "32 Bit-Mode". (V-592
                    InsideMacintosh)*/

#define MMU24  {char mode; mode = false32b; SwapMMUMode(&mode);}
  /* schaltet MacII um von "32 Bit-Mode" auf "24 Bit-Mode". (V-592
                    Inside Macintosh)*/
```

Mit dem Aufruf von "MMU32" schaltet man den Mac also um in "32-Bit-Mode". Ebenso wichtig wie das Einschalten von "MMU32" ist aber sofort nach dem Zugriff auf die Micron-Karte das Zurückschalten auf den "24-Bit-Mode", da viele "User-Interface-Funktionen" im "32-Bit-Mode" unkorrekt arbeiten und unvorhergesehene Ergebnisse produzieren.

---

[21] Siehe [21],[22]

"32-Bit-Mode" bedeutet 8 Hexa-Ziffern zur eindeutigen Adressierung. Die erste Ziffer braucht man für das Ansprechen der Micron-Karte, je nach gewählter Nummer der Einschubstelle (0x9-0xE) ergibt das eine Adresse (bei uns z.B. 0xB) von 0xBx xx xx xx.

b. MacVEE

Damit Micron und MacVEE einwandfrei laufen, müssen die folgenden Bedingungen durch die MacVEE-Karte erfüllt sein. Die MacVEE-Karte muss unbedingt im ersten "Slot" des VME-Crates untergebracht sein. Auf der Karte müssen einige Schalter und Brücken richtig gesetzt werden (Abbildung III.7).

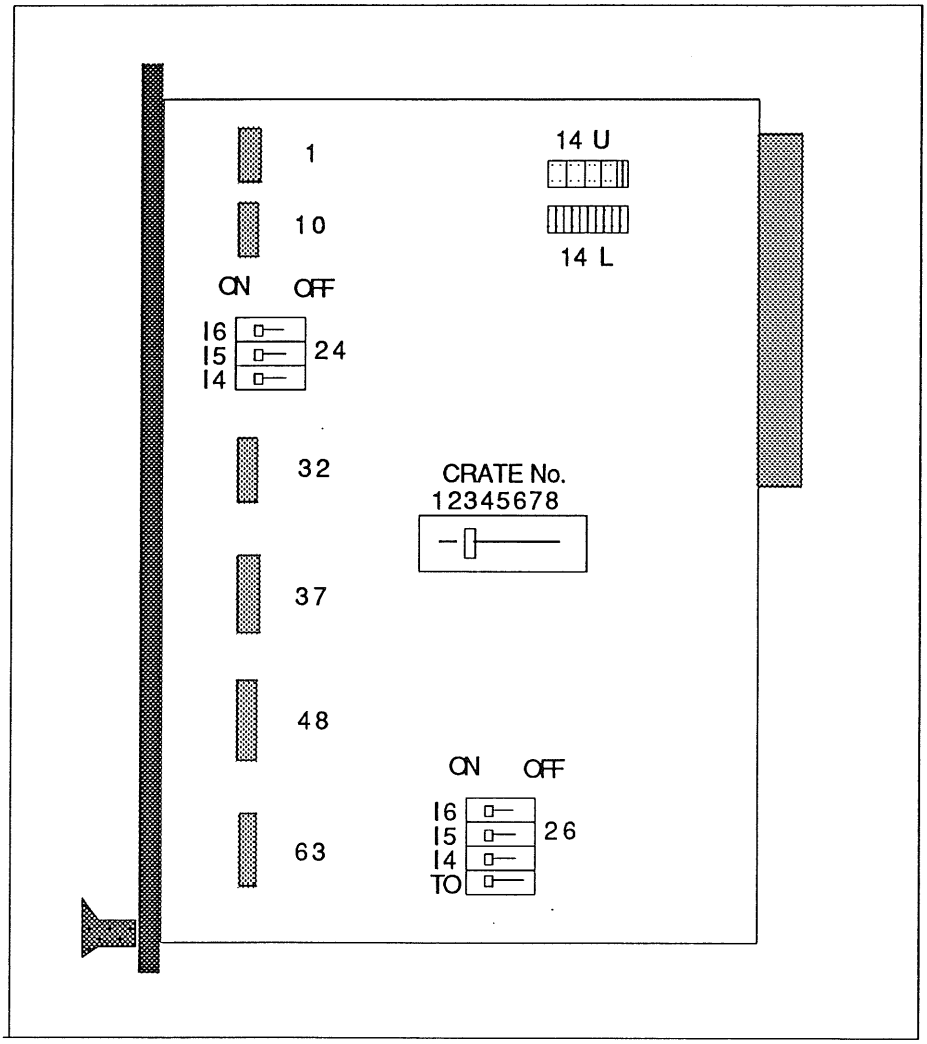


Abbildung III.7 Brücken für MacVEE

Der wichtigste davon ist sicherlich die Einstellung der "Crate"-Nummer, in der das MacVEE sitzt (bei uns "Crate" Nr. 0). Da die zweite Hexa-Ziffer für die Angabe der "Crate"-Nummer reserviert ist, lautet die Adresse 0xB0 xx xx xx ("Crate"-Nummer - 1 muss eingesetzt werden).

Die Schalter I4, I5, I6 und TO wurden auf "ON" gesetzt (siehe MacVEE-User-Manual[22]). Die Schalter 14U und 14L wurden in der "System Controller Configuration" benützt, und zwar so, dass ein "RESET" von der MICRON-Karte (MacII) einen "SYSRESET" im ganzen VME-Crate generiert.
Um die Funktionstüchtigkeit der Verbindung Mac - VME-Crate zu überprüfen, hat sich eine von W.J.Haynes geschriebene Programmbibliothek[23] bestens bewährt. Vor allem die Befehle "SLOT_STATUS", "SLOT_DEScription" und "CRATE_DEScripton" geben einen Überblick, was vom Mac überhaupt gesehen und angesprochen werden kann.

[22] Siehe [22] Seite 26/27
[23] Siehe [23]

III.4 Digitales "In/Out-Modul"

Um mit einem Computer direkten Einfluss auf Eingänge- oder Ausgänge zu nehmen, sie auf Ein oder Aus zu setzen bzw. zu lesen, braucht man ein "I/O-Interface" in unserem Fall ein digitales "I/O-Modul" der Firma XYCOM (XVME-200), das speziell für eine Verwendung in einem VME-Crate konzipiert wurde.
Insgesamt stellt dieses "I/O-Modul" (DIO) 32 digitale In/Out-Kanäle (TTL) zur Verfügung. Am Rande sei erwähnt, dass die Anwendungsmöglichkeiten dieser Karte weit über die von uns gewünschten Anforderungen hinausgeht, werden doch volle VME-Bus "Interruptfähigkeit" und zwei 24-Bit "Timer" zur Verfügung gestellt.
Das Herzstück dieser Karte sind zwei MC68230 Prozessoren, die in einem seperaten Manual beschrieben werden[24]. Jeder dieser "Chips" (PI/T#1,PI/T#2) stellt zwei Anschlüsse zu 8 Bit zur Verfügung, insgesamt also vier Anschlüsse zu 8 Bit (= 32 Bit insgesamt) mit den Namen A1,B1 bzw. A2,B2. Jeder dieser vier "Data-Ports" kann unabhängig von den anderen benutzt werden und besitzt einen eigenen 8-Bit "Transceiver", der unter anderem die Datenflussrichtung bestimmt. Die Verbindung nach aussen wird mit zwei 100 poligen Steckern sichergestellt.
Ein dritter Anschluss auf jedem "Chip" (C1,C2) ist als "Kontroll-Port" für "Interrupt-Handling", "Timer-Operationen" und "Data-Port" Richtungsbestimmung konzipiert. Dieser Anschluss hat keine Verbindung nach aussen und wird vom Computer aus je nach gewünschter Funktionsart programmiert.

[24] Siehe [12]

Eine Übersicht über den Aufbau der ganzen Karte gibt Abbildung III.8.

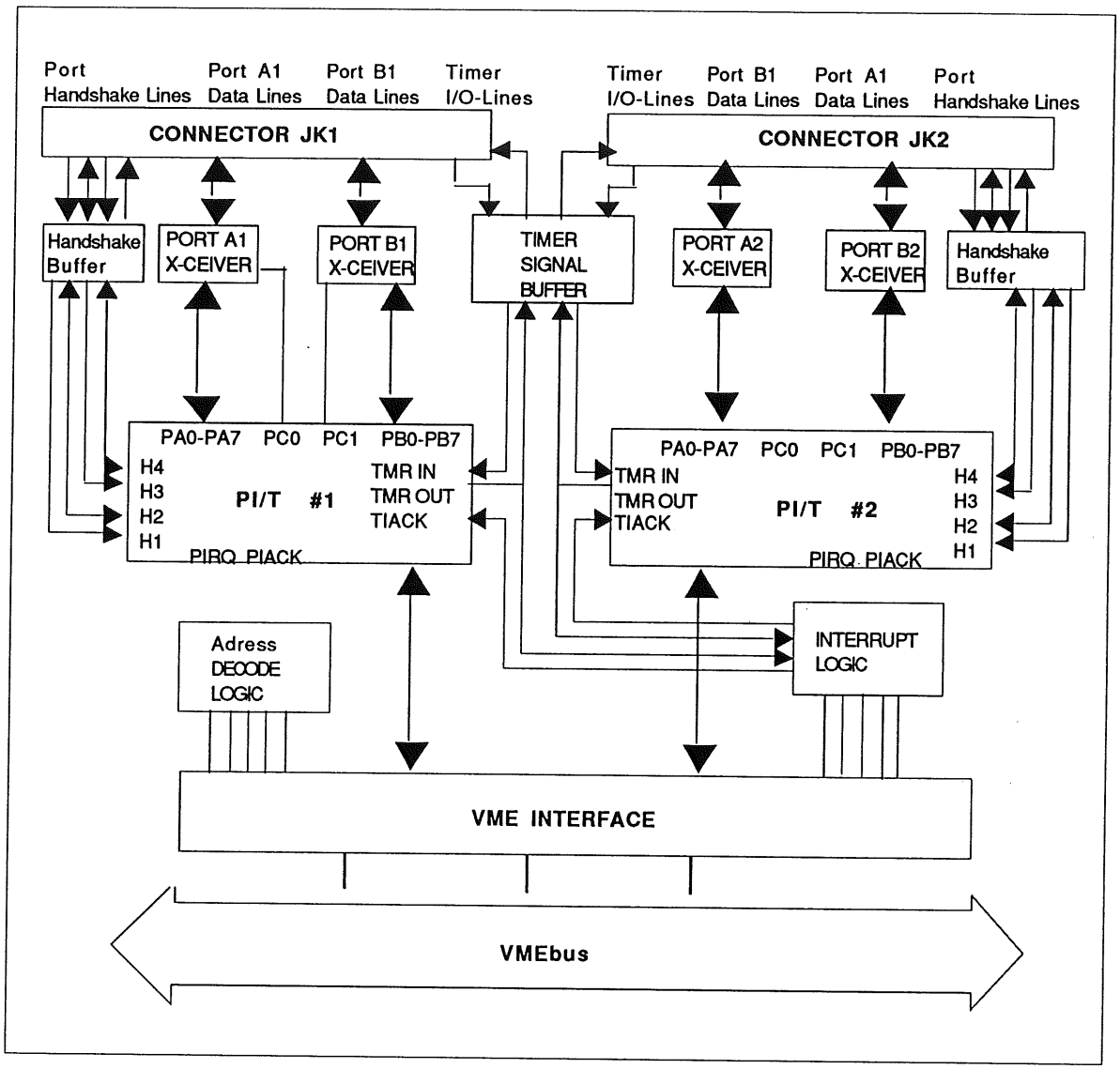


Abbildung III.8 Übersicht über "I/O-Modul"

Die Karte kann grundsätzlich in vier verschiedenen Arten betrieben werden:

Mode 0
(Unidirectional 8Bit Mode): Jeder der vier "Ports" kann entweder als "Input" oder als "Output" programmiert und verwendet werden.

Mode 1
(Unidirectional 16-Bit Mode): Immer zwei "Ports" werden zusammengefasst und dann gemeinsam als "Output" oder "Input" programmiert und verwendet.

Mode 2 (Bidirectional 8-Bit Mode):	Jeder "Port" kann zugleich als "Input" wie auch als "Output" verwendet werden.
Mode 3 (Bidirectional 16-Bit Mode):	Analog können zwei zusammengefasste "Ports" sowohl als "Input" wie auch als "Output" verwendet werden.

Jeder dieser "Modes" kennt nun noch verschiedene "Submodes", die diverse Spezifikationen erlauben, wie die Art des "Data-Bufferings"[25] .
Um die oben aufgeführten Betriebsarten zu wählen, sind zusätzlich sogenannte "Handshake-Linien" (H1-H4) vorhanden. Ausserdem können einzelne dieser Linien (H2,H4) dazu verwendet werden, den Zustand sogenannter "Handshake-Pins" (H2S,H4S) auf den Verbindungssteckern gegen aussen anzugeben.
Auf der Karte müssen ausserdem Brücken gesetzt werden (siehe Abbildung III.9).

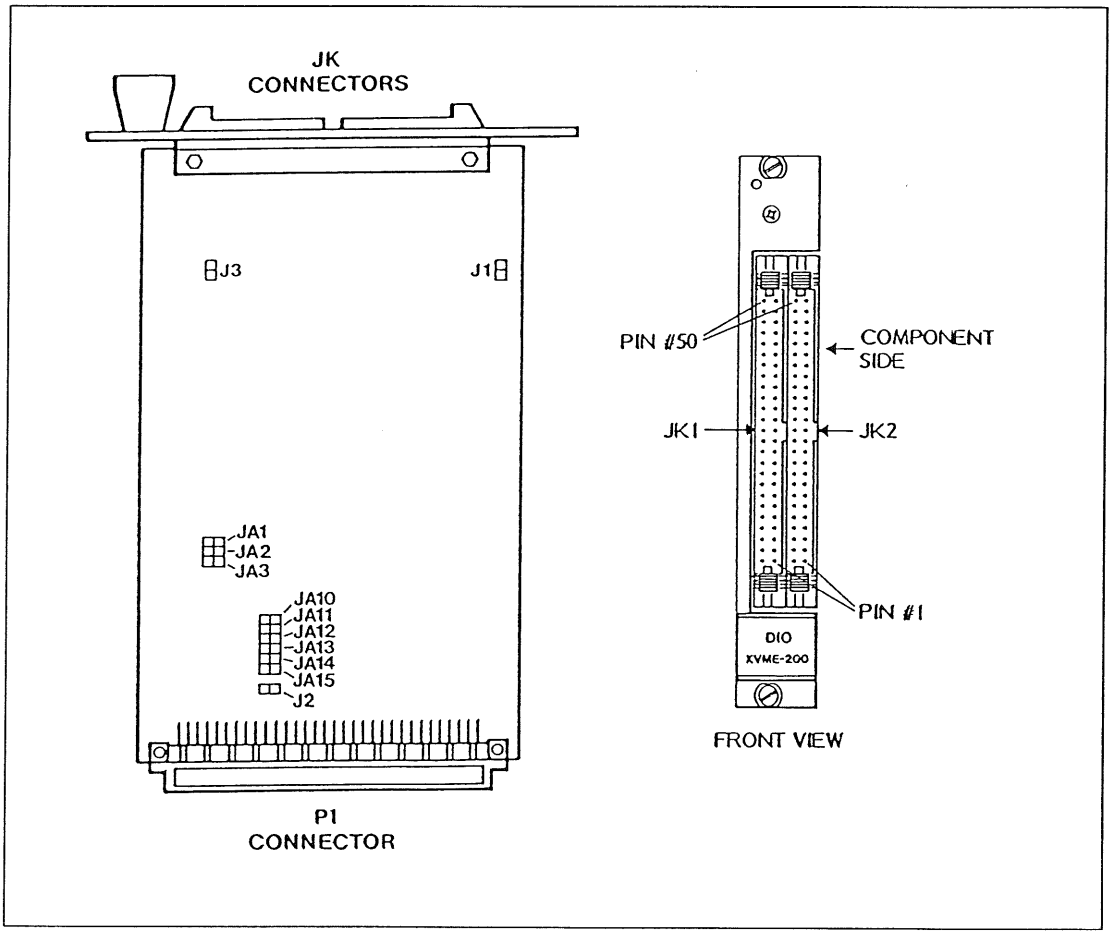


Abbildung III.9 Brücken "I/O-Modul"

Die genaue Position hängt wiederum vom gewählten Betriebsmode ab.

---

[25] Siehe [12]

Für das normale Auslesen der FADC-Karte brauche ich ein einzelnes "Output-Bit" (A0) und ein "Bit", dass mir zeigt, ob von aussen seit dem letzten Auslesen ein Signal angekommen ist (H2S).
Für den Betrieb bei der Absoluteichung brauche ich zusätzlich 8 "Output-Bit" und 3 "Input-Bits".
Ich initialisiere darum die "In/Out-Karte" wie folgt:

JK1 (Stecker 1):	Port A:	Output
	Port B:	Input
JK2 (Stecker 2):	Port A:	Output
	Port B:	Input
H2		asserted edge sensitiv input (H2S wird gesetzt, wenn in H2 ansteigende Flanke ankommt)

Das ergibt folgende Steckerbelegung:

J1:	out
J2:	out
J3:	in
JA1-JA3	in
JA10-JA15	in,in,out,in,in,out (Basisadresse der Karte: 0x9000)

Obige Initialisierung ist in Anhang VI.4 ausprogrammiert.
Dieser Einstellung liegt die Idee zugrunde, "JK1" für die reinen Belange des einfachen Auslesens des FADCs zu verwenden, wärenddem "JK2" allein für die BIFT-Karte reserviert bleibt.
Gebe ich bei dieser Einstellung in "Pin H2" das von der Elektronik beim Durchgang eines Teilchens durch die Driftkammer generierte Signal ein, so kann ich dieses sehr kurze Setzen dieses Kanales durch repetitives Auslesen des "H2S Status-Bits" feststellen und mit dem Auslese-Prozess beginnen. Ist der Mac mit dieser Arbeit fertig, so kann er das mit Setzten von "A0" nach aussen mitteilen.
Ich möchte dieses kurz anhand einiger Programm-Sequenzen illustrieren:
Wie erwähnt, kann man die Basisadresse mittels Brücken einstellen und erhält darum mit der VME-Adresse zusammen: 0x B0 7F 90 00

```
#define IoBasAdress 0xB07F9000          /*Basis-Adresse der
                                        I/O-Karte*/
```

In einer Schlaufe wird mittels "Event_Stop()" kontrolliert, ob ein Ereignis eingetroffen ist, und das allfällig nötige Auslesen des FADCs wird aufgerufen.

```
do
{
      if (Event_Stop())                 /* Ist ein Stop Signal gekommen? */
            {
                  Auslesen();           /* wenn ja auslesen */
                  Anz_Auslesen+=1;      /* Anzahl der Auslesungen um 1
                                        /erhöhen*/
            }
} while (Ende);                         /* Schlaufe abbrechen, wenn
                                        Programmabbruch */
```

Die Funktion "Event_Stop()" macht nichts anderes, als "H2S" auszulesen:

```
short  Event_Stop()
{
      byte nofi;
      byte *IoAdress;

      IoAdress = (byte *) IoBasAdress;
      MMU32;
      nofi=IoAdress[PSR];                    /* Portstatusregister
                                             /auslesen*/
      MMU24;
      if ((nofi&0x2)?1:0)                    /* Kontrolle ob Bit 2 = 1 */
            {
                  return(TRUE);
            }
      else
            {
                        return(FALSE);
            }

}
```

"H2S" ist das zweite "Bit" des "Port-Status-Registers". Das Register hat also den Wert 2, wenn "H2" vom "Photo-Multiplier" gesetzt wurde und ist gleich 0, wenn kein Ereignis eingetroffen ist.
In dieser Sequenz sehen wir auch das erste Mal das Arbeiten mit "Pointern" innerhalb von C:
"IOAdress" (Oder IoBasAdress) ist als "Pointer" auf einen Datentypen "Byte" (unsigned char) deklariert. Das "Port-Status-Register" liegt aber an der Adresse "IOAdress + PSR"(= 27). Die einfachste Möglichkeit der Adressierung ist darum die "Array"-Schreibweise "IOAdress[PSR]".
Das Löschen des Inhaltes der Register bewerkstelligt man mit dem Einlesen vom Wert 1 ins entsprechende "Bit".

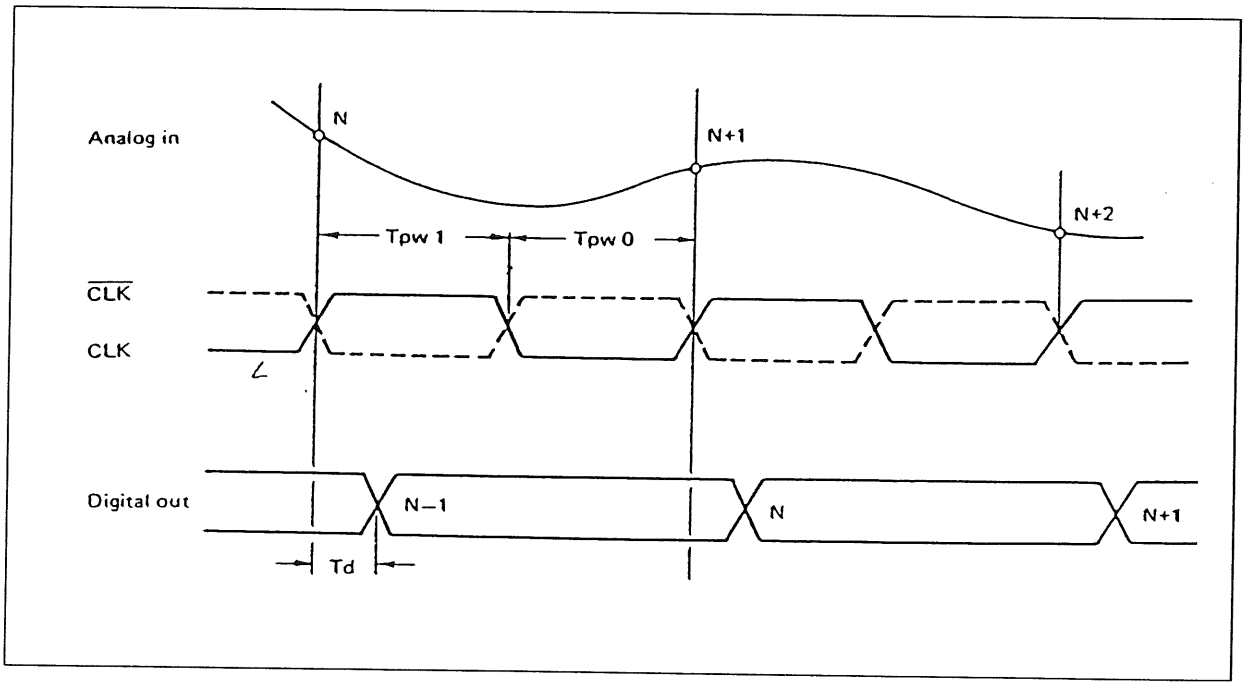
```
H2S_loeschen()
{

      byte *IoAdress;

      IoAdress = (byte *) IoBasAdress;
      MMU32;
      IoAdress[PSR2]=0x02;

      MMU24;
}
```

III.5 FADC-Karte

Die von uns verwendete FADC-Karte[26] F1001 wurde in Hamburg speziell für die Erfordernisse bei HERA gebaut. Es handelt sich um ein 16 Kanal-System, das um FADC-Bausteine vom Typ SONYCX20116 aufgebaut ist. Die Bausteine ermöglichen eine Wandlungsrate von 100MHz und verfügen über eine

[26] Flash Analog Digital Converter

Auflösung von 8-Bit. Zu jedem Kanal ist ein Speicher mit einer Tiefe von 256 Worten auf der Karte untergebracht. Der FADC ist als VME-Modul konzipiert und verfügt darum über zwei VME-Anschlüsse auf der Rückseite. Zudem ist dort auch ein 96-Pol-Stecker angebracht, über den die analogen Eingangssignale der FADC-Karte zugeführt werden. Auf der Frontseite kann mittels des gleichen Steckertypes auf dieses analoge Eingangssignal zugegriffen werden. Eine Anschluss-Tabelle der Stecker ist in Anhang VI.5 abgebildet. Für die Vergrösserung des dynamischen Bereiches sorgt eine Eingangsbeschaltung mit einer nichtlinearen Kennlinie [27].
Grosse Probleme hatten wir mit der Form des "Clock"-Signales. Als Generator benutzten wir aus Kostengründen einen "NIM"-Einschub, der die von Sony für den "FADC-Chip" verlangten Anforderungen nur knapp erfüllt (Abbildung III.10):



$\overline{\text{CLK}}$	:= Clock input pin, ECL level	
CLK	:= Inverse clock input pin, ECL level	
Clock Pulsweite	: Tpw1	min 7.5 ns
	Tpw0	min 2.5 ns

Abbildung III.10 "Clock"-Signal für FADC-Chip

Durch ein sehr kurzes Verbindungskabel zwischen "Clock-Generator" und FADC ist das System aber betriebsfähig.
Die Einstellung der Brücken auf der Karte können dem Anhang VI.6 entnommen werden.
Wie erwähnt, können 16 analoge Signale (X0 - Y7) im FADC eingelesen werden. Jedes Mal, wenn ein "Clock-Signal" gegeben wird (alle 10 ns, da 100MHz

[27] Siehe IV.1

FADC), wird an jedem Draht die momentan herrschende Spannung mit Referenzspannungen verglichen, und ein entsprechender digitaler Wert (zwischen 0-256) wird im "RAM" des jeweiligen Einganges abgelegt; das heisst, um genauer zu sein, im "RAM" mit der Adresse Draht[Counter]. "Counter" wird nun um eins erhöht und beim nächsten "Clock" werden die nächsten 8 Bit des Ringspeichers beschrieben. Das passiert aber alles nur, wenn "Sample" aktiv ist.
Vorausgesetzt, dass "Sample" inaktiv ist, besteht das Auslesen des FADCs aus zwei wesentlichen Komponenten: Zuerst einmal muss der momentane Stand des "Counters" ausgelesen werden. So wie die Adress-Brücken bei uns gesteckt sind, liegt der "Counter" bei der FADC-Adresse 0x 41 00 00 (die 4 kann mit Brücken frei gewählt werden). Die entsprechende C-Funktion hat darum die Form:

```
void ReadCounter(a)

ModuleRecord  *a;
{
      MMU32;
      a->Counter=*((short*)0xB0410000);
      MMU24;
}
```

Die zweite Komponente besteht im eigentlichen Auslesen der gespeicherten Daten der 16 Drähte.
Um die Übergangszeit so kurz als möglich zu halten, liest man nun nicht alle 16 Drähte à 8 Bit einzeln , sondern immer zwei Drähte zusammen (16 Bit := word:=unsigned short) aus. Es ist später kein Problem, diese zwei Werte wieder zu trennen. Man wendet also die Drahtauslesefunktion nur 8 Mal an:

```
void ReadData(a)

ModuleRecord *a;
{
      short b;
      WireRecord *wire;
      for(b=0;b<8;b++)  /* es werden 16 Draehte ausgelesen */
      {
            wire = a->NextWire[b];
            ReadWire(wire,b,a->Counter);
      }
}
```

Damit beim Auswerten die hier ausgelesen Daten die gewohnte Form haben (Daten beginnen bei RAM 0, enden bei RAM 255), wird "i" zuerst als "256-Counter" gesetzt. Der Index "i" läuft nun als erstes von "256-Counter+1" bis 255 (ist 8 Bit lang), wird dann zu 0 und das Auslesen wird mit "i"="256-Counter" gestoppt. Ausgelesen werden jedesmal 16 Bit, auf die der "Pointer" "adr" zeigt. "adr" wird gleich anschliessend mit dem Befehl "adr++" erhöht, und zwar um vier 8-Bit-Adressen (= zeigt auf übernächsten Speicherplatz vom Typ word), da "adr" als vom Typ "Pointer" auf "lword" (32-Bit) deklariert ist. Das hängt mit der "Memory-Map" des FADCs zusammen. Kürzen wir die einzelnen "Bits" der Adresse wie folgt ab: FFFF. 000S. MMMM. CCRR. RRRR. RRBB. Die ersten vier "Bits" können wie erwähnt als Brücken gesetzt werden, und ich habe diesem "Memory-Start" den Wert 4 gegeben. "S" ist ein

"Status-Bit", das normalerweise 0 ist, aber bei speziellem Auslesen wie "ReadCounter" oder "ReadStatus" 1 gesetzt wird. Die "R" nun bestimmen den momentanen Adress-Wert der "RAM" (8 Bit-256 verschiedene Adressen) des jeweiligen Drahtes. Will man nun diesen Adresswert um eins erhöhen, so muss man die gesamte Adresse um vier erhöhen.
Wir lesen also jedes Mal 16 Bit zusammen aus, und zwar mit BB=00 oder 10. Damit lesen wir immer die Y- und X-Werte (links und rechts ausgelesen) eines jeweiligen Drahtes aus, da BB folgenden Einfluss auf die Adresse hat [28]:

BB=	00:	Channel	Y1, Y3, Y5, Y7
BB=	10:	Channel	Y0, Y2, Y4, Y6
BB=	01:	Channel	X1, X3, X5, X7
BB=	11:	Channel	X0, X2, X4, X6

Die übrigbleibende Bestimmung der Adresse wird nun noch durch CC vorgenommen, gilt doch:

CC=	00:	Channel	0,1	CC=	10:	Channel	4,5
CC=	01:	Channel	2,3	CC=	11:	Channel	6,7

Mit der Ausleseanweisung in untenstehendem Programm wird obiges zusammengefasst, von je einem Draht die Werte links und rechts zusammen ausgelesen und in einer 16-Bit-Variablen abgespeichert. Das passiert mit allen 256 möglichen "RAM"-Adressen dieser zwei Werte:

```
void ReadWire(c,b,Counter)
WireRecord  *c;
short b,Counter;
{
 register lword *adr;
 register byte i,end;

 i=end=256-Counter;      /*  Es  werden  256  Werte  pro  Draht  in
Schieberegister gespeichert*/
 adr=(lword*)((0xB0000000)|(0x20<<17)|((b/2)<<10)|((b%2)?0:2));[29]
 MMU32;
 do
  {
    c->Daten[++i]=*(word*)(adr++);/*wird adr vom Typ lword erhoeht,so
wird zur Adresse 4 */
      /*dazugezaehlt*/
  }while (i!=end);
 MMU24;
}
```

[28] Siehe [25]

[29] (b%2)?0:2 ergibt 0, wenn b%2 ungleich 0, ergibt 2 wenn b%2=0

III.6 Die Vieldraht-Proportionalkammer

Die von mir verwendete Vieldraht-Proportionalkammer[30] wurde von P.Eschle gebaut und ist ausführlich in seiner Diplomarbeit[31] beschrieben. Ich beschränke mich darum auf das Wesentlichste.
Ursprünglich bestand die Kammer aus vier Ebenen mit je einer Potentialebene (Aluminiumfolie) und einem Rahmen mit 160 parallel zueinander gespannten Signaldrähten, die 1 mm auseinander liegen. Im gasgefüllten Raum zwischen den Potentialebenen driften die von ionisierten Teilchen freigesetzten Elektronen analog Kap II zu den Signaldrähten. Wiederum ist die durch Gasverstärkung auftretende Elektronenlawine als Stromimpuls messbar.
Für unsere Zwecke mussten wir die Kammer in zwei unabhängige Hälften teilen. Jede dieser zwei neuen Kammern besteht jetzt also aus zwei Ebenen mit je 160 Signaldrähten (Abbildung III.11):

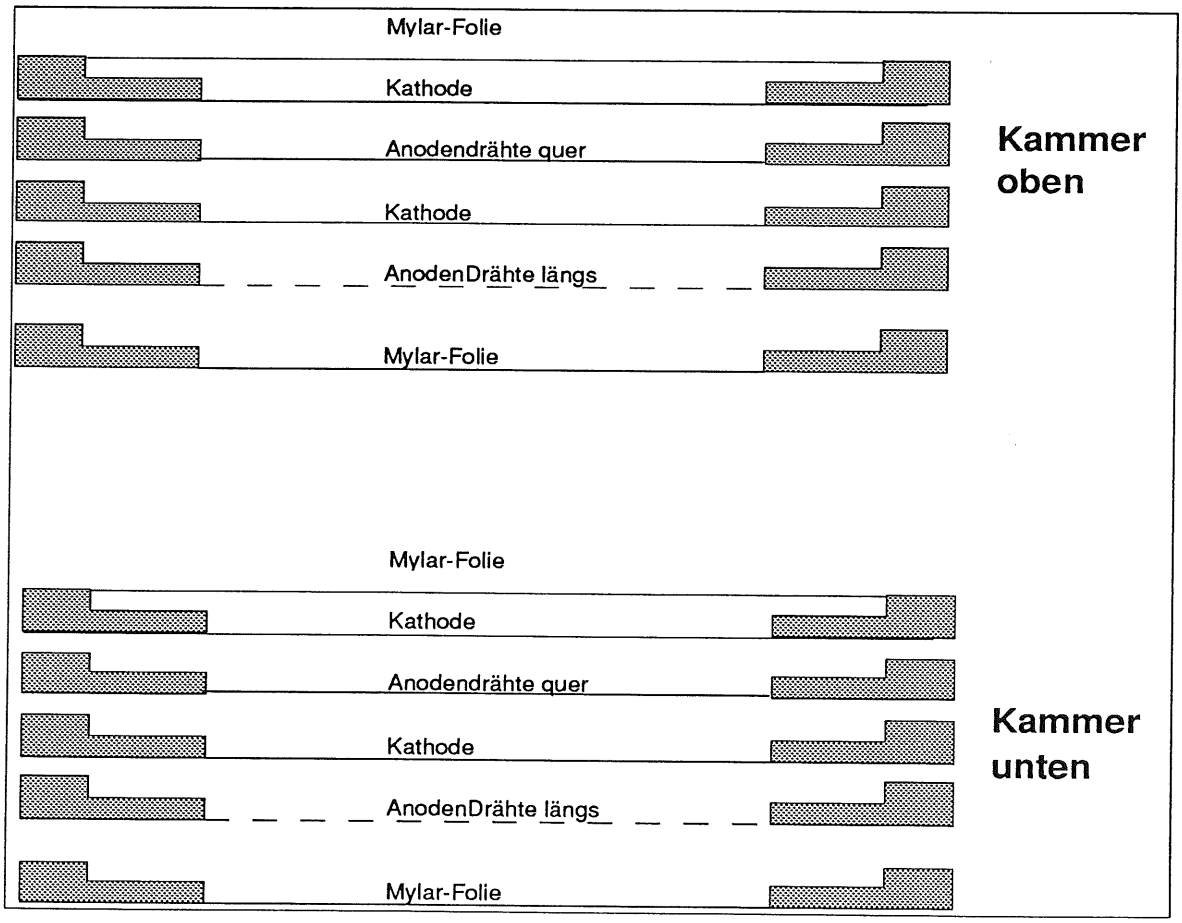


Abbildung III.11 Querschnitt durch Proportional-Kammer

[30] Multi Wire Proportional Chamber
[31] Siehe [10]

Die Signaldrähte sind auf 43-polige Flachbandstecker herausgeführt. Die Potentialebenen liegen auf negativer Hochspannung (3600 V) und die Signaldrähte auf Erdpotential. Das Feldlinienbild am Rande der Kammer hat die Form von Abbildung III.12:

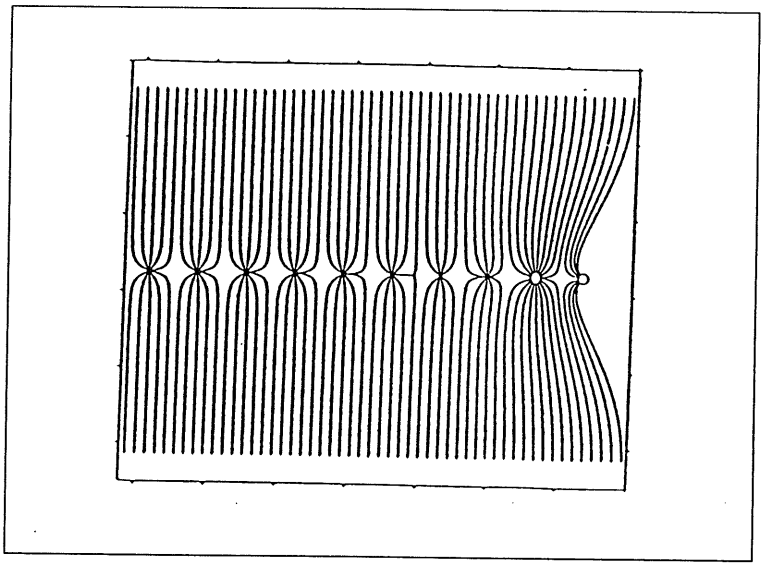


Abbildung III.12 Feldlinienverlauf der Kammer

Die Kammer ist während längerer Zeit betrieben worden und liefert einwandfrei die geforderte Ortsauflösung von 1mm.

III.7 PCOSII, BIFT-Karte und VME-Interface

Die Spannungsimpulse auf den Signaldrähten in der Kammer müssen verstärkt und gespeichert werden, bis der Mac sie ausliest. Es werden dazu PCOSII-Vorverstärker von LeCroy benützt.
Der Mac liest die Vorverstärker nicht selber aus, sondern überlässt diese zeitkritische Arbeit einer Ausleseelektronik.
Das selber gebaute PCOSII-Interface ist kein selbständiges VMEbus-Modul, sondern benötigt eine BIFT-Karte als Pufferspeicher. Auch wird es über die in III.4 beschrieben I/O-Karte kontrolliert (nicht über den VMEbus).

Der Aufbau ist in Abbildung III.13 abgebildet:

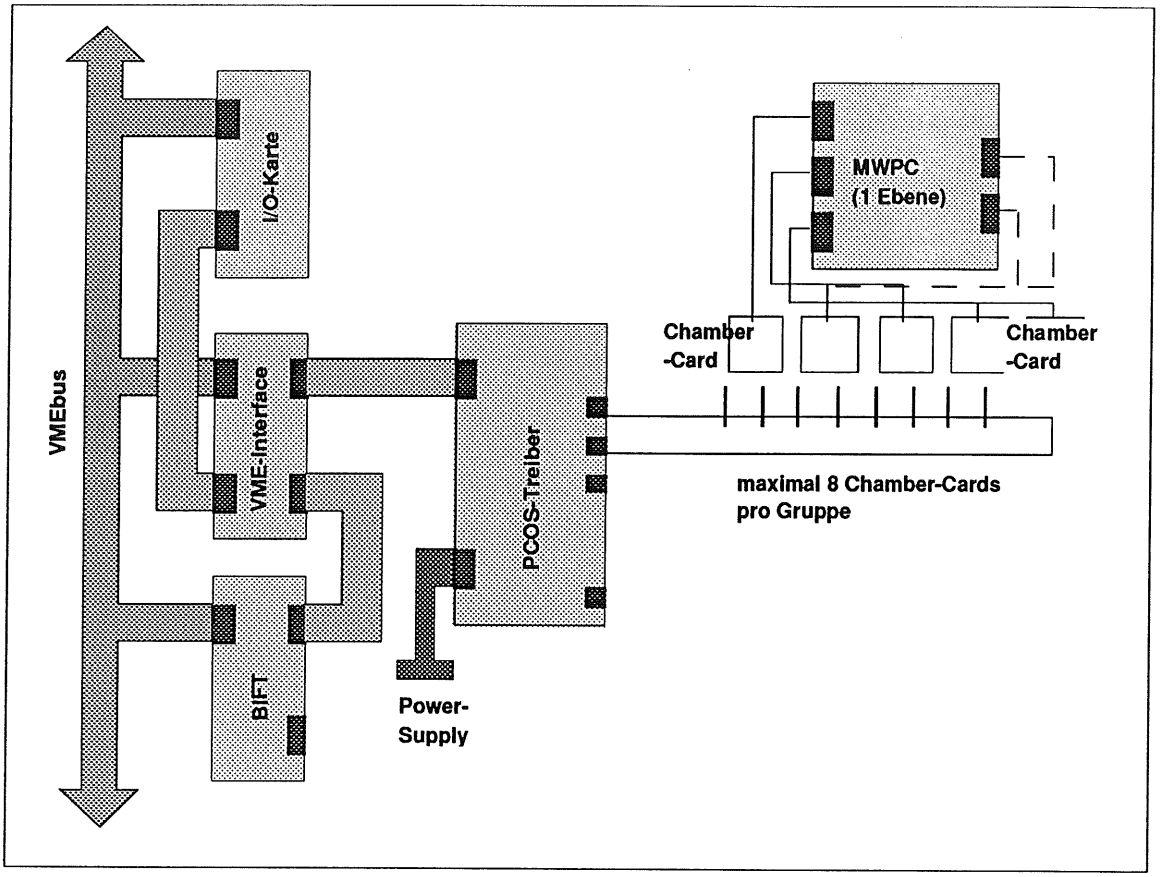


Abbildung III.13 Aufbau des Proportional-Kammer Auslesens

Für das Prinzip der Ausleseelektronik verweise ich auf die Arbeit von P. Eschle[32]. In der Bift-Karte werden für jedes Ereignis und jede Ebene die Zahlen der jeweils gezündeten Drähte gespeichert, die dann vom Mac via VMEbus ausgelesen werden können.

[32] Siehe [10]

III.8 Schaltung der Elektronik für Normalbetrieb

Da wir beim Betrieb unserer Versuchsanordnung vor allem mit Teilchen aus der Höhenstrahlung arbeiten (Rate 1 Teilchen/s), ist die elektronische Schaltung nicht sehr zeitkritisch, und es mussten darum nicht lange Optimierungsüberlegungen angestellt werden.
Die prinzipielle Schaltung ist in Abbildung III.14 dargestellt:

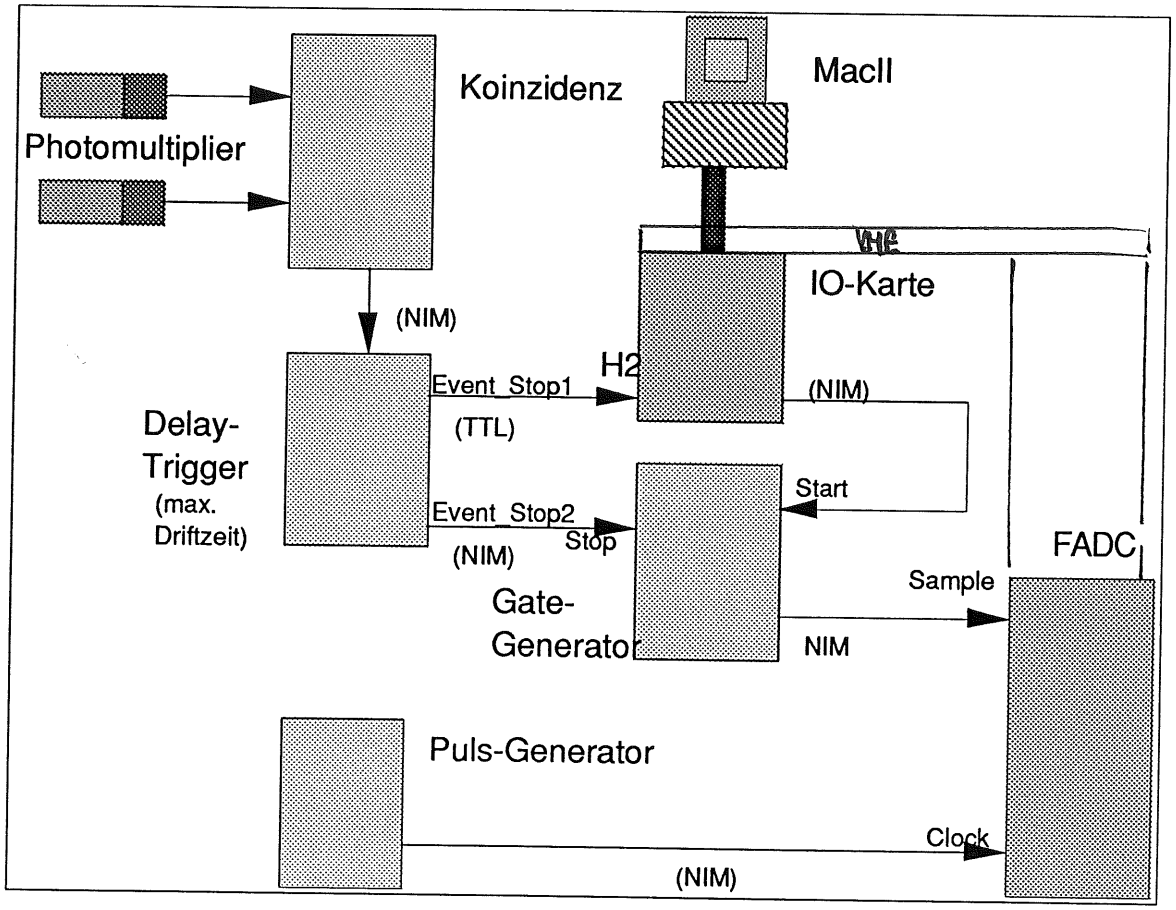


Abbildung III.14 Schaltung der Elektronik für Normalbetrieb

Durchquert ein Teilchen die Versuchsanordnung, so wird es durch die Signale der Szintillatoren registriert, wobei die Pulse der Photoröhren nach einem Diskriminator in Koinzidenz geschaltet werden. Das entsprechende Signal wird um 1.8 µs verzögert, um maximale Driftzeiten der Elektronen zu den Anodendrähten zu berücksichtigen. Diese Verzögerung muss kleiner sein als die dem maximalen Speicherplatz des FADCs entsprechende Zeit von 2.56 µs. Nach diesen 1.8 µs muss verhindert werden, dass die relevanten Daten im FADC überschrieben werden. Ein erster Puls (Event_Stop2) führt darum zum Starteingang eines "Gate-Generators". Das Kabel zum "Sample" Eingang des FADCs ist mit dem "Anti"-Eingang dieses Generators verbunden. Im Normalzustand ist dieser "Anti"-Eingang aktiv, der FADC nimmt Daten auf. Wird der "Gate-Generator" aber mit "EVENT_Stop2" gestartet, so

wird die Datenaufnahme verhindert, bis ein Stopsignal eintrifft. Dieses Stopsignal wird vom MacII via der IO-Karte (Bit A0) generiert, wenn er das Auslesen des FADCs beendet hat.
Begonnen wird das Auslesen, wenn das Programm während seiner Hauptschlaufe feststellt, dass in der IO-Karte das "H2-Bit" von "Event_Stop1" gesetzt wurde.
Selbstverständlich muss das Programm dafür sorgen, dass das "H2-Bit" wieder gelöscht wird, bevor A0 losgeschickt wird und der normale Sample-Zyklus wieder beginnt.
In Abbildung III.15 ist der zeitliche Ablauf der Signalfolge nochmals dargestellt:

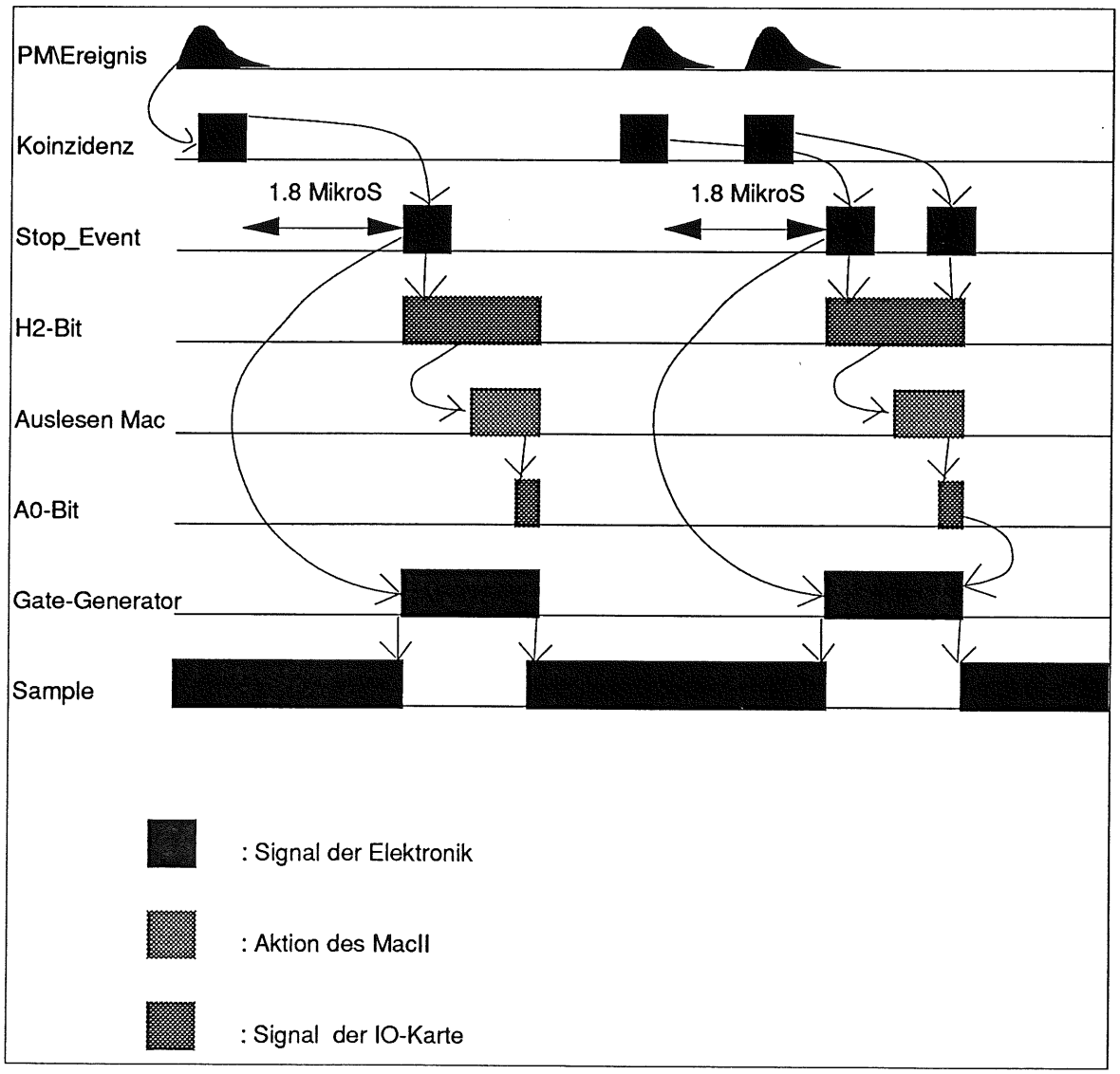


Abbildung III.15 Zeitverhältnisse Normalbetrieb

Mit einer simulierten Ereignisrate von 100 KHz haben wir versucht, die benötigten Zeiten für das Auslesen und Speichern der Daten abzuschätzen. Betreibt man das Ausleseprogramm ohne die ausgelesenen Daten auf der Festplatte abzuspeichern (siehe III.9), so kann man in drei Minuten ca. 4430

"Ereignisse" auslesen. Das bedeutet, dass pro Sekunde maximal 25 Ereignisse ausgelesen werden können, bzw. dass die Elektronik (inkl. Computer) ca. 0.04 Sekunden für das Auslesen eines "Ereignisses" benötigt. Speichert man die Daten auf der Festplatte in "Files" zu fünf Ereignissen (siehe III.9) , so kann in drei Minuten noch 680 Mal ausgelesen werden, was einer Auslesera-te von 3.8 Ereignissen/Sekunde oder 0.26 Sekunden/Ereigniss entspricht. Für das alleinige Speichern eines einzigen Ereignisses auf die Festplatte ergibt sich so die Zeitdauer von ca. 0.22 Sekunden.

III.9 Programm für den Normalbetrieb

Das ganze Programm ist in eine übliche Mac-Umgebung (siehe III.2.i.b) eingebettet. Neben den Menus, die unbedingt nötig sind für das Auslesen und Darstellen der FADC-Daten, ist ein kleiner "Text-Editor" programmiert, der die Möglichkeit gibt, Files zu beschreiben und zu speichern.
Alles Wesentliche ist aber im Menu "FADC" und "DatenAufbereiten" abrufbar.
Die Struktur der Daten, die man abspeichert, ist so aufgebaut, dass in einem File fünf Ereignisse untergebracht sind. Diese Zahl kann man verändern, indem man die Konstante "Anz_Daten_in_File" entsprechend anpasst. Nach oben ist die Anzahl allerdings begrenzt, da die Kapazität der "RAM" nicht mehr als 20 Ereignisse verträgt.
Für die einzelnen Ereignisse haben die 8 Drähte folgende Form:

NWORD	IFLAGS	IDFADC	ISTART	Daten[0]	Daten[1]	.....	.....	Dater

NWORD:	Ist eine Angabe (unsigned short) über die Grösse dieses Daten-Arrays in Einheiten von 16 Bit, in unserem Fall also 260
IFLAGS:	wird 0 gesetzt
IDFADC:	besteht aus vier Hexa-Ziffern: 0x ABCD A:= Crate-Nummer, bei uns 0 B:= Modul-Nummer, bei uns 0 C:= 0 D:= Wire-Nummer, von 0-7
ISTART:	0
Daten[i]:	Messdaten aus FADC, 16 Bit lang, die ersten 8 Bit entsprechen Draht rechts ausgelesen, die zweiten 8 Bit links ausgelesen.

Diese Datenform entspricht dem offiziellen Format, das bei HERA vorgeschrieben wird. Zusammenfassend kann man also feststellen, dass in einem File:

(Anz_Daten_in_File*8*260) unsigned short = (5*8*260*2) Byte=**20.8** KByte

gespeichert werden.
Das Flussdiagramme (Abbildung III.16) gibt einen Überblick über das Programm,von dem einige Abschnitte schon bekannt sind und dessen wichtigste Funktionen im folgenden kurz beschrieben werden.

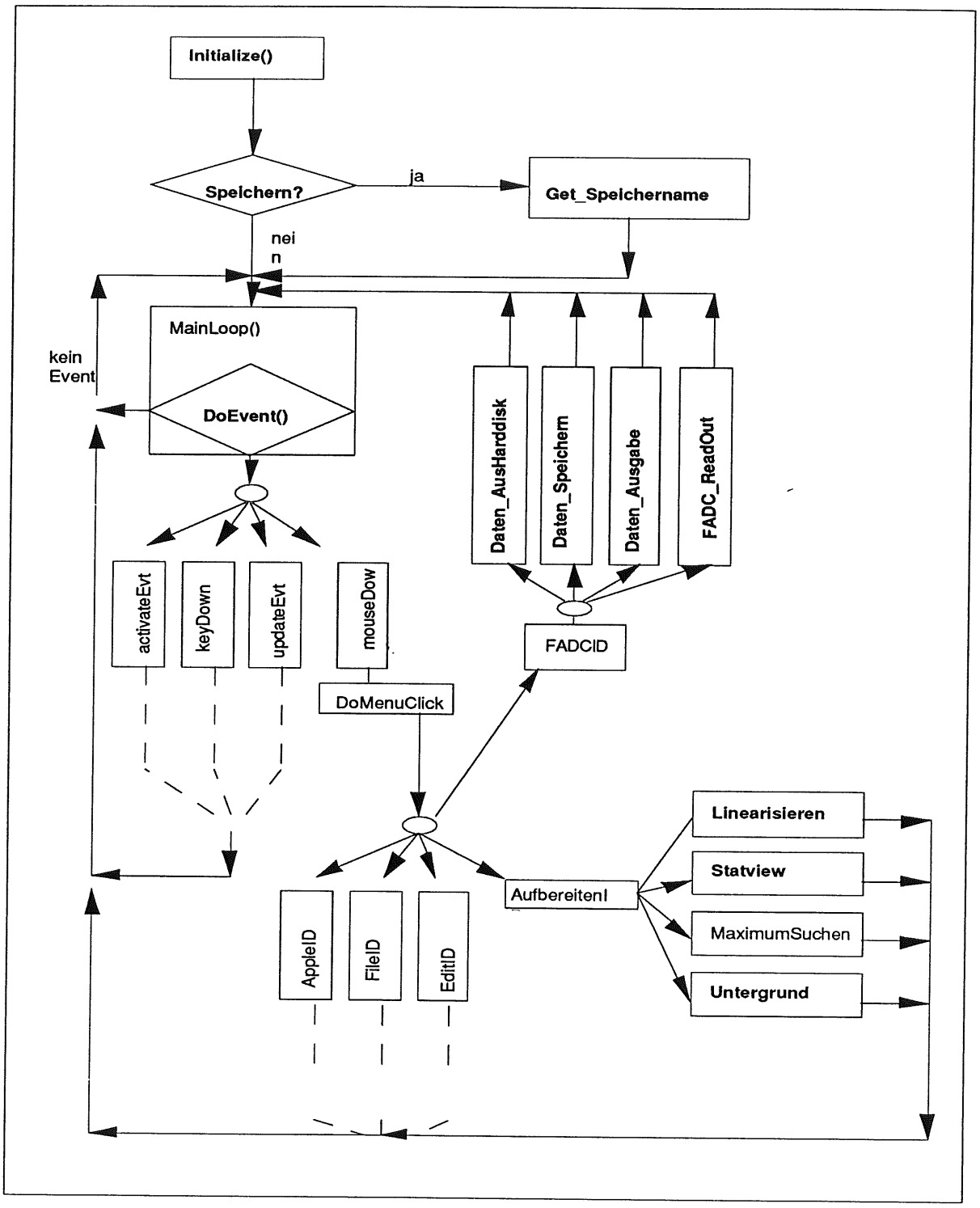


Abbildung III.16 Flussdiagramm von FADC_ReadOut;

FADC-ReadOut Setzt den Betrieb und das Auslesen der FADC-Karte in Gang, bis die Stop-Taste mit der Maus gedrückt wird. Hat man vorher gewünscht, die Daten auf der Harddisk zu speichern, wird das nun nach "Anz_Daten_in_File"-Ereignissen ausgeführt.

Daten_Ausgabe	Mit dieser Funktion können die Daten auf dem Bildschirm dargestellt werden, je nach Wunsch grafisch oder in Zahlen. Man kann bei beiden Varianten wählen, welches Ereignis (zwischen 0 und Anz_Daten_in_File) und welche Drahtnummer (links oder rechts) man anschauen möchte. Bei der grafischen Option kann man auch die Variante wählen, dass alle vier Drähte eines Ringes auf dem Bildschirm erscheinen.
Daten_AusHarddisk	So kann man bereits abgespeicherte Files wieder in die RAM einlesen und sie von neuem bearbeiten.
Daten_Speichern	Ruft man diese Funktion auf, so erscheint auf dem Bildschirm ein Fenster mit der Frage, ob man die anfallenden Daten auf die Harddisk speichern will. Drückt man die OK-Taste, erscheint eine Dialog-Box, mit der man den gewünschten File-und Verzeichnis-Namen für die Daten eingeben kann.
Get_Speichername	Diese Funktion wird selbständig am Anfang des Programmes aufgerufen und macht das gleiche wie Daten_Speichern.
Linearisieren	Damit werden alle Daten, die sich momentan im RAM befinden, linearisiert. So kann angeschaut werden, welche Form die anfallenden Daten wirklich haben und wie sie auch später verarbeitet werden.[33]
Statview	Diese Funktion verändert die gewünschten Daten in eine Form, die von dem Statistikprogramm "Statview" verarbeitet werden kann. Der Name des Files, in das diese aufbereiteten Daten geschrieben werden, kann neu gewählt werden.
Untergrund	Diese Funktion führt einen Algorithmus zur Unterdrückung des Rauschens mit allen sich in der "RAM" befindlichen Daten durch.[34]

Der Quelltext des Programmes ist unter dem Namen FADC_ReadOut.c und FADC_ReadOut.h gespeichert. Diesen Files sind ausführliche Kommentare beigefügt, so dass es nicht schwierig sein sollte, das Programm zu verstehen.

[33] Siehe IV.1
[34] Siehe IV.2

III.10 Schaltung der Elektronik für Absoluteichung

Die Ausleseelektronik der Proportional-Kammer kennt eine externe "Trigger"-Schaltung, die das Auslesen der Vorverstärker ohne Eingriffe eines Computers starten soll. Zum Sperren des "Triggers" dient die Leitung "Mastergate", die über die Eingänge "MG-SET" und "MG-RESET" vom "Trigger" her gesetzt werden kann. Vom Computer her kann der Ausgang "Mastergate" entweder ganz gesperrt oder, wenn er vom "Trigger" zurückgesetzt wurde, wieder aktiviert werden.Um das Auslesen der Proportional-Kammer zu starten, ist damit folgende Sequenz notwendig: Der Computer setzt das "Mastergate" und gibt damit den "Trigger" frei. Wird der Durchgang eines Teilchens durch die Photoröhre registriert, sperrt sich der "Trigger" (MG-RESET) und schickt ein "Strobe"-Signal an die Vorverstärkerkarten. Damit bewahren die Karten den momentanen Zustand. Hat auf mindestens einer Karte ein Draht angesprochen, so werden mit "LOAD" die Schieberegister geladen und mit "START-READOUT" das Auslesen in die BIFT-Karte gestartet. Nach dem Auslesen der BIFT-Karte durch den Mac werden die Vorverstärker durch ihn gelöscht und das "Mastergate" wieder aktiviert. Kurz zusammengefasst kann man die "Prop-Elektronik" wie in Abbildung III.17 darstellen :

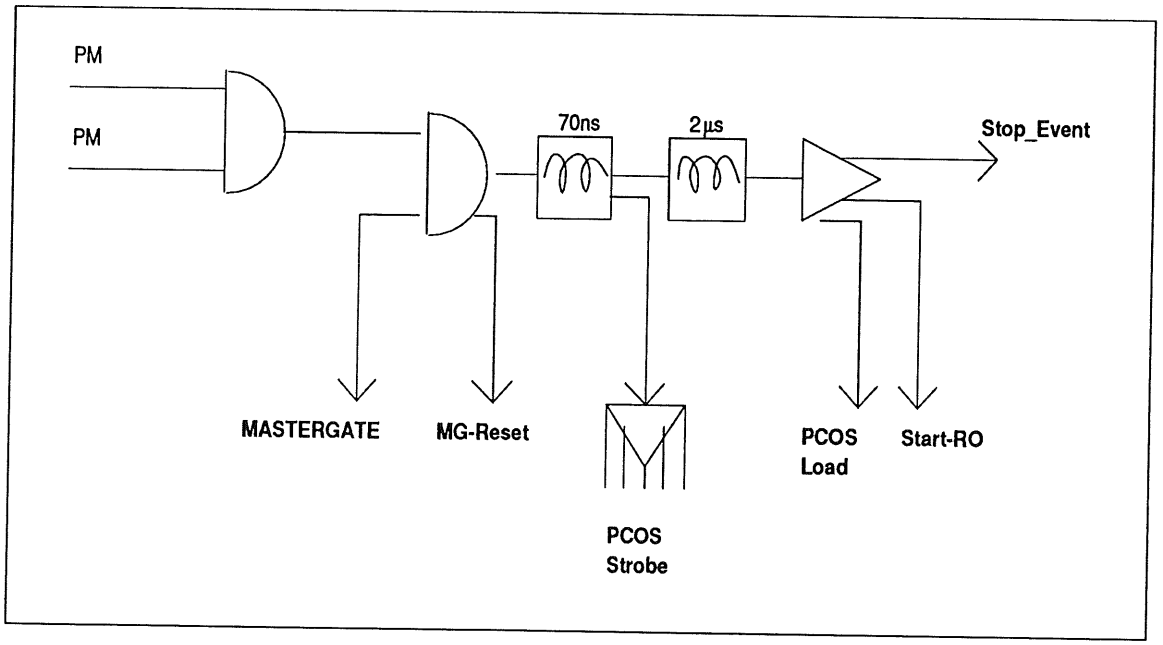


Abbildung III.17 Schaltung Absoluteichung

Setzt man obigen Ablauf mit der Elektronik der FADC-Karte zusammen, so sieht das wie folgt aus: (Abbildung III.18)

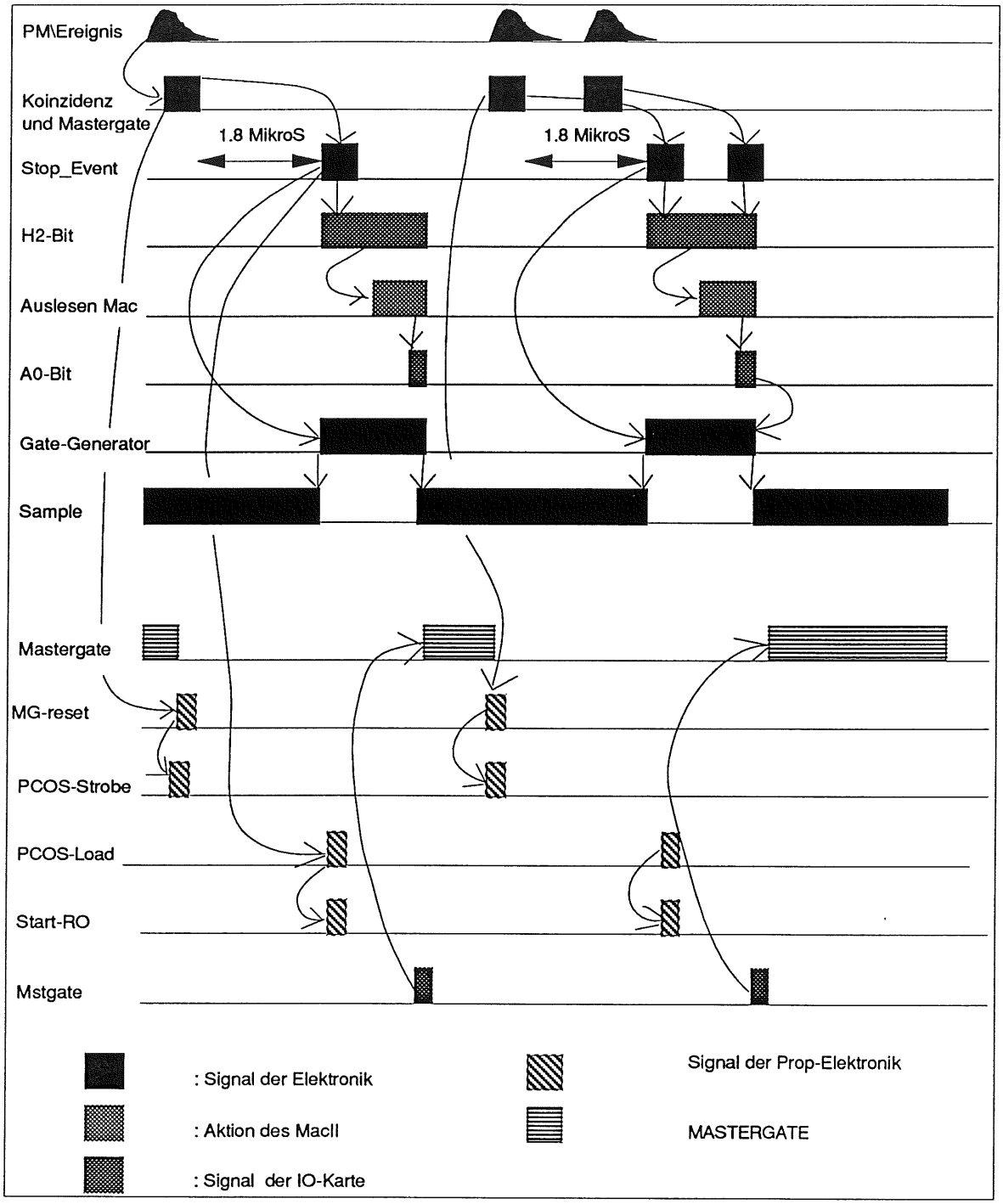


Abbildung III.18 Zeitverhältnisse Elektronik für Absoluteichung

Auch bei dieser Schaltung wurde wiederum versucht, mit einer simulierten Ereignisrate von 100 KHz die benötigten Zeiten für das Auslesen und Speichern der Daten abzuschätzen. Da nun auch die Daten der Proportionalkammer verarbeitet werden müssen, werden die benötigten Zeiten im Vergleich zum Normalbetrieb grösser.
Betreibt man das Ausleseprogramm ohne die ausgelesenen Daten auf der Festplatte abzuspeichern (siehe III.12), so kann man in drei Minuten ca. 1865 "Ereignisse" auslesen. Das bedeutet, dass pro Sekunde maximal 10 Ereignisse

ausgelesen werden können, bzw. dass die Elektronik (inkl. Computer) ca. 0.10 Sekunden für das Auslesen eines "Ereignisses" benötigt. Speichert man die Daten auf der Festplatte in "Files" zu fünf Ereignissen (siehe III.12) , so kann in drei Minuten noch 431 Mal ausgelesen werden, was einer Ausleserate von 2.4 Ereignissen/Sekunde oder 0.41 Sekunden/Ereigniss entspricht. Für eine Speicherung der Daten eines einzigen Ereignisses auf die Festplatte ergibt sich so die Zeitdauer von ca. 0.3 Sekunden.

III.11 Auslesen der BIFT-Karte

Der Programmausschnitt, der das Auslesen der Bift-Karte vornimmt, ist nachfolgend abgebildet:

```
void  ReadMWC(MWC_Daten MWC)

      /* liest Daten von MWC und speichert sie in RAM */
{
      register byte i,a,w,x,y,z;
      word  NrEvents=0;
      word  Daten=0;
      byte  DrahtNr;
      byte  Ebene;
      word  *bift =(word *) BIFT_BASE;

      bift_intern();
      MMU32;
      NrEvents=bift[BIFT_BAC];
      MMU24;
      if (NrEvents==0)
            {
                  Puls_IO_A(Bift,Clear);        /* Clear pulsen */
                  bift_reset();                 /* loescht Bift-Karte*/
                  bift_extern();                /* erlaubt Einlesen von
                                                Daten */
                  return;
            }
      else
            {
                  MMU32;
                  bift[BIFT_BAC]=0;
                  MMU24;
                  for (i=1;i<=NrEvents;i++)
                        {
                              MMU32;
                              Daten=bift[BIFT_BUF];
                              MMU24;
                              Ebene=(Daten&0x00FF);
                              DrahtNr=((Daten&0xFF00)>>8);
                              MWC->Ebene[Ebene-1]->DrahtNr[a]=DrahtNr;
                        }

                  Puls_IO_A(Bift,Clear);        /* Clear pulsen */
                  bift_reset();                 /* loescht Bift-Karte*/
                  bift_extern();                /* erlaubt einlesen von
                                                Daten */
            }
}
```

Die Funktion "bift_reset()" löscht die Bift-Karte, "bift_intern()" und "bift_extern()" verbieten und erlauben das Einlesen von Daten, sie sind im Anhang ausgedruckt.
Ebenfalls sind die verschiedenen Register der Bift-Karte im Anhang VI.8 ausgeschrieben.
"Bift" ist ein Pointer auf "word" (8 Bit), es werden darum immer 8 Bits ausgelesen. Die Basisadresse "BIFT_BASE" ist auf der Karte manuel einstellbar, ich habe 0xB0FF1100 eingestellt.

Am Anfang der obigen Prozedur muss die Funktion "bift_intern()" aufgerufen werden, damit die Bift-Karte überhaupt ausgelesen werden kann. Anschliessend wird aus dem Register "bift[BIFT_BAC]" die Anzahl der gespeicherten Daten ausgelesen. Ist diese Anzahl gleich 0, ist etwas schief gelaufen und die Funktion wird verlassen, nachdem die Karte mit "bift_reset()" gelöscht und mit "bift_extern()" wieder auf Einlesen umgestellt wurde. Sind aber Daten vorhanden, so werden sie nacheinander mit dem Befehl "Daten=bift[BIFT_BUF]" ausgelesen. Die ersten zwei Bits der Variablen "Daten" beinhaltet nun die Drahtnummer, die letzten beiden Bits die dazugehörige Ebene, derjenigen Drähte, welche in der Proportionalkammer angesprochen haben. Nach Beendigung des Auslesens werden wiederum die Bift-Karte mit "bift_reset()" gelöscht und mittels "bift_extern()" auf Einlesen umgeschaltet. Mit der Anweisung "Puls_IO_A(Bift,Clear)" wird ein Signal durch die I/O-Karte auf die Prop-Elektronik gegeben, welche die Vorverstärker löscht. Das Setzen des "Mastergates", das natürlich für den weiteren Betrieb unumgänglich ist, wird erst nach dem Senden von "A0", also nachdem der Mac sämtliche Arbeiten ausgeführt hat, durchgeführt.

III.12 Programm für die Absoluteichung

Obwohl einige Änderungen durch das zusätzliche Auslesen der Bift-Karte auftreten, bleibt die Grundstruktur von "FADC_ReadOut" erhalten. Es wird nun einfach zusätzlich noch die Funktion "ReadMWC" aufgerufen, die im vorherigen Abschnitt besprochen wurde.

```
void Auslesen(void)

{

      IoAdress = (byte *) IoBasAdress;
      ReadCounter(currentModule);          /* Der Wert des Counters wird
                                           ein gelesen*/
      ReadData(currentModule);             /* Die Daten des FADCs werden
                                           eingelesen */
      ReadMWC(current_MWC);                /* Die Daten der BIFT-Karte
                                           werden eingelesen */
      H2S_loeschen();                      /* Nach dem Auslesen muss
                                           Status Bit geloescht werden*/
      MMU32;
      IoAdress[PADR2] = 0;                 /* Ausgeben eines Pulses auf
                                           Pin1 der I/O-Karte Stecker2
                                           (A0)*/
      IoAdress[PADR2] = 0x03;              /* lösche auch Vorverstarker
                                           nocheinmal */
      IoAdress[PADR2] = 0x08;              /* Wichtig, dass Mastergate
                                           =1 bleibt */
      MMU24;
}
```

Mit den Anweisungen "IoAdress[PADR2]" über die I/O-Karte wird "A0" gepulst und das "Mastergate" gesetzt.
Lässt man das Programm laufen, taucht ein neues Menu auf: "MWC". Mit ihm kann man die ausgelesenen Daten der Bift-Karte anschauen. Sonst bleibt alles beim Alten.

Lässt man das Programm die Daten auf der Festplatte speichern, so wird neben den bekannten Datenfiles der FADC-Daten ein File mit dem gleichen Namen und der Endung (Extension) "_MWC" mit den Proportional-Kammer-Daten kreiert.
Um die Übersichtlichkeit zu steigern, wurde der Quelltext nun in 6 Files aufgeteilt:

MWC.c und MWC.h	Beinhaltet vor allem das Auslesen der FADC-Karte und die Mac-Umgebung, die Main-Routine befindet sich ebenfalls darin.
Bift.c und Bift.h	Hier sind alle Funktionen, die das Auslesen, Speichern und Steuern der Bift-Karte überwachen, gespeichert.
Updatedata.c und Updatedata.h	Hier sind alle Funktionen, die das verändern der Daten beinhalten, gespeichert.

IV. Datenauswertung

Die eigentlichen Messungen mit der FADC-Karte und dem Prototypen gestalteten sich sehr schwierig. Grosse Probleme hatten wir mit Untergrund-Rauschen, das sämtliche Messungen überlagerte.
Damit unser Analyseprogramm überhaupt einzelne Ereignisse erkennen konnte, mussten wir darum die gemessenen Werte zuerst mit einer speziellen "Anti-Noise"-Funktion aufbereiten. Diesen Ablauf werde ich in IV.2. beschreiben.
In IV.1. sind Messungen bezüglich der Nichtlinearität des FADCs beschrieben und eine diese Tatsache berücksichtigende Funktion wird an die Messdaten angepasst.
In IV.3. wird die eigentliche Messung, deren Ergebnisse das Ziel dieser Arbeit sind, erläutert und dargestellt.

IV.1 Testmessungen mit der FADC-Karte

Um den Messbereich eines FADCs zu vergrössern, kann man ihn "nicht-linear" konzipieren. In diesem Fall wird etwa folgende Kennlinie erwartet (Abbildung IV.1):

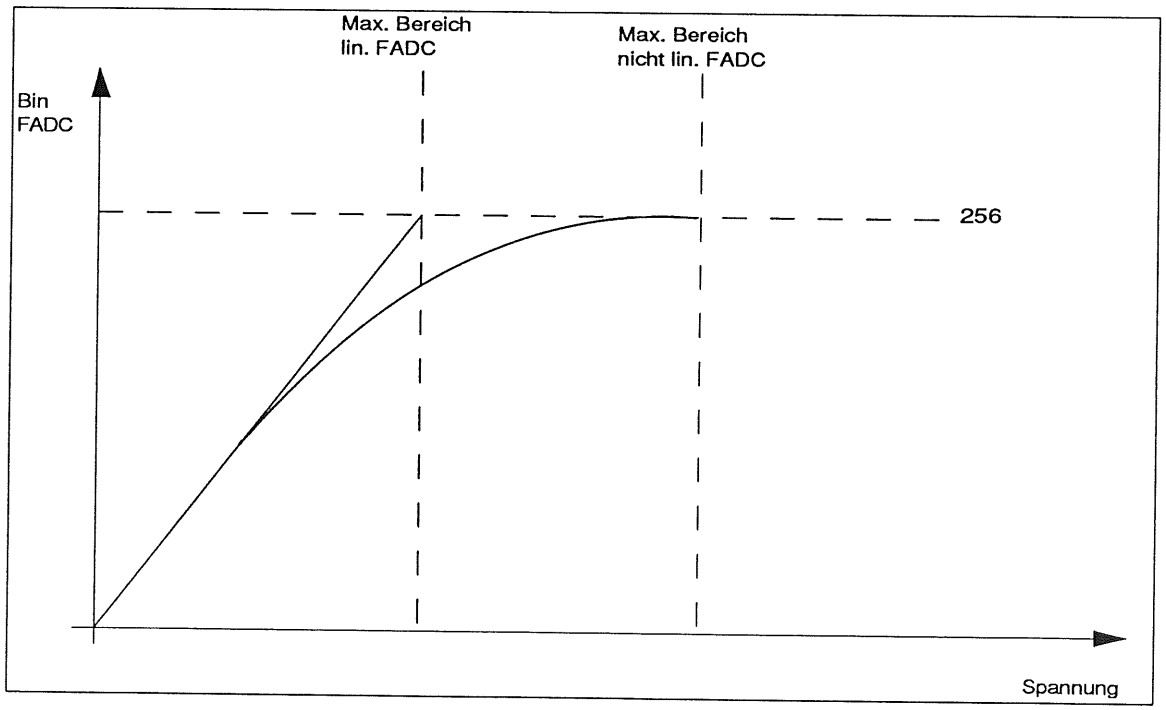


Abbildung IV.1 Kennlinie Nicht-Linearer-FADC

Da man für die Auswertung der Signale der Driftkammer die wahren Spannungswerte braucht, ist es wichtig, die vom FADC gelieferten Daten linearisieren zu können. Man muss also den Verlauf der oben dargestellten Kurve explizit kennen.
Durch Ergebnisse aus Messungen mit früheren Typen der gleichen Baureihe haben wir folgende Funktion angesetzt:

$$i(l) = \frac{a \cdot 256 \cdot l}{2560 + bl} \qquad [1]$$

wobei i(l):= Binhöhe FADC
l:= Spannung
a,b:= Konstanten

Um die Konstanten a und b bestimmen zu können, habe ich zu verschiedenen Zeiten eine Rechteck-Spannung von 30ns Breite in der Höhe (Vo) variiert und dem FADC als Eingangssignal zugeführt. (Abbildung IV.2)

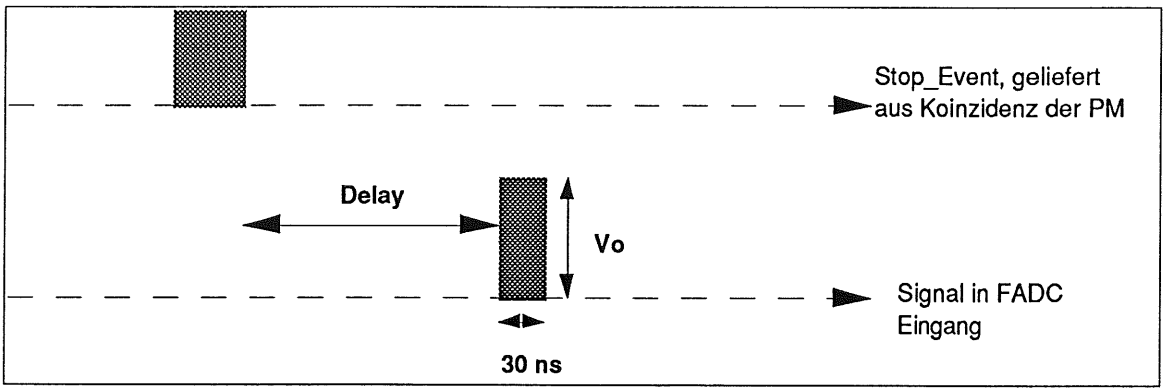


Abbildung IV.2 Schaltung Testsignal

In AbbildungIV.3 sind beispielsweise die resultierenden Daten des

FADCs aufgetragen bei einem Eingangspuls von 1.7 V Höhe und einem "Delay" von 1µs:

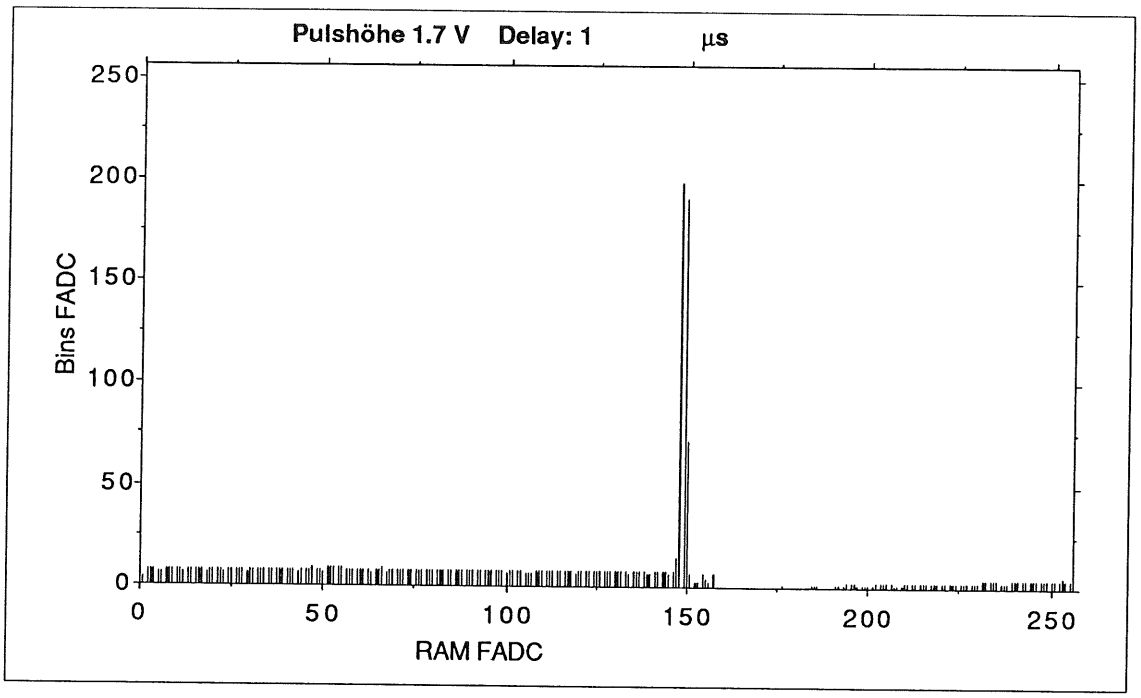


Abbildung IV.3 Auswertung Testsignal

Trägt man nun das vom FADC gelieferte Maximum gegen die Spannung auf, erhält man Messpunkte wie in Abbildung IV.4. In der gleichen Abbildung sieht man ebenfalls die an diese Messpunkte anpepasste Kurve [1]:

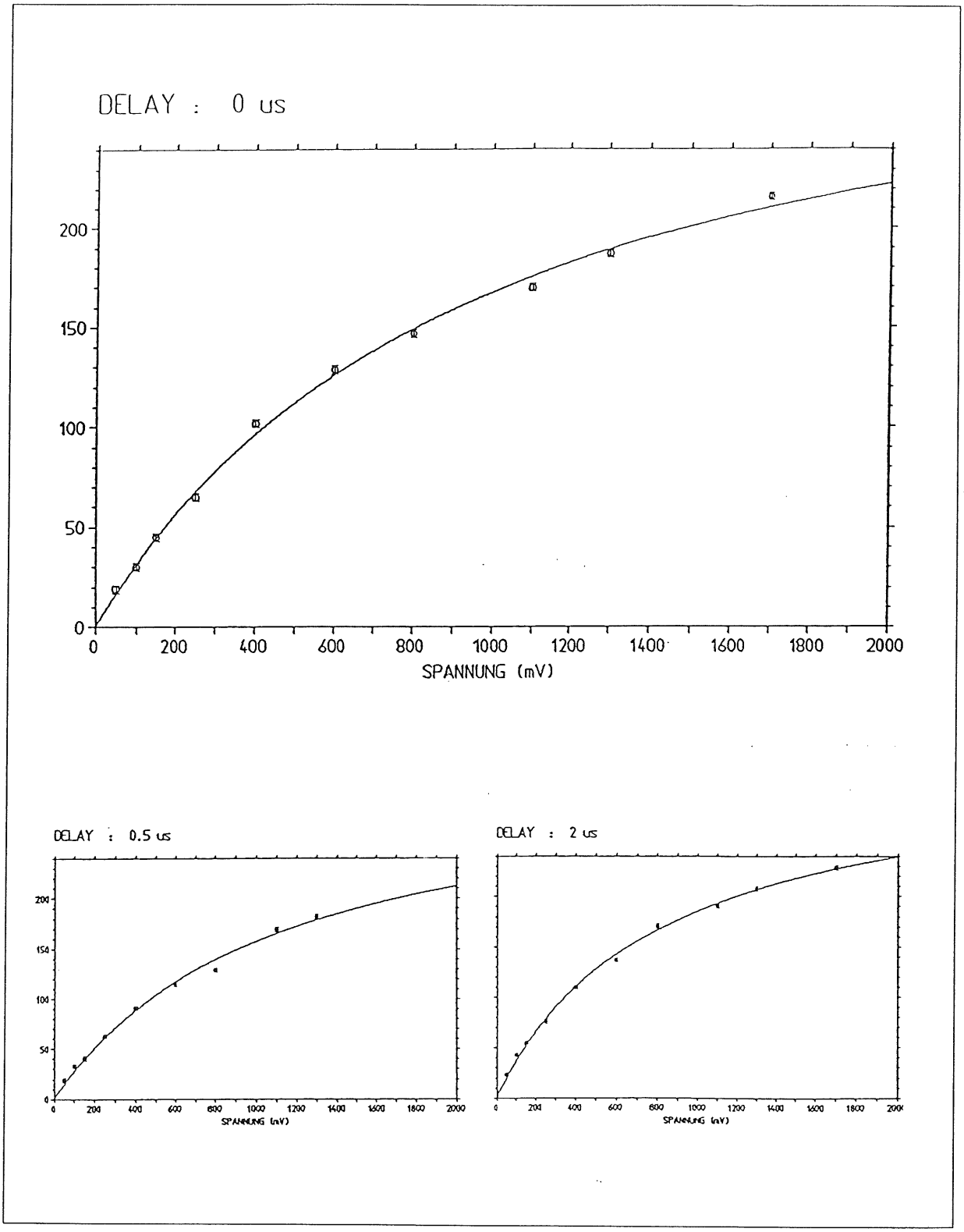


Abbildung IV.4 Angepasste Kennlinie durch Maxima des Testsignales

Folgende Ergebnisse werden so erhalten:

Delay [µs]	a	b
0.0	3.41(7)	2.6(1)
0.5	3.05(7)	2.4(1)
1.0	3.46(7)	2.9(1)
1.5	3.07(7)	2.5(1)
2.0	4.12(7)	3.1(1)

Daraus kann man die mittleren Werte für a und b berechnen:

a= 3.4(2) **b=2.7(1)**

Neben a und b liefert die gleiche Messanordnung mit eingestelltem "Delay" = 0 zusätzlich noch die Kanalnummer für den Nullpunkt der Zeit (d.h. die Durchstosszeit des Teilchens durch die Apparatur [Photoröhren]), sofern man einen kleinen "Offset", der durch die Verzögerung in der Elektronik (Kabellänge, ca 10 ns) und die Reaktionszeit der Photoröhren (ca 30 ns) entsteht, unberücksichtigt lässt. Mit unserer Apparatur erhält man so für den Nullpunkt **Kanal 45** (49-Offset; Offset=4).
Am Rande sei erwähnt, dass Eingangsspannungen der Pulse, die grösser als 2 V sind, also über der vom Hersteller angegebenen maximal möglichen Eingangsspannung liegen, ein Nachschwingen des FADCs hervorrufen (siehe Abbildung IV.5):

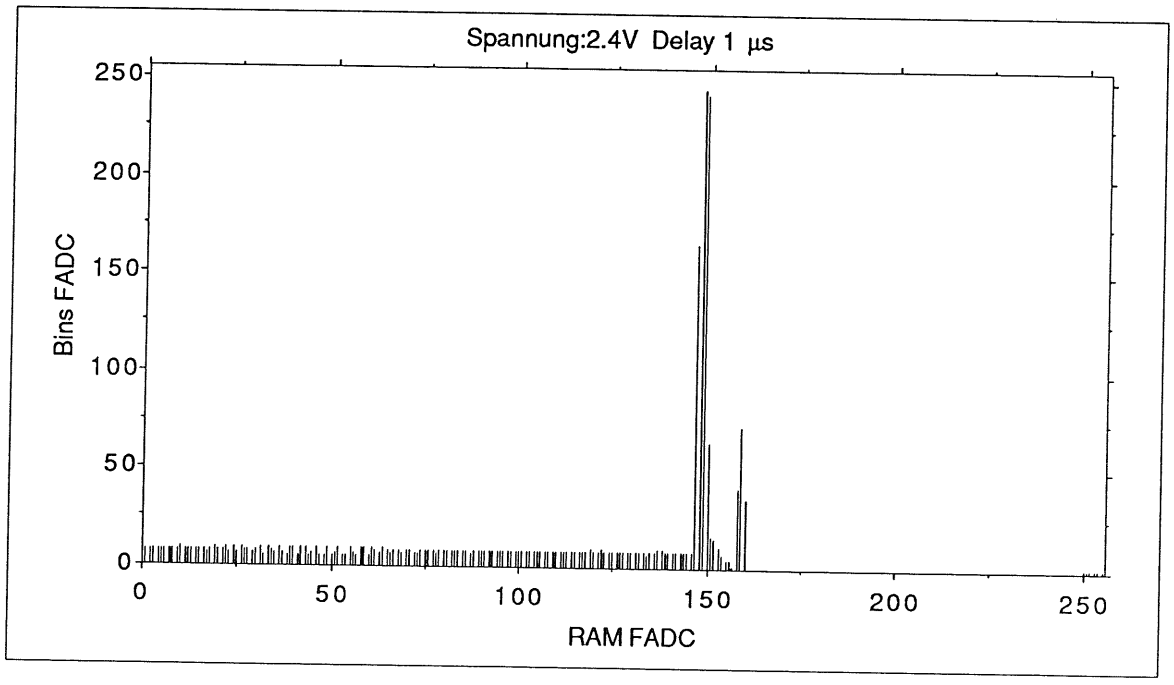


Abbildung IV.5 Überschwingen des FADC's

IV.2 Messungen am Prototypen der Driftkammer

Sämtliche Messungen, die in dieser Arbeit besprochen werden, sind mit Höhenstrahlung durchgeführt worden. Dabei handelt es sich um eine Strahlungsquelle, die bei Wechselwirkungen hochenergetischer kosmischer Teilchen[35] mit Atomen der Erdatmosphäre entsteht. Rund 75% der am Erdboden ankommenden Teilchen sind Myonen. Abbildung IV.6 zeigt das Impulsspektrum dieser Teilchen oberhalb 0.2 GeV/c:

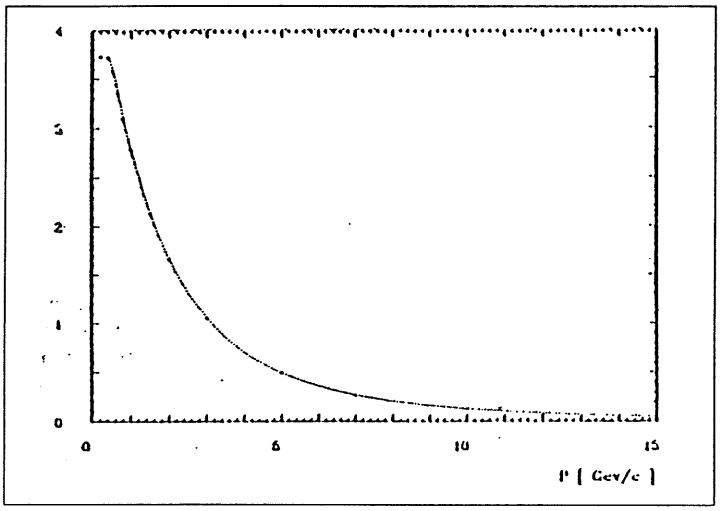


Abbildung IV.6 Impulsspektrum der Höhenstrahlung

IV.2.i Rauschen

Wie bereits erwähnt, ergaben sich bei der Durchführung von Messungen grosse Probleme mit störendem Rauschen des Untergrundes. Nach intensiven und zeitraubenden Bemühungen gelang es uns, dieses Rauschen auf 150 mV zu senken. In Abbildung IV.7 sieht man einerseits typisches Rauschen (mit einem digitalen KO aufgenommen), andererseits Resultate des Auslesens des FADCs ohne auf Ereignisse zu triggern.

[35] 86% Protonen, 12,7% α–Teilchen, 1,3% schwere Kerne

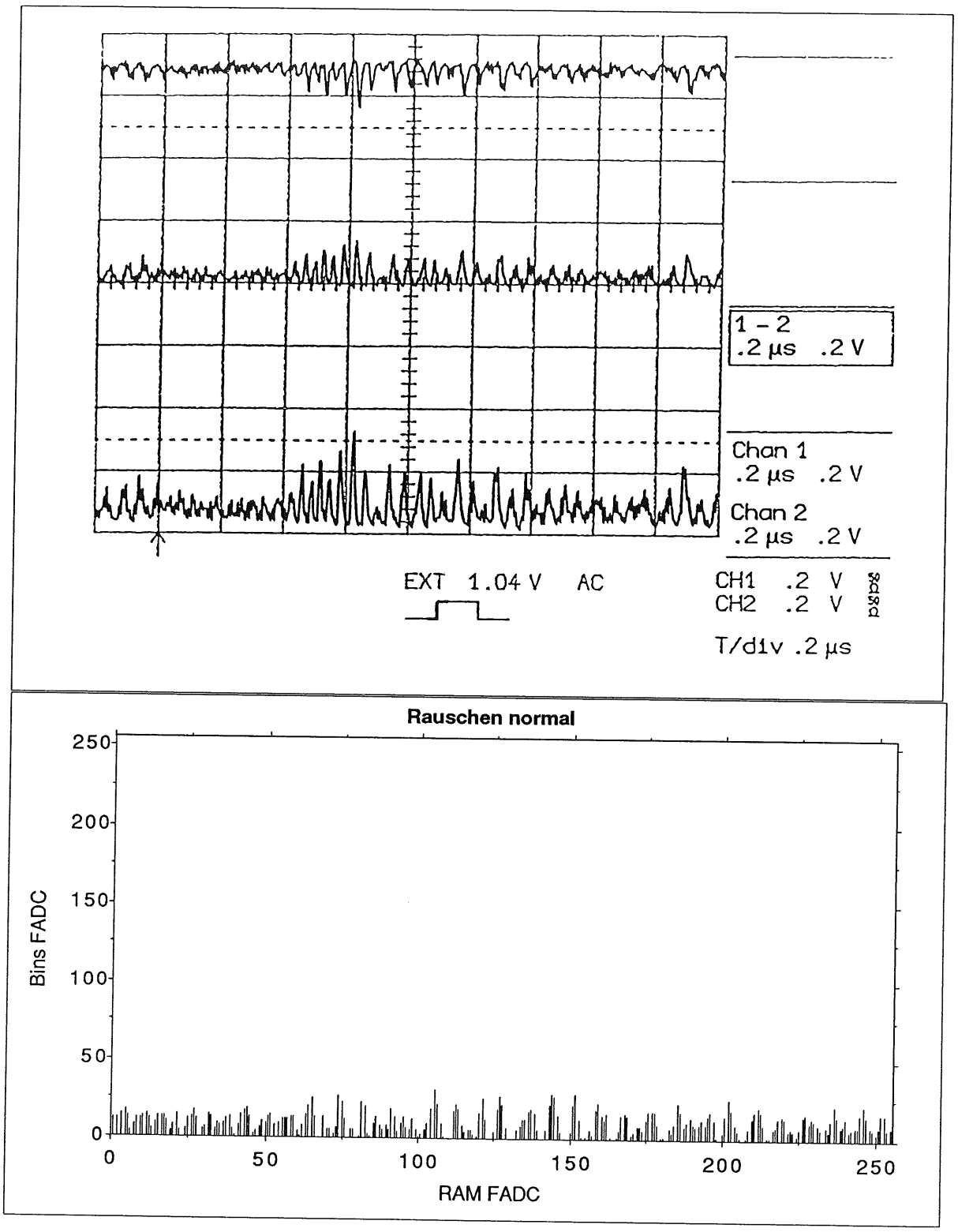


Abbildung IV.7 Durchschnittliches Rauschen

Nicht selten kommt es aber aus unerklärlichen Gründen vor, dass das Rauschen wie in Abbildung IV.8 aussieht:

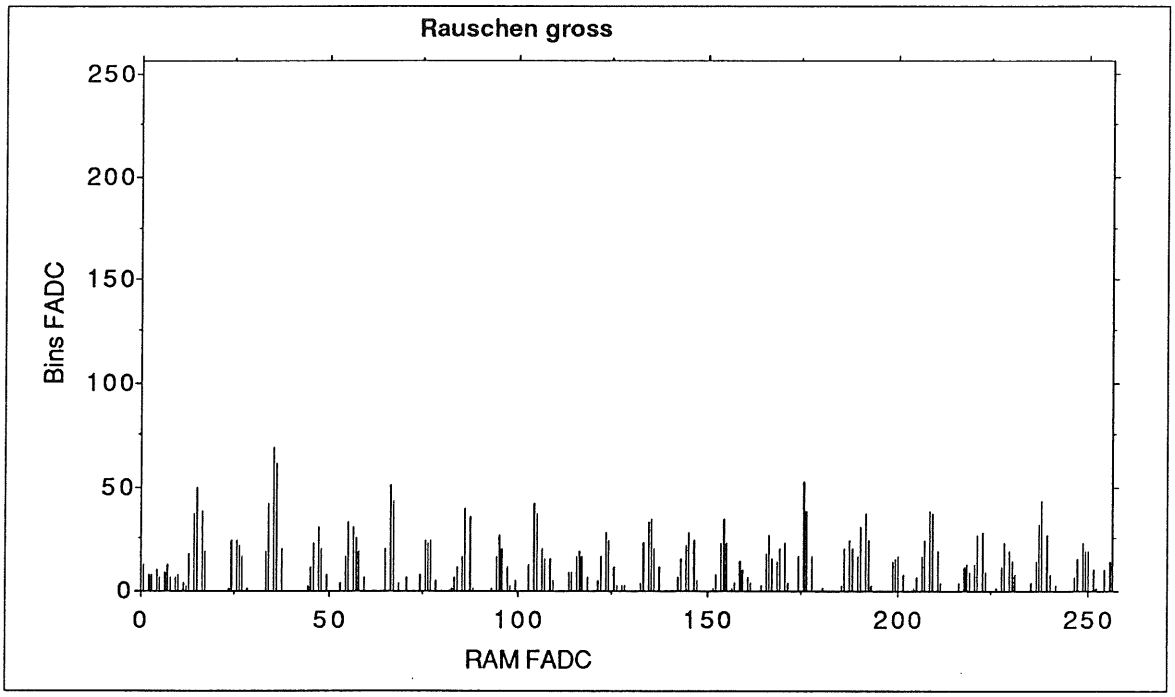


Abbildung IV.8 Grosses Rauschen;

Da alleine schon das Eigenrauschen der Vorverstärker gleich nach der Kammer ca. 50 mV[36] beträgt, dürften wir mit einem Rauschen von 150 mV an der Grenze des Möglichen sein.
Da aber immer noch grosse Probleme bestanden, Signale von Rauschen zu trennen (vor allem softwaremässig), blieb uns nichts anderes übrig, als mit einem Computerprogrammm zu versuchen, das Rauschen zu unterdrücken.
Prinzipiell kann man einen Peak des Untergrundrauschens und einen Peak eines Kammerereignisses mittels der Pulslänge Δt unterscheiden: ein Ereignissignal ist mindestens **40 ns** (vier Kanäle FADC) breit.
Der Algorithmus der Rauschunterdrückungsfunktion besteht demzufolge darin, nachzuschauen, ob der FADC-Wert über einer einstellbaren Schwelle liegt. Wenn nicht, setzt man den FADC-Wert an dieser Stelle gleich dem Schwellenwert. Liegt er aber über der Schwelle so kontrolliert man, wie lange das der Fall ist: Bei einer Zeitdauer grösser als Δt (variierbar) handelt es sich um einen echten Puls, und man verändert nichts. Ist die Zeitdauer aber kleiner als Δt, so setzt man alle Werte während dieser Zeit Δt gleich der Schwellenhöhe und arbeitet bei Kanal $\Delta t+1$ analog dem obigen Prozedere weiter. Der Algorithmus in Pseudocode sieht wie folgt aus:

[36] Testmessung von K.Esslinger

```
i=0
do
      {
            if Wert[i]<Threshold
                {
                  Wert[i]=Threshold
                  i=i+1
                }
            else
                {
                  a=i
                  do
                        {
                              a=a+1
                        } while Wert[a]> Threshold
                  Länge=a-i
                  if Länge< Δt
                        {
                              for (l=i to l=a)  Wert[l]=Threshold
                        }
                  else
                        {
                              i=a+1
                        }
                }
      } while i<256
```

Baut man diese Routine in die Datenaufbereitung ein, so erhält man eine recht grosse Effizienz für die Signalsuche des eigentlichen Analyseprogrammes (siehe IV.2.2).

IV.2.ii Datenaufbereitung

Im folgenden möchte ich anschaulich den kompletten Ablauf der Datenaufbereitung von der Messung bis hin zur eigentlichen Analyse zeigen. Abbildung IV.9 zeigt ein typisches Signal der Driftkammer, das mit einem digitalen Oszilloskop am Frontausgang des FADCs gemessen wurde.

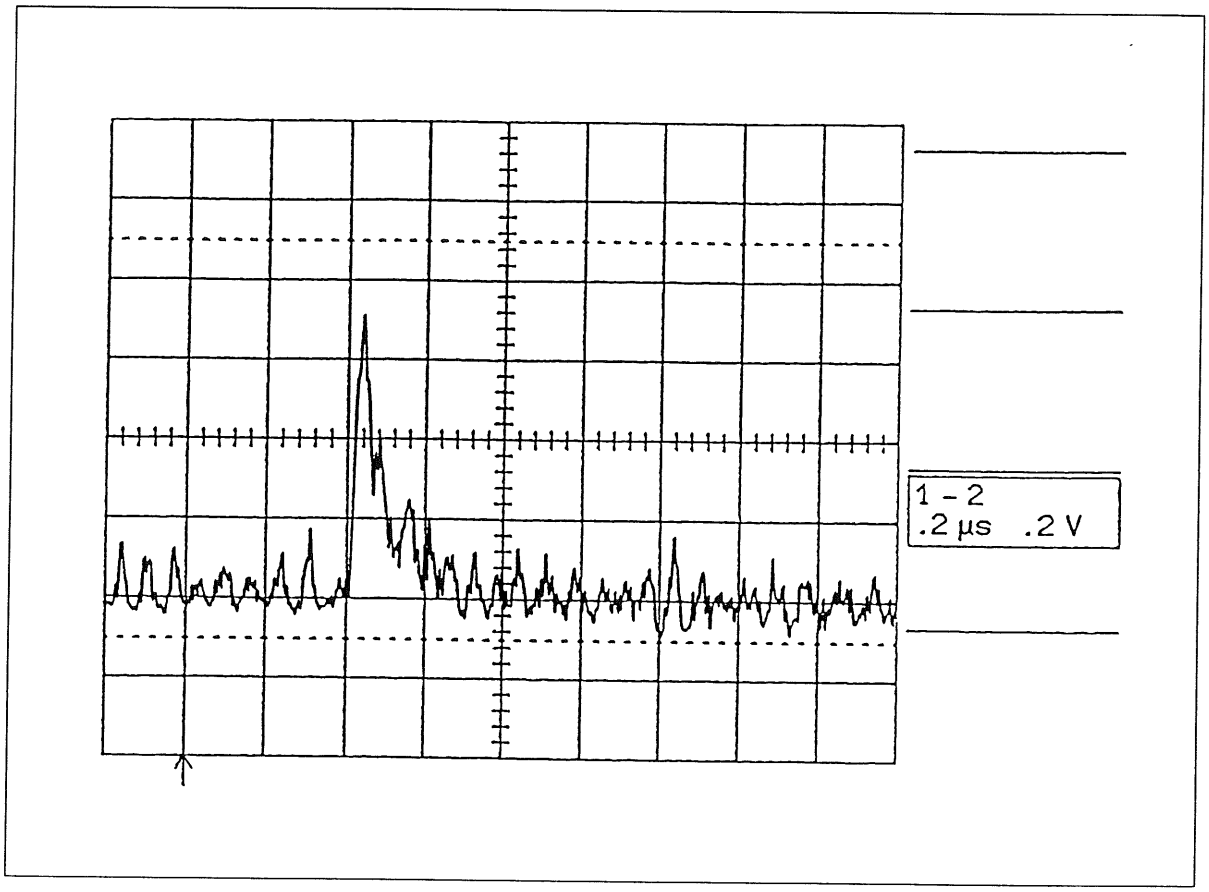


Abbildung IV.9 Signal der Driftkammer mit Digital-Oszilloskop gemessen;

Das gleiche Ereignis ist in Abbildung IV.10 als digitalisierter Ausgang des FADCs gezeichnet:

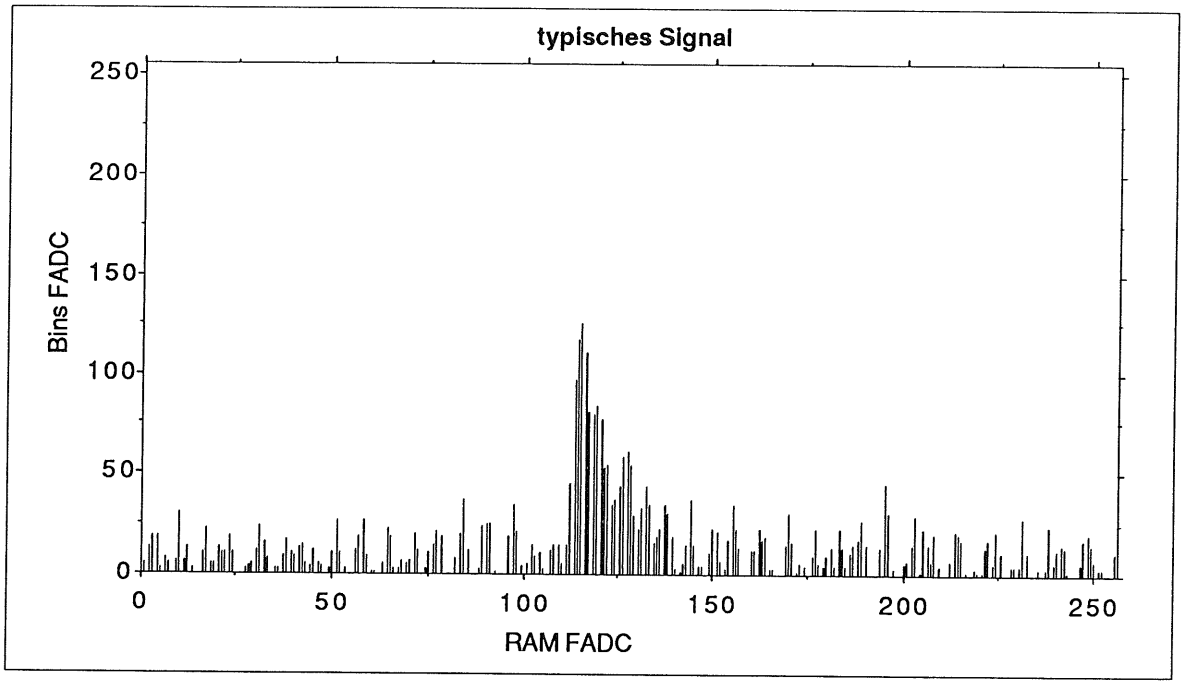


Abbildung IV.10 Puls mit FADC normal

Wird die Rauschunterdrückungsfunktion darauf angewendet, so haben die Werte folgende Form (Abbildung IV.11):

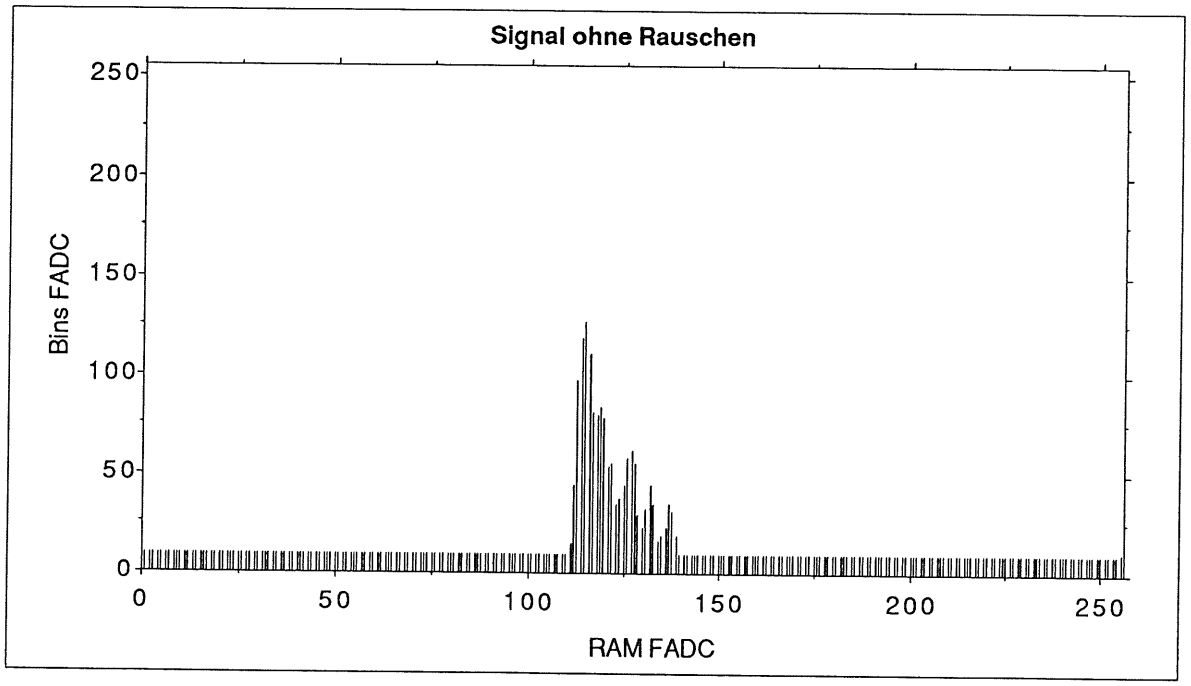


Abbildung IV.11 Puls nach "Anti-Noise Funktion"

Am Schluss der Datenaufbereitung wird die Kurve nun noch linearisiert (Abbildung IV.12). Werte, die grösser sind als 256 werden in dieser Darstellung aus programmtechnischen Gründen abgeschnitten.

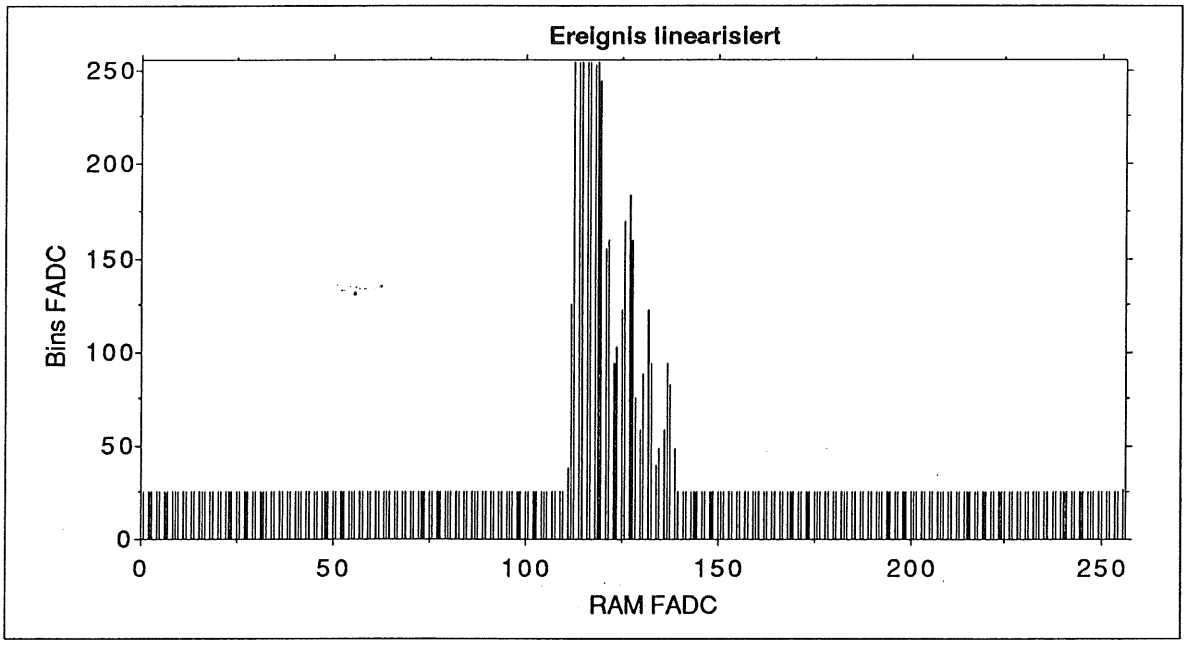


Abbildung IV.12 Puls linearisiert

Der Versuch, die Reihenfolge zu ändern hat sich nicht bewährt, da die Anti-Rauschfunktion, auf die linearisierten Daten angewendet, einige unwahre Ereignisse feststellt (siehe Abbildung IV.13).

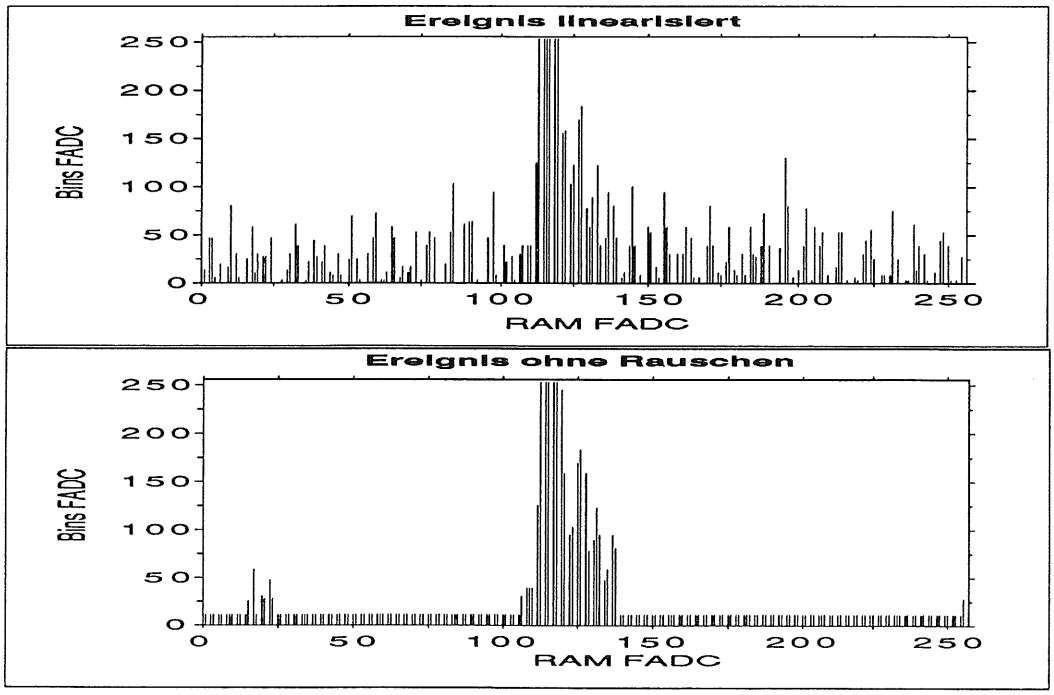


Abbildung IV.13 Umgekehrte Reihenfolge

IV.2.iii Geometrie und Koordinatensystem, Programm qtarn

Um ein vorläufiges Bezugssystem für die Driftkammer einführen zu können, betrachten wir einen seitlichen Querschnitt durch die Kammer (Abbildung IV.14):

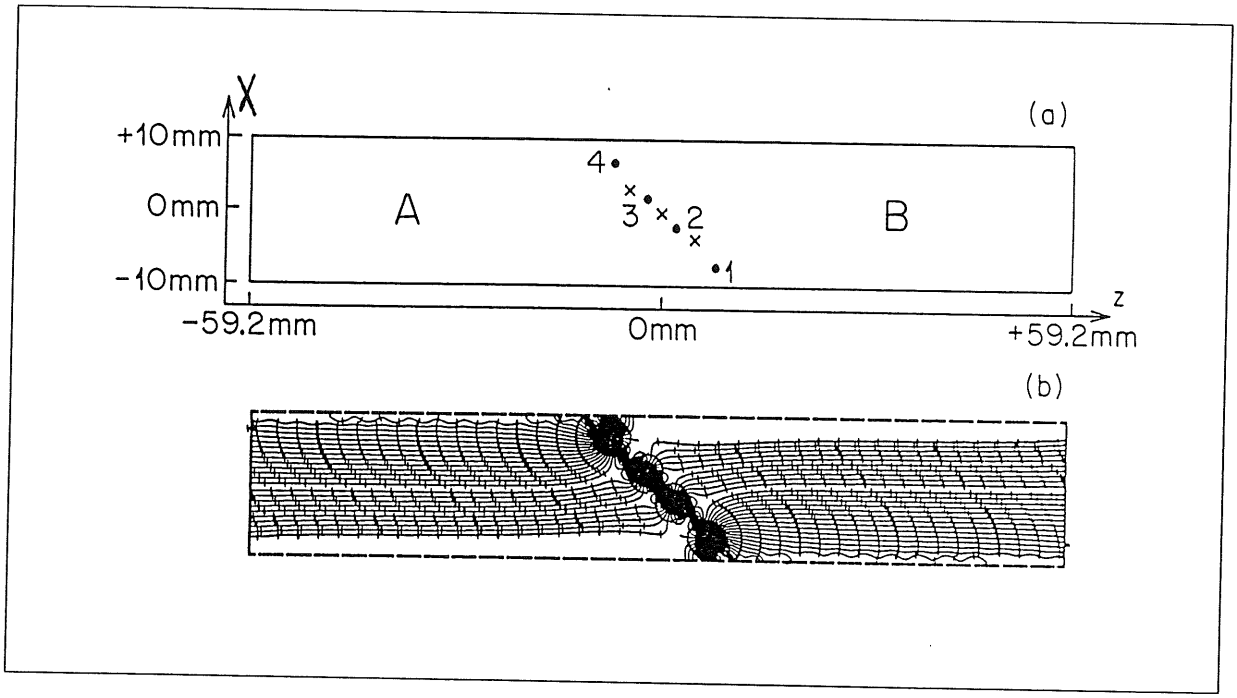


Abbildung IV.14 Koordinatensystem Driftkammer und Bild der Feldlinien (aus [13])

Resultate der Driftkammerdaten-Analyse werden nun sämtliche bezüglich obigem Koordinatensystem angegeben. Unter Driftkammerdaten sind von nun an die aus der Driftkammer ausgelesenen Werte aller vier zur Zelle 1 gehörenden Drähte gemeint, also insgesamt 8 * 256 Werte (an beiden Enden, links und rechts ausgelesen).
Nach Untergrundfilter und Linearisierung wird versucht, auf jedem Draht unabhängig Signale zu finden.
Die gewählte Anordnung der Feldlinien (Abbildung IV.14) sorgt dafür, dass bei einer Teilchenspur durch den Teil A der Kammer die Drähte 3, 2 und 1, bei einer Spur durch Teil B aber die Drähte 0, 1 und 2 ansprechen.
Ist ein Signal gefunden, so wird die Zeit t_L bzw. t_R und eine Amplitude A_L bzw. A_R berechnet (A ist das Integral des Pulses bis zum jeweiligen Maximum). Hier nun kann ein erster Schnitt vorgenommen werden: Ist nämlich der Zeitunterschied $\Delta t = |t_L - t_R|$ grösser als ein vorgegebener Wert, so gehören die zwei Pulse auf dem gleichen Draht nicht zum gleichen Ereignis und müssen weggeworfen werden. Ist alles in Ordnung, so wird eine mittlere Zeit

$t=\frac{t_L+t_R}{2}$ und aus der Asymmetrie der Pulshöhen bzw. Ladungsverteilungen A ein $\varphi=\frac{A_L-A_R}{A_L+A_R}$ berechnet (Abbildung IV.15).

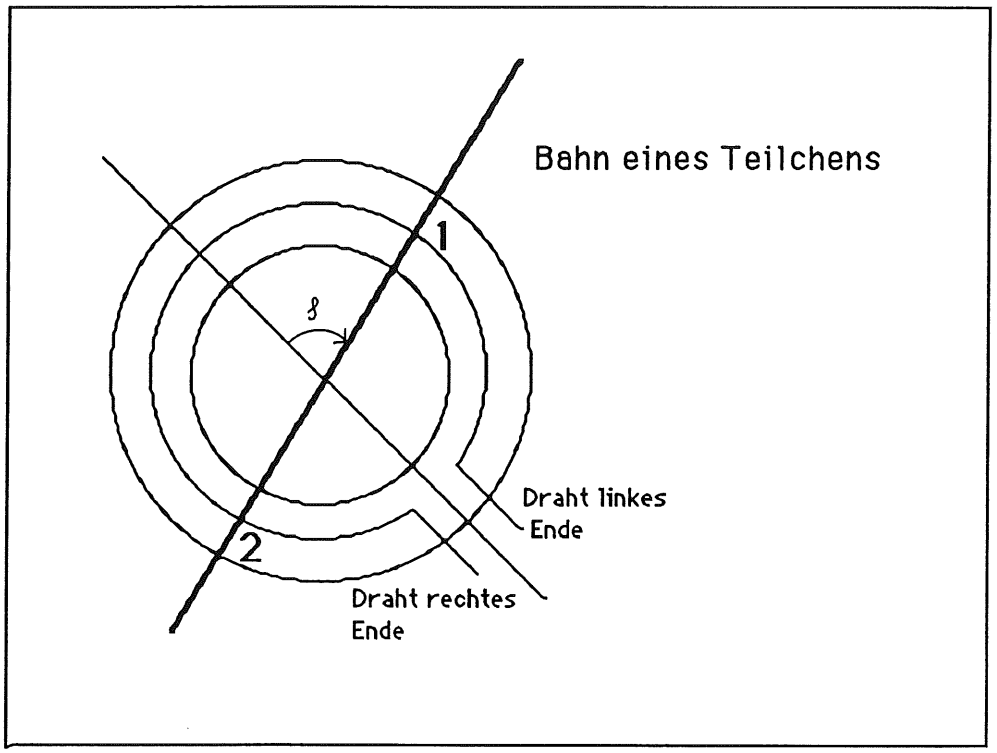


Abbildung IV.15a) Charge Division $A_{L1}>A_{L2}$ $A_{R2}>A_{R1}$

Je näher der Durchstosspunkt eines Teilchens bei einem Drahtende liegt, desto grösser wird die Amplitude A (der Widerstand R des Drahtes wird kleiner, V=R*I, I konstant). Betrachten wir in obigem Beispiel Durchstosspunkt 1, so ist sicherlich A_L grösser als A_R, umgekehrt im Durchstosspunkt 2. Die Grösse φ misst daher den Azimutalwinkel in Einheiten von π, mit allerdings beschränkter Auflösung ($\Delta\varphi\cong 0.3$).

Die Grössen t und φ können nun in Koordinatenangaben transformiert werden, was in der Arbeit von P. Robmann ref[13] beschrieben und im Programm "**qtarn**" ausgeführt ist.

Für das in Abbildung IV.15 gezeigte Ereignis erhält man die in Abbildung IV.16 schematisch skizierte Pulsform.

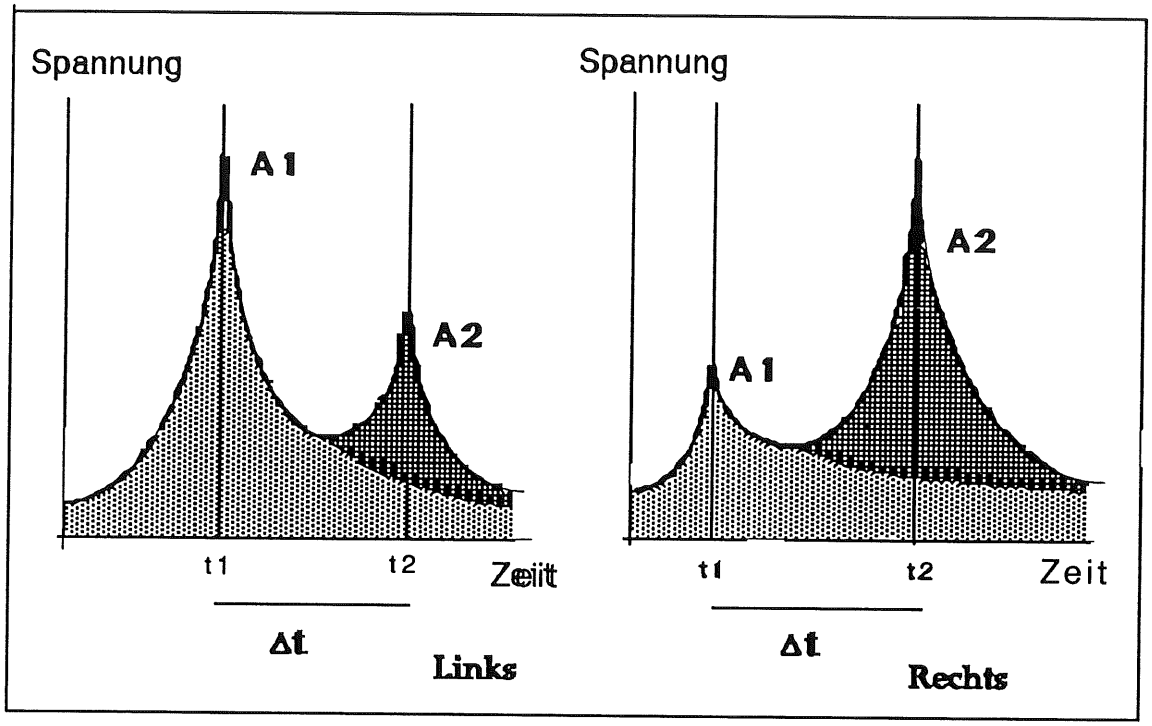


Abbildung IV.16 Optimale Pulsform

In diesem bevorzugten Fall kann man die Pulse A1 (und A2) an beiden Enden des Drahtes auslesen, verwenden und erhält ein z-Koordinate für den Durchstosspunkt 1 (und Durchstosspunkt 2). Da nicht in jedem Fall 1 und 2 so gut getrennt sind, muss vermieden werden, dass eine falsche Zuordnung auftritt (z.B. A_{L1} mit $A_{R2,.....}$), da dann weder die Ladungsverteilung noch die Puls-Zeit korrekt sind. Auch eine vollständige Überdeckung der beiden Pulse würde bereits genügen, das Ergebnis zu verfälschen, da z.B. die gesuchte Amplitude von A1 sicherlich kleiner ist als diejenige von A1 und A2 zusammen.
Damit diese Probleme nicht auftauchen, müssen zwei Bedingungen erfüllt sein:

- Die Pulse A1 und A2 müssen mindestens eine Zeit $\Delta t_{min} \cong 300ns \cong$ 30 Kanäle voneinander gestreut sein. Das bedeutet, dass sich die Driftstrecken von A1 und A2 um mindestens $\Delta z_{min}=v_d*\Delta t_{min}$ unterscheiden müssen (Abbildung IV.15.b).

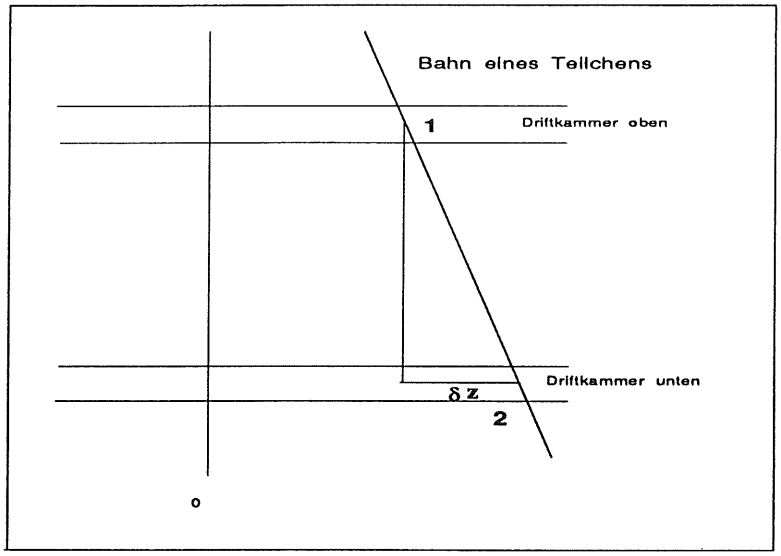


Abbildung IV.15b) Seitenansicht für die Berechnung von α_{min}

Daraus folgt zwingend, dass die Spur der Teilchen nicht auf einer Geraden senkrecht zur Driftkammer liegen dürfen. Für den kritischen Winkel gilt:

$$\arctan(\alpha_{min}) = \frac{\Delta z_{min}}{\text{Durchmesser Driftkammer}} = \frac{v_d * \Delta t_{min}}{l} = \frac{300ns*0.03mm/ns}{373mm}$$

$\rightarrow \alpha_{min} = 1.4^o$

- Ferner muss das Analyse-Programm überhaupt in der Lage sein, zwei oder mehrere Pulse auf dem gleichen Drahtende getrennt behandeln zu können. Dieses Problem ist im jetzt vorhandenen Algorithmus noch nicht vollständig gelöst, es kann daher in ungünstigen Fällen noch passieren, dass die beiden Pulse A1 und A2 nicht auseinandergehalten werden.

Der Wert für φ ist dann natürlich unsinnig, und das Ereignis sollte nicht berücksichtigt werden. Dieses prinzipielle Problem unserer Messapparatur taucht beim eigentlichen Messvorgang in Hamburg nicht auf, da der Ursprung der zu messenden Teilchen in der Mitte der Kammer liegt.
Eine Lösung dieses Dilemmas in unserer Versuchsanordnung könnte darin bestehen, die Driftkammer so zu drehen, dass das Auslesen der Drähte genau in der Senkrechten liegt, denn dann wären A_L und A_R für Durchstosspunkt 1 annähernd gleich gross und für Durchstosspunkt 2 annähernd 0 bzw. sehr gross, was eine Identifizierung massiv erleichtern würde (siehe Kap.IV.5.2).

IV.2.iv Programm "analdrift"

Die als Ergebnis der Zuordnung (t -> z) im Programm "qtarn" erhaltenen Punkte sind für die Drähte 1 und 2 nicht eindeutig, da für jeden Wert von t zwei z-Werte geliefert werden. Zu einer Teilchenspur gehören also maximal

5 Punkte, es ist aber ohne weiteres möglich und sogar der Normalfall, dass vereinzelte Punkte fehlen (Inneffizienz der Kammer) und dafür weitere Werte, herrührend vom zweiten Durchstosspunkt oder durch kurze Zeit später eintreffende Teilchen, bei einer Auswertung berücksichtigt werden müssen. Alle diese Punkte werden nun im Programm "analdrift" zu möglichen Teilchenspuren kombiniert.
In einem ersten Schritt werden alle Kombinationen von zwei und mehr Punkten, die mindestens von zwei verschiedenen Drähten stammen, zusammengestellt. Dabei werden Punktekombinationen, mit einer Winkeldifferenz $\Delta\varphi$(Draht i,Draht j) über einer wählbaren Schwelle ("deltaphicut") verworfen, da diese Kombinationen nicht vom gleichen Durchstosspunkt herrühren.
Bei den restlichen Punkten wird nun auf derjenigen Seite der Signaldrähte eine Gerade gefittet, auf der mehr Punkte vorhanden sind (z entweder immer >0 oder immer <0). Da der Idealfall von drei Punkten auf der gleichen Seite nur sehr selten vorkommt, und damit ein statistischer Test, wie gut diese Punkte auf einer Geraden liegen, nur selten angewendet werden kann, dürften nach wie vor recht häufig schlechte bis völlig falsche Werte in unsere Auswertung eingehen (siehe Kap. V.5.2). In Abbildung IV.17.a) ist ein Beispiel für einen einfachen Fall, bei dem man mit einer statistischen Prüfung (χ^2-Test) leicht feststellen kann, ob die drei Punkte auf der rechten Seite zur gleichen Spur gehören. Im Fall der in Abbildung IV.17.b) gezeichneten Punkteanordnung hat man dagegen keine Möglichkeit zu überprüfen, ob die beiden Punkte auf der rechten Seite tatsächlich zum gleichen Durchstosspunkt und zur gleichen Teilchenspur gehören.

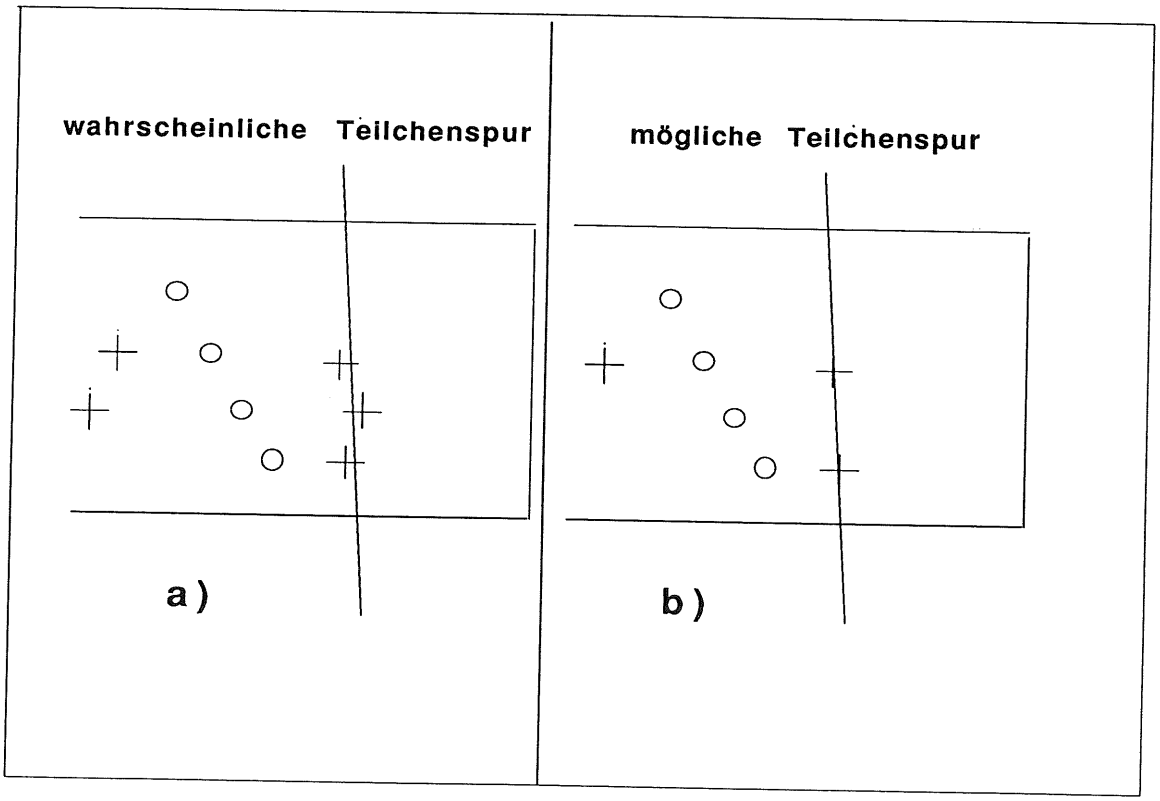


Abbildung IV.17 Beispiel für Geradenkonstruktion

Die gefundenen Spuren sind durch den Achsenabschnitt z (x=0) und die Neigung α charakterisiert.

IV.3 Messungen an der Proportionalkammer

Die Messungen mit der Proportionalkammer dienen, wie im letzten Abschnitt ausgeführt wurde, dazu, den Durchstosspunkt einer Teilchenbahn durch die Driftkammer, nun aber unabhängig von dieser, genauer zu bestimmen. Darum sind die Berechnungen und Programme völlig getrennt von der Driftkammerauswertung.

IV.3.i Geometrie und Koordinatensystem

Auch hier wollen wir nun wieder zuerst ein Koordinatensystem als Bezugspunkt wählen. Wir betrachten die Apparatur von vorne. Als Nullpunkt wurde der von der Driftkammer gebildete Kreismittelpunkt gewählt. Zufällig fällt dieser mit der dritten Ebene der Proportionalkammer auf die gleiche Höhe (Abbildung IV.18).

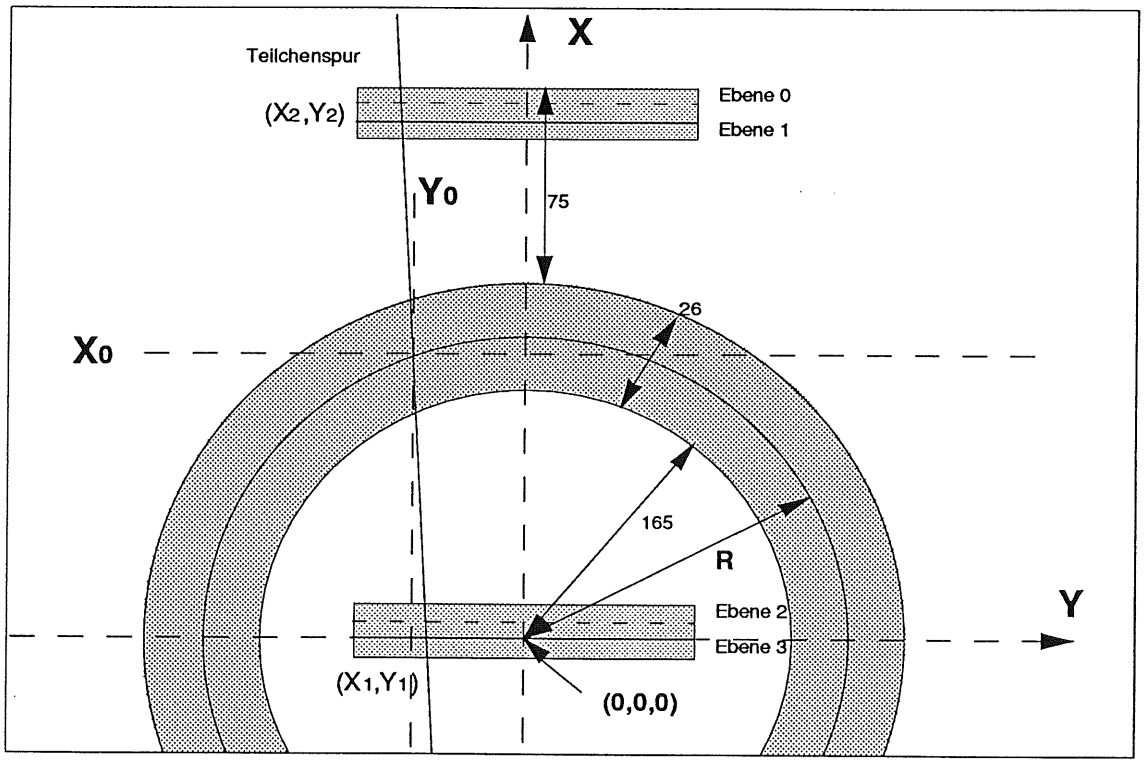


Abbildung IV.18 Koordinatensystem der Proportionalkammer von vorne

Sämtliche Angaben in obiger Abbildung sind in Millimeter gemacht.
Als ersten Schritt wollen wir den Punkt (X_0,Y_0) berechnen. Ausgangspunkt sind die bekannten Grössen (Y_2,X_2) bzw. (Y_1,X_1), welche von der Proportionalkammer mit einem Fehler von 1mm geliefert werden.

(X_0,Y_0) liegen auf dem Schnittpunkt von zwei Kurven:

- Auf einem Kreis, den die Driftkammer bildet $\Rightarrow x^2 + y^2 = R^2$
- auf einer Geraden, die durch die Teilchenspur gebildet wird $\Rightarrow y = a*x + b$

Leicht lassen sich nun die Werte für $a = \frac{Y_2-Y_1}{X_2-X_1}$ und $b=Y_1 - a*X_1$ bestimmen.
Benützt man nun noch die quadratische Kreisgleichung, erhält man:

$$X_{0_{12}} = \frac{-ab \pm \sqrt{a^2R^2-b^2+R^2}}{a^2+1}$$

Im folgenden werden wir nur noch die positive Wurzel verwenden (Lösung für oberen Schnittpunkt).
Das nächste Ziel ist die Bestimmung von Z_0. Dabei brauchen wir X_0, um angeben zu können, auf welcher Höhe die Driftkammer durchquert wurde (Abbildung IV.19):

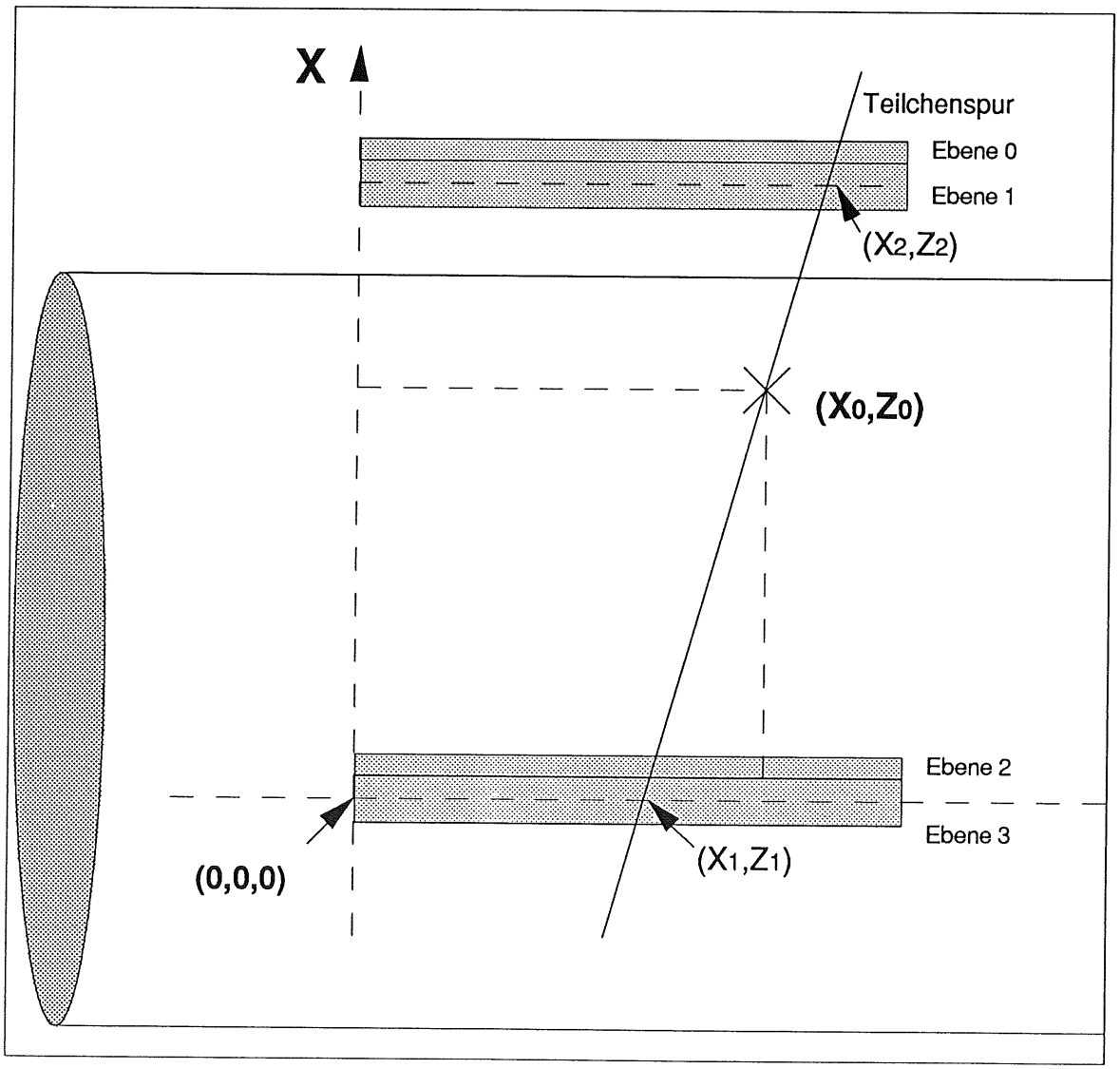


Abbildung IV.19 Koordinatensystem Proportionalkammer seitlich

Z_0 muss auch hier wieder auf einer Geraden liegen: $z = a * x + b$

Aus den Kammerdaten (X_1,Z_1) bzw. (X_2,Z_2) lassen sich $a = \frac{Z_2-Z_1}{X_2-X_1}$ und $b=Z_1 - a*X_1$ berechnen.

Die von uns gesuchte zentrale Grösse Z_0 , der Abstand des Durchstosspunktes der Teilchenspur durch die Driftkammer vom vorderen Rand, erhält man durch Einsetzen aller obigen Grössen:

$$\mathbf{Z_0 = a * x_0 + b}$$

Wir haben für die Koordinaten der Proportionalkammer einen Fehler angenommen, der kleiner ist als 1mm, darum ist klar, dass auch der Fehler von $\mathbf{Z_0}$ kleiner sein muss als 1mm.

Bei der Berechnung von Z_0 wird für jedes registrierte Ereignis in jeder der vier Ebenen separat nach den Drähten mit einem Signal gesucht. Hat mehr als ein Draht angesprochen, wird kontrolliert, ob es sich um nebeneinander liegende Drähte handelt. Ist dies der Fall, so wird der Mittelwert der Drähte fverwendet, wobei die maximale erlaubte Anzahl auf 50 beschränkt werden kann. Zweideutige Ereignisse mit mehr als einem (x,y) Kreuzungspunkt in jeder Kammer werden ebenfalls verworfen.

IV.4 Absoluteichung der Driftkammer

Setzt man nun alle diese in den vorherigen Abschnitten vorgeführten Schritte zusammen, so sind wir unserem Ziel, einer Methode für die Absoluteichung einer Driftkammer, schon sehr nahe. Bevor ich in iV.4.ii. die erhaltenen Resultate diskutiere, möchte ich im nächsten Abschnitt noch einmal einen Gesamtüberblick über die entwickelte Analysesoftware geben.

IV.4.i Programmaufbau

Der ganze Aufbau lässt sich mit folgendem Schema veranschaulichen (Abbildung IV.20):

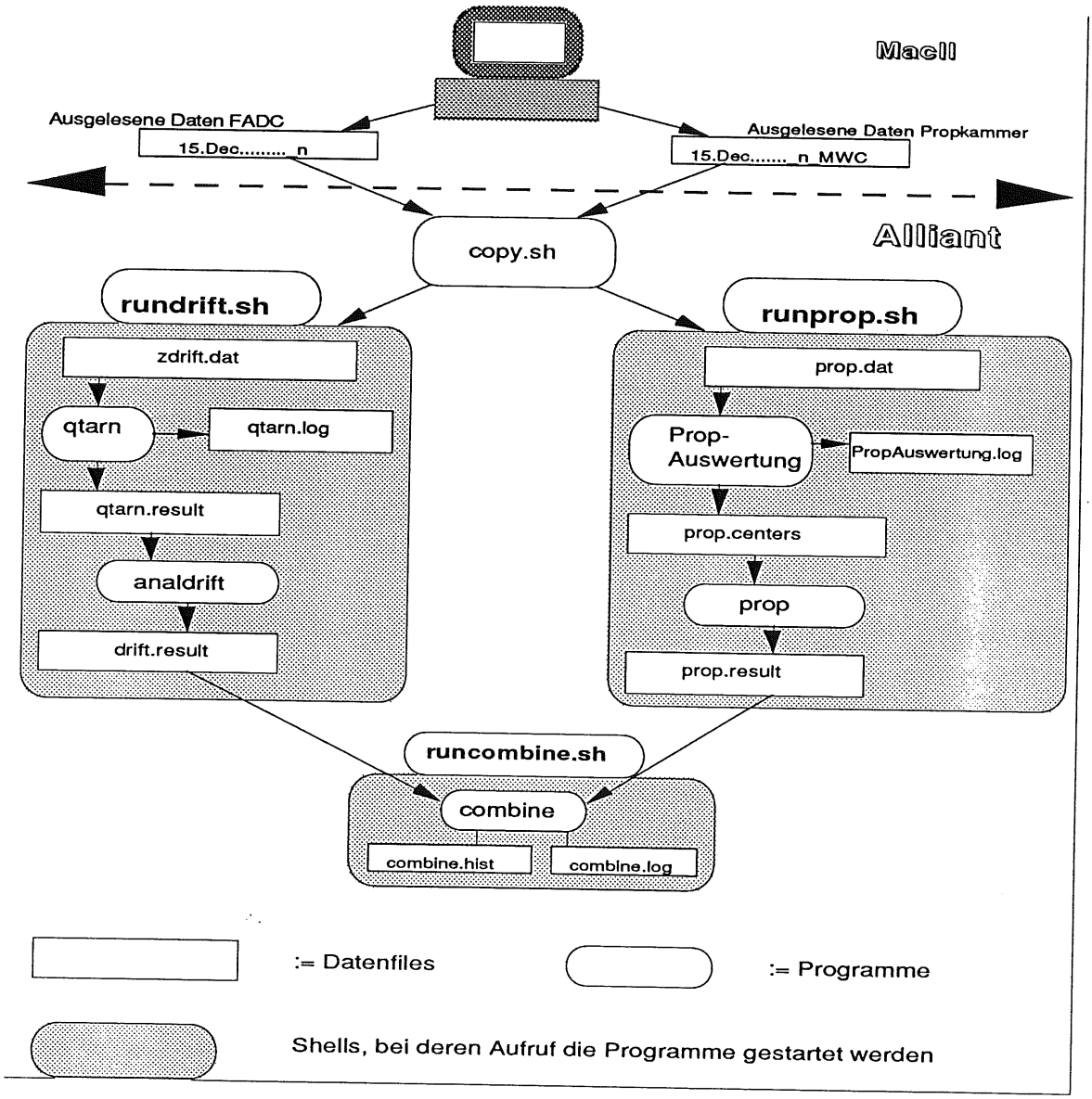


Abbildung IV.20 Programm-Aufbau

Die Form der einzelnen Datenfiles sind im Anhang VI.2 genau aufgelistet.

In untenstehender Liste sind die Variablen und die Optionen für die einzelnen Programmstufen aufgeführt:

Programmstufe	**Was man eingeben kann und muss**
copy.sh	Namen und Anzahl der zu verarbeitenden Files
PropAuswertung	Anzahl benachbarter Kammerdrähte, die erlaubt sind
rundrift.sh	Pulsform der Ereignisse zeichnen Rauschunterdrückung: Minimale Länge für Puls Rauschunterdrückung: Schwelle für Puls Max. Zeitdifferenz zwischen Pulsen von linker und rechter Seite des Drahtes Obere Grenze für χ^2 bei Geradenanpassung mit mehr als zwei Punkten Obere Grenze für $\Delta\varphi$ zwischen verschiedenen Punkten
runcombine.sh	Schnitt auf χ^2 Unterer Wert für φ minimale Anzahl der geforderten Drähte (2 oder 3) maximale Anzahl der Spuren pro Ereignis maximaler Betrag des Winkels in der Proportionalkammer

Das aussehen dieser Programmteile ist in Anhang VI.3 aufgelistet.

IV.4.ii Resultate

Wie in IV.4.i. ausführlich beschrieben ist, kann man mittels einstellbarer Schnitte die Datenanalyse stark beeinflussen. Ich möchte darum bei einer Messung mit insgesamt 1850 Ereignissen die Auswertung mit verschieden gewählten Werten für diese Schnitte darstellen und die Auswirkungen zeigen.
Für die erste Einstellung habe ich folgende Werte gewählt:

Anzahl nachfolgender Kammerdrähte der Proportional-Kammer, die erlaubt sind [# DrähteProp]	2
Rauschunterdrückung: Minimale Länge für Puls [Länge]	5 Kanäle (50 ns)
Rauschunterdrückung: Schwelle für Puls [Threshold]	15
Maximale Zeitdifferenz zwischen Pulsen von linker und rechter Seite des Drahtes [Δt]	2 Kanäle (20 ns)
Obere Grenze für χ^2 bei Geradenanpassung mit mehr als zwei Punkten [χ^2]	2.0
Obere Grenze für $\Delta\varphi$ zwischen den verschiedenen Punkten [$\Delta\varphi$]	0.2

Unterer Wert für φ [φ]	0
Minimale Anzahl der geforderten Drähte (2 oder 3) [#Drähte]	2
Maximale Anzahl der Spuren pro Ereignis [#Spuren]	2
Maximaler Betrag des Winkels der Proportional-kammer [α]	0.18 (10.3°)

Mit dieser Einstellung haben 125 Events bis zum Schluss der Analyse überlebt. In der nachfolgenden Liste wird aufgeführt, bei welchem Schnitt die Ereignisse verlorengehen:

eingelesene Ereignisse	1849
Ereignisse, die den #Spuren-Schnitt überlebt haben:	459
Spuren mit einem vorhandenen Wert für z-Drift:	569
Spuren, die den χ^2-Schnitt überlebt haben:	569
Spuren, die den #Drähte-Schnitt überlebt haben:	569
Spuren, die den φ-Schnitt überlebt haben:	419
Spuren mit einem vorhandenen Wert für z-Prop:	191
Spuren, die den α-Schnitt überlebt haben:	125

In Abbildung IV.21 sind die gemessenen z-Prop-Werte als Funktion der z-Drift-Daten aufgetragen:

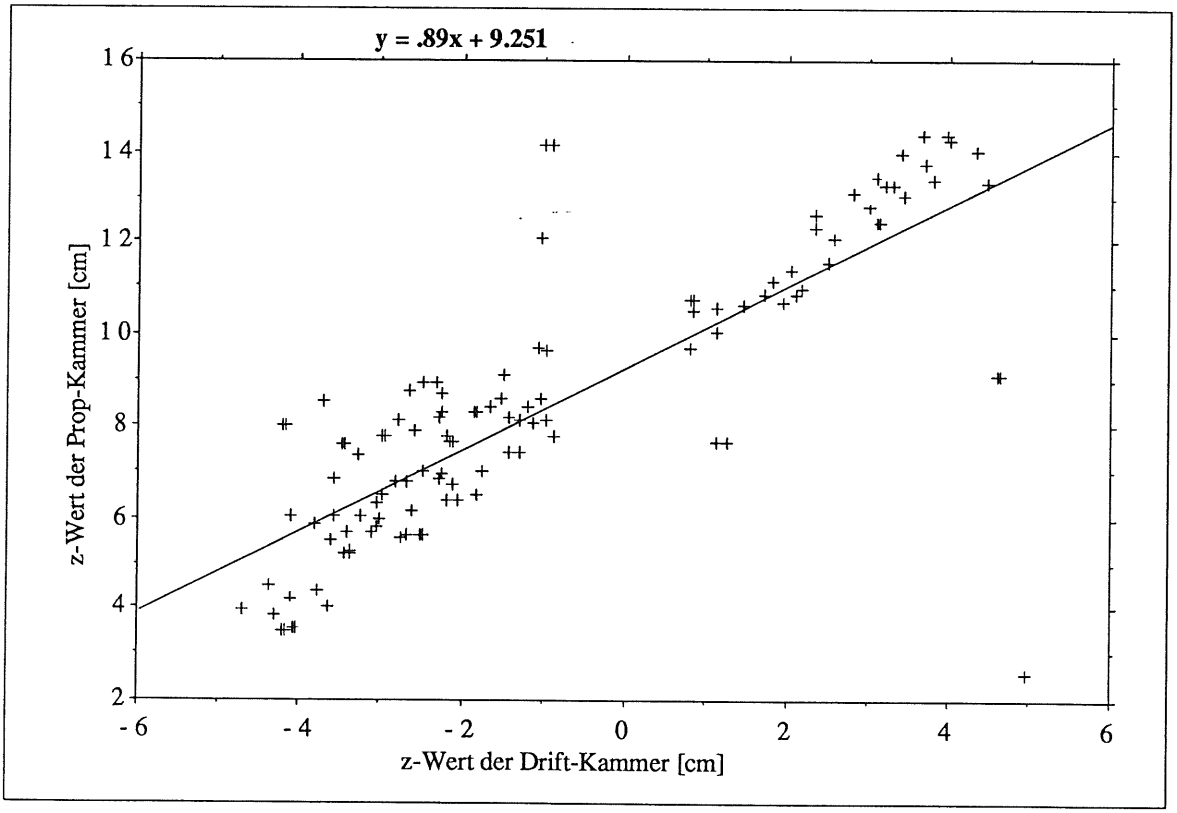


Abbildung IV.21 z-Prop als Fkt von z-Drift

Gleichzeitig ist in Abbildung IV.19 eine durch diese Punkte angepasste Gerade zu sehen. Folgende Werte wurden durch das "Fit-Programm" ausgegeben:

Steigung a	Achsenabschnitt b [mm]	c	σ_a	σ_b [mm] [35]
0.89	92.5	295	0.05	1.7

Ändert man bei dieser Einstellung die Schwelle der Pulshöhe im vernünftigen Rahmen, so ändert sich an diesen Zahlen nichts.
Geht man aber dazu über, die minimal verlangte Anzahl von zwei auf drei Drähte zu erhöhen, sieht das Bild wie in Abbildung IV.22 aus:

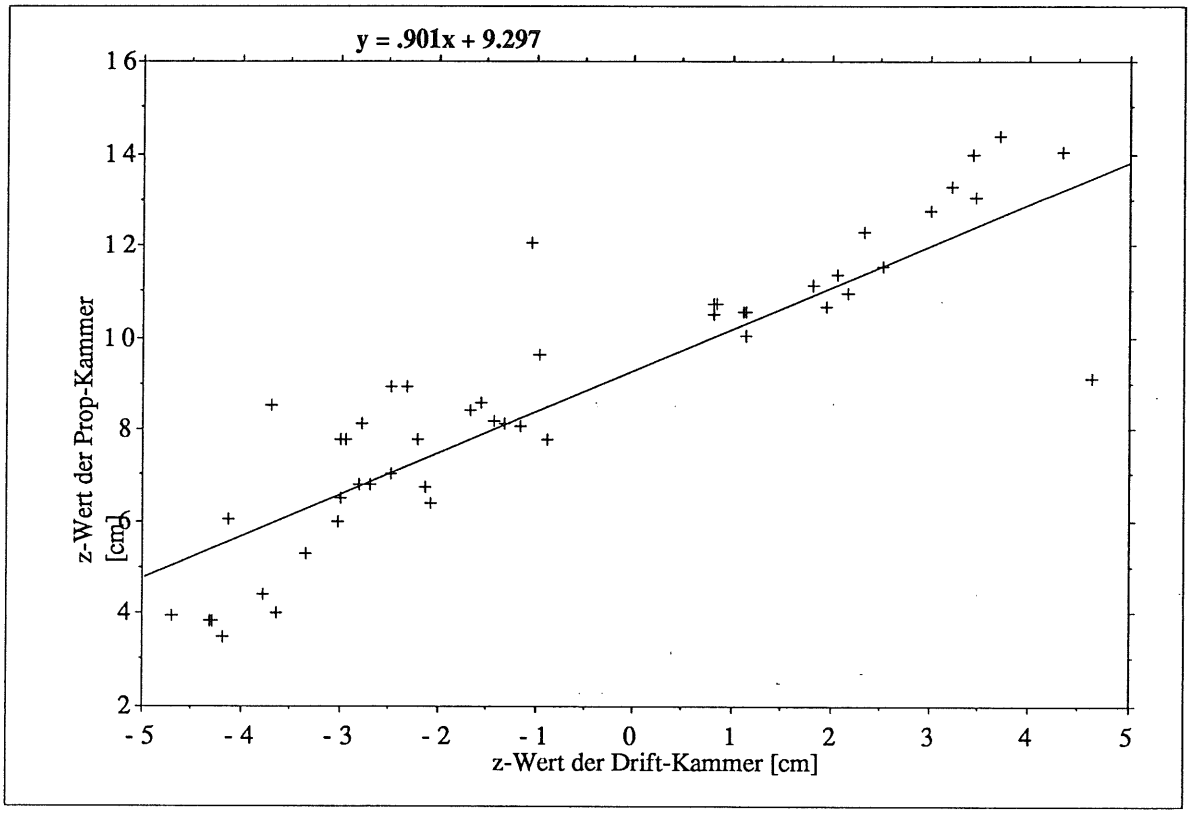


Abbildung IV.22 z-Prop als Fkt von z-Drift, minimal 3 Drähte

Die gesuchten Werte haben nur wenig geändert:

Steigung a	Achsenabschnitt b [mm]	c	σ_a	σ_b [mm] [35]
0.90	93.0	220	0.03	0.6

[35] Siehe Anhang VI.1
[35] Siehe Anhang VI.1

Als nächstes wurde der maximal erlaubte Proportional-Kammer-Winkel auf 5^0 verkleinert (Abbildung IV.23).

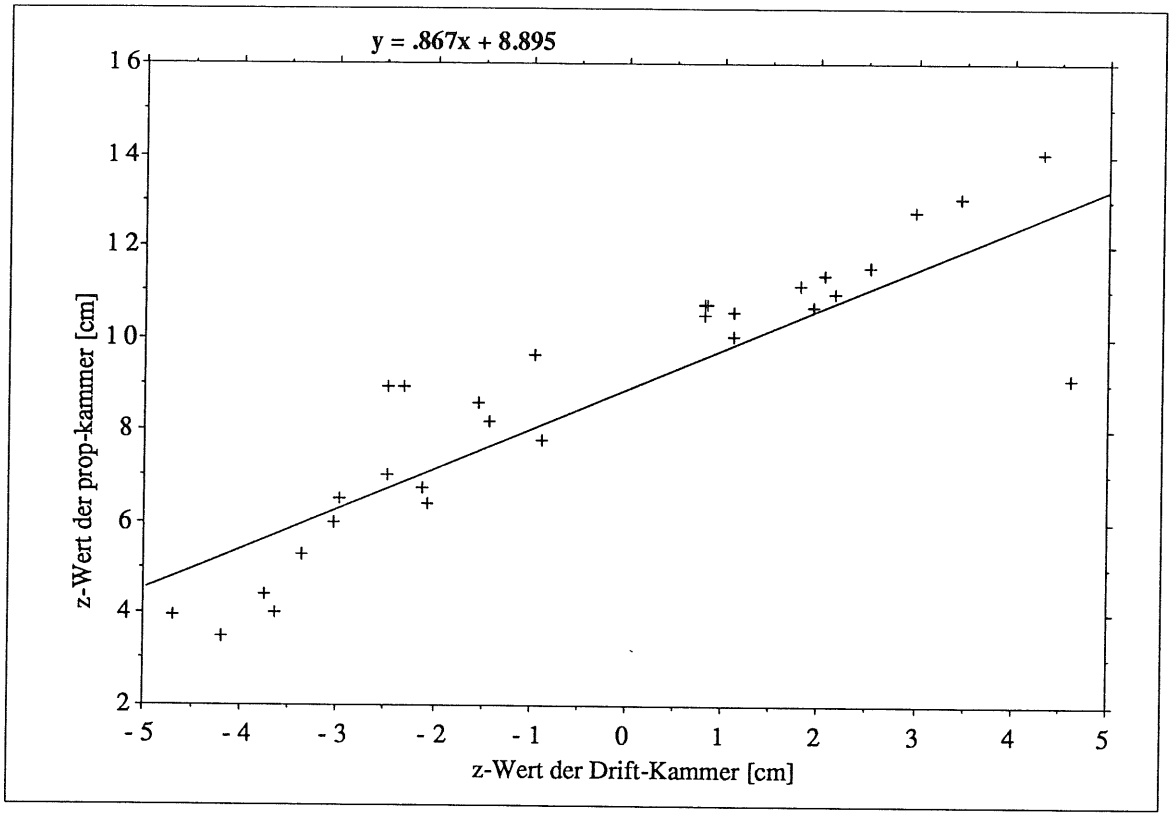


Abbildung IV.23 z-Prop als Fkt von z-Drift, Prop-Winkel<5^0

Die gefitteten Daten betragen:

Steigung a	**Achsenabschnitt b [mm]**	**c**	σ_a	σ_b [mm] [35]
0.87	89.0	213	0.01	0.3

Mit der Verschärfung der Bedingungen für die Grösse des Prop-Winkels und der Anzahl gefeuerter Drähte haben massiv weniger Teilchen die Analyse überlebt. Hier wiederum eine Liste:

eingelesene Ereignisse	1849
Ereignisse, die den #Spuren-Schnitt überlebt haben:	459
Spuren mit einem vorhandenen Wert für z-Drift:	569
Spuren, die den χ^2-Schnitt überlebt haben:	569
Spuren, die den #Drähte-Schnitt überlebt haben:	258
Spuren, die den φ-Schnitt überlebt haben:	185
Spuren mit einem vorhandenen Wert für z-Prop:	81
Spuren, die den α-Schnitt überlebt haben:	31

[35] Siehe Anhang VI.1

Beim Betrachten der Daten fällt auf, dass vor allem die Werte für z-Prop = 90 aus dem Rahmen fallen. Anhand der gefitteten Geraden stellt man fest, dass an der Stelle z-Prop=90 der Wert z-Drift=0 sein sollte. Von der Funktionsweise der Driftkammer her ist es klar , dass beim Durchgang von Teilchen durch die von den Signaldrähten aufgespannte Ebene (z-Drift=0) Problemen auftauchen können. Ich habe daher bei der nächsten Auswertung alle Werte z-Prop=90 weggelassen (Abbildung IV.24):

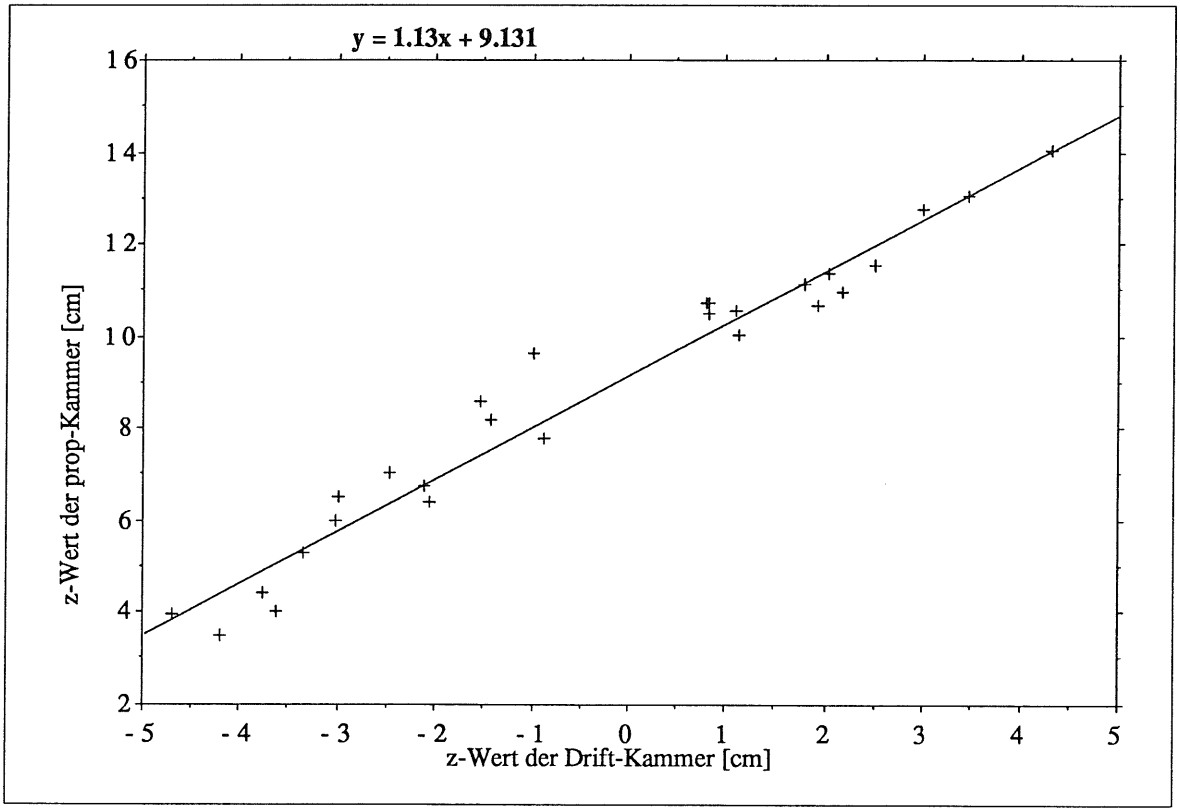


Abbildung IV.24 z-Prop als Fkt von z-Drift, ohne Werte z-Prop=90

Die dabei gefitteten Werte betragen:

Steigung a	Achsenabschnitt b [mm]	c	σ_a	σ_b [mm] [35]
1.13	91.3	52	0.05	1.4

[35] Siehe Anhang VI.1

Als letztes Bild in meiner Versuchsreihe habe ich den Wert für χ^2 auf 1.5 verschärft (Abbildung IV.25):

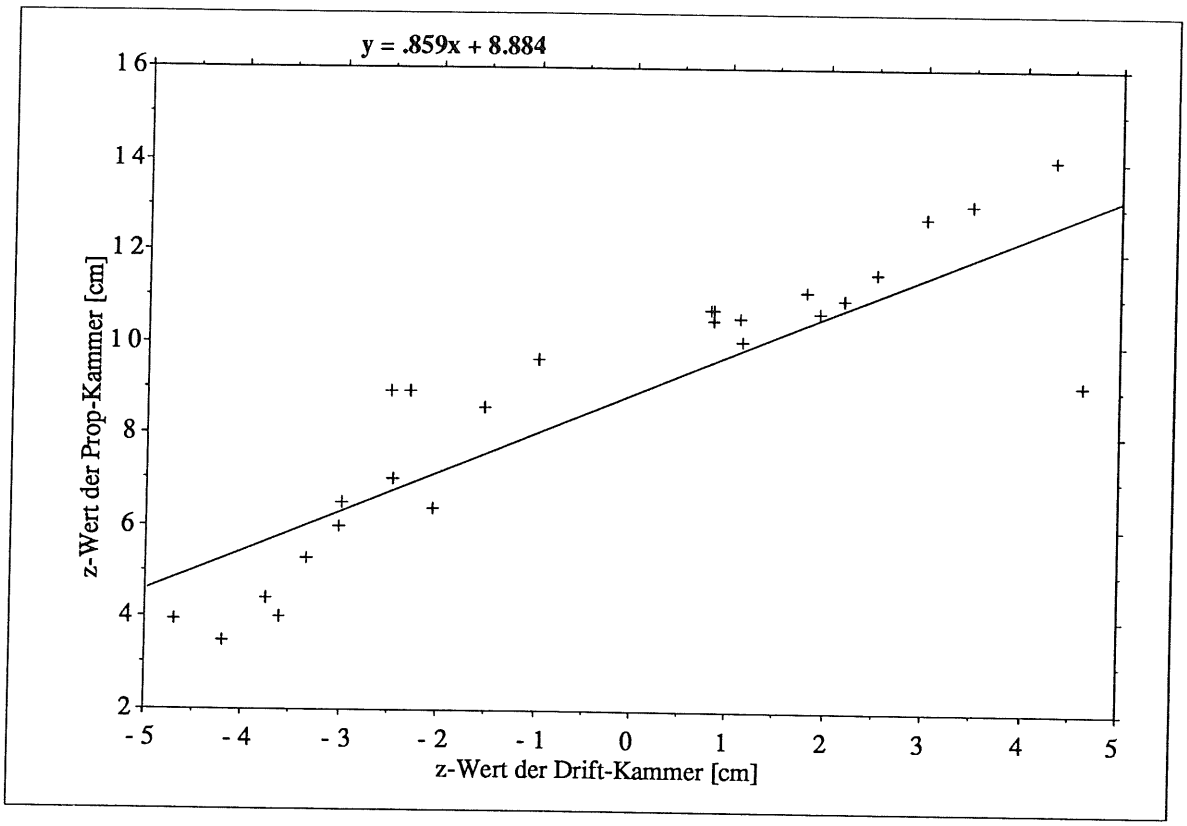


Abbildung IV.25 z-Prop als Fkt von z-Drift, χ^2 =1.5

Die gefitteten Daten:

Steigung a	Achsenabschnitt b [mm]	c	σ_a	σ_b [mm] [35]
0.86	88.2	230	0.01	0.2

Die gleiche Kurve zeigt Abbildung IV.26 nochmals mit den weggelassenen Events um z-Drift=0 herum:

[35] Siehe Anhang VI.1

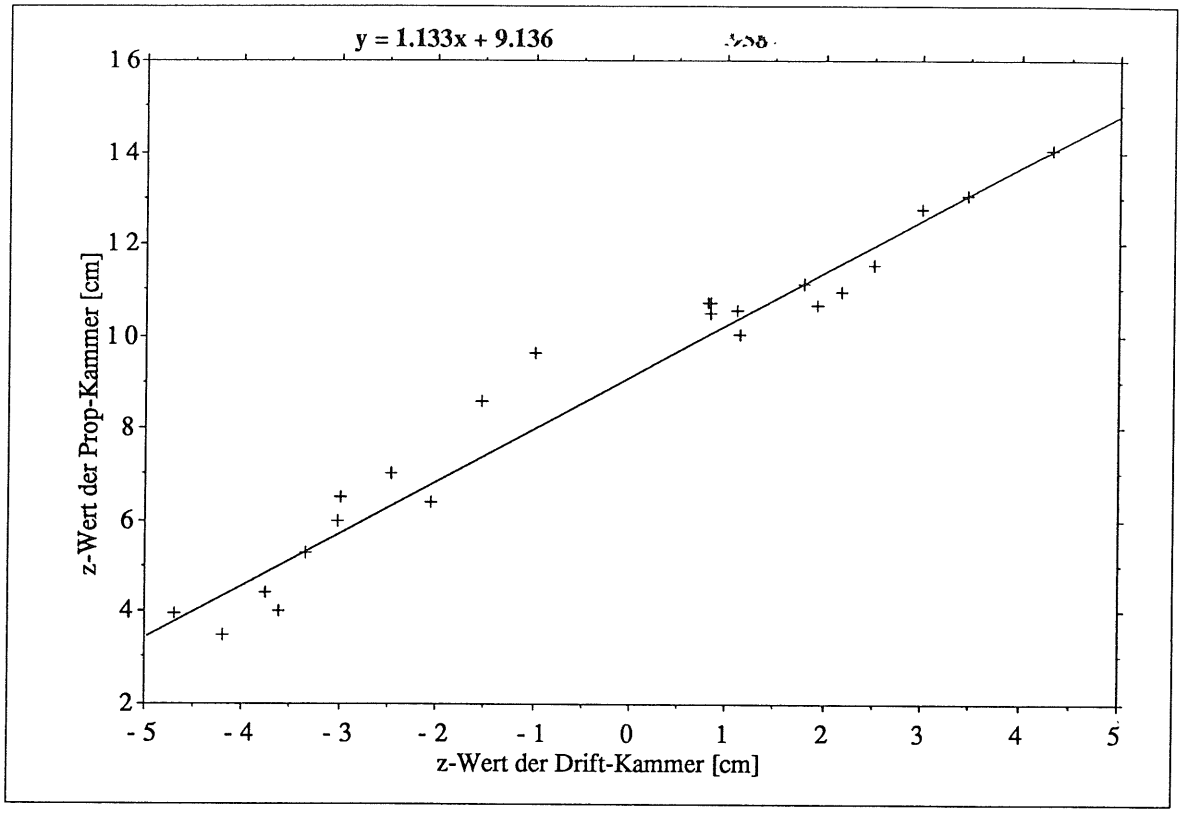


Abbildung IV.26 z-Prop als Fkt von z-Drift, χ^2 =1.5, ohne Werte z-Prop=90

Auch hier habe ich die erhaltenen Werte aufgelistet:

Steigung a	Achsenabschnitt b [mm]	c	σ_a	σ_b [mm] [35]
1.13	91.4	36	0.04	0.1

IV.4.iii Diskussion der Resultate

Die erhaltenen Messwerte sind doch etwas überraschend. Wenn wir von der Annahme ausgehen, dass bei den Proportional-Kammer-Werten kein grosser systematischer Fehler vorliegt (die Kammer war schon viele Male bei anderen Experimenten in Betrieb), so liegen doch selbst in der letzten Messreihe grosse Abweichungen der Driftkammer-Werte von den Sollwerten vor (Abweichungen bis zu 5 mm). Für eine Kammer, die Präzisions-Messresultate liefern sollte (Fehler kleiner als 0.3mm), sind diese Abweichungen inakzeptabel.
Das Entstehen dieser Ungenauigkeiten bei den von der Driftkammer gelieferten Koordinaten-Werte kann mit einer zuwenig effizienten Analyse der ausgelesenen Daten erklärt werden. Betrachtet man ein ideales Ereignis, erwartet man bei der Analyse der Driftkammer-Werte als Ergebnis fünf z-Werte, die zu einer möglichen Teilchenspur kombiniert werden müssen (siehe V.2.iv).

[35] Siehe Anhang VI.1

In den meisten Fällen sollten zudem fünf zusätzliche Werte, die vom zweiten Durchstosspunkt der Kammer stammen, auftauchen (siehe IV.2.iii). Berücksichtigt man die Möglichkeit, dass einzelne Pulse zu klein sind, um ausgewertet werden zu können, so müsste trotzdem die Gesamtzahl der pro Ereignis erhaltenen z-Daten um den Wert 8 schwanken. Vergleicht man diese Zahl mit der tatsächlich gemessenen Anzahl von ca. 5 Punkten pro Ereignis für den z-Wert, so stellt man ein Missverhältnis fest.
Dieses Missverhältnis lässt zwei Schlüsse zu: Entweder können die Messdaten mit dem vorhandenen Analysealgorithmus nicht genügend aufgelöst werden (Inneffizienz der Datenanalyse) oder die Messapparatur misst mit einer zu kleinen Auflösung (Inneffizienz der Messapparatur). Die Effizienz der Kammer wurde von P. Robmann [13] überprüft und sollte darum einwandfrei gewährleistet sein. Daher liegt der Schluss nahe, dass man in einer nächsten Phase der Arbeit versuchen muss, das Analysieren der gemessenen Daten und dabei vor allem die Suche der Maximas (Driftzeit) zu optimieren. Es ist unumgänglich, dass man das in Abschnitt IV.2.iii geschilderte Problem löst, da sonst keine genaueren Messresultate erwarten werden können.
Trotz all diesen Problemen kann man feststellen, dass die verwendete Tabelle für die Driftgeschwindigkeiten im wesentlichen stimmt, da die Steigung der Geraden in Abbildung IV.24 (im wesentlichen: verwendete Driftgeschwindigkeit/Solldriftgeschwindigkeit) nahezu gleich eins ist. Damit kann man auch dem erhaltenen Achsenabschnitt, also dem Nullpunkt der Driftkammer relativ zum Nullpunkt der Proportionalkammer, recht grosses Vertrauen schenken. Zudem kam eine Handmessung (Meterband) auf ein ähnliches Ergebnis (nur weniger genau).

V. Zusammenfassung

Im ersten Teil dieser Arbeit wurden sämtliche Komponenten, die zum Betrieb und der Auswertung einer z-Driftkammer gehören, zusammengesetzt und zum Laufen gebracht. Der Schwerpunkt lag dabei einerseits bei der Entwicklung eines Programmes zur Steuerung der Datenauslesung eines neuentwickelten Analog-Digital-Konverters, auf der anderen Seite bei der Inbetriebname der Analyse-Software auf dem Grossrechner Alliant. Es ist nun der vollständige Ablauf von der Messung mit der Kammer bis zur Ausgabe einer Ortsangabe eines Teilchenstrahles sichergestellt.
In einer zweiten Phase wurde versucht, eine neue Methode zur Eichung der von einer Driftkammer gemessenen Ortsangabe auszuarbeiten. Damit kann der Zusammenhang zwischen dem von der Driftkammerdatenanalyse gelieferten Nullpunkt und dem tatsächlichen Nullpunkt der Kammer bestimmet werden. Dabei findet eine Überprüfung der gebrauchten Tabelle mit den Zusammenhängen zwischen Driftzeit und Teilchenspur (im wesentlichen Tabelle der Driftgeschwindigkeit) statt. Bei der von uns entwickelten Methode wird die absolute Ortsangabe (Bahn des zu messenden Teilchens) durch eine 1mm-Proportionalkammer gemessen.
Grundsätzlich ist die Methode brauchbar, es bestehen aber noch durch Untergrund-Rauschen bedingte Probleme beim Auffinden und Auswerten der Signale der Teilchenspuren. Dadurch können falsche oder ungenaue Driftzeiten entstehen, die das Gesamtbild verfälschen.
Für die Messung mit dem Prototypen einer Driftkammer haben wir festgestellt, dass die verwendete Tabelle für die Driftgeschwindigkeiten nur leicht modifiziert werden muss. Trägt man die erhaltenen z-Werte der Driftkammer als Funktion der tatsächlichen Werte (von Proportional-Kammer) auf, so erhält man eine Steigung von 1.13(4). Da diese Steigung dem Verhältnis gewählte Driftgeschw./Soll-Driftgeschw. entspricht, ist die verwendete Tabelle sicherlich brauchbar.

VI. Anhang

VI.1 Überlegungen zur Fehlerrechnung

Bei einer Geradenanpassung versucht man den Wert $\chi^2 = \sum \frac{(y_{gem}-y_{theo})^2}{\sigma^2}$ zu minimalisieren. Dabei gilt für jeden gemessen Wert y_{gem}, dass er mit einem Fehler σ behaftet ist. Der optimale Wert für χ^2 ist die Anzahl der Freiheitsgrade f (bei einer Geraden gleich n-2).

Der genaue Wert für σ ist bei unserer Messung unbekannt; in dem verwendeten Programm wird er darum vorerst einmal als konstant angenommen und gleich 1 gesetzt ($=\sigma_p$).

Dadurch wird in unserem Programm ein Wert χ^2_p minimalisiert, der verschieden ist vom eigentlichen χ^2.

Die absoluten Werte der Steigung a und des Achsenabschnittes b sind unabhängig vom gewählten σ_p, nicht aber die Fehler σ_a und σ_b.

Was wir für die Bestimmung dieser Fehler also brauchen, ist einen "vernünftigen " Wert für σ. Wir verwenden die Bedingung, dass das χ^2 seinen Erwartungswert annehmen soll:

$$\chi^2(\sigma)=f$$

Man macht nun den Ansatz: $\sigma=\sqrt{c}*\sigma_p \Rightarrow f=\frac{1}{c}*\chi^2_p \Rightarrow \quad c=\frac{\chi^2_p}{f}$

Für σ_a und σ_b gilt dann:

$$\sigma_a=\sqrt{c}*\sigma_{a_p} \qquad \text{und} \qquad \sigma_b=\sqrt{c}*\sigma_{b_p}$$

VI.2 Figurenverzeichnis

VI.3 Steckerbelegung der I/O-Karte

Da Kanal JK1 defekt ist, gibt es eine kleine Änderung im Bezug auf die Beschreibung in Kapitel 2. Es muss nun alles über Kanal JK2 abgewickelt werden:

Pin JK2	Bedeutung des Anschlusses
PA0	Meldung des Macs, dass fertig ausgelesen
PA1	Clear Bift
PA2	Load Bift
PA3	Mastergate Bift
PA4	Testdata Bift
PA5	ExtStart Bift
PA6	ExtReset Bift
PA7	ExtClock Bift

VI.4 Register Offsets der I/O-Karte

```
/* Defines : XVME-200 IO-Board Stecker JK1*/
/*****************************************/


#define IoBasAdress 0xB07F9000  /*Basis-Adresse der I/O-Karte,die mit-
tels Stecker eingestellt werden kann*/

/* Port offsets ,offiziel aus Manuel entnommen*/
#define PGCR            1
#define PSRR            3
#define PADDR           5
#define PBDDR           7
#define PCDDR           9
#define PIVR            11
#define PACR            13
#define PBCR            15
#define PADR            17
#define PBDR            19
#define PAAR            21
#define PBAR            23
#define PCDR            25
#define PSR             27
#define TCR             33
#define TIVR            35
#define CPR             39
#define TSR             53


/* Defines : XVME-200 IO-Board SteckerJK2 */
/*****************************************/

/* Port offsets ,offiziel aus Manuel entnommen*/
#define PGCR2           65
#define PSRR2           67
#define PADDR2          69
#define PBDDR2          71
#define PCDDR2          73
#define PIVR2           75
#define PACR2           77
#define PBCR2           79
#define PADR2           81
#define PBDR2           83
#define PAAR2           85
#define PBAR2           87
#define PCDR2           89
#define PSR2            91
#define TCR2            97
#define TIVR2           99
#define CPR2            103
#define TSR2            117

#define PVECT1          128
#define PVECT2          132
#define TVECT1          136
#define TVECT2          137

#define IO_CNF_HPORT      0x20
#define IO_CNF_DATAPORT   0x80
```

VI.5 Initialisierung der I/O-Karte

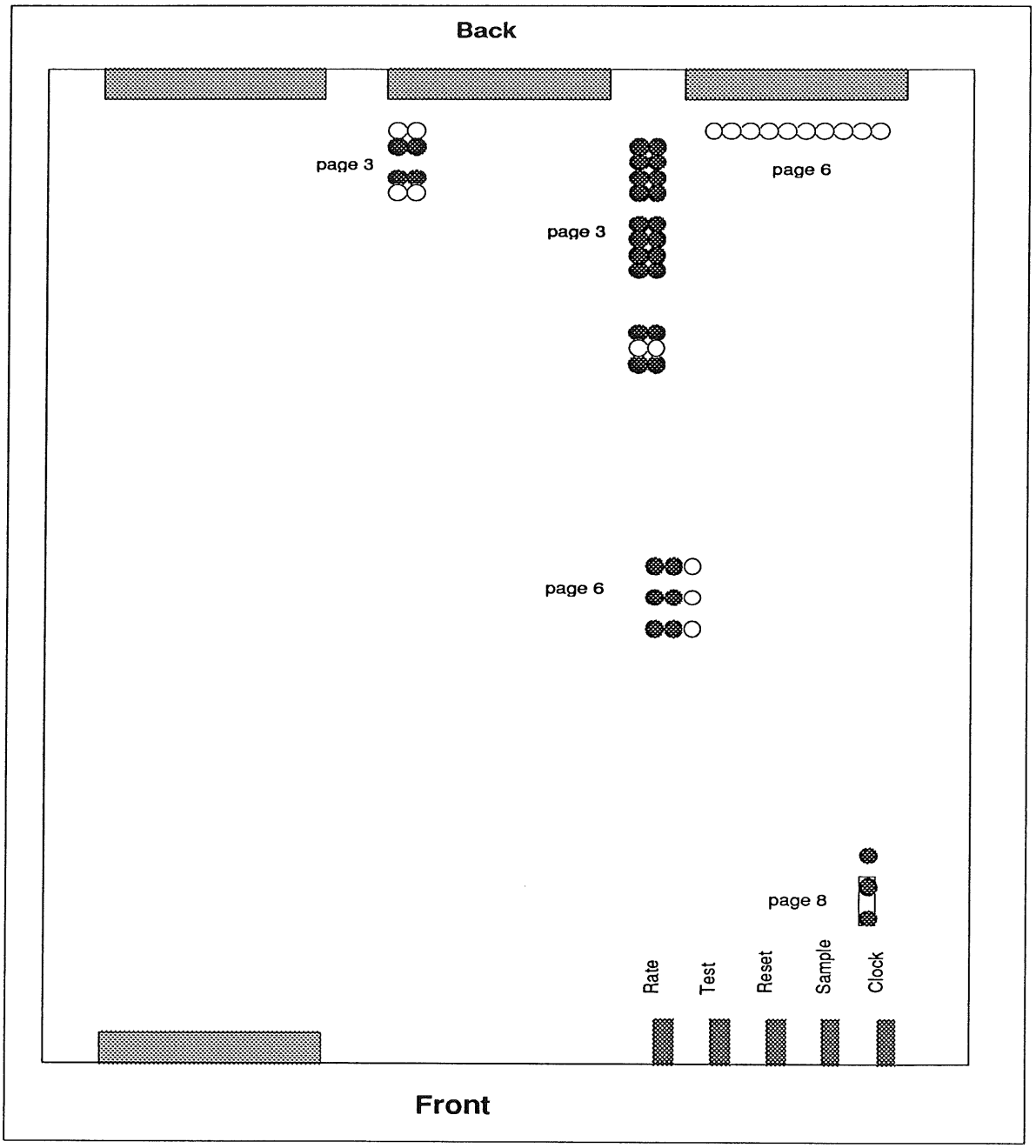
```
void Init_IO_Module()

/* Diese Procedure initialisiert die I/O-Karte.Kanal A wird auf out-
put,Kanal B auf input programmiert.Beide Kanaele werden ungepuffert ge-
lesen, das heisst sie kommen nicht in einen Latch.Pin H2 wird als
asserted edge sensitiver input programmiert. Das Status Bit H2S im Port
Status Register zeigt an, ob ein Signal angekommen ist. Geloescht wird
H2S, indem 1 eingelesen wird.Saemtliche Interrupts sind ausgeschal-
tet.*/

{
      IoAdress=(byte*)IoBasAdress;
                              /*Einlesen der Basisadress:0xB07F9000*/
      MMU32;
      IoAdress[PACR] = 0;     /*Port A Control Register:Submode erst
                              bestimmen,wenn H12,H34 */
      IoAdress[PBCR] = 0;     /*in PGCR festgelegt sind*/
      IoAdress[TCR]  = 0;      /*Timer Control Register:kein Timer
                              interrupt disable */
      IoAdress[PCDR] = 0xFF;
      IoAdress[PCDDR]= 0x1B;  /* Port C Data direction Register: Pins
                              Nr.0 + 1 werden auf output gestellt,
                              damit in PCDR Transceiverrichtung der
                              Kanaele A+B eingestellt werden kann*/
      IoAdress[PGCR] = 0x3F;  /*Port General Control Register: Mode
                              00,H34 enabled(Port B eingeschaltet),H12
                              enabled(Port A eingeschaltet)*/
                              H4-H0 Sense=0(Pin ist auf high volage
                              level wenn negated, auf low level voltage
                              wenn asserted) */
      IoAdress[PSRR] = 0x00;  /* no Interrupt support*/
      IoAdress[PCDR] = 0xfc;  /*Transceiverrichtung von A+B auf Out
                              put*/
      IoAdress[PADDR]= 0xff;  /* Port A DataDirection: Output */
      IoAdress[PBDDR]= 0xff;  /* Port B DataDirection: Output  */
      IoAdress[PIVR] = 128;   /* Port Interrupt Vektor Register: bein
                              haltet die upper order six bits der vier
                              Interruptvektoren */
      IoAdress[PACR] = 0x00;  /* Port A Control Register:Submode:00, H2
                              Control:100 (edge sensitive status
                              input,H2S is set on an asserted edge) */
      IoAdress[PBCR] = 0x00;  /* Port B Control Register:Submode:00, H4
                              Control:100 (edge sensitive status
                              input,H4S is set on an asserted edge) */
      IoAdress[PSR]  = 0xff;  /* Port Status register 0 setzen (ist
                              Flip Flop spule)*/
      MMU24;

}
```

VI.6 Pinbelegung der Eingänge des FADC F1001

Pin	Row C	Row B	Row A
1	VGND	VGND	VGND
2	X2 **15**	VGND	X0 **7**
3	X2* **16**	VGND	X0* **8**
4	VGND	VGND	VGND
5	X6	VGND	X4
6	X6*	VGND	X4*
7	VGND	VGND	VGND
8	Y2 **31**	VGND	Y0 **23**
9	Y2* **32**	VGND	Yo* **24**
10	VGND	VGND	VGND
11	Y6	VGND	Y4
12	Y6*	VGND	Y4*
13	VGND	VGND	VGND
14	TPE,X **3**	+5.0V **5**	TP0.X **39**
15	TPE.X* **4**	+5.0V **6**	TP0.X* **40**
16	VGND	VGND	VGND
17	TPE.Y	-5.2V **37**	TP0.Y
18	TPE.Y*	-5.2V **38**	TP0.Y*
19	VGND	VGND	VGND
20	X3 **19**	VGND	X1 **11**
21	X3* **20**	VGND	X1* **12**
22	VGND	VGND	VGND
23	X7	VGND	X5
24	X7*	VGND	X5*
25	VGND	VGND	VGND
26	Y3 **35**	VGND	Y1 **27**
27	Y3* **36**	VGND	Y1* **28**
28	VGND	VGND	VGND
29	Y7	VGND	Y5
30	Y7*	VGND	Y5*
31	VGND	VGND	VGND
32	VGND	VGND	VGND

Es ist nur Ring 1 angeschlossen. Die Zahlen besiehen sich auf die Pinnummer des Steckers bzw. die Nummer auf dem Vorverstärker nach der Driftkammer.

VI.7 Jumpereinstellung der FADC-Karte



VI.8 Register Offsets und spezielle Funktionen der Bift-Karte

```
/* Bift-Karte Definitionen */
/***************************/

/* Basis-Adresse der Bift Karte, kann mittels Jumper eingestellt wer-
den. Sie befindet sich im short I/O-Bereich des VME */

#define BIFT_BASE 0xB0FF1100


/* Adressen der verschiedenen Register */

#define BIFT_TIM  0           /* Timer */
#define BIFT_CON  1           /* darin befinden sich 16 Control Lines*/
#define BIFT_STAT 1           /* kann man Control Lines monotoren */
#define BIFT_BAC  2           /* Anzahl der Daten die in BIFT.BUF ein
                              gelesen wurden*/
#define BIFT_CYC  3           /* kann man einstellen, wieviele Daten
                              man auslesen will */
#define BIFT_BUF  4           /* Darin befinden sich die 16 Bit Daten*/
#define BIFT_Mac  A
#define BIFT_MAP  C
#define BIFT_AMR  10


/* Konstante um Kontroll Lines von BIFT.CON zu setzen */

#define BIFT_ALL  0x0004
#define BIFT_C0         0x0200
#define BIFT_C1         0x0800
#define BIFT_C2         0x1000
#define BIFT_S0         0x0200
#define BIFT_S1         0x0400
#define BIFT_S2         0x0800
#define BIFT_S3         0x1000
#define BIFT_S4         0x2000
#define BIFT_START      0x0400
#define BIFT_BUFRD      0x4000


#define bift_intern()
      {MMU32;bift[BIFT_CON]=(BIFT_ALL|BIFT_BUFRD);MMU24}
            /* bift_intern setzt Daten Richtung auf output -> nur aus
            lesen moeglich */

#define bift_extern()
      {MMU32;bift[BIFT_CON]=(BIFT_ALL|BIFT_START);MMU24}
            /* bift_extern setzt Daten Richtung auf input -> nur ausle
            sen moeglich */

#define bift_reset()
      {MMU32;bift[BIFT_BAC]=0;bift[BIFT_CYC]=0;bift[BIFT_CON]=0;MMU24}
            /* loescht Inhalt der Bift Karte */
```

VI.9 Form der Datenfiles

qtarn.result

```
     file  event    draht   time        phi          z          x
neues event
     0     0        1       101.563     0.173        -3.025     -0.380
     0     0        1       101.563     0.173        3.348      -0.020
     0     0        2       238.125     0.219        -0.823     0.148
     0     0        2       238.125     0.219        0.470      0.253
     0     0        3       1045.313    0.350        -3.640     0.810
     0     0        3       045.313     0.350        -200.000   -100.000
neues event
neues event
          0 2       3       1380.625    0.422        -4.611     0.808
          0 2       3       1380.625    0.422        -200.000   -100.000
neues event
          0         3       3           812.813      -0.386     -2.969
     0.810
          0         3       3           812.813      -0.386     -200.000
     -100.000
neues event
neues event
neues event
          0 6       2       466.875     0.219        -1.495     0.046
          0 6       2       466.875     0.219        1.154      0.312
          0 6       3       361.250     0.186        -1.676     0.754
          0 6       3       361.250     0.186        -200.000   -100.000
```

drift.result

Bei einer langen Zeile haben die Werte folgende Bedeutung:

1= z an Stelle x=0

2= α

3= χ^2

4= Anzahl Drähte für Fit

5= Draht 0...3(1=benutzt,0=nicht)

6= φ der benutzten Drähte

Sonst gilt :
1= event
2= Anzahl Tracks

```
     0                  0
     0                  0
     0                  0
     0                  0
     2                  0
     3                  0
     3                  0
     3                  0
     6                  1
     -1.483             -0.2503          999.000          2
     0 0 1 1            2.000            2.000            0.219          0.186
```

```
     6                0
     8                0
     8                0
     8                0
    11                0
    11                0
    13                0
    13                0
    15                1
  -2.009             -0.2314          999.000          2
0 0 1 1              2.000            2.000            0.381          0.432
    16               1
  -1.868             -0.2390          999.000          2
0 0 1 1              2.000            2.000            0.199   0.247
```

prop.result

event	x	y	z	Winkel
0	0.000000	0.000000	2000.000	-10.00000
1	0.000000	0.000000	2000.000	-10.00000
2	176.7219	59.59541	72.69941	0.1602677
3	0.000000	0.000000	2000.000	-10.00000
4	179.2786	-51.39490	30.87844	-0.2678419
5	0.000000	0.000000	2000.000	-10.00000
6	0.000000	0.000000	2000.000	-10.00000
7	0.000000	0.000000	2000.000	-10.00000
8	0.000000	0.000000	2000.000	-10.00000
9	0.000000	0.000000	2000.000	-10.00000
10	0.000000	0.000000	2000.000	-10.00000
11	0.000000	0.000000	2000.000	-10.00000
12	0.000000	0.000000	2000.000	-10.00000
13	178.0024	55.65409	79.26303	-5.2583061E-02

Output combine:

jeweils 1.Zeile : Werte von Driftkammer:

event z Mittelwert φ Winkel in Grad

jeweils 2.Zeile: Werte Proportional-Kammer

event z y Winkel in Grad

39	3.459000	0.1793333	-11.50500
39	138.9234	-48.19493	7.284029
59	-2.087000	0.2420000	-30.16052
59	67.00141	5.451355	-2.368027
89	-3.000000	0.1783333	-27.94890
89	59.08542	-49.70645	-1.507437
200	3.009000	0.2333333	-6.285352
200	127.0788	-13.74597	1.507437
301	-1.648000	0.2183333	-13.85413
301	83.40991	-49.78431	-9.601991
328	3.714000	0.2016667	-21.02757
328	142.9597	-49.58150	5.795891
358	-2.966000	0.2616667	-9.173061
358	64.62888	-20.97639	4.941859

combine.hist :

folgende Daten sind geplottet:

```
 .  NO                        TITLE                          ID  B/C  ENTRIES
DIM   NCH
A      LOWER       UPPER       ADDRESS LENGTH .
 .    1  zprop vs. zdrift                                     1  32        50
2  X    5
0  -0.100E+02   0.100E+02       99429   2362 .
Y    4
0   0.000E+00   0.160E+03       97224   2196 .
 .    2            PROJECTION X                                  32        50
1  X    5
0  -0.100E+02   0.100E+02       92374     72 .
 .    3            PROJECTION Y                                  32        50
1  Y    4
0   0.000E+00   0.160E+03       92165     62 .
 .    4  alpha vs. zdrift                                     2  32        50
2  X    5
0  -0.100E+02   0.100E+02       97199   2150 .
Y    3
6  -0.900E+02   0.900E+02       95202   1988 .
 .    5            PROJECTION X                                  32        50
1  X    5
0  -0.100E+02   0.100E+02       92301     72 .
 .    6            PROJECTION Y                                  32        50
1  Y    3
6  -0.900E+02   0.900E+02       92106     58 .
 .    7  alpha der z-drift                                    10  32        50
1  X    3
6  -0.900E+02   0.900E+02       95178     83 .
 .
 .    8  alpha der prop                                       11  32        50
1  X    3
6  -0.900E+02   0.900E+02       95094     82 .
 .    9  phi der z-drift                                      15  32        50
1  X    5
0  -0.100E+01   0.100E+01       95010     96 .
 .   10  chisq der z-drift                                    20  32        50
1  X    4
0   0.000E+00   0.100E+02       94911     87 .
 .   11  y der Prop.kammer                                    30  32        50
1  X    5
0  -0.100E+03   0.100E+03       94822     97 .
 .   12  zprop vs. yprop                                     31  32        50
2  X    5
0  -0.100E+03   0.100E+03       94722   2362 .
Y    4
0   0.000E+00   0.160E+03       92517   2196 .
 .   13            PROJECTION X                                  32        50
1  X    5
0  -0.100E+03   0.100E+03       92228     72 .
 .   14            PROJECTION Y                                  32        50
1  Y    4
0   0.000E+00   0.160E+03       92043     62 .
 .   15  ntracks per event, alle drift events                100  32       550
1  X    2
0   0.000E+00   0.200E+02       92489     71 .
```

VI.10 Form der Shells

Copy.sh:

```
#
# copy eventfiles
rm -f zdrift.dat
rm -f prop.dat
@ i = 0
while ($i < 95)
        @ i += 1
        cat ~kindler/qta/Hoehen2/20.Dec.89_16.14.48_{$i} >>zdrift.dat
        cat ~kindler/qta/Hoehen2/20.Dec.89_16.14.48_{$i}_MWC >>prop.dat
#       echo $i
end
#
```

rundrift.sh:

```
#
  setenv FOR002 zdrift.dat
  setenv FOR003 qtarn.log
  setenv FOR004 qtarn.result
#
# moegliche switches: d   mach bildli
#                     l   minimale laenge fuer puls
#                     t   Schwelle fuer puls
#                     f   filenummer
#                     z   max zeitdiff zw linker und rechtem FADC
#                         (Einheit : 1/32 Kanal)
  qtarn l5 t30 f0 z64
#
  echo "qtarn o.k., jetzt 0"
#
# moegliche switches
#  c : oberer chisq Schnitt, nur fuer Spuren mit mehr als 2 hits
#  d : obere deltaphi Schnitt, fuer alle Spuren
#
  0 d0.2 c2. > drift.result
  exit(0)
```

runprop.sh

```
#
PropAuswertung prop.dat prop.centers > PropAuswertung.log
#
#
echo "PropAuswertung o.k, jetzt prop "
setenv FOR002 prop.centers
prop > prop.result
#
```

runcombine.sh

```
#
setenv FOR001 drift.result
setenv FOR002 prop.result
#
setenv FOR007 combine.hist
#
# moegliche switches: c    Schnitt auf chisq   (real)
#                     p    Schnitt auf phi unterer Wert (real)
#                     w    minimale Anzahl wires (2 oder 3) (integer)
#                     t    maximale Zahl der Tracks pro event (integer)
#                      a    maximaler Betrag des propkammer winkels (real)
# (Winkel im Bogenmass, 0 = senkrecht)
#
combine  p0. w3 t2 a0.087
#
```

VI.11 Literaturangabe

[1]	G.Wolf	HERA: Physics, Machine and Experiments DESY preprint 86-089	Hamburg	1986
[2]	P.Truöl	HERA-Physik Vorlesungsunterlagen	Zürich	1989
[3]	R.Schmidt	Vorbereitende Studien zum Bau und Betrieb des Full-Size-Prototypen Diplomarbeit	Hamburg	1989
[4]	L.Russek	Untersuchungen an einer kleinen Driftkammer im Magnetfeld mit der kosmischen Höhenstrahlung Diplomarbeit	Hamburg	1989
[5]	W.Fleck	Bestimmung der Doppelspurauflösung von Driftkammern Diplomarbeit	Hamburg	1989
[6]	P.Robmann	Entwicklung und Bau eines zylindrischen Driftkammer-Prototyps für den H1 Detektor Diplomarbeit	Zürich	1988
[7]	G.Merki	Planung und Test einer Driftkammer Diplomarbeit	Zürich	1987
[8]	F.Sauli	Principles of Operation of Multiwire Proportional and Drift Chambers CERN 77-09	CERN	1977
[9]	K. Kleinknecht	Detektoren für Teilchenstrahlung Teubner Studienbücher	Stuttgart	1987
[10]	P.Eschle	Computerunterstützte Datenerfassung und Datenanalyse für ein Channeling-Experiment (Diplomarbeit)	Zürich	1988
[11]	Zeitschrift Elektronik	Der VMEbus Franzis-Verlag	München	1986
[12]	Motorola	MC68230 Parallel Interface/Timer	Texas	1983
[13]	P.Robmann	Construction and Analysis of a Prototype z-Drift Chamber for the H1 Experiment NuclearInstruments and Methods in Physics Research A277 368-378	Zürich	1988
[14]	Addison-Wesley	Inside Macintosh Volume I-V Addison-Wesley Publishing Company,Inc.	California	1985
[15]	Hayden	Macintosh Revealed Volume I-III Addison-Wesley Publishing Company,Inc.	Indiana	1988

[16]	ResEdit Version 1.2	Apple Computer, Inc	California	1989
[17]	Apple Computer	Macintosh Programmer's Workshop Development Environment Volume I-IV	Cupertino USA	1989
[18]	H.Schildt	C The Complete Reference Osborne McGraw-Hill	Berkeley	1987
[19]	Kelley/Pohl	C Grundlagen und Anwendungen Addison-Wesley Publishing Company,Inc.	Bonn	1984
[20]	H.Schildt	C Made Easy Osborne McGraw-Hill	Berkeley	1885
[21]	B.G.Taylor	TheMICRON User Manual EP Division	CERN	1988
[22]	B.G.Taylor	The MacVEE Hardware User Manual EP Division	CERN	1988
[23]	W.J.Haynes	VME_TOOLS	DESY	1989
[24]	H.Klär	Triggerbare Signalgeneratoren für einen Pipline TDC (Diplomarbeit)	DESY	1989
[25]	E.Lohrmann	Hochenergiephysik Teubner Studienbücher	Stuttgart	1979
[26]	H1 Collaboration	Technical Proposal for the H1 Detector	Hamburg	1986
[27]	.....	Private Auskunft		

VI.12 Verdankung

Ich möchte allen Personen, die mir beim Zustandekommen dieser Arbeit geholfen haben, herzlich danken. Stellvertretend für alle, die mir während den letzten 14 Monaten in irgend einer Art hilfreich zur Seite standen, möchte ich folgende Personen namentlich erwähnen:

Prof. P.Truöl:	Für die Vergabe des Themas.
J.Riedlberger:	Es war eine Freude, mit ihm Fussball zu spielen.
P.Robman:	Er hat mich in einer schwierigen Situation über den Berg gezogen.
B.Pümpin:	Ihre Hilfe am Schluss war Gold wert.
S.Eichenberger:	Ich werde ihn doch noch vom Mac überzeugen.
N.Scaltri:	Sie hat zu Hause alle meine erlebten Frustrationen (meistens) geduldig ertragen.
W.Fässler:	Er hatte für alle "kleinen" Probleme immer eine Lösung.
K.Müller:	Es war schön, mit ihr zusammen Physik zu studieren.
P.Eschle:	Er hat mir geduldig seine Proportional-Kammer erklärt (und noch vieles mehr).

Vor allen anderen aber möchte ich Ueli Straumann danken. Ohne ihn wäre diese Arbeit nie entstanden. Er hat mir eine Ahnung davon gegeben, was ein guter Physiker alles wissen und können sollte. Zusätzlich konnte man mit ihm nach der Physik auch noch ein Glas Bier trinken, was ich sehr zu schätzen wusste.